ROBUST MULTICRITERIA OPTIMIZATION OF SURFACE LOCATION ERROR AND MATERIAL REMOVAL RATE IN HIGH-SPEED MILLING UNDER UNCERTAINTY

By

MOHAMMAD H. KURDI

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
UNIVERSITY OF FLORIDA

2005

To my Mom and Dad.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Tony Schmitz for his advice and generous financial support of my research. I would like to thank Dr. Haftka for his expert advice and inspiring questions. I would like to thank Dr. Mann for introducing me to the field of time finite elements. I would like to thank the committee members Dr. Schmitz, Dr. Haftka, Dr. Mann, Dr. Schuller and Dr. Akcali for their advice, time and effort.

In completing my research I was lucky to be a member of the Machine Tool Research Center where I had the opportunity to work with intelligent and hard working graduate students. I would like to thank all fellow members for their helpful suggestions and interactions. Also, I would like to thank Ms. Christine Schmitz for taking the time to edit the dissertation draft.

I would like to thank my wife Carolina for her continued support and encouragement. I would like to thank my daughters Alanis and Alia for bringing laughter and joy to my life. Finally, I would like to thank my mom and dad for their endless encouragement and support.

TABLE OF CONTENTS

page

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

A	slotting tranformation matrix
C_x	modal damping in x-direction
C_y	modal damping in y-direction
E	number of elements
$F(\vec{x})$	vector of objective functions
F^0	utopia point
$\overline{F}$	average cutting force
$\overline{F}_x$	average cutting force in x-direction
$\overline{F}_{xc}$	average normal cutting force in x-direction
$\overline{F}_{xe}$	average normal edge cutting force in x-direction
$\overline{F}_y$	average cutting force in y-direction
$\overline{F}_{yc}$	average tangential cutting force in y-direction
$\overline{F}_{yc}$	average tangential edge cutting force in y-direction
$\mathbf{I_Q}$	Identity matrix of size Q
$\boldsymbol{K_c}$	cutting force coefficients matrix defined in Appendix A
K_t	tangential cutting force coefficient
K_n	normal cutting force coefficient
K_{te}	edge tangential cutting force coefficient
K_{ne}	edge normal cutting foce coefficient
K_x	modal stiffness in x-direction
K_y	modal stiffness in y-direction
L	sample size
M_x	modal mass in x-direction
M_y	modal mass in y-direction
MRR	material removal rate
N	number of teeth on the cutting tool
PSO	particle swarm optimization

Q	number of experimental runs
R	cutter radius
R^2_{adj}	adjusted coefficient of determination
SLE	surface location error
SQP	sequential quadratic programming
U	used for utility function
U_e	expanded uncertainty
$\vec{X}$	feasible design space
$\vec{X}(t)$	two-element position vector
X_i	milling model ith input parameter
Y_i	vector of observations in ith response
$\vec{Z}$	feasible criterion space
Z_i	$Q \times p_i$ matrix of rank p_i
a	radial depth of cut
b	axial depth of cut
b_j	set of goals for objective functions
b_{lim}	maximum stable axial depth
c	chip load
d_j	deviation from the goals
e	MRR constraint limit
$\vec{f}_0$	cutting force coefficients vector defined in Appendix A
f_{SLE}	surface location error objective function
f_{MRR}	material removal rate objective
g_j	set of inequality constraints
g_λ	absolute value of maximum characteristic multiplier
h_l	set of equality constraints
h	step size used to estimate numerical derivative
p	parameter in exponential weighted criterion
p_i	rank of Z_i matrix
r	number of response variables
r_{xy}	correlation coefficient between x and y
u_c	combined standard uncertainty
$\vec{w}$	vector of weights (preferences)
x	x-direction

$\vec{x}$	vector of design variables
y	y-direction
α	parameter used in homotopy method
$\boldsymbol{\beta}$	vector of unknown constant parameters
$\hat{\boldsymbol{\beta}}$	unbiased estimate of $\boldsymbol{\beta}$
δ	spindle speed perturbation (half of bandwidth)
ε	absolute error limit
ε_i	random error vector associated with i^{th} response
Δ	variance-covariance matrix
$\vec{\lambda}$	system characteristic multipliers
ϕ	radial depth angle
ϕ_{st}	radial depth angle at start of cut
ϕ_{ex}	radial depth angle at end of cut
Ω	spindle speed
Σ	covariance matrix
σ_{ij}	i, j element of covariance matrix Σ
$\hat{\sigma}_{ij}$	estimate of σ_{ij}
τ	tooth passing period
ζ	damping factor

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

ROBUST MULTICRITERIA OPTIMIZATION OF SURFACE LOCATION ERROR AND MATERIAL REMOVAL RATE IN HIGH-SPEED MILLING UNDER UNCERTAINTY

By

Mohammad H. Kurdi

August 2005

Chair: Tony L. Schmitz
Cochair: Raphael T. Haftka
Major Department: Mechanical and Aerospace Engineering

High-speed milling (*HSM*) provides an efficient method for accurate discrete part fabrication. However, successful implementation requires the selection of appropriate operating parameters. Balancing the multiple process requirements, including high material removal rate, maximum part accuracy, chatter avoidance, and adequate surface finish, to arrive at an optimum solution is difficult without the aid of an optimization framework.

Despite the attractive gain in productivity that *HSM* offers, full realization of the benefits is dependent on the proper selection of cutting parameters. Parameters selected must achieve the required productivity while maintaining an acceptable accuracy. Milling models are used to aid in the proper selection of these cutting parameters. They provide information on whether a cutting condition is stable and/or predict the surface accuracy. However, this selection is rather tedious, costly and time consuming and might not even

provide an optimum solution. Parameters are selected based on experience until a point is found that provide the productivity and surface accuracy required. Difficulties encountered in this selection process include sensitivity of surface accuracy to cutting parameters, uncertainties in several parameters in the milling model and the computational effort needed to account for stability and surface accuracy. Therefore, balancing the multiple requirements, including high material removal rate, minimum surface location error and chatter avoidance, to arrive at an optimum solution is difficult without the aid of optimization techniques.

In this dissertation a robust optimization algorithm that accounts for the inherent process uncertainty and surface location error sensitivity is developed. Two optimization criteria are considered, namely, surface location error and material removal rate under the stability constraint. The trade off curve of surface location error versus material removal rate is calculated for the mean values of input parameters, as well as for a confidence level in the stability boundary. An experimental validation of the robust optimization algorithm is also conducted, including an experimental validation of the variation of the cutting forces as a function of spindle speed. The confidence level in the axial depth limit and surface location error prediction is found using two methods: 1) sensitivity analysis; and 2) sampling methods. The sensitivity study highlights the most significant factors affecting process stability and surface location error. The effect of input parameters correlation is included in the confidence level predictions using Monte Carlo and Latin Hyper-Cube sampling methods.

CHAPTER 1
INTRODUCTION

Justification of Work

Intense competition in manufacturing places a continuous demand on developing cost-effective manufacturing processes with acceptable dimensional accuracy. High-speed milling, *HSM*, offers these benefits provided appropriate operating parameters are selected. Some typical applications include, but are not limited to, orthopedic surgery [1], end milling (pocketing) of airframe panels [2] and ball end milling of stamping dies [3, 4] in automotive manufacturing.

Despite the attractive gain in productivity that *HSM* offers, full realization of the benefits is dependent on the proper selection of cutting parameters. Parameters selected must achieve the required productivity while maintaining an acceptable accuracy. Milling models are used to aid in the proper selection of these cutting parameters. They give us information on whether a cutting condition is stable and/or they predict the surface accuracy. However, this selection is rather tedious, costly and time consuming and might not even provide an optimum solution. Parameters are selected based on experience until a point is found that provides the productivity and surface accuracy required. Difficulties encountered in this selection process include sensitivity of surface accuracy to cutting parameters, uncertainties in several parameters in the milling model and the computational effort needed to account for stability and surface accuracy. Therefore, balancing the multiple requirements, including high material removal rate, f_{MRR} ,

minimum surface location error $|f_{SLE}|$ and chatter avoidance, to arrive at an optimum solution, is difficult without the aid of optimization techniques.

Literature Review

The literature review proceeds with a summary of previous implementations of optimization methods in machining, with particular attention to high-speed milling and multi-objective optimization. Also, a review of milling models for stability and surface location error is provided.

Optimization in Machining

Previous research in machining process optimization [5] has focused on mathematical modeling approaches to determine optimal cutting parameters with regard to various objective functions. Three main objectives have been recognized: 1) maximum production rate or minimum cycle time [6-9]; 2) minimum cost [10-21]; and 3) maximum profit [12, 22], or a combined criterion based on a weighted sum of these [23, 24].

The machining optimization problem can be formulated using deterministic and probabilistic approaches [11, 25]. Several optimization techniques were used to handle both formulations. For the deterministic approach they include linear and nonlinear programming techniques [9, 15, 26, 27], while for the probabilistic approach chance-constrained programming can be used [17, 28]. Other optimization techniques used in machining include graphical optimization [12, 22], polynomial geometric programming [6, 18-20, 29, 30], geometric programming [10] based on quadratic posylognomials (QPL) [31], goal programming with linear [32, 33] and nonlinear [34] goals, fuzzy optimization [35], and global search methods such as particle swarm optimization [21] and simulated annealing [16].

The machining optimization literature can also be classified according to different constraints and design variables handled. Several authors [7, 14] considered cost optimization for single-pass milling and turning [10, 17, 19, 20, 29]. The range of constraints considered are machine tool constraints, such as cutting speed and feed rate, tool dynamics constraints such as cutting force, power and stability, and product constraints such as surface roughness. In reference, [17] some of the constraints considered are of probabilistic nature. Also, multi-pass peripheral and end milling to maximize production rate are considered [8] under a range of constraints with relevance to rough milling such as the machine tool limiting power, torque, feed force and feed-speed boundaries while in another work. In addition to the previous constraints, arbor rigidity and deflection are used [6].

High-speed Milling Optimization

Few references are found on optimization of high-speed milling. The concept of adaptive learning (polynomial network) [16] is used to construct a machining model. Simulated annealing was then used to minimize production cost for rough high-speed machining operations for three cutting condition parameters namely cutting speed, chip load and axial depth of cut. A similar study was done for low speed milling [21] where an artificial neural network was used to build the machining model. However, particle swarm optimization was used to optimize production cost under machine, tool and product constraints.

Multi-objective Optimization

Multi-objective optimization (MOO) addresses the issue of competing objectives using concepts first introduced by Edgeworth [36] then expanded and developed by Pareto [37], the French-Italian economist who established an optimality concept in the

field of economics based on multiple objectives. A Pareto front [38] is generated that allows designers to trade off one objective against another.

In the area of machining, Jha [24] studied two objective function optimization based on cost and rate of production where example constraints were machine power, cutting speed limitations, depth of cut, and table feed. The two objectives were combined using weights. Koulams [28] studied single-pass machining considering the influence of tool chatter failure where a tool failure probability function effect was added as a penalty cost function to the objective function.

Stability and Surface Location Error

As explained earlier, the full exploitation of *HSM* demands mathematical models to predict stability and surface location error. An unstable milling process is caused by a phenomenon called chatter. Among the first to describe chatter is Taylor, [39] who described chatter as "the most obscure and delicate of all problems facing the machinist." Chatter [40] is a self-excited vibration that occurs if the chip width is too large with respect to the dynamic stiffness of the system. It causes undulations in the machined surface (poor surface finish) and could result in tool breakage. Extensive work has been done to generate stability boundaries or lobes. The lobes define a region below which chatter is nonexistent. Two approaches are used to generate these lobes: 1) analytical [41] with a continuous cutting model or with an interrupted cutting model [42]; and 2) time domain simulation [43, 44].

Surface location error is defined as the error in the placement of the milling cutter teeth when the surface is generated. This error depends on the interaction of work-piece/tool dynamic stiffness and the cutting forces. The correct prediction of this error depends on correct prediction of the cutting forces and resulting deflections. Mechanistic

models can be used to estimate these forces. The cutting force is found by summing the forces acting on incremental sections of a helical cutting edge [45, 46], then the surface location error is computed based on the static stiffness of the tool [47]. However, the effect of the deflection of the cutter on the cutting forces is not included. In an improvement of the previous model, the static deflection is fed back to correct the cutting forces [48, 49]. A more realistic regenerative force model [50] considered the effect of undulations in the surface generated by previous tooth passage on the next tooth passage. In this model the dynamic deflection of the tool imprints waviness on the generated surface. Using time domain simulation, surface location error, cutting forces and stability lobes are predicted. An improvement on this model considered [42, 51-53] interrupted cutting as a factor influencing the stability lobes and surface location error. A newly developed method uses time finite element analysis (TFEA) to model the governing time delayed differential equation [54-58]. Regenerative cutting forces and dynamic deflection of the tool are all implicitly included in the governing differential equation. The advantage of this method is that it concurrently provides surface location error and stability information on the milling process in a semi-analytical manner. In this method the governing differential equation is modeled by dividing the time in the cut into a number of elements, where displacement and velocity continuity are enforced between each element. A discrete linear map is formed by mapping the time in the cut to free vibration. The eigenvalues of the discrete map determine the stability boundaries, whereas fixed points of the dynamic map determine surface location error (f_{SLE}).

Scope of Work

The purpose of this dissertation is to use optimization as a tool to efficiently determine preferred and robust operating conditions in *HSM*, considering multiple objectives. Although known optimization methods and machining models will be applied, there are a number of innovative aspects of this research. First, proper formulation of the objective functions to account for practical application of the preferred conditions is necessary. The formulation should account for uncertainty in the milling model and sensitivity of objective(s) to process variables. Uncertainty has not previously been considered. Second, two objectives are simultaneously optimized: surface location error $|f_{SLE}|$ and material removal rate, f_{MRR}. Stability and side bounds of design variables are considered as constraints. Prior research has focused only on the empirical tool life, not the unavoidable milling dynamics and the inherent limitations they impose. The tradeoff curve (Pareto front) [38, 59] of f_{MRR} and $|f_{SLE}|$ is generated based on nominal experimental model parameters. Experimental case studies are conducted to verify the validity of the Pareto front. The uncertainty in the milling model is addressed using Monte Carlo simulation and/or sensitivity analysis, where a confidence interval is applied to the stability limit. The uncertainty of different input parameters such as cutting force coefficients, tool/work-piece dynamic parameters and milling process parameters are considered in the uncertainty prediction. This uncertainty is used in the selection of a robust design that would allow a venue for the practical application of the stability lobe theory at the shop floor.

The dissertation organization proceeds as follows: Chapter 2 gives a general description of multi-objective optimization; Chapter 3 describes Pareto front generation formulation of the optimization problem, optimization methods and case studies; Chapter 4 provides the uncertainty analysis of stability and surface location error; Chapter 5 describes the robust optimization algorithm and presents some practical case studies to verify stability lobes and selected design points on the Pareto front. Chapter 6 summarized the results and outlines future work in this area.

CHAPTER 2
MULTI-OBJECTIVE OPTIMIZATION

Fundamental Concepts in Multi-Objective Optimization

Optimization is an engineering discipline where extreme values of design criteria are sought. However, quite often there are multiple conflicting criteria that need to be handled. Satisfying one of these criteria comes at the expense of another. Multi-objective optimization deals with such conflicting objectives. It provides a mathematical framework to arrive at an optimal design state which accommodates the various criteria demanded by the application.

This chapter begins with a comparison of single- and multiple-objective optimization. Next, the definition of the multi-objective optimization problem and terms are explained. Then, a summary of multi-objective optimization methods is presented. Finally, reasons are given for the choice of the multi-objective optimization method.

Single and Multi-objective Optimization

In single objective optimization one is faced with the problem of finding the optimum of the objective function. For example considering the decision making involved in an investment (Figure 1). There are several possible designs in the feasible domain (A, B and 1-6). These designs are mapped from the design space Figure 1 (b) into the criteria space Figure 1 (a). In the design space there are two design variables x_1 (spindle speed) and x_2 (axial depth) where the feasible domain is limited by the

constraint. If we are only concerned about profit with no regard to risk (profit is our single objective), then point B would correspond to the maximum profit optimum design. A risk averse investor would choose *risk* as an objective function. The optimum design for the risk objective would correspond to point A. Depending on the objective function, constraints, and design variables, different techniques are used to solve for the single-objective optimum. However, in multi-objective optimization, a vector of objectives needs to be optimized. For the investment example, two objectives are considered. In this two objective case there is no unique optimum, rather a set of optimum solutions is found. In Figure 1**,** for instance, points A, B and 5-6 are all candidate solutions. Depending on the decision maker's risk aversion, a single solution can be chosen from that set.

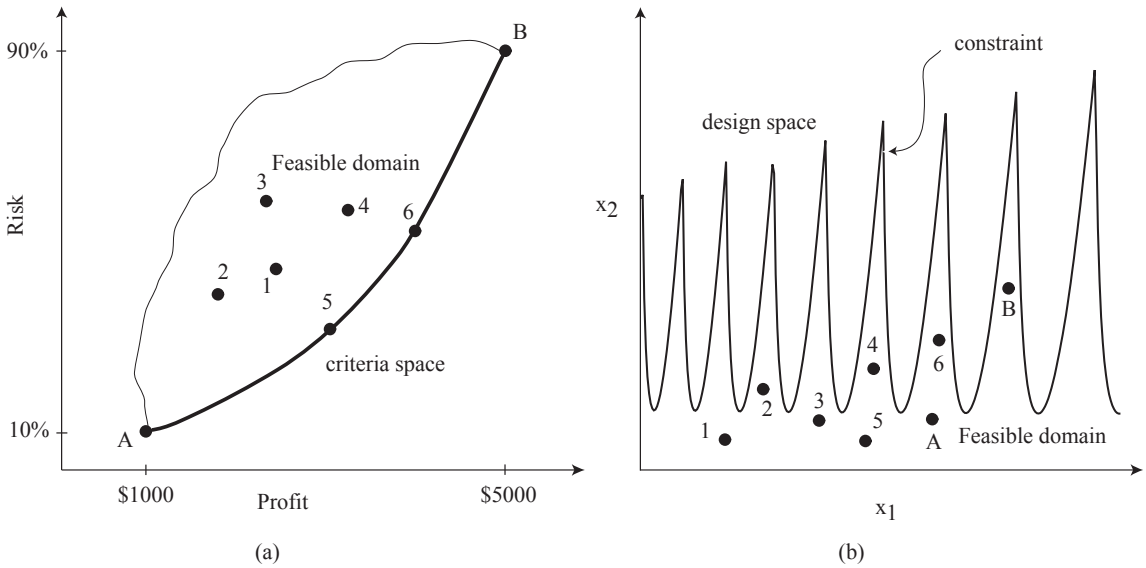

Figure 1. (a) Typical Pareto front in the criteria space (b) Design variables x_1 and x_2, and constraint in the design space.

The similarity between single- and multi-objective optimization makes it possible to use the same optimization algorithms as for the single-objective case. The only

required modification is to transform the multi-objective problem into a single one. This may be accomplished in a number of ways, such as introducing a vector of preferences, $\vec{w}$, to get a single objective as a weighted sum, or by solving one of the objectives for a different set of limits on the other objectives [60-62]. In any case, a set of optimal solutions are found rather than a single one. It is worth noting, however, that when the objective functions are non-conflicting, the optimal set reduces to a single solution rather than a set. This can be related to the commodity example. For instance, if we want to maximize both cost and quality, then solution B is the only one.

Definition of Multi-Objective Optimization Problem

The mathematical representation of the multi-objective optimization problem is formulated as follows:

$$\begin{aligned} &\textit{Minimize } F(\vec{x}) = \left[F_1(\vec{x}), F_2(\vec{x}), ..., F_k(\vec{x})\right]^T \\ &\textit{subject to } g_j(\vec{x}) \le 0, \quad j = 1, 2, ..., m \\ &\qquad\qquad h_l(\vec{x}) = 0, \quad l = 1, 2, ..., e \end{aligned} \quad , \tag{2.1}$$

where subscript *k* denotes the number of objective functions F, *m* is the number of inequality constraints and *e* is the number of equality constraints; and $\boldsymbol{x} \in E^n$ is the vector of design variables, where *n* is the number of independent design variables.

Definition of Terms

The *feasible design space (inference space),* $\vec{X}$ *, i*s defined as the set of design variables that satisfy the constraint set, or

$$\left\{\vec{x} \middle| g_j(\vec{x}) \le 0, \quad j = 1, 2, ..., m; \text{ and } h_l(\vec{x}) = 0, \quad l = 1, 2, ..., e \right\}. \tag{2.2}$$

The *feasible criterion space,* $\vec{Z}$, (often called the cost space or attainable set) *i*s defined as the set of cost functions $\vec{F}(\vec{x})$ such that $\vec{x}$ maps to a point in the feasible design space $\vec{X}$ or $\left\{\vec{F}(\vec{x})\middle|\vec{x}\in\vec{X}\right\}$.

The *preferences* refer to the decision maker's opinion in terms of points in the criterion space. The preferences can be set *a priori* (before solution set is obtained) or *a posteriori* (after solution set is obtained).

The *preference function* is an abstract function of points in the criterion space which perfectly satisfies the decision maker's preferences.

The *utility function* is an amalgamation of individual utility functions of each objective that approximates the preference function, which typically cannot be expressed in mathematical form. The formation of a utility function requires insight into the physical aspects of each objective. This may require finding the *Pareto front* (explained next) in order to properly formulate the utility function.

A *utopia point* is a point $F^0 \in Z^k$ that satisfies $F_i^0 = minimum\left\{F_i(\vec{x})\middle|\vec{x}\in\vec{X}\right\}$ for each $i = 1, 2..., k$.

Pareto Optimality

The multi-objective optimization problem has more than one global optimum. The predominant concept in defining an optimal point is that of Pareto optimality [37] which is defined as follows: a point, $x^* \in \vec{X}$, is Pareto optimal if there does not exist another point, $x \in \vec{X}$, such that $\vec{F}(x) \le \vec{F}(x^*)$, and $F_i(x) < F_i(x^*)$ for at least one function. That is the set of Pareto optimal points dominates any other optimal set. This can be defined by the *domination* relation [60], where a vector $\vec{x}_1$ dominates a vector $\vec{x}_2$ if: $\vec{x}_1$ is

at least as good as $\vec{x}_2$ for all the objectives, and $\vec{x}_1$ is strictly better than $\vec{x}_2$ for at least one objective. To better understand the domination relation, or Pareto optimality, an example is provided [63] (Figure 2). A two-objective problem of maximizing f_1 and minimizing f_2 is addressed. Table 1 presents the set of solutions, classified with respect to each other. A solution P is designated as +,- or = depending on whether it is better, worse or equal to a solution Q for the corresponding objective. For example, comparing solutions A and B, we find that solution A is worse for f_1 (maximizing f_1) compared to B, therefore it is designated as (-) for objective f_1. Also, comparing objective f_2 we find that solution A is worse than B (-). Now for a solution to belong to the non-dominated set it must be as good as the other solutions for both objectives and it must be strictly better for at least one objective. Considering solution A in Figure 2 we see it is worse than all other solutions (dominated); solution B is also worse than C for both objectives (dominated). Solution C is not dominated by point E (couple (+,-) at the intersection of the row E and the column C) and it does not dominate point E (couple (-,+) at the intersection of the row C and the column E), therefore points C and E are non-dominated. Solution D is worse than C for both objectives therefore solution D is dominated.

Multi-objective Optimization Methods

As explained earlier the solution to a multi-objective optimization problem is a Pareto optimal set that gives a tradeoff between the different objective functions considered. Depending on the decision maker's preferences, a solution is selected from that set. Therefore multi-objective optimization methods can be categorized according to how the designer articulates his preferences (by order or by importance of objectives). This includes three cases: *a priori*, *a posteriori,* and progressive articulation of

preferences. A brief overview of the methods used is outlined. For a detailed description of the methods the reader is referred to reference [64].

Table 1. Classification of solutions

Solutions	A	B	C	D	E
A		(-,-)	(-,-)	(-,-)	(-,-)
B	(+,+)		(-,-)	(-,=)	(-,=)
C	(+,+)	(+,+)		(+,+)	(-,+)
D	(+,+)	(+,=)	(-,-)		(-,=)
E	(+,+)	(+,=)	(+,-)	(+,=)	

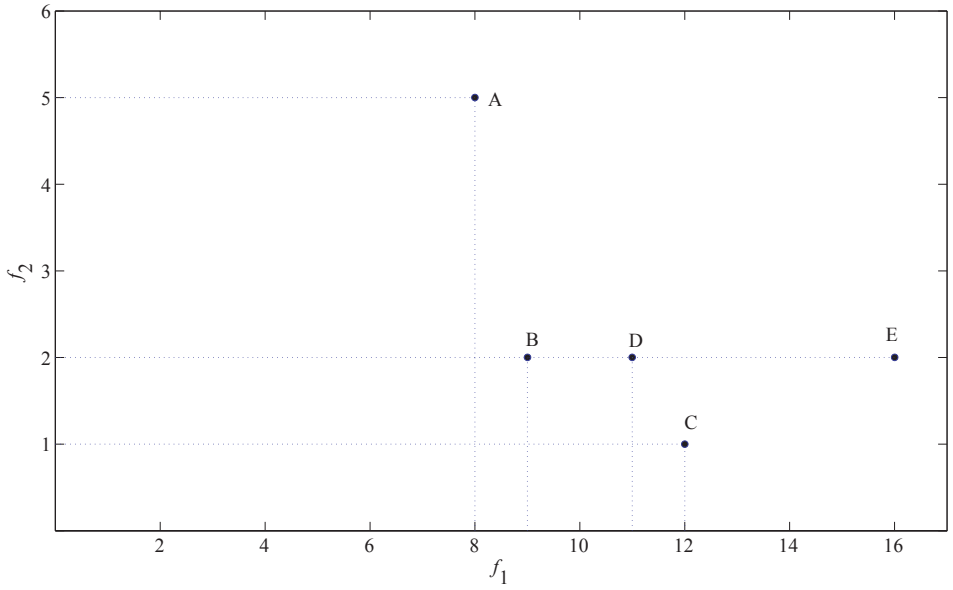

Figure 2. Pareto optimality and domination relation.

Methods with *a Priori* Articulation of Preferences using a Utility Function

In these methods, the decision maker's preferences are incorporated as parameters in terms of a utility function *a priori*. Typically these parameters can be coefficients, exponents, constraint limits, etc. These parameters determine the tradeoff of objectives before implementation of the optimization method. The optimum solution found would

reflect the tradeoff made *a priori*. Depending on whether the solution found turned out to satisfy the preferences or not, the decision maker can re-adjust the parameters to get a better solution. However the beauty of these methods is that they do not require doing a multi-objective optimization problem since the *a priori* preferences and utility function reduce the optimization to a single one.

Weighted global criteria method

In this method, all objective functions are combined to form a single utility function. The weighted global criterion is a type of utility function U in which parameters are used to model preferences. The simplest form of a general utility function can be defined as

$$U = \sum_{i=1}^{k} w_i \left[F_i(x) \right]^P, \qquad F_i(x) > 0 \forall i, \quad \text{or} \tag{2.3}$$

$$U = \sum_{i=1}^{k} \left[w_i F_i(x) \right]^P, \qquad F_i(x) > 0 \forall i, \tag{2.4}$$

where $\vec{w}$ is the vector of weights set by the decision maker such that $\vec{w} \geq 0$ and $\sum_{i=1}^{k} w_i = 1$. The difference between the two above formulations is related to conditions required for Pareto optimality. Complete discussion can be found in reference [64].

Weighted sum method

This is a special case of the weighted global criteria method in which the exponent *P* is equal to one; that is,

$$U = \sum_{i=1}^{k} w_i F_i(x). \tag{2.5}$$

The method is easy to implement and guarantees finding the Pareto optimal set, provided the objective function space is convex. However, a uniformly distributed set of

weights does not necessarily find a uniformly distributed Pareto optimal set, which makes it difficult to obtain a Pareto solution in a desired region of the objective space.

Exponential weighted criterion

It is defined as follows:

$$U = \sum_{i=1}^{k} \left(e^{pw_i} - 1 \right) e^{pF_i(x)}, \tag{2.6}$$

where the argument of the summation represents an individual utility function for $F_i(x)$.

Weighted product method

To avoid transforming objective functions with similar significance and different order of magnitude, one may consider the following formulation [65]:

$$U = \prod_{i=1}^{k} \left[F_i(x) \right]^{w_i}, \tag{2.7}$$

where w_i are weights indicating the relative significance of the objective functions.

Conjoint analysis

This method [66, 67] uses a concept borrowed from marketing, where a product is characterized by a set of attributes, with each attribute having a set of levels. An aggregated utility function is developed by direct interaction with the customer/designer; the designer is asked to rate, rank order, or choose a set of product bundles. In engineering design studies, we can assume that people will choose their most preferred product alternative. Conjoint analysis takes these sets of attributes and converts them into a utility function that specifies the preferences that the customer has for all of the product's attributes and attribute levels. The advantage of this method is that it automatically takes into account marginal diminishing utility (i.e., no cost is expended in a design that does not really have practical utility).

Methods with *a Priori* Articulation of Preferences without using a Utility Function

Lexicographic method

Here the objective functions are arranged in a descending order of importance [68]. The highest preference objective is optimized with no regard to the other objectives, and then a single objective problem is solved consecutively (in order of preference of objectives) for a set of limits on the optimums of previously solved for objectives. This can be defined as

$$\begin{aligned} &\textit{Minimize } F_i(x) \\ &\textit{subject to } F_j(x) \le F_j(x_j^*), \quad j = 1,2,...,i-1, i>1, \\ &i = 1,2,...,k \end{aligned} \tag{2.8}$$

where *i* represents the function's position in the preferred sequence and $F_j(x_j^*)$ represents the optimum of the j[th] objective function found in the j[th] iteration.

Goal programming methods

Here, goals b_j are specified for each objective function $F_j(\vec{x})$ [69]. Then the total deviation from the goals, $\sum_{j=1}^{k}|d_j|$, is optimized, where d_j is the deviation from the goal b_j for the j^{th} objective.

Methods for an *a Posteriori* Articulation of Preferences

The inability of the decision maker to set preferences *a priori* in terms of a utility function makes it necessary to generate a Pareto optimal set after which an *a posteriori* articulation of preferences is made; such methods are sometimes referred to as *cafeteria* or *generate-first-choose-later.* These methods however require the generation of the Pareto optimal set which may be prohibitively time consuming. It is worth noting that repeatedly solving the weighted sum approaches presented earlier can be used to find the

entire Pareto optimal solution for convex criteria space; however, these methods fail to provide an even distribution of points that can accurately represent the Pareto optimal set.

Bounded objective function method

In this method [70], the single most important objective function, $F_s(\vec{x})$, is minimized, while all other objective functions are added as constraints with lower and upper bounds such that $l_i \le F_i(\vec{x}) \le \varepsilon_i, \quad i = 1, 2, ..., k, i \ne s$. A variation of this method is the $\varepsilon-$constraint [71] or trade-off method in which the lower bound l_i is excluded and the Pareto optimal set is obtained using a systematic variation of ε_i. This method is particularly useful in finding the Pareto optimal solution for convex or non-convex objective spaces alike. However, choice of the constraint vector $\vec{\varepsilon}$ must lie within the minimum and maximum of the objective function considered; otherwise, no feasible solution will be found. Also the distribution of the Pareto optimal solution will usually be non-uniform for the objective function(s) minimized.

Normal boundary intersection (NBI) method

This method provides a means for obtaining an even distribution of Pareto optimal points for a consistent variation in parameter vector of weights [72, 73], even with a non-convex Pareto optimal set (a deficiency found in weighted sum method). For each parameter weight the NBI problem is solved to find an optimum point that intersects the criteria feasible space boundary, however, for non-convex problems, some of the solutions found can be non Pareto optimal. Details of the method can be found in the references.

Normal constraint (NC) method

This method uses normalized objective functions with a Pareto filter to eliminate non-Pareto optimal solutions [74]. The individual minima of the normalized objective functions are used to construct the vertices of the utopia hyper-plane. A sample of evenly distributed points on the utopia hyper-plane is found from a linear combination of the vertices with consistently varied weights in criterion space. Each Pareto optimal point is found by solving a separate normalized single-objective function with additional inequality constraints for the remaining normalized objective functions.

Homotopy method

In this method the convex combination of bi-objective functions $(1-\alpha) f_1 + \alpha f_2$ is optimized for an initial value of the parameter α. Then homotopy curve tracking methods are used to generate the Pareto optimal solution curve for $\alpha \in [0,1]$ whenever the curve is smooth [75, 76] or even non-smooth [62, 77] at points corresponding to changes in the set of active constraints.

Choice of Optimization Method

The ease of implementation of the ε-constraint method [71] for a bi-objective problem makes it a good candidate method. In this method, one of the objectives is optimized for systematic variation of limits ($\varepsilon_1, \varepsilon_2 ..., \varepsilon_i$) on the second objective. A uniform distribution of the Pareto optimal set can be found for the constrained objective. There is no limitation on the convexity or non-convexity of the objective space in finding the Pareto optimal set. However, choice of the constraint set of limits ($\varepsilon_1, \varepsilon_2 ..., \varepsilon_i$) must lie within the minimum and maximum of the objective function considered; otherwise, no feasible solution would be found. In our case the material removal rate (f_{MRR}) and

surface location error (f_{SLE}) are the bi-objective criteria. The material removal rate objective would be a better choice for the constrained objective, since the set of limits ($\varepsilon_1, \varepsilon_2 ..., \varepsilon_i$) of f_{MRR} constraint can be more easily constructed according to designer's preference, whereas that would be difficult for the f_{SLE} objective.

CHAPTER 3
MILLING MULTI-OBJECTIVE OPTIMIZATION PROBLEM

Introduction

In this chapter, a description of the milling problem and solution method used to solve the mathematical model is presented. Two optimization methods of interest are briefly described. These methods are then applied to the multi-objective optimization problem and a discussion of results is provided.

Milling Problem

Milling Model

The schematic for a two degree-of-freedom (2-DOF) milling process is shown in Figure 3 (repeated here). With the assumption of either a compliant tool or a structure, a summation of forces gives the following equation of motion:

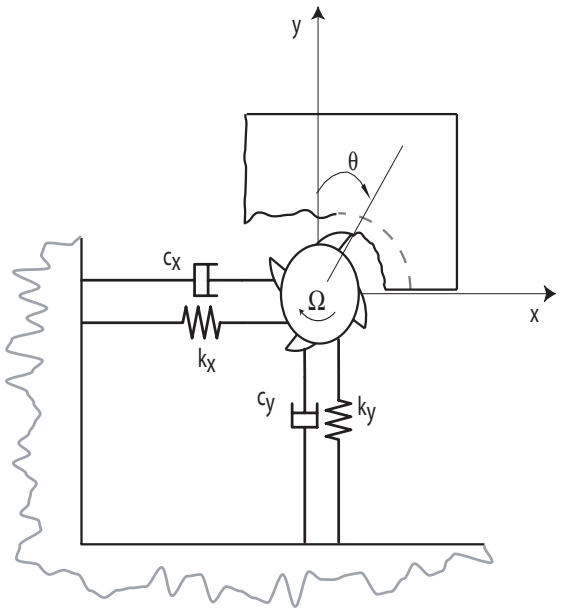

Figure 3. Schematic of 2-DOF milling tool

$$\begin{bmatrix} m_x & 0 \\ 0 & m_y \end{bmatrix}\begin{Bmatrix} \ddot{x}(t) \\ \ddot{y}(t) \end{Bmatrix}+\begin{bmatrix} c_x & 0 \\ 0 & c_y \end{bmatrix}\begin{Bmatrix} \dot{x}(t) \\ \dot{y}(t) \end{Bmatrix}+\begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix}\begin{Bmatrix} x(t) \\ y(t) \end{Bmatrix}=\begin{Bmatrix} F_x(t) \\ F_y(t) \end{Bmatrix}, \tag{3.1}$$

where the terms m_x, c_x, k_x and m_y, c_y, k_y are the modal mass, viscous damping, and stiffness terms, and F_x, and F_y are the cutting forces in the x and y directions, respectively. A compact form of the milling process can be found by considering the chip thickness variation and forces on each tooth (a detailed derivation is provided in references [54-58] and Appendix A):

$$\boldsymbol{M}\ddot{\vec{X}}(t)+\boldsymbol{C}\dot{\vec{X}}(t)+\boldsymbol{K}\vec{X}(t)=\boldsymbol{K_c}(t)b\left(\vec{X}(t)-\vec{X}(t-\tau)\right)+\vec{f}_o(t)b\ , \tag{3.2}$$

where $\vec{X}(t)=\begin{bmatrix} x(t) & y(t) \end{bmatrix}^T$ is the two-element position vector and $\boldsymbol{M}$, $\boldsymbol{C}$, and $\boldsymbol{K}$ are the 2x2 modal mass, damping, and stiffness matrices, $\boldsymbol{K_c}$ and $\vec{f}_0$ (function of the cutting force coefficients) are defined in Appendix A, b is the axial depth of cut, $\tau = 60/(N\Omega)$ is the tooth passing period in seconds, Ω is the spindle speed given in rev/min (rpm), and N is the number of teeth on the cutting tool. As shown in Eq. (3.2), the milling model is dependent on modal parameters of the tool/work-piece combination and the cutting force coefficients.

Solution Method

As described in Chapter 1, a solution of Eq. (3.2) can be completed using numerical time-domain simulation [43, 44, 50] or the semi-analytical TFEA [54-58]. Compared to the first approach, TFEA can obtain rapid process performance calculations of surface location error, f_{SLE}, and stability. The computational efficiency of TFEA compared to conventional time-domain simulation methods makes it the most attractive candidate for use in the optimization formulation. In this method a discrete linear map is generated that

relates the vibration while the tool is in the cut to free vibration out of the cut. Stability of the milling process can be determined using the eigenvalues of the dynamic map, while surface location error (see Appendix A) is found from the fixed points of the dynamic map. Details can be found in references [54-58]. An added advantage of TFEA is that it provides a clear and distinct definition of stability boundaries (i.e., eigenvalues of the milling equation with an absolute value greater than one identify unstable conditions).

Problem Specifics

In this section, the calculation of the stability boundary is analyzed, the continuity of surface location error and stability boundary is addressed, TFEA convergence is described, and sensitivity of the milling model to cutting force coefficients is defined.

Stability Boundary

In order to find the axial depth limit, b_{lim}, of neutral stability at corresponding input parameters, the bi-section method is used in the TFEA algorithm to solve for b_{lim} at which the maximum characteristic multiplier is equal to one (stability limit)

$$g_{\lambda} = \max\left|\vec{\lambda}\right| \le 1 \tag{3.3}$$

where $\vec{\lambda}$ is the eigenvalues of the dynamic map. An absolute error is used as a criterion for convergence

$$\frac{b_i - b_{i-1}}{b_i} \le \varepsilon \tag{3.4}$$

where ε corresponds to the error tolerance and b_i is the root corresponding to $\max\left|\vec{\lambda}\right| = 1$ at iteration *i*. The value of ε is set based on the numerical accuracy required in the calculation of b_{lim}. A value of $\varepsilon = 1e-3$ can be adequate for the calculation of b_{lim}.

Surface location error and stability boundary: C^1 discontinuity

Correct use of an optimization method depends on its limitations. Gradient-based methods, for example, depend on C^1 continuity (the first derivative of the function is continuous) of the objective functions (f_{MRR} and $|f_{SLE}|$) and stability constraint (Eq. (3.3)). The objective f_{MRR} is defined analytically in Eq. (3.6), where it is clear that it is C^1 continuous. However, the $|f_{SLE}|$ and stability (g_λ) functions are only found numerically using TFEA. A graphical description of both functions provides some insight into the continuity of these functions. Figure 4 depicts the variation of f_{SLE} and $|f_{SLE}|$ as a function of spindle speed for a typical set of cutting parameters. Although f_{SLE} is C^1 continuous in the region where it is defined (stable region), $|f_{SLE}|$ is C^1 discontinuous. This can be easily verified analytically by considering the functions $f(x) = x$ and $f(x) = |x| = \{x \ for \ \ x > 0 \text{ and } -x \ for \ x < 0\}$. The absolute function is clearly C^1 discontinuous at $x = 0$. The same argument can be made for the near-zero $|f_{SLE}|$ range shown in Figure 4. In Figure 5 the variation of stability function g_λ versus spindle speed shows lobe peaks where C^1 (slope) discontinuity of g_λ is also observed. C^1 discontinuity makes convergence of gradient-based optimization algorithms near the discontinuity rather difficult. This requires the use of multiple initial guesses in order to converge to even a local optimum.

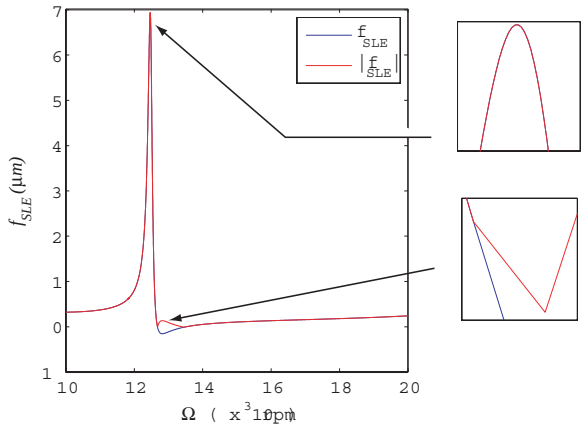

Figure 4. Surface location error and its absolute. Discontinuity of the absolute surface location error is apparent in the lower insert.

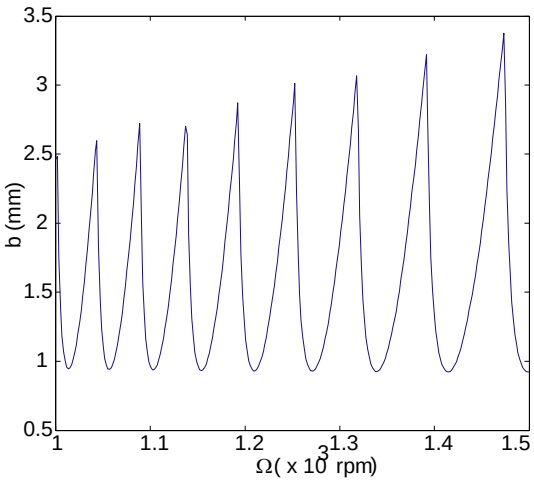

Figure 5. A typical stability boundary. The cusps where C^1 discontinuity in the stability boundary are depicted.

TFEA convergence

The convergence of TFEA depends on the cutting parameters. A higher number of elements must be used when convergence is not achieved. Either $|f_{SLE}|$ or the stability boundary g_λ can be used to check for convergence. A typical procedure to test for the convergence of finite element meshes is to compare the change in the estimated value (g_λ or $|f_{SLE}|$) as the number of elements is increased (mesh refinement). In Figure 6, the

dependence of convergence on the spindle speed is shown for a randomly selected cutting condition of 5% radial immersion (percentage of radial depth of cut to tool diameter) and 18 mm axial depth. As seen in Figure 6, the flawed convergence for a small number of elements (=1) would give the impression of a sufficient number of elements. However, further increasing the number of elements (=12) shows poor convergence for the low speed. This can be due to the fact that as the spindle speed decreases, the time in the cut increases, which requires a higher number of elements to achieve convergence. The fact that the optimization algorithm will pick milling parameters within the design space makes it necessary to choose a rather high number of elements to ensure convergence anywhere in the design space. However, a penalty in computational time is incurred.

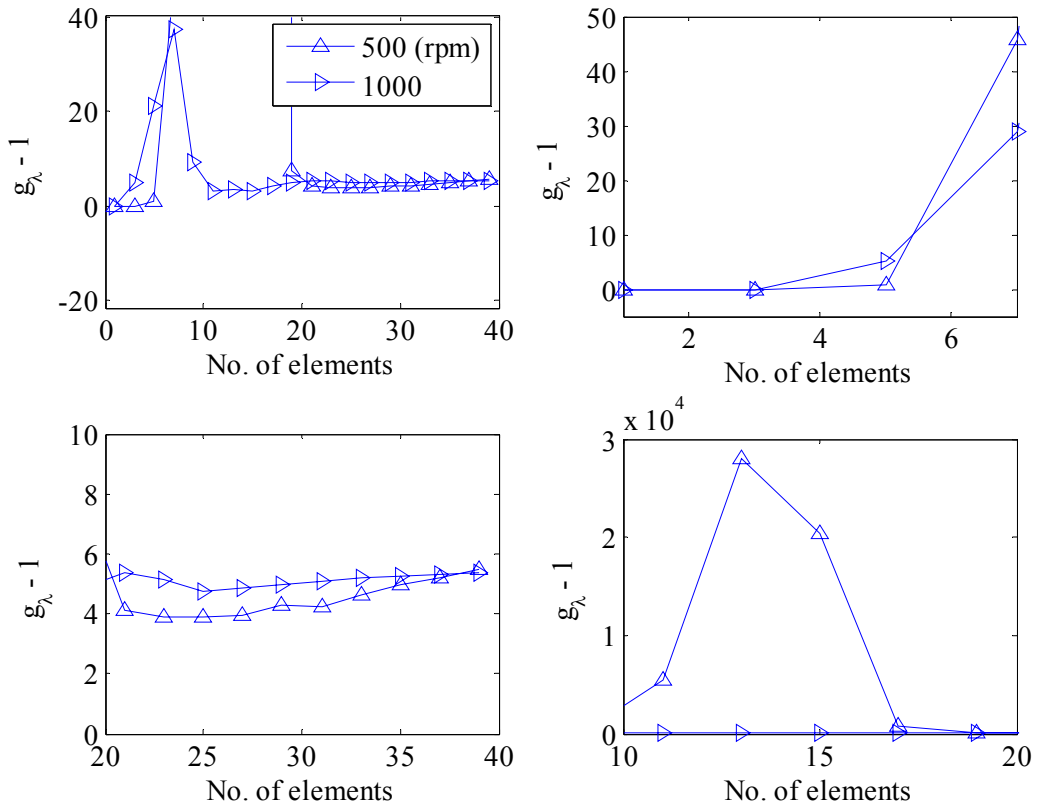

Figure 6. Convergence of stability constraint for 5% radial immersion and different spindle speeds for an 18 mm axial depth. We can see that convergence at lower speeds require substantially more number of elements.

Optimization Method

Optimization methods can be categorized according to the searching method used to find the optimum [78]. They are either *direct* where only the values of the objective function and constraints are used to guide the search strategy, or *gradient-based*, where first and/or second order derivatives guide the search process. Particle swarm optimization (*PSO*) and sequential quadratic programming (*SQP*) will be used to test the feasibility of both methods, respectively, for the problem at hand.

Particle Swarm Optimization Technique

Particle swarm optimization is an evolutionary computation technique developed by Kennedy and Eberhart [79, 80]. It can be used for solving single or multi-objective optimization problems. To find the optimum solution, a swarm of particles explores the feasible design space. Each particle keeps track of its own personal best (pbest) fitness and the global best (gbest) fitness achieved during design space exploration. The velocity of each particle is updated toward its pbest and the gbest positions. Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward pbest and gbest.

In order to accommodate constraints, Xiaohui *et al.* [79] presented a modified particle swarm optimization algorithm, where *PSO* is started with a group of feasible solutions and a feasibility function is used to check if the newly explored solutions satisfy all the constraints. All the particles keep only those feasible solutions in their memory while discarding infeasible ones.

Sequential Quadratic Programming (*SQP*)

The basic idea of this method is that it transforms the nonlinear optimization problem into a quadratic sub-problem around the initial guess. The nonlinear objective function and constraints are transformed into their quadratic and linear approximations. The quadratic problem is then solved iteratively and the step size is found by minimizing a descent function along the search direction. Standard optimization algorithms may be used to solve the quadratic sub-problem.

Usually *SQP* leads to identification of only local optima. In order to better converge to the global optimum, a number of initial guesses is used to scan the design space and the optimum of these local optima is close to the global optimum.

Problem Formulation

In this section, the multi-objective optimization problem is defined and then a description of the tradeoff method is given. The problem solution is then presented in the order it has been addressed in the robust optimization section. Finally, discussion of the simulation results is provided.

Problem Statement

The problem of minimizing surface location error $|f_{SLE}|$ and maximizing material removal rate f_{MRR} is stated as follows:

$$\begin{aligned} & min\left[\left|f_{SLE}\left(a,b,c,N,\Omega\right)\right|,-f_{MRR}\left(a,b,c,N,\Omega\right)\right], \\ subject\ to: \quad & g_{\lambda}\left(b,\Omega\right)=max\left|\vec{\lambda}\left(a,b,N,\Omega\right)\right|\leq 1 \end{aligned} \tag{3.5}$$

where g_{λ} is the stability constraint obtained from the dynamic map eigenvalues, f_{SLE} is found from the fixed points, and the mean f_{MRR} is given as:

$$f_{MRR}=abcN\Omega\,, \tag{3.6}$$

where a, b, c, N and Ω are radial depth of cut, axial depth of cut, feed per tooth (chip load), number of teeth, and spindle speed, respectively (Figure 7). From Eq. (3.5) it can be seen that only the stability constraint is not a function of the feed per tooth. In Eq.(3.5) , f_{SLE} and f_{MRR} are explicitly stated as a function of cutting conditions (a, b, c, N and Ω). This reflects the relative ease by which these conditions can be adjusted to achieve optimality of the objectives.

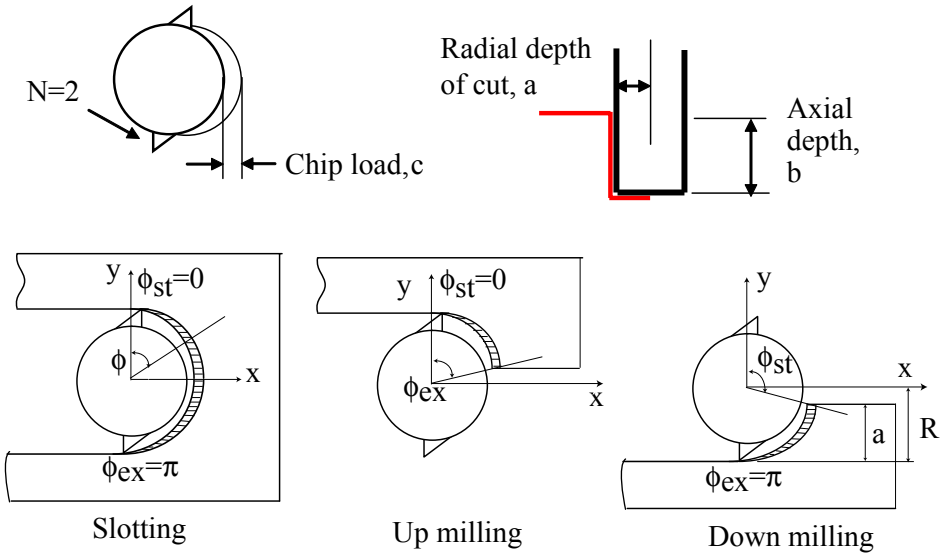

Figure 7. Schematic of milling cutting conditions and various types of milling operations.

Tradeoff Method

To address the multi-objective problem the constraint method is used, where the two-objective problem is transformed into a single objective problem of minimizing one objective with a set of different limits on the second objective. Each time the single objective problem is solved, the second objective is constrained to a specific value until a sufficient set of optimum points are found. These are used to generate the Pareto front [38] of the two objectives. In the case that f_{SLE} is chosen as the objective function to be minimized then Eq. (3.5) is transformed to:

$$\begin{aligned} & min\left|f_{SLE}\left(a,b,c,N,\Omega\right)\right|, \\ subject\ to: \quad & -f_{MRR}\left(a,b,c,N,\Omega\right)\le e_i, \qquad for\ i\ =\ 1\ldots k \\ & g_\lambda\left(a,b,N,\Omega\right)=max\left|\vec{\lambda}\left(a,b,N,\Omega\right)\right|\le 1, \\ & for\ a\ series\ of\ k\ selected\ limits\ (e)\ on\ f_{MRR}. \end{aligned} \tag{3.7}$$

Where the cutting conditions: a, b, c, N and Ω are the design variables. On the other hand if f_{MRR} is chosen as the objective function to be maximized, then Eq. (3.5) is transformed to:

$$\begin{aligned} & min-f_{MRR}\left(a,b,c,N,\Omega\right), \\ subject\ to: \quad & \left|f_{SLE}\left(a,b,c,N,\Omega\right)\right|\le e_i, \qquad for\ i\ =\ 1\ldots k \\ & g_\lambda\left(a,b,N,\Omega\right)=max\left|\vec{\lambda}\left(a,b,N,\Omega\right)\right|\le 1, \\ & for\ a\ series\ of\ k\ selected\ limits\ (e)\ on\ \left|f_{SLE}\right|. \end{aligned} \tag{3.8}$$

It should be noted that applying Eq. (3.7) using the *SQP* method is more straightforward than Eq. (3.8). The reason is that in order to use a number of initial guesses along the $|f_{SLE}|$ contour in Eq. (3.8), the axial depth corresponding to that $|f_{SLE}|$ needs to be found, whereas in Eq. (3.7) the axial depth can be explicitly expressed found as shown in Eq. (3.6).

Robust Optimization

Problem solution

In the first iteration of the problem, only axial depth (b) and spindle speed (Ω) are considered as design variables. Other cutting conditions are held fixed (Table 2) for a down milling cut. Modal parameters for a single degree-of-freedom tool with one dynamic mode in x and y directions are used (Table 2). The nominal values of the tangential (K_t) and normal (K_n) cutting force coefficients are 550 N/mm^2 and 200 N/mm^2, respectively. The *SQP* method is used to find the Pareto front using the formulation in Eq. (3.7). Here $|f_{SLE}|$ is minimized for a set of limits on f_{MRR}. As mentioned earlier, the

SQP method is a local search method that is highly dependent on C^1 continuity of the objective function and constraints. To obtain a global optimum, a number of initial guesses are used along each f_{MRR} constraint limit. A set of optimum points are obtained for these initial guesses. The minimum of these optimum points is nominated as a global optimum. The number of initial guesses is increased and another run of the optimization simulation is made to check the validity of that global optimum.

Table 2. Cutting conditions and modal parameters for Tool used in optimization simulations

M (kg)	C (Ns/m)	K (N/m)	
0.056 0 0 0.061	3.94 0 0 3.86	1.52×10^6 0 0 1.67×10^6	
Tool diameter (mm)	c (mm)	a (mm)	N
19.05	0.178	0.76	2
K_t (N/m^2)	K_n (N/m^2)	K_{te} (N/m)	K_{ne} (N/m)
550 x 10^6	200 x 10^6	0	0

In this formulation, the minimum $|f_{SLE}|$ points were found to favor spindle speeds where the tooth passing frequency is equal to an integer fraction of the system's natural frequency (Figure 8), which corresponds to the most flexible mode (these are the traditionally-selected 'best' speeds which are located near the lobe peaks in stability lobe diagrams). Because f_{SLE} can undergo large changes in value for small perturbations in Ω at these optimum points, the formulation provided in Eqs. (3.7) and (3.8) leads to optima which are highly sensitive to spindle speed variation (Figure 8) . To show the sensitivity of these optimum points, a typical optimum point is superimposed on a graph of $|f_{SLE}|$ vs. Ω in Figure 9. It is seen that the optimum point is located in a high f_{SLE} slope region.

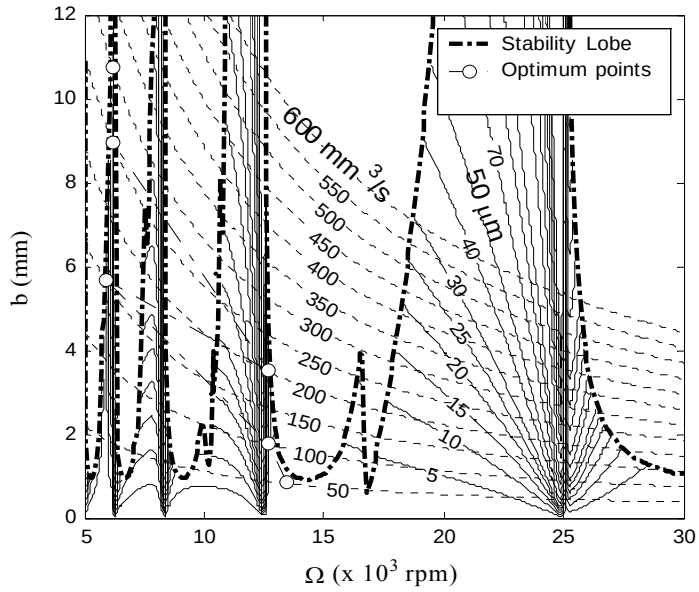

Figure 8. Stability, $|f_{SLE}|$ and f_{MRR} contours with optimum points overlaid. The figure shows that optimum points occur in regions sensitive to spindle speed variation (Table 2).

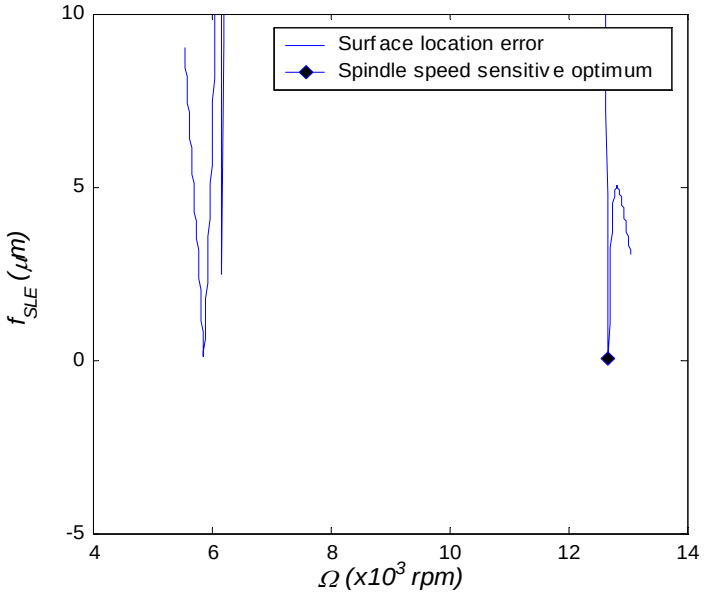

Figure 9. A typical optimum point found; optimum point sensitivity with respect to spindle speed is apparent (Table 2).

Reformulation of problem

The optimization problem was redefined in order to avoid convergence to spindle speed-sensitive optima. Two approaches were applied: 1) an additional constraint was added to the $|f_{SLE}|$ slope; and 2) the f_{SLE} objective was redefined as the average of three perturbed spindle speeds. The latter proved to be more robust than the former. This is due to the difficulty in setting the value of the $|f_{SLE}|$ slope constraint *a priori*. The spindle speed perturbed form of the problem transforms Eqs. (3.7) and (3.8) to

$$\begin{aligned} & min \left[\frac{\left|f_{SLE}\left(b,\Omega+\delta\right)\right| + \left|f_{SLE}\left(b,\Omega\right)\right| + \left|f_{SLE}\left(b,\Omega-\delta\right)\right|}{3} \right], \\ subject\ to: \quad & -f_{MRR}\left(b,\Omega\right) \le e_i, \qquad for\ i\ =\ 1 \ldots k \\ & \left\{ g_\lambda\left(b,\Omega-\delta\right) \cap g_\lambda\left(b,\Omega\right) \cap g_\lambda\left(b,\Omega+\delta\right) \right\} \le 1, \\ & for\ a\ series\ of\ selected\ limits\ (e)\ on\ f_{MRR}, \end{aligned} \tag{3.9}$$

and

$$\begin{aligned} & min -f_{MRR}\left(b,\Omega\right), \\ subject\ to: \quad & \left[\frac{\left|f_{SLE}\left(b,\Omega+\delta\right)\right| + \left|f_{SLE}\left(b,\Omega\right)\right| + \left|f_{SLE}\left(b,\Omega-\delta\right)\right|}{3} \right] \le \varepsilon_i, \quad for\ i\ =\ 1 \ldots k \\ & \left\{ g_\lambda\left(b,\Omega-\delta\right) \cap g_\lambda\left(b,\Omega\right) \cap g_\lambda\left(b,\Omega+\delta\right) \right\} \le 1, \\ & for\ a\ series\ of\ selected\ limits\ (\varepsilon)\ on\ average\ perturbed\ |f_{SLE}|, \end{aligned} \tag{3.10}$$

where δ is the spindle speed perturbation selected by the designer (a typical value for our analyses was 50 rpm). A study of spindle speed perturbation selection is provided in the next section.

The validity of the perturbed $|f_{SLE}|$ average as a convergence criteria can be seen in Figure 10. In this figure the perturbed average $|f_{SLE}|$ is plotted with $|f_{SLE}|$, where points A and B correspond to highly and moderately spindle speed-sensitive $|f_{SLE}|$, respectively.

The average perturbed $|f_{SLE}|$ at point A (high slope point) is shown to be higher than at point B. Therefore, using the perturbed average $|f_{SLE}|$ as an objective function criteria can avoid convergence to spindle speed sensitive $|f_{SLE}|$ (such as $|f_{SLE}|$ region near point A).

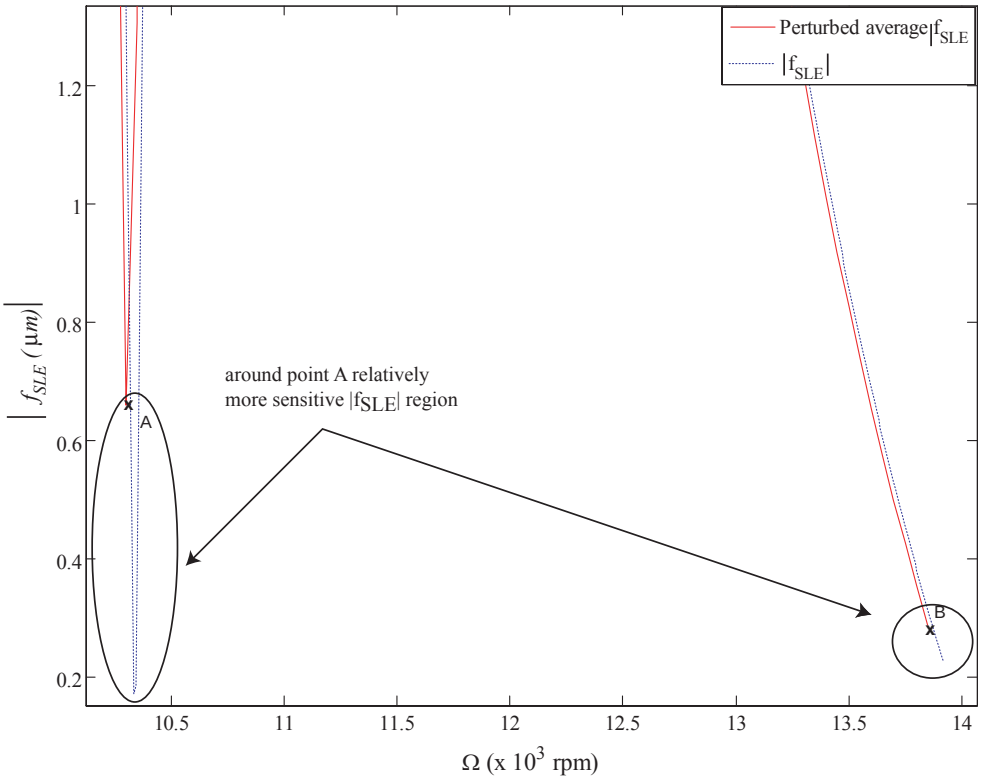

Figure 10. Perturbed average of $|f_{SLE}|$ validation as optimization criterion that avoids spindle speed sensitive $|f_{SLE}|$. Shown in the figure are points A (close to steep slope region of $|f_{SLE}|$) and B (close to moderate slope region of $|f_{SLE}|$), the perturbed average of $|f_{SLE}|$ near A is higher than at B. Therefore, using the perturbed average as an optimum criterion is valid.

The *SQP* method is used to solve Eqs. (3.9) and (3.10). In case Eq. (3.9) is implemented then initial guesses of Ω and b (design variables) are made along the f_{MRR} contour. In the other case (Eq. (3.10)) the initial guesses of Ω and b are made along $|f_{SLE}|$

contour. The number of initial guesses along the constraint is made such that convergence is towards a global optimum. The initial guesses for the spindle speed are increased in 625 rpm increments for the corresponding spindle speed range considered. Also, the *PSO* method is used to solve Eq. (3.8). When using *PSO*, the optimum points do not tend to converge to spindle speed sensitive optimums. Therefore, there is no need to solve the reformulated form of the problem in *PSO*. This leads to a fewer number of evaluations of $|f_{SLE}|$ and is a computationally more efficient optimization method.

A comparison of the three optimization schemes is shown in Figure 11 and Figure 12. Figure 11 shows the optima for each approach superimposed on the corresponding stability lobe diagram. In Figure 12, the Pareto fronts for the three methods are shown. The optimum points found using the two *SQP* formulations closely agree with the *PSO* method (Figure 12).

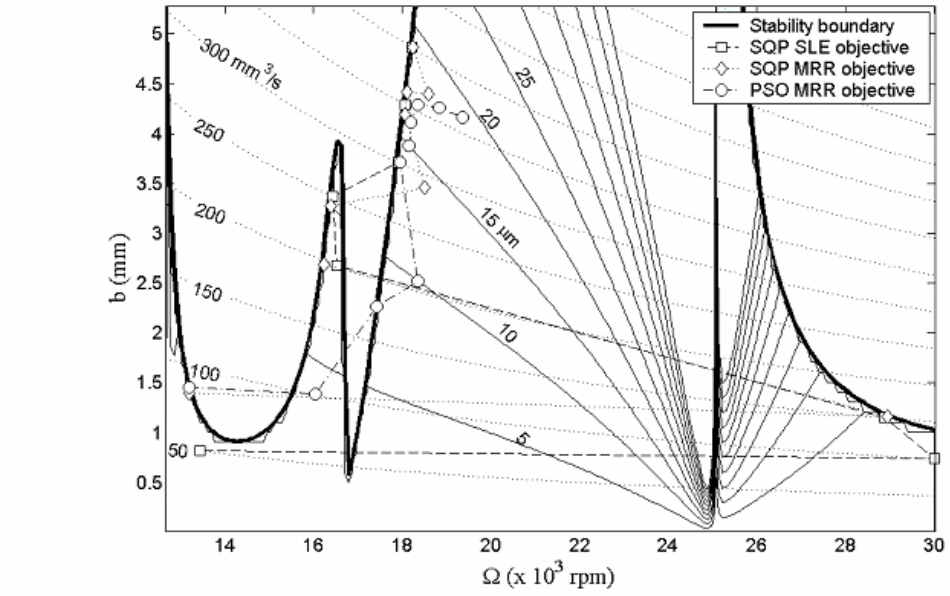

Figure 11. Stability, $|f_{SLE}|$, and f_{MRR} contours with optimum Pareto front points found using *PSO* and *SQP* (average perturbed spindle speed formulation). The figure shows that optimum points are not in regions sensitive to spindle speed (Table 2).

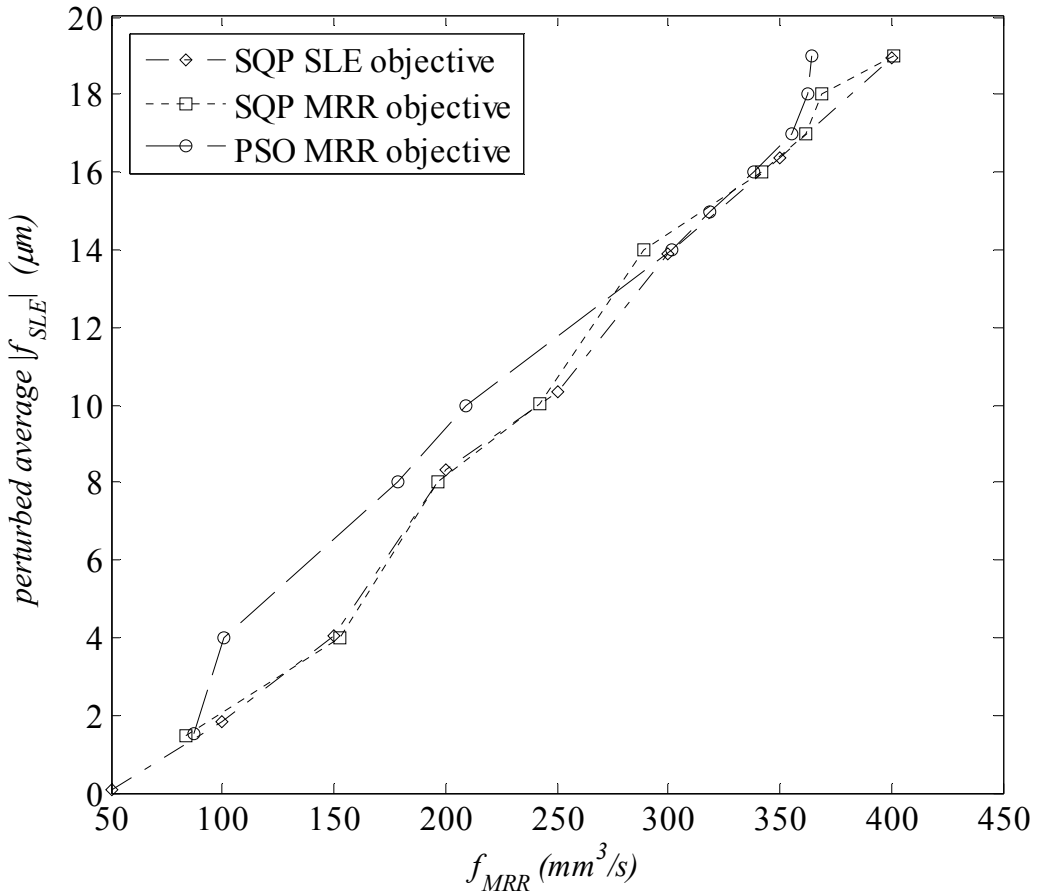

Figure 12. Pareto front showing optimum points found using three optimization algorithms/formulations; the same trends are apparent. However, the *SQP* methods required additional computational time (Table 2).

Although the *PSO* points show the same trend, some improvement in the fitness is still possible relative to the *SQP* results. Because the *PSO* search inherently avoided optimum points that are spindle speed insensitive, there is no need to use average perturbed f_{SLE} as with *SQP*, which leads to a decreased number of f_{SLE} evaluations in *PSO*. However, narrow optimum points may go undetected when using *PSO*.

As noted, when comparing the Pareto fronts in Figure 12, it is seen that the *PSO* approach did not converge to the same fitness as *SQP* method. A check of the optimum points which correspond to a value of $|f_{SLE}| = 4$ μm, for example, shows that *PSO* converged to 100 mm^3/s, while *SQP* converged to 150 mm^3/s. To better understand this

result, the design space was divided between the two design vectors, *b* and *Ω*, for *SQP* and *PSO* using a factor, *a*, that was normalized between 0 and 1. The *PSO* and *SQP* optimums were normalized to *a* = 0 and 1, respectively. Next, the stability constraint (g_λ), f_{MRR}, and $|f_{SLE}|$ were plotted against that ratio. In Figure 13 it is seen that discontinuities exist in the $|f_{SLE}|$ constraint and the first derivative of the eigenvalue constraint within this region. Although *PSO* is not significantly affected by a discontinuity in the derivative constraint, it can be affected by a discontinuity of the $|f_{SLE}|$ constraint, where the discontinuity tends to narrow the search region of the swarm.

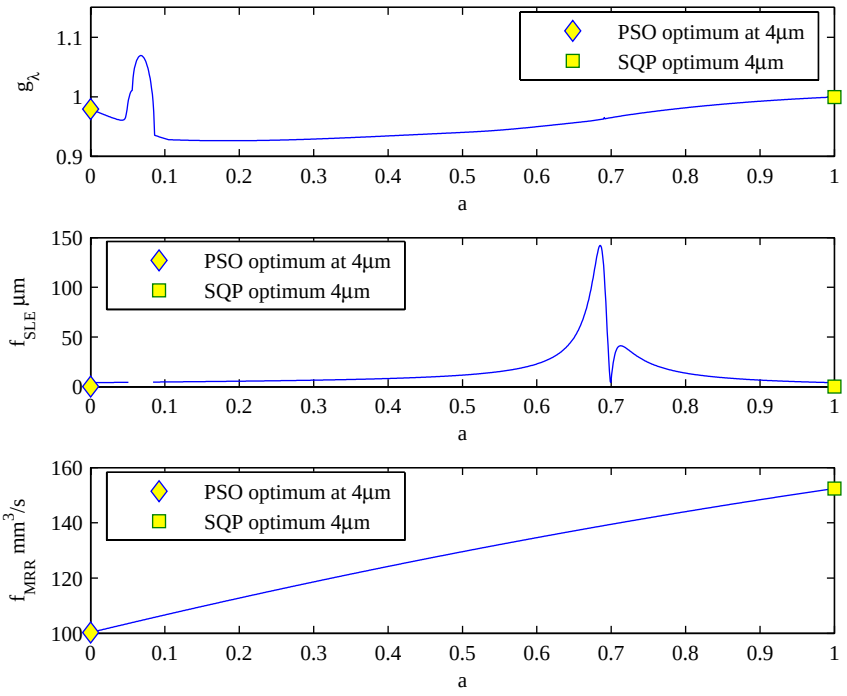

Figure 13. Variations in the eigenvalues, surface location error, and removal rate for *PSO* and *SQP* optima, where f_{MRR} is the objective for both. The discontinuities in the surface location error cause *PSO* to not converge on the *SQP* optimum.

Bi-objective space

In this section, the bi-objective domain (the feasible space of the objective functions) of average perturbed $|f_{SLE}|$ and f_{MRR} for the set of input parameters listed in Table 3 for an up milling case is provided. Figure 14 shows the objective contours in the design space of spindle speed (Ω) and axial depth (b). The respective bi-objective space is shown in Figure 15 and Figure 16. In Figure 15 the contours of constant axial depth are shown, while the contours of constant spindle speeds are shown in Figure 16. These figures give an idea of the feasible design and bi-objective space. It can be seen that the bi-objective feasible space can be non-convex (not all points on a straight line connecting two points in the feasible domain belong to that domain). This makes the choice of using the tradeoff method as a multi-objective optimization approach a suitable one, since this method can handle both convex and non-convex problems. A good observation can be made from Figure 15, where it can be seen that for the high f_{MRR} region with high b values, the relative sensitivity of $|f_{SLE}|$ increases compared to the lower f_{MRR} region.

Table 3. Cutting conditions, modal parameters and cutting force coefficients used in bi-objective space simulations

M (kg)	C (Ns/m)	K (N/m)	
0.44 0 0 0.35	83 0 0 90	4.45×10^6 0 0 3.55×10^6	
Tool diameter (mm)	c (mm)	a (mm)	N
25.4	0.1	21.8	1
K_t (N/m^2)	K_n (N/m^2)	K_{te} (N/m)	K_{ne} (N/m)
700 x 10^6	20 x 10^6	46 x 10^3	33 x 10^3

Selection of spindle speed perturbation bandwidth

In Figure 10, it was shown that the average perturbation of $|f_{SLE}|$ provided an adequate optimization criteria. However, the choice of the spindle speed perturbation step size or bandwidth, 2δ, depends on the designer preference. Any spindle speed perturbation in $|f_{SLE}|$ would avoid convergence to sensitive $|f_{SLE}|$ optima. Depending on the machining center spindle drive accuracy, the perturbation bandwidth can be set accordingly. The average perturbed $|f_{SLE}|$ contours of 100 and 300 rpm bandwidth are shown in Figure 17 (use Table 3 parameters). The high slope region of average $|f_{SLE}|$ in the 100 rpm bandwidth case is replaced by higher values of average $|f_{SLE}|$, making the optimization formulation favor insensitive spindle speed $|f_{SLE}|$.

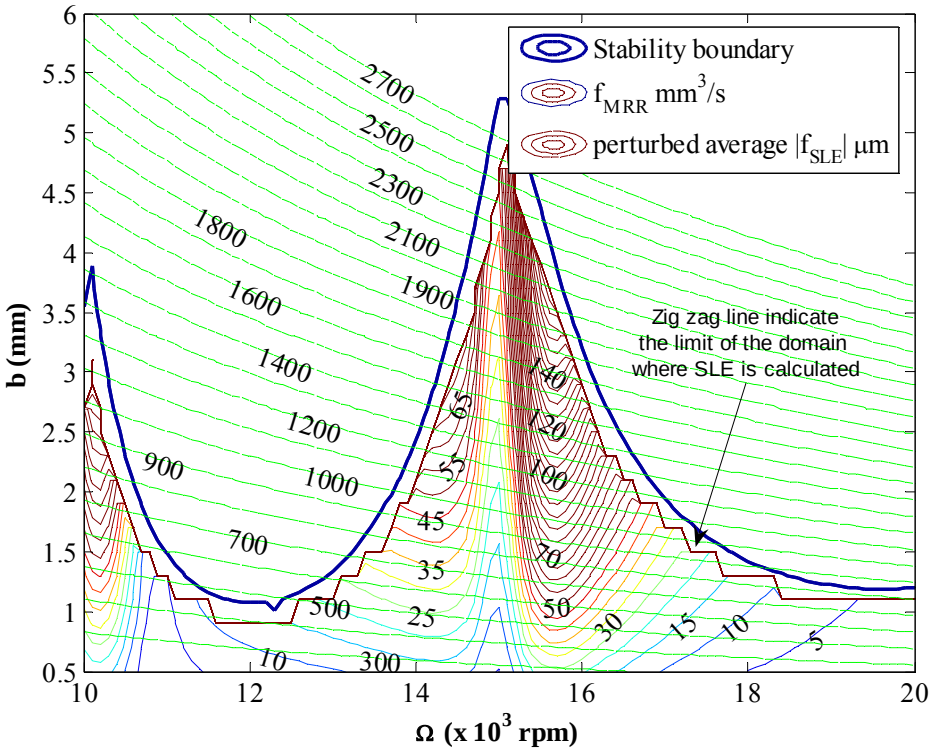

Figure 14. Average surface location error contours for 300 rpm bandwidth perturbation, stability boundary and material removal rate (see Table 3).

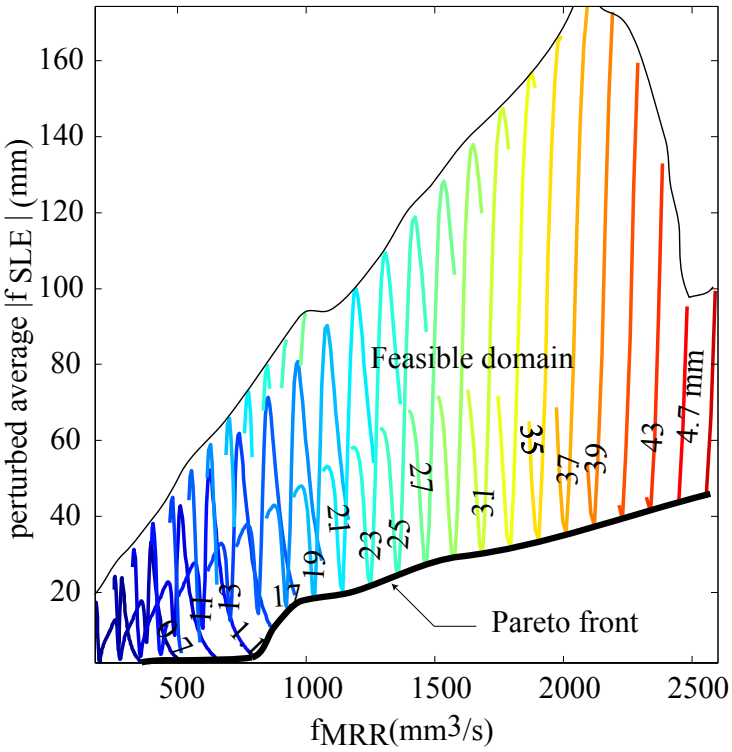

Figure 15. Feasible domain. Contour lines corresponding to constant axial depth in the stable region in the bi-objective space (see Table 3).

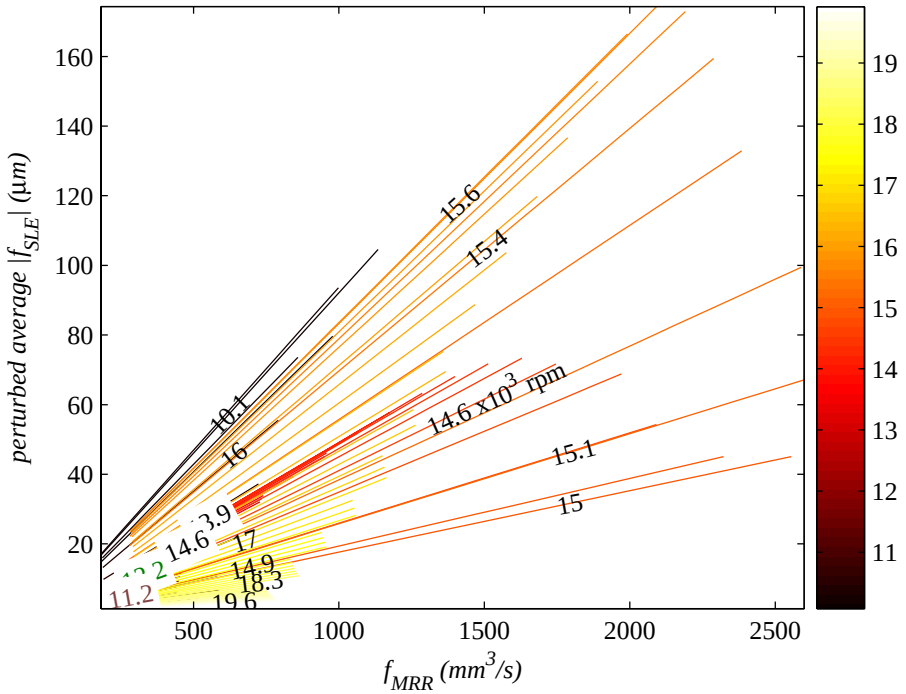

Figure 16. Contour lines corresponding to constant spindle speed in feasible region of bi-objective space (see Table 3).

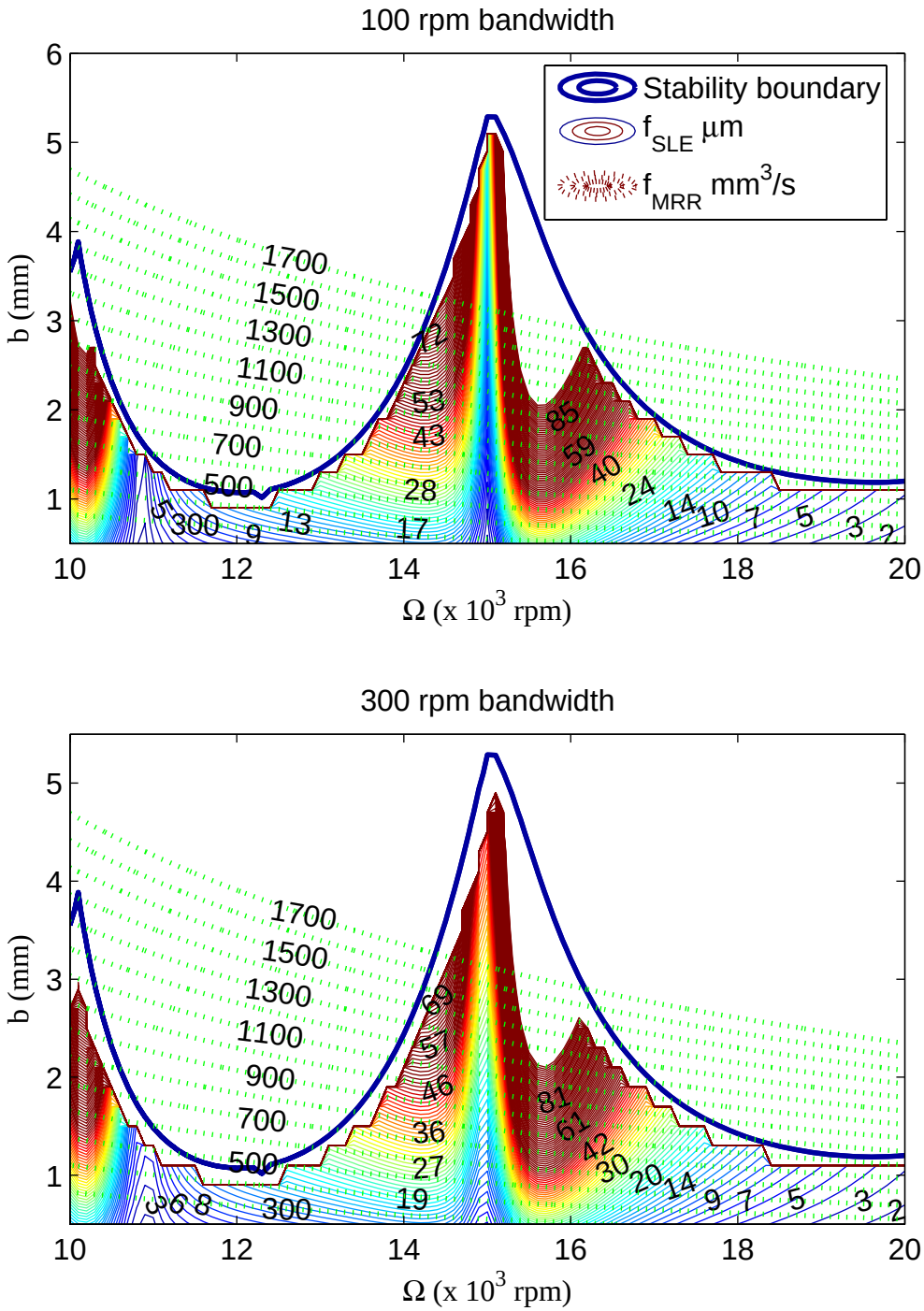

Figure 17. Average surface location error contours for 100 and 300 rpm band width, stability boundary and material removal rate contours (see Table 3).

Case Studies

As opposed to the previous analysis of two design variables (Ω and b), two cases of an added third design variable were analyzed. The first one was for radial immersion (a) and the second one was for chip load (c). These cases are compared to the two design variable case.

Radial immersion (a)

Previous simulations considered spindle speed and axial depth of cut as design variables. Another simulation was completed using radial immersion as a third design variable for an up milling cut. It was compared to a two design variable case where radial immersion was held constant at 0.508 mm in a 25.4 mm tool (Table 4). Figure 18 shows the Pareto front for these two cases. It is seen that adding radial immersion as a third design variable improved the value of perturbed average $|f_{SLE}|$ with respect to the constant radial immersion case. The optimum radial immersion found was 0.58 mm for all optimum points up to 500 mm^3/s. In both simulations the same spindle speed perturbation ($\delta = 170$ rpm) was used. As seen in Figure 18, a better calculation of the Pareto front (smoother than Figure 12) is found by using small increments in the spindle speed (each 100 rpm) initial guesses. However, the $|f_{SLE}|$ found in Figure 18 appear to be unrealistically small which may warrant further analysis.

Table 4. Cutting conditions, modal parameters and cutting force coefficients used in radial immersion case study

M (kg)	C (Ns/m)	K (N/m)	
$\begin{matrix} 0.25 & 0 \\ 0 & 0.23 \end{matrix}$	$\begin{matrix} 34.4 & 0 \\ 0 & 27.0 \end{matrix}$	$\begin{matrix} 1.30\times10^6 & 0 \\ 0 & 1.20\times10^6 \end{matrix}$	
Tool diameter (mm)	c (mm)	a (mm)	N
25.4	0.1	0.508	2
K_t (N/m^2)	K_n (N/m^2)	K_{te} (N/m)	K_{ne} (N/m)
700 x 10^6	210 x 10^6	0	0

Chip load (*c*)

To study the effect of chip load on surface location error, it is added as a third design variable in addition to spindle speed and axial depth. The parameters used in this study are listed in Table 5 for a down milling case. For the two design variable case (0.1 mm/tooth chip load), the Pareto optimal points are found for two different bandwidths, 100 rpm and 400 rpm, respectively (Figure 20). It is noted that Ω of the optimum points is almost constant up to 700 mm^3/s (31,325 rpm) where it changes to another almost constant Ω (29,500 rpm) for the higher f_{MRR} range. Also the effect of bandwidth size does not show significant effect on the optimum points found. Figure 19 shows the Pareto front for a constant chip load of 0.1 mm/tooth compared to the three design variable case, where the chip load (3rd design variable) side constraints are from 0.01 mm/tooth to 0.2 mm/tooth. An improvement in the average perturbed $|f_{SLE}|$ can be seen. It should be noted here that for the latter case, Ω is also found to be constant (31,325 rpm same as two design variable case) while the chip load increased from 0.16 to 0.2 mm/tooth. The effect of adding the chip load is therefore seen as an improvement in the fitness of average perturbed $|f_{SLE}|$ objective, where further improvement is possible while eliminating the need to switch to a lower speed (29,500 rpm) where the $|f_{SLE}|$ error is much higher. This explains the agreement between the two design variables case and three design variable case in the f_{MRR} range below 600 mm^3/s. When higher f_{MRR} is needed the two design variable case fails to account for the f_{MRR} constraint at the same spindle speed. However the three design variable case (with chip load) can accommodate this by increasing the chip load while keeping the spindle speed unchanged. This makes the $|f_{SLE}|$ in the three design variable case substantially lower.

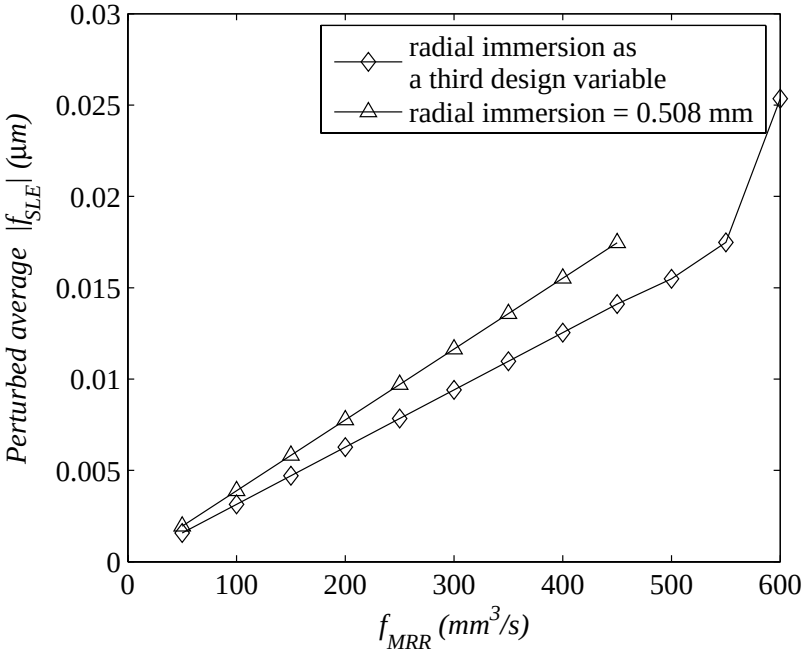

Figure 18. Pareto front for spindle speed and axial depth as design variables with radial immersion 0.508 mm, compared to the case where radial immersion is added as a third design variable. The optimum radial immersion for the latter case is 0.58 mm up to 500 mm^3/s (see Table 4).

Table 5. Milling cutting conditions, modal parameters and cutting force coefficients used in chip load study case

M (kg)	C (Ns/m)	K (N/m)	
0.027 0 0 0.03	7 0 0 2	1.0×10^6 0 0 1.6×10^6	
Tool diameter (mm)	c (mm)	a (mm)	N
12.7	0.1	0.635	2
K_t (N/m^2)	K_n (N/m^2)	K_{te} (N/m)	K_{ne} (N/m)
600 x 10^6	180 x 10^6	0	0

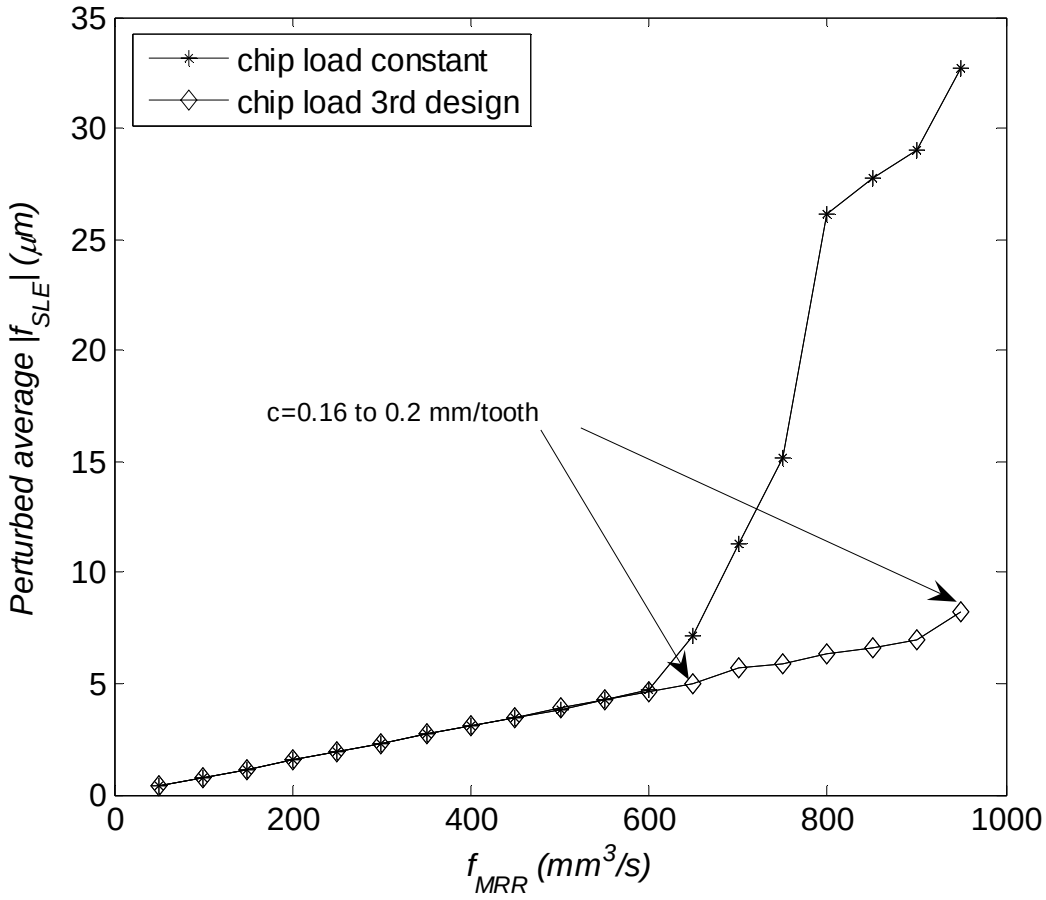

Figure 19. Pareto front using chip load as a third design variable compared to spindle speed and axial depth as design variables. For the three design variable case, an improvement in the average surface location error can be seen (see Table 5).

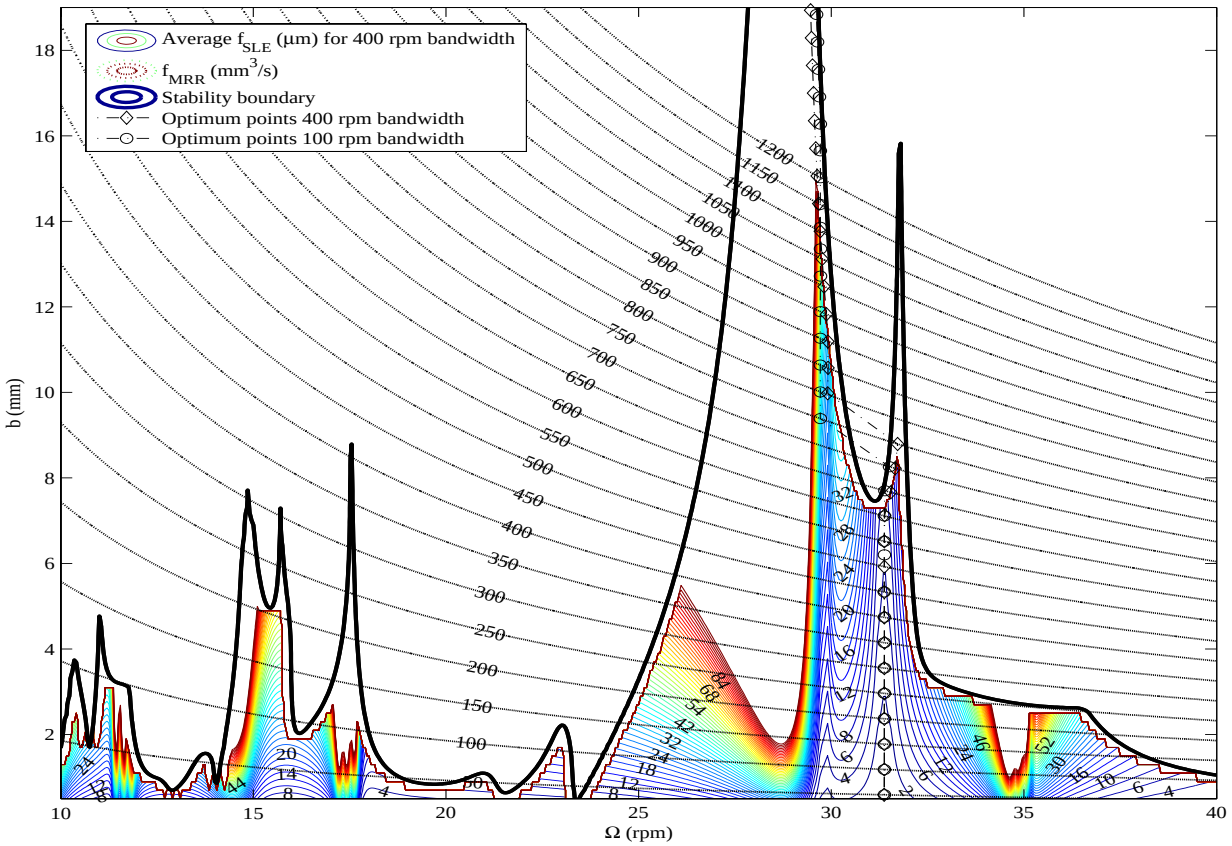

Figure 20. Stability, perturbed average $|f_{SLE}|$, and f_{MRR} contours with optimum Pareto front points found using 100 rpm and 400 rpm bandwidth. This case study shows the difficulty in selecting optimum points based on experience (Table 5).

Discussion

The formulations provided in Eqs. (3.9) and (3.10) proved adequate in finding the Pareto optimal set insensitive to spindle speed variation, provided an appropriate number of initial guesses is made. Also, the Eq. (3.9) formulation is easier to apply using the *SQP* method, where the initial guesses are made along the f_{MRR} contour.

The generation of the Pareto front for the multi-design variable case can be rather time-consuming. However, if the designer is given that freedom of choice, it might be a necessity. For example, the effect of adding chip load or radial immersion as a third design variable gave a substantial improvement in the surface location error in comparison to the two design variable case. This is counterintuitive to using a lower value of *c* or *a* as means of reducing the surface location error.

The effect of spindle speed perturbation bandwidth on the sensitivity of optimum points is rather complex. Qualitatively, in Figure 17 it is shown that increasing the bandwidth from 100 rpm to 300 rpm had the same effect of increasing the value of $|f_{SLE}|$ near the sensitive region. Further investigation is needed to establish a quantitative relation between bandwidth and sensitivity of optimum points.

CHAPTER 4
UNCERTAINTY ANALYSIS

In Chapter 3, optimization was used to find preferable designs for two objectives: material removal rate (*MRR*) and surface location error [48, 81, 82] (*SLE*), with a Pareto front, or tradeoff curve, found for the two competing objectives. Although the milling model used in the optimization algorithm is deterministic (time finite element analysis), uncertainties in the input parameters to the model limit the confidence in the optimum predictions. These input parameters include cutting force coefficients (material- and process-dependent), tool modal parameters, and cutting conditions. By accounting for these uncertainties it is possible to arrive at a robust optimum operating condition.

In previous studies [83-85], uncertainty in the milling process was handled from a control perspective. The uncertainty in the cutting force was accommodated using a control system. The force controller was designed to compensate for known process effects and accounted for the force-feed nonlinearity inherent in metal cutting operations. In this study, the uncertainties in the milling model are estimated using sensitivity analysis and Monte Carlo simulation. This enables selection of a preferred design that takes into account the inherent uncertainty in the model *a priori*.

This chapter begins with a description of the milling model and continues with a discussion of stability lobes and surface location error analysis with regard to their numerical accuracy. Sensitivity analysis is discussed in the next section. Then, case studies for the numerical accuracy of the sensitivities of the maximum stable axial depth, b_{lim}, and *SLE* are presented for a typical two degree-of-freedom tool. This enables us to

carry out the stability lobe and surface location error sensitivity analysis in the next two sections. Sensitivity is used to determine the effect of input parameters on b_{lim} and *SLE*. This enables the determination of which parameter(s) is the highest contributor to stability enhancement and *SLE* reduction. The uncertainties in b_{lim} and *SLE* predictions are then calculated using two methods 1) the Monte Carlo simulation; and 2) the use of numerical derivatives of the system characteristic multipliers to determine sensitivities. The uncertainty in axial depth effects a reduction in the *MRR*, and the *SLE* uncertainty provides bounds on *SLE* mean expected value. This allows robust optimization that takes into consideration both performance and uncertainty.

Milling Model

A schematic of a two degree-of-freedom milling tool is shown in Figure 21. The tool/work-piece dynamics and cutting forces are used to formulate the governing delay differential equation for the system. Solution of the delay differential equation is found using time finite element analysis (TFEA) [54-56]. This method provides the means for predicting the milling process stability and quality (*SLE*). However, the uncertainty in the input parameters to the solution method places an uncertainty on the stability and *SLE* prediction. These parameters are divided into two groups; 1) uncertainty from lack of knowledge of the tool modal matrices, *K*, *C* and *M*, and the cutting force coefficients (mechanistic force model); and 2) uncertainty in other machining parameters, such as spindle speed, chip load and radial depth. To estimate the parameters in the former, modal testing is used to measure the dynamic parameters while cutting tests are completed to estimate the cutting force coefficients. In the modal parameter estimation the peak amplitude method is used to fit the measured frequency response function. In this method [86, 87], the peak of the magnitude of the frequency response function

corresponds to the natural frequency. From this the half power frequencies are used to estimate the damping ratio. Table 6 lists the mean modal values for 25.4 mm diameter endmill having a 12° helix angle with 114 mm overhang length and the corresponding cutting force coefficients for 6061 aluminum (assuming a mechanistic force model, see Chapter 5). The cutting conditions are also listed in the table. These parameters will be used in the simulations in this chapter for a down milling cut.

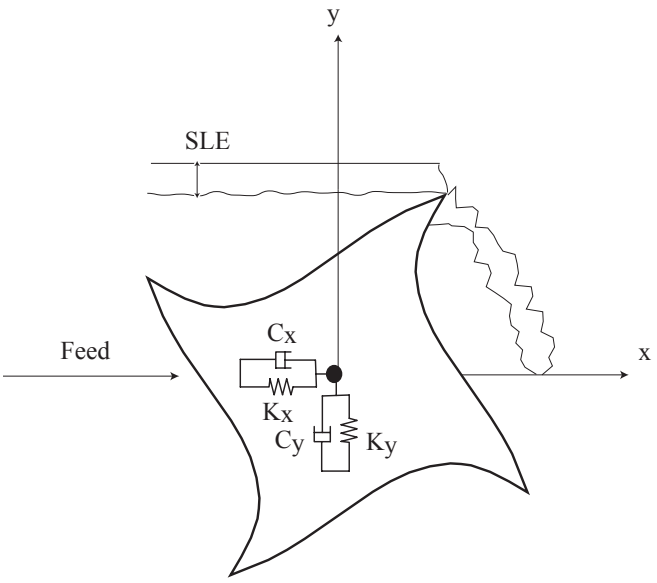

Figure 21. Schematic of 2-D milling model. Surface location error (SLE) due to phasing between cutting force and tool displacement is also shown.

Table 6. Cutting force coefficients, modal parameters and cutting conditions of milling process.

	M (kg)	K (N/m x10^6)	C (N.s/m)	ζ
x	0.44	4.45	83	0.030
y	0.44	3.55	90.9	0.036
K_t (N/m^2 x10^6)	K_n (N/m^2 x10^6)	K_{ne} (N/m x10^3)	K_{te} (N/m x10^3)	
600	180	6	12	
Tool diameter (mm)	radial depth,a (mm)	chip load, c (mm/tooth)	N	
25.4	0.508	0.1	1	

Stability and Surface Location Error Analysis

The stability lobes are used to represent the stable space of axial depth (b) and spindle speed Ω of a milling process. In TFEA [54-57], a discrete map is used to match the tool-free vibration while out of the cut, with the tool vibration in the cut. The system characteristic multipliers ($\vec{\lambda}$) of the map provide the stable cutting zone where $\max\left|\vec{\lambda}\right|$ is less than one.

TFEA provides a field of $\max\left|\vec{\lambda}\right|$ in the design space of b and Ω. The limit of stability, b_{lim} can be found using root-finding numerical techniques. Here we use the bi-section root-finding method. The convergence criterion of the bi-section method should account for the amplification of numerical noise induced by sensitivity estimation. It should be noted that the number of elements affects the accuracy of the estimation.

For calculation of *SLE* in TFEA, the numerical noise is only due to the number of elements. In this section we will discuss the effect of both the convergence criterion and the number of elements on the sensitivity estimation of b_{lim} and *SLE*.

Bi-section Method Convergence Criterion

As described in Chapter 3 the axial depth limit, b_{lim}, was calculated using the bi-section method (Eq. **(3.4)**). Although a relatively large value of ε can be adequate for the calculation of the stability lobes, a tighter limit is needed to calculate the sensitivities. This is attributed to amplification of numerical noise in the derivative calculation. This comparison is made in the Case Studies section.

Number of Elements

The accuracy of TFEA prediction of stability and *SLE* is highly dependent on the number of elements used. The effect of the number of elements is even more apparent

when calculating the sensitivity of the prediction, where a higher number of elements is needed to eliminate numerical noise from the sensitivity calculation.

Numerical Sensitivity Analysis

The sensitivity of axial depth to input parameters $\left(\partial b / \partial X_i\right)$ is cumbersome to compute analytically using the TFEA method; therefore, a numerical derivative is used by implementing a small perturbation.

Factors which affect accurate calculation of sensitivity to inputs include: 1) central difference truncation error; and 2) step size selection. Therefore, a balance needs to be achieved in determining the sensitivity that provides a stable estimate of the sensitivity while maintaining computational efficiency. In the following, we describe these factors and their consideration in the calculation of stability and SLE sensitivities.

Truncation Error

The central difference method is used in the sensitivity calculation. The formula for this method is

$$\frac{\partial b}{\partial X_i} = \frac{b_1 - b_{-1}}{2h} + O\left(h^2\right), \tag{4.1}$$

where h denotes the step size in input parameter X_i, $b_1 = b\left(X_i + h\right)$, $b_{-1} = b\left(X_i - h\right)$ and $O(h^2)$ is the 2nd order truncation error. A higher order formula with 4th order truncation error $O(h^4)$ can also be used. However, as shown in Eq. (4.2), it is two times more computationally expensive than Eq. (4.1),

$$\frac{\partial b}{\partial X_i} = \frac{-b_2 + 8b_1 - 8b_{-1} + b_{-2}}{12h} + O\left(h^4\right). \tag{4.2}$$

In order to help decide whether the higher truncation error formula need be applied (Eq. (4.2)), the sensitivity of b_{lim} with respect to modal stiffness K_x is calculated as a function of step size h. This comparison is made in the Case Studies section.

Step Size

The step size, h, in Eqs. ((4.1) and (4.2)) should be carefully chosen. This is especially important when there is numerical noise in the calculated b_{lim} due to the convergence criterion (Eq. (1)). The step size should be large enough to be out of the numerical noise range, however, not so large that the non-linear variation in the output (b_{lim} or *SLE*) takes effect. The following section illustrates this idea.

Case Studies

In this section, numerical estimations of the sensitivity are made based on different variations of convergence criterion, number of elements, sensitivity analysis formula (Eq. (4.1) and Eq. (4.2)), and step size. The comparisons are made for a 10 krpm spindle speed, 10 elements and $\varepsilon = 3x10^{-4}$ unless otherwise noted. The logarithmic derivative can be used in making these comparisons by evaluating the percentage of change in an output (axial depth, b) due to a percentage change in the input, X_i. It is expressed as

$$\frac{\partial \ln(b)}{\partial \ln(X_i)} = \frac{X_i}{b}\frac{\partial b}{\partial X_i} \tag{4.3}$$

To illustrate the effect of convergence criterion, the logarithmic derivative of b_{lim} with respect to M_x (the X direction modal mass) is calculated for two error limits as a function of step size percentage $(\%h = \Delta X_i / X_i \times 100)$, see Figure 22. It can be seen that a tighter error limit nearly eliminates the numerical noise in the derivative calculation.

The effect of the number of elements on *SLE* sensitivity is illustrated in Figure 23, where the *SLE* sensitivity with respect to K_x is calculated. The $\partial SLE / \partial K_x$ is used to illustrate the effect of the number of elements because it is known that the *SLE* does not depend on the K_x stiffness (tool feeding direction being the *x*-axis). Therefore $\partial SLE / \partial K_x = 0$, which would amplify and illustrate more clearly the effect of the number of elements on the sensitivity estimation. The higher number of elements provides a larger stable region of sensitivity. It should be noted that the 2nd order finite difference method is used in this sensitivity comparison and the bi-section convergence criterion is not applicable here since *SLE* is found from fixed points of the dynamic map (see Eq. **(A.18)** in Appendix A) when the cutting conditions provide a stable cut.

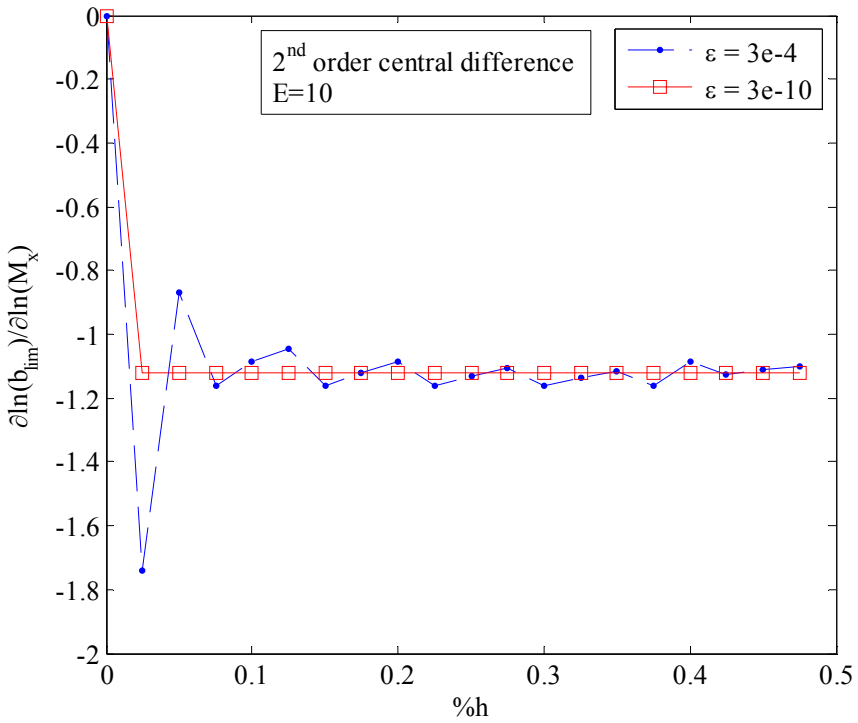

Figure 22. The effect of error limit in the bisection method on numerical noise in the sensitivity calculation (see Table 6).

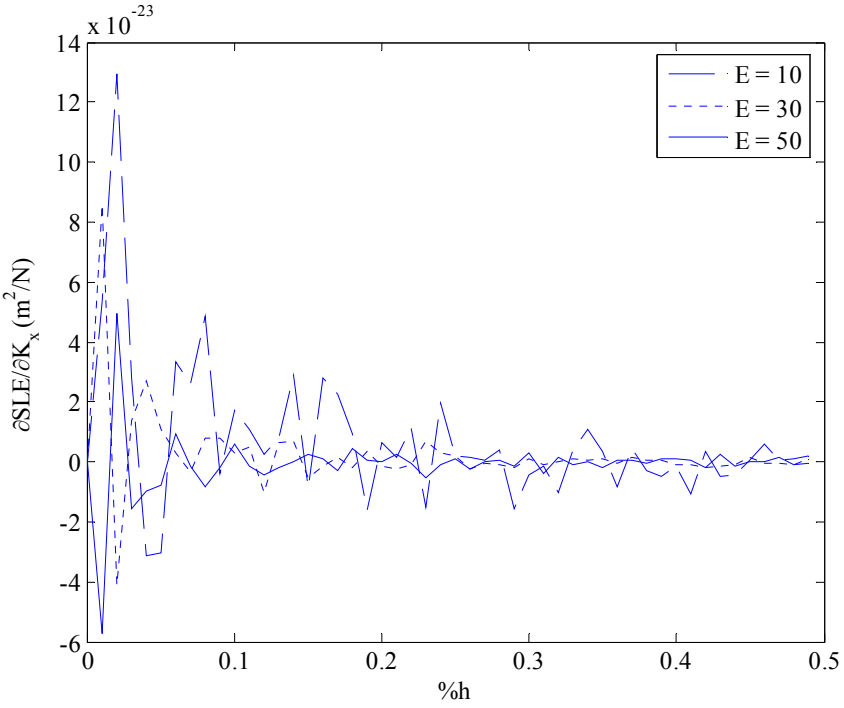

Figure 23. Sensitivity of *SLE* with respect to K_x. The higher number of elements, E, provides more stable sensitivity estimation. The second order finite difference formula is used here (see Table 6).

Figure 24 shows the effect of the central difference truncation error. A finite step size percentage is needed to reach a stable value of the derivative for both formulas. It can be seen that Eq. (4.2) gives a wider range of step sizes at which the sensitivity calculation is stable. However, the improved stability range, or reduction in numerical noise, is not significant to sacrifice computational efficiency for its usage.

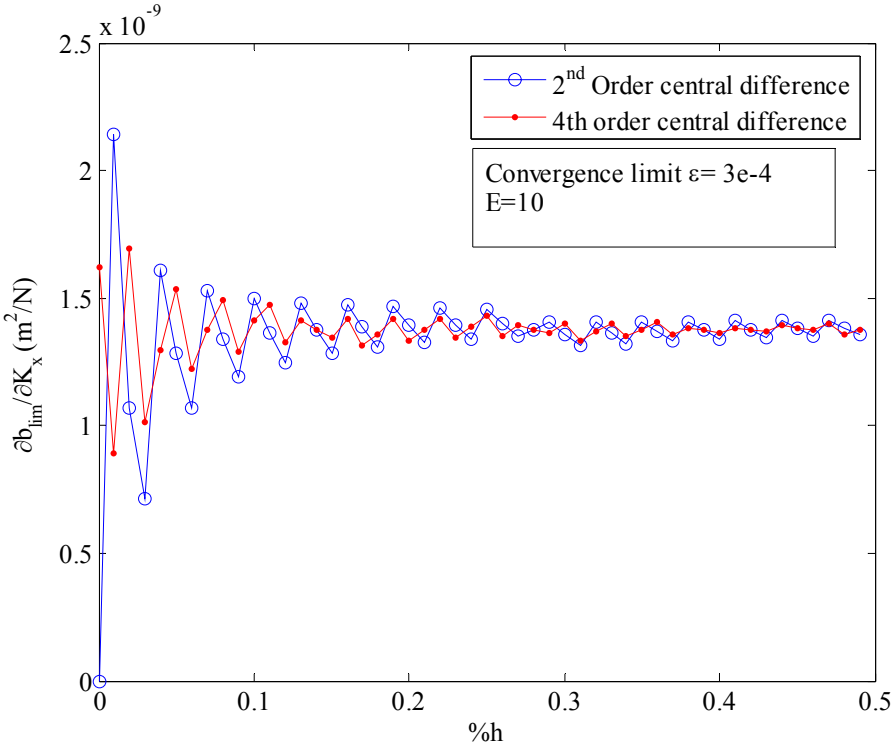

Figure 24. Comparison between 2nd and 4th order central difference formulas. The 4th order formula shows a wider stable region for step size, but higher computation time (see Table 6).

The importance of step size selection can be illustrated by Figure 25, which shows the logarithmic derivative of axial depth with respect to input parameters versus step size percentage. It can be seen that the step size should be chosen high enough to be out of the numerical noise range but not so high so that the non-linear variation is included (in this range of $\%h$ only Ω is non-linear). The figure also indicates the relative sensitivity of axial depth to each input parameter, spindle speed having the largest effect followed by modal mass and stiffness.

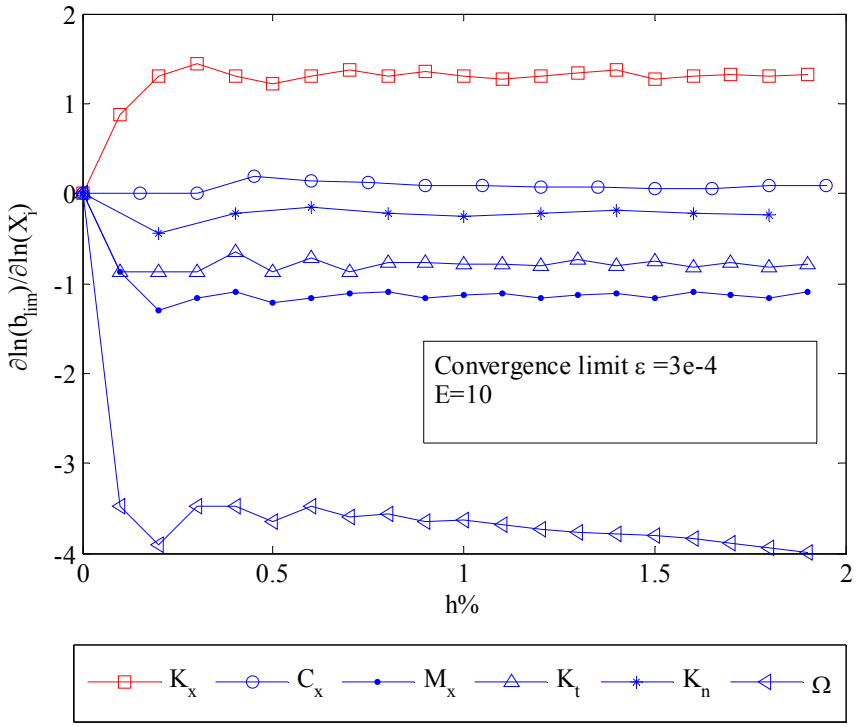

Figure 25. The logarithmic derivative of axial depth with respect to input parameters versus step size percentage (see Table 6).

From Figure 24 and Figure 25 it can be seen that h=0.2% provides a stable sensitivity estimation. To verify that a typical step size of 0.2%, convergence limit $\varepsilon = 3\times10^{-4}$, E=10, and the 2nd order finite difference approximation give correct calculation of sensitivity, the variations of b to modal parameters and cutting coefficients are plotted in Figure 26 and Figure 27, respectively. Also, the slope predicted using Eq. (4.1) with h=0.2% is superimposed on the same plot. The suitable selection of h is indicated by the tangency of the predicted slope to the functional variation. On the other hand, it can be seen that when the variation is linear, the linear approximation can be accurate for a large variation of the input parameter.

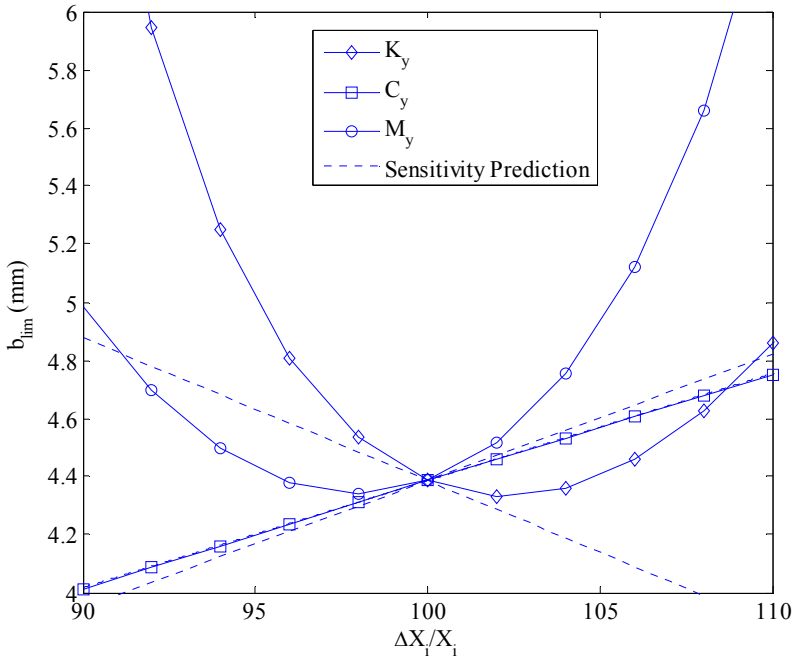

Figure 26. The variation of axial depth b_{lim} with respect to a 10% change in nominal input parameters. The sensitivity of b_{lim} with respect to each parameter is superimposed. Linearity and non-linearity of $b_{lim}(X_i)$ can be observed (see Table 6).

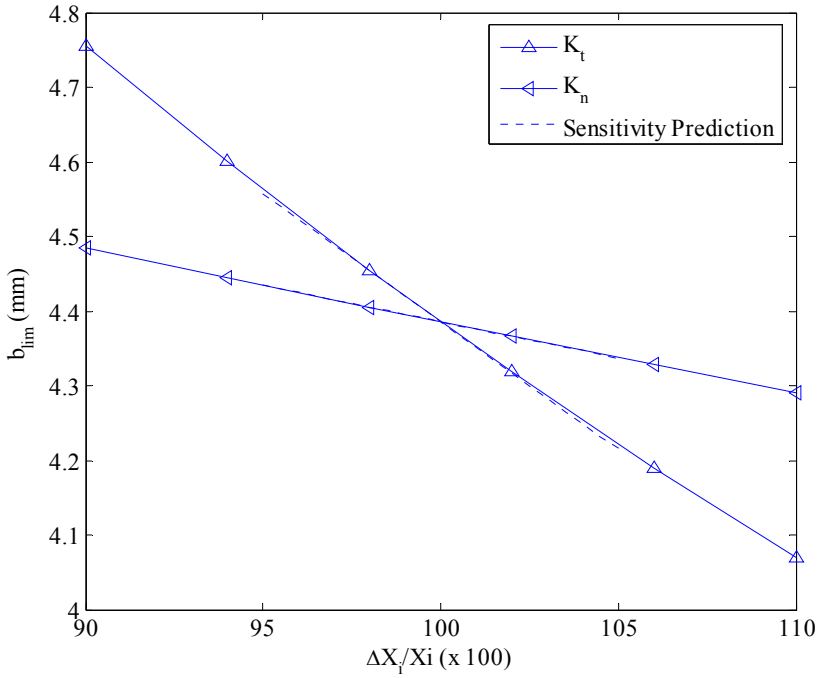

Figure 27. The variation of b_{lim} with respect to a 10% change in K_t and K_n. The sensitivity of b_{lim} with respect to each parameter is superimposed. Linearity of $b_{lim}(Xi)$ can be observed (see Table 6).

Stability Sensitivity Analysis

In this section, calculations of the sensitivity of b_{lim} to the input parameters are provided. The parameters used in the sensitivity calculations are provided in Table 7. In Figure 28 a comparison between the sensitivities of stiffness, *K*, and modal mass, *M*, are compared in the *x* (feed) and *y*-directions of the tool. As can be seen in the figure, the sensitivities in the *x* and *y*-directions are comparable in magnitude; however, the sensitivity in the *y*-direction is inaccurate near discontinuities in the system characteristic multipliers. This will be explained in the Uncertainty section with a graphic depicting these discontinuities.

Table 7. Parameters used in sensitivity analysis.

h (%)	E	Central difference	ε
0.2	10	2nd order	3×10^{-4}

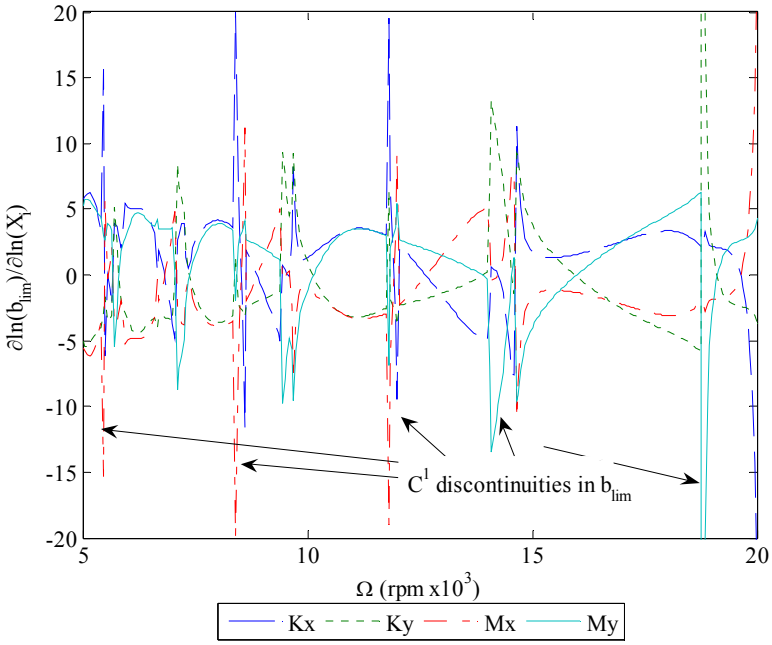

Figure 28. Sensitivity of axial depth blim to changes in modal mass *M* and modal stiffness *K* in the *x* and *y*-directions (see Table 6).

In Figure 29, the effect of damping on the stability is shown to be minimal compared to the modal stiffness and mass. This is a somewhat counter-intuitive result, but can be explained by regeneration (undulations in the cut surface experienced by the tooth in the current cut that are caused by the tooth vibration in the previous cut), which is a primary physical phenomenon that causes instability. The modal mass and stiffness have a great effect on the system's natural frequency, which has a significant effect on regeneration. This also explains the result shown in Figure 30, where the sensitivity of axial depth b_{lim} to a change in spindle speed is significant and comparable to modal mass and stiffness. The effect of cutting force coefficients is shown in Figure 31, where the tangential cutting force coefficient, K_t, has more effect on the axial depth limit than the normal direction coefficient, K_n.

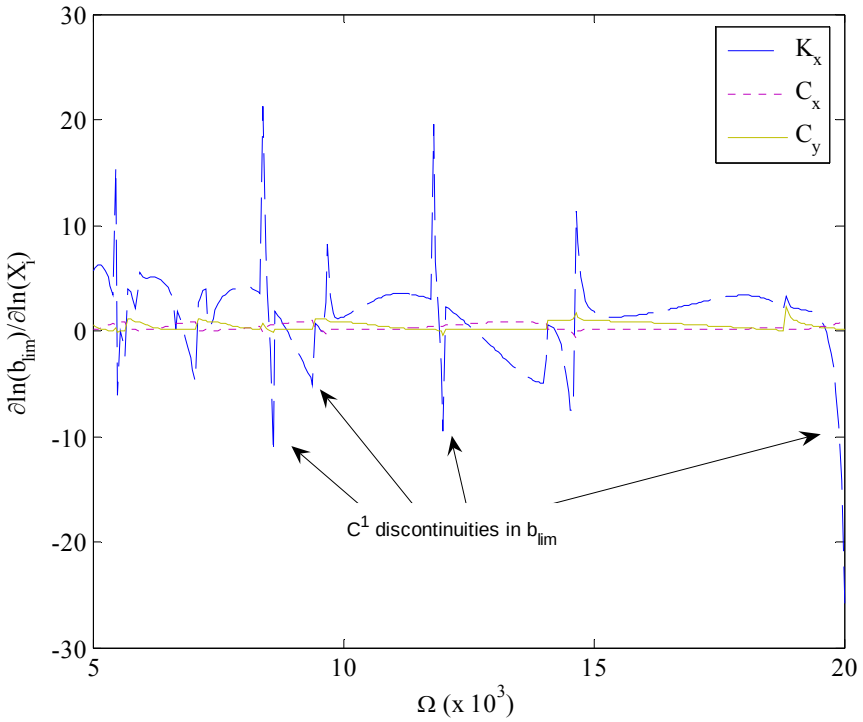

Figure 29. Sensitivity of axial depth b_{lim} to changes in modal damping C in the x and y-directions. The damping sensitivity is compared to modal stiffness sensitivity in the x-direction (see Table 6).

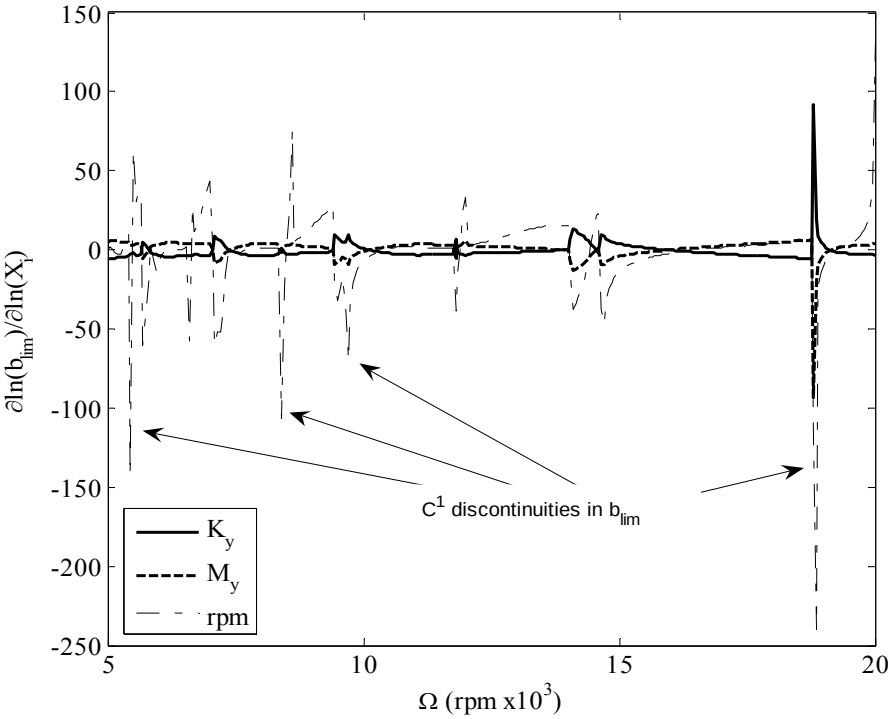

Figure 30. Sensitivity of axial depth b_{lim} to changes in spindle speed. The spindle speed sensitivity is compared here to the modal mass and stiffness in y-direction (see Table 6).

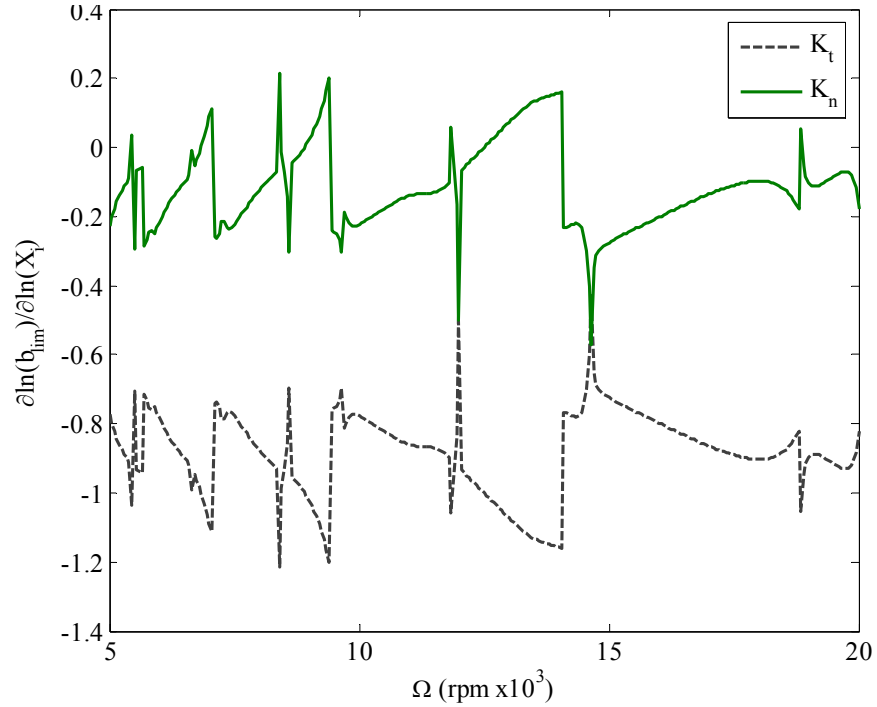

Figure 31. Sensitivity of axial depth b_{lim} to changes in force cutting coefficients in the tangential K_t and normal directions K_n. Higher sensitivity can be seen for K_t (see Table 6).

Surface Location Error Sensitivity Analysis

The sensitivity of surface location error, *SLE*, to changes in input parameters is examined here. The parameters listed in Table 6 are used with b=1 mm and down milling case. In Figure 32, the sensitivity of *SLE* to changes in modal parameters in the *y*-direction is shown. Again, it can be seen that changes in K_y and M_y contribute more than C_y to a change in *SLE*. In Figure 33, the effect of cutting force coefficients is shown, where it is observed that the highest contributors to *SLE* sensitivity are K_t and K_{te}. Also, in Figure 34, *SLE* sensitivity to spindle speed and radial depth, r_{step}, is shown. Substantial sensitivity to spindle speed can be seen. This is due to the dependence of *SLE* on the relationship between the tool point frequency response and the selected spindle speed. As the spindle speed changes, it tracks different parts of the response.

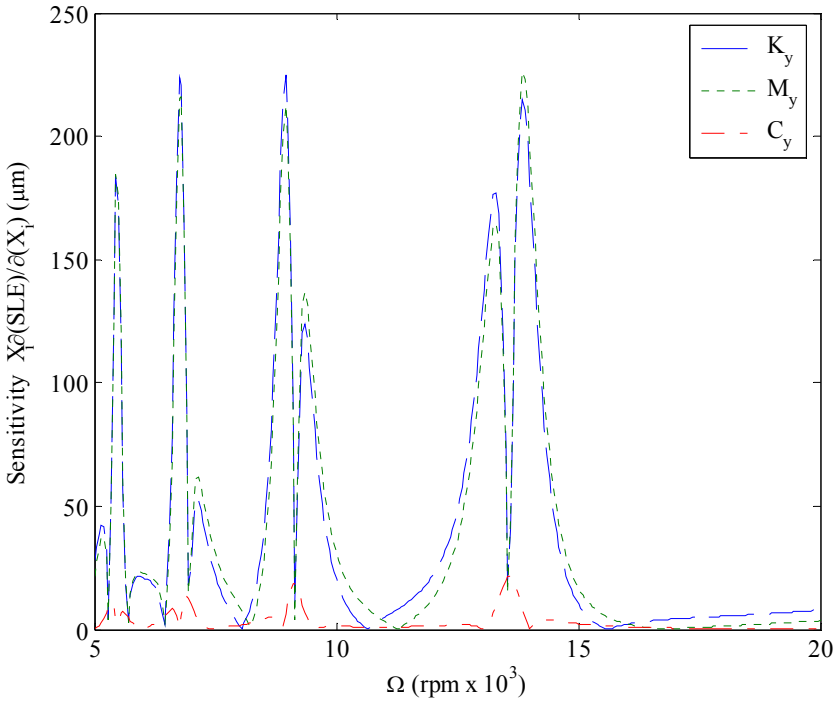

Figure 32. Sensitivity of surface location error *SLE* to changes in modal parameters in *y*-direction (see Table 6).

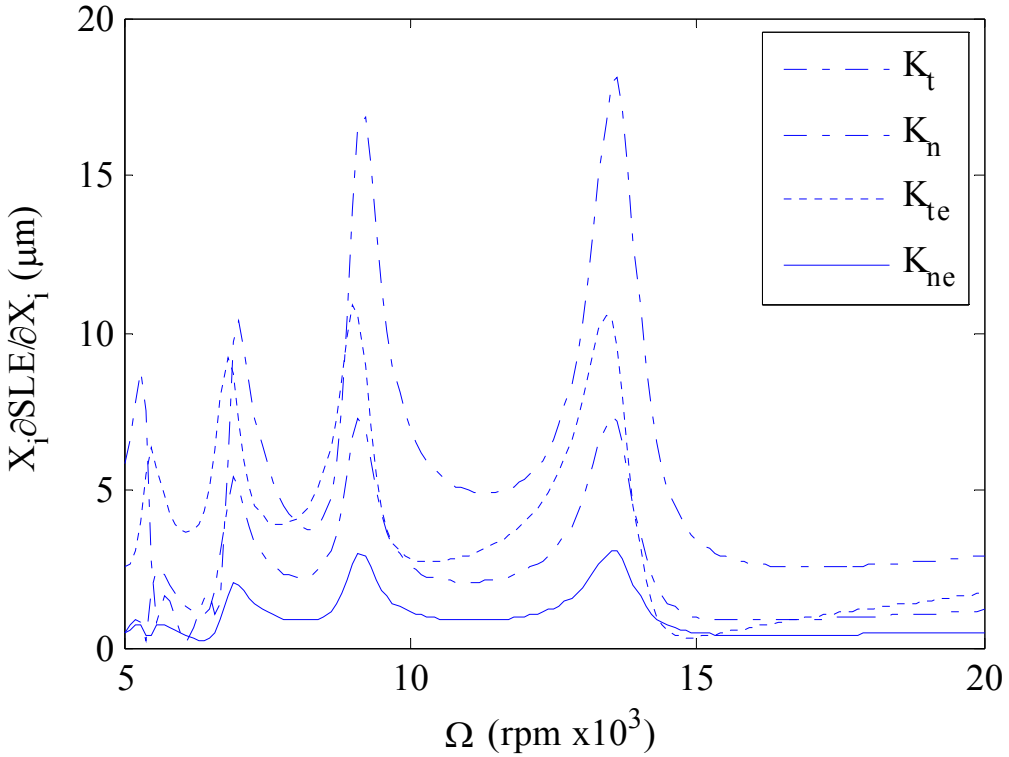

Figure 33. Sensitivity of *SLE* to cutting force coefficients (see Table 6).

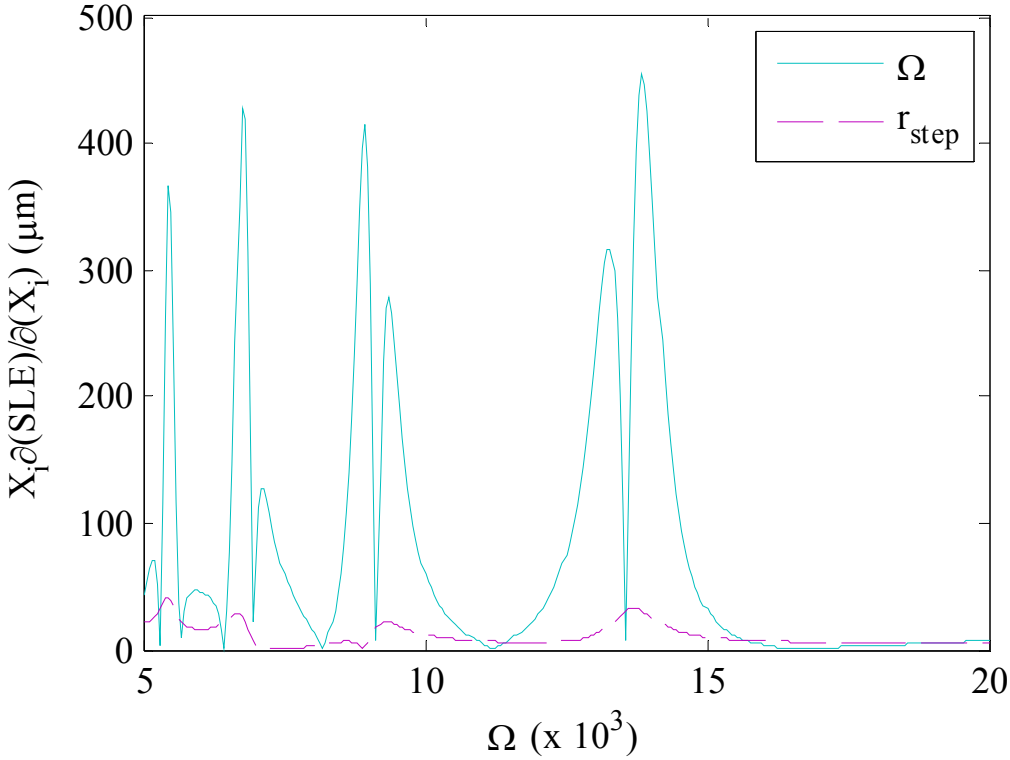

Figure 34. Sensitivity of *SLE* to spindle speed and radial depth of cut (see Table 6).

Uncertainty of Stability Boundary and Surface Location Error

Input Parameters Correlation Effect

The correlation between the input parameters can have significant effect on the prediction of uncertainty. Neglecting the correlation can result in erroneous estimation of the uncertainty, especially when the input parameters are highly correlated. Inclusion of the covariance matrix between parameters is necessary in this case. The input parameters can be classified into three groups: dynamic modal parameters of the tool (work-piece assumed rigid), cutting force coefficients and machining parameters (e.g., radial step and spindle speed). In Chapter 5, estimation of the correlation between parameters of the first two groups is explained and used in the uncertainty prediction.

The combined standard uncertainty u_c can be found using sensitivities of output (b_{lim} or SLE) to input parameters. For the case of axial depth limit, u_c is given as [88]:

$$u_c\left(b_{\lim}\right)^2 = \sum_{i=1}^{m}\left(\frac{\partial b_{\lim}}{\partial X_i}u\left(X_i\right)\right)^2 + 2\sum_{i=1}^{m-1}\sum_{i=1}^{m}\frac{\partial b_{\lim}}{\partial X_i}\frac{\partial b_{\lim}}{\partial X_j}u\left(X_i, X_j\right), \tag{4.4}$$

where $u(X_i)$ refers to the standard uncertainty in the input parameter X_i, $u(X_i,X_j)$ is the estimated covariance between parameters X_i and X_j, and m is the number of input parameters. The degree of the correlation between X_i and X_j is characterized by the correlation coefficient

$$r\left(X_i, X_j\right) = \frac{u\left(X_i, X_j\right)}{u\left(X_i\right)u\left(X_j\right)}. \tag{4.5}$$

In the Monte Carlo and Latin Hype-Cube sampling methods (described next), the multivariate normal distribution can be used to estimate the confidence level, in which case the covariance matrix between parameters controls the random sampling procedure.

Monte Carlo Simulation

The combined standard uncertainty, u_c, of the stability boundary (b_{lim}) and surface location error (*SLE*) can be predicted using Monte Carlo simulation. In this method, a random sample of size *n* is selected from the population of each input parameter. A normal distribution of the input parameters is assumed. In the sample *n*, the nominal value and standard deviation of each input parameter are used to generate the sample. The axial depth limit and surface location error are then calculated using TFEA for each point in the sample. The standard deviation of the predicted b_{lim} and *SLE* is then calculated from sample output for the range of spindle speeds of interest. It should be noted here that in doing so, no correlation between the input parameters is assumed, which is a common, and sometimes *erroneous*, approach.

To illustrate the effect of uncertainty in the input parameters on stability boundary uncertainty, standard uncertainties of 5%, 0.5%, 0.001% and 0.5% are assigned to nominal values of the cutting force coefficients, modal parameters, radial step, and spindle speed, respectively. The values of the standard uncertainties assigned correspond to practical variation in the parameters involved. The parameters are assumed to be uncorrelated here. A sample size of 1000 is used. The stability boundary uncertainty is found, as shown in Figure 35, for one standard deviation interval around the neutral stability boundary.

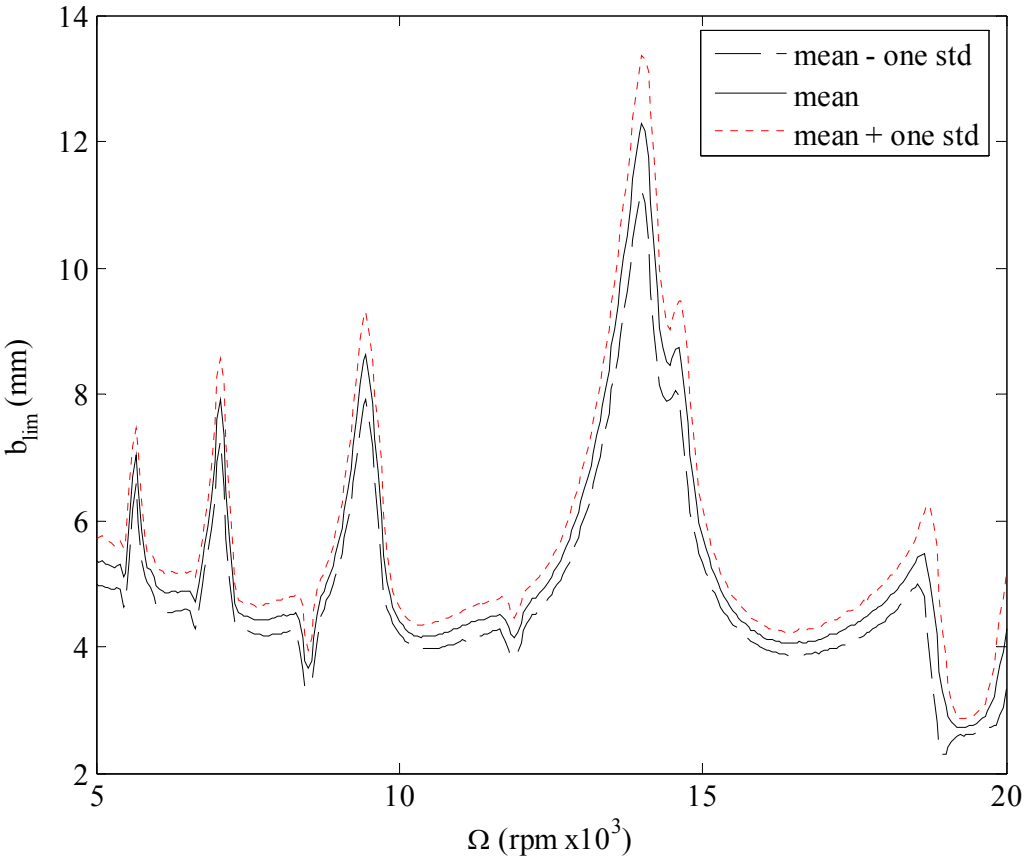

Figure 35. Confidence in stability boundary due to input parameters uncertainties using Monte Carlo simulation (see Table 6).

Sensitivity Method

The combined standard uncertainty u_c in axial depth limit while neglecting correlation between input parameters can be obtained from Eq. (4.4) as

$$u_c\left(b_{\lim}\right)=\sqrt{\sum_{i=1}^{m}\left(\frac{\partial b_{\lim}}{\partial X_i}u\left(X_i\right)\right)^2}\,, \tag{4.6}$$

where $u(X_i)$ refers to the standard uncertainty in the input parameter X_i (same used for Monte Carlo method), and m is the number of input parameters. Although this relation assumes no correlation between input parameters it should be noted that cutting force

coefficients (K_t, K_n, K_{te}, K_{ne}) and modal parameters (K, C, M) may be correlated in practice.

The same standard uncertainty is assumed in the input parameters as in previous sections and the confidence level in axial depth limit is calculated for an interval of ± 2 $u_c(b_{lim})$. Figure 36 shows the close agreement found using the two methods. However, it should be noted that the sensitivity method can be inaccurate near points where the function (b_{lim}) is C^1 discontinuous. Figure 37 shows the direct correspondence between the inaccurate sensitivity and C^1 discontinuity in $\vec{\lambda}$. The C^1 discontinuity in b_{lim} leads to inaccurate estimation of $u_c(b_{lim})$ (see Eq. (4.6)).

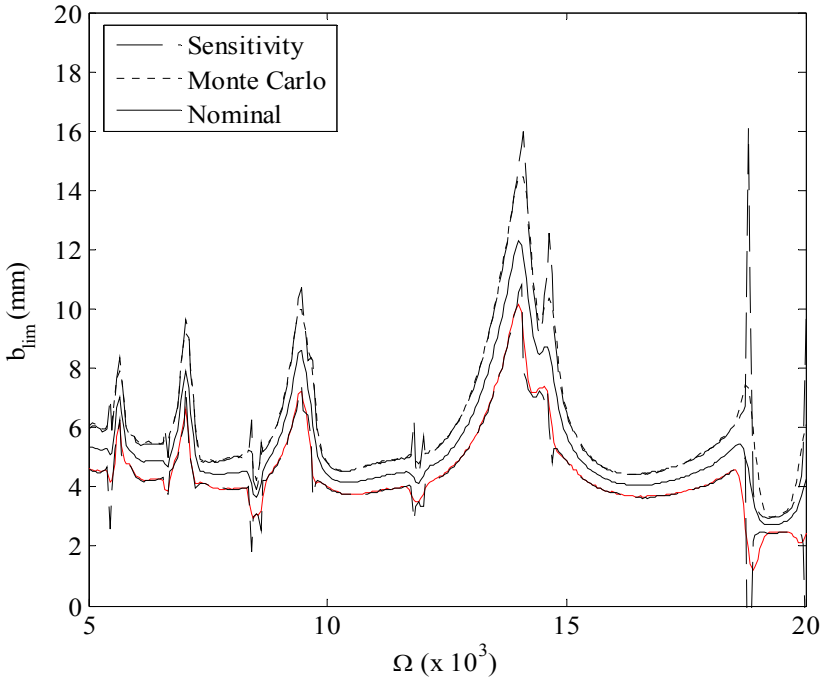

Figure 36. Uncertainty boundary in axial depth limit using two standard deviation confidence interval. Uncertainty is calculated using sensitivity method and Monte Carlo method (see Table 6).

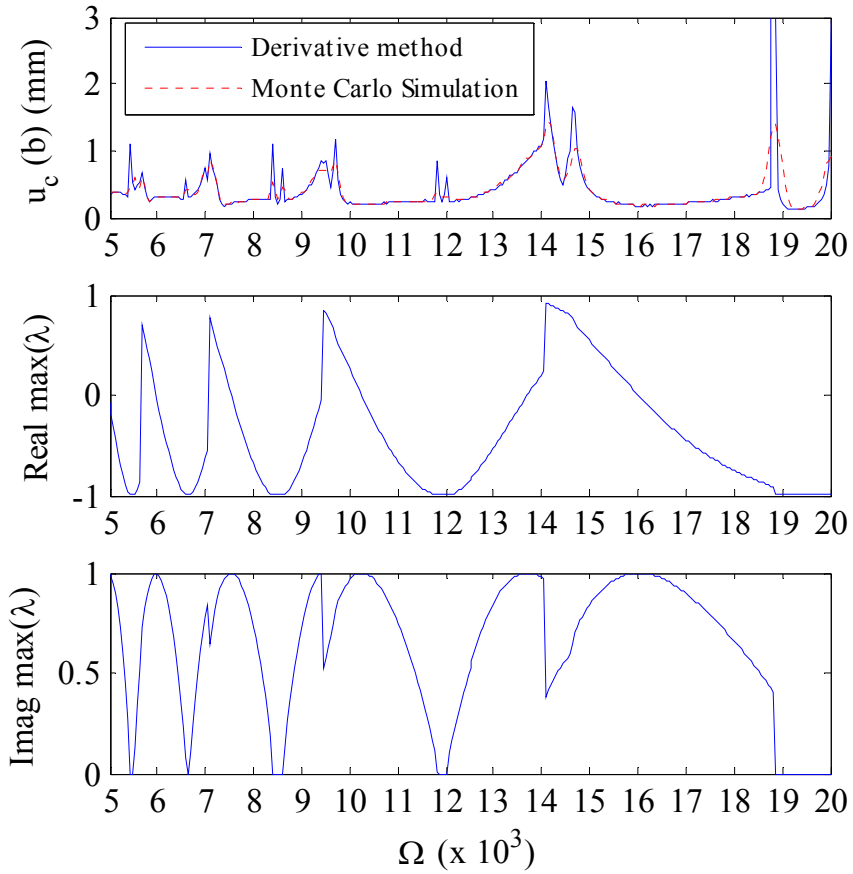

Figure 37. Uncertainty in axial depth using sensitivity and Monte Carlo methods. Inaccuracies in the sensitivity method can be seen near C^1 discontinuity in the real and imaginary part of system characteristic multipliers (see Table 6).

It should be noted here that predicting the uncertainty by Eq. (4.6) uses a linear approximation. The standard uncertainties assumed earlier are small where the linear approximation is still valid. However, if the uncertainties in the input parameters are large, then that linear approximation is no longer valid. In this case, simple random sampling methods (such as Monte Carlo simulation) are more appropriate.

The surface location error uncertainty is found similarly using both methods. However, as shown earlier (see Figure 32 and Figure 34), the *SLE* sensitivities are accurate and do not depend on the characteristic multipliers' continuity. Since the *SLE* is only defined for stable cutting conditions (see Eq. (A.18) in Appendix A) and explains

the close prediction of uncertainty in SLE using sensitivity and Monte Carlo methods (Figure 38).

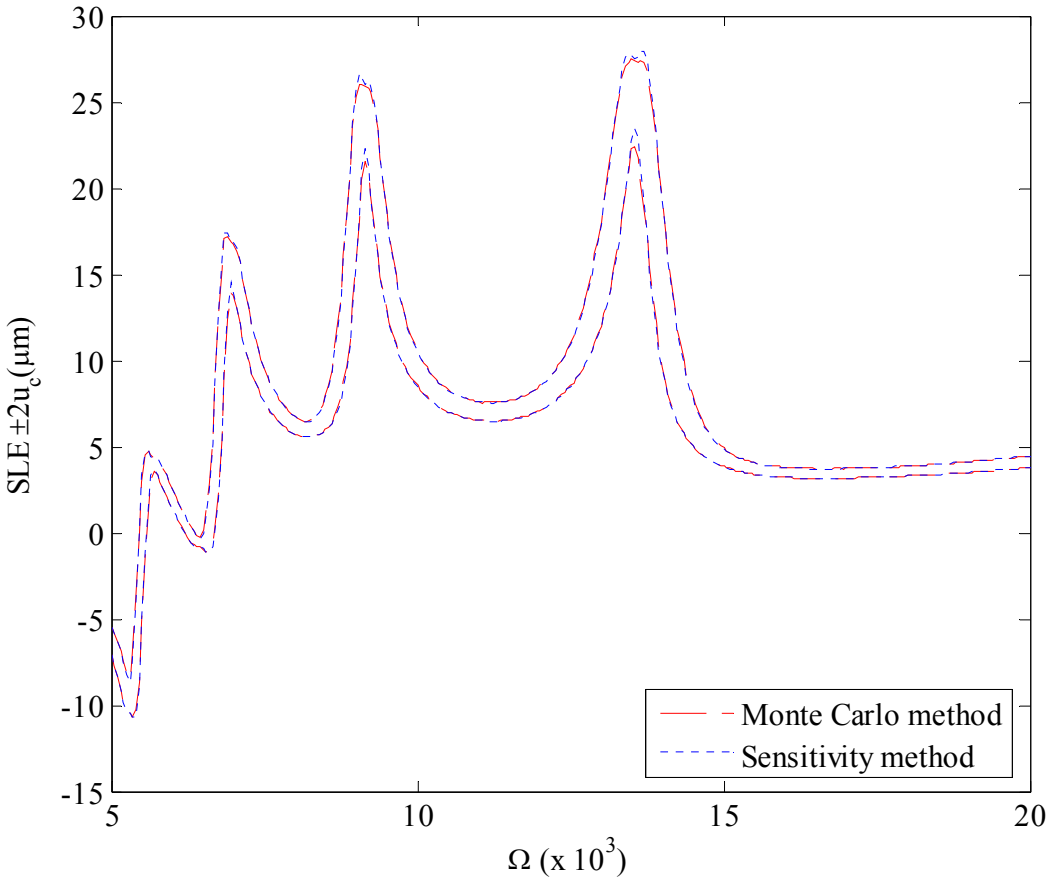

Figure 38. Surface location error uncertainty with two standard deviation confidence interval on the nominal *SLE*. Close agreement is observed (see Table 6).

Latin Hyper-Cube Sampling Method

This method was originally proposed as a variance reduction technique [89] in which the estimated variance is asymptotically lower than with simple random sampling (Monte Carlo method) [90, 91]. That is, for a sample size *L*, this method gives a lower estimate of the output variance than is possible with the Monte Carlo method. The basic idea of this method is that each value (or range of values) of a variable is represented in the sample, no matter which value turns out to be the most important. In this way, the

sampling distribution is divided into a number of strata with a random selection inside each stratum. The Latin Hyper-Cube method will be used in Chapter 5 for predicting the standard combined uncertainty of the stability and surface location error cutting tests in that chapter.

Robust Optimization under Uncertainty

In order to account for uncertainty in the axial depth stability limit, the safety factor design analogy is used here. The deterministic optimization algorithm implemented in Chapter 3 (Eq. (3.9)), repeated here, can be modified to account for the axial depth uncertainty.

$$\begin{aligned} & min\left[\frac{\left|f_{SLE}\left(b,\Omega+\delta\right)\right|+\left|f_{SLE}\left(b,\Omega\right)\right|+\left|f_{SLE}\left(b,\Omega-\delta\right)\right|}{3}\right], \\ subject\ to: \quad & -f_{MRR}\left(b,\Omega\right)\le e_i, \qquad for\ i\ =\ 1\ldots k \\ & \left\{g_\lambda\left(b,\Omega-\delta\right)\cap g_\lambda\left(b,\Omega\right)\cap g_\lambda\left(b,\Omega+\delta\right)\right\}\le 1, \\ & for\ a\ series\ of\ selected\ limits\ (e)\ on\ f_{MRR}, \end{aligned} \tag{4.7}$$

Therefore, the axial depth b used in the stability constraint is set equal to an uncertainty inflated value. That is, b is replaced by $b+ U_e$, where $U_e = ku_c\left(b\right)$ is the expanded uncertainty, k is a factor that estimates the uncertainty confidence interval and $u_c(b)$ is the combined standard uncertainty in the axial depth. Therefore Eq. (4.7) becomes

$$\begin{aligned} & min\left[\frac{\left|f_{SLE}\left(b,\Omega+\delta\right)\right|+\left|f_{SLE}\left(b,\Omega\right)\right|+\left|f_{SLE}\left(b,\Omega-\delta\right)\right|}{3}\right], \\ subject\ to: \quad & -f_{MRR}\left(b,\Omega\right)\le e_i, \qquad for\ i\ =\ 1\ldots k \\ & \left\{g_\lambda\left(b+U_e,\Omega-\delta\right)\cap g_\lambda\left(b+U_e,\Omega\right)\cap g_\lambda\left(b+U_e,\Omega+\delta\right)\right\}\le 1, \\ & for\ a\ series\ of\ selected\ limits\ (e)\ on\ f_{MRR}, \end{aligned} \tag{4.8}$$

Discussion

In this chapter, the sensitivities of axial depth limit and surface location error to model input uncertainties were studied. Numerical estimation of the sensitivities can be challenging, where several factors contribute to the accuracy of the estimation. Step size is one of the significant factors that affect the accuracy of the estimation.

The sensitivity analysis aids in identifying the relative contribution of the milling model input parameters to the sensitivity of either axial depth limit or surface location error. For the case of axial depth, the spindle speed, followed by modal stiffness and mass, is the most significant contributor. In the case of cutting force coefficients, the tangential cutting force coefficient is found to contribute more to the sensitivity than the normal cutting force coefficient. As for the surface location error sensitivity, the same trend can be observed. However, for the cutting force coefficients, the edge tangential cutting force coefficient significantly contributes to the *SLE*.

The uncertainty in axial depth and surface location error was predicted using two methods: the sensitivity method and the Monte Carlo simulation approach. Comparable agreement is shown. However, the sensitivity method is more efficient computationally. For example, in the case of *SLE* uncertainty prediction, Monte Carlo simulation required 9.34 hours, while the sensitivity method needed only 0.26 hours (36 times more efficient). It is noted that for the $u_c(SLE)$ case, when the milling parameters are well into the stable region, the accuracy of the sensitivity method is not sacrificed at the cost of efficiency as is the case for $u_c(b)$ at discontinuities in the characteristic multipliers.

Finally, the optimization algorithm introduced in Chapter 3 was modified to account for confidence levels in the axial depth limit. This allows robust optimization to

account for inherent uncertainty in the mean values of the input parameters. In Chapter 5 an implementation of this algorithm is demonstrated.

CHAPTER 5
EXPERIMENTAL RESULTS

The milling model accuracy depends on reliable determination of cutting force coefficients and tool or work-piece modal parameters. These values are found experimentally and their uncertainties contribute to the uncertainty of the model prediction. In this chapter, the experimental procedure used to determine these parameters is described and then the optimization algorithm is executed using the experimentally determined input parameters to find the Pareto optimal points. Another set of experiments is completed to validate/invalidate these optimal points. Using the optimization algorithm, the strength and weakness of the mathematical model or solution method can be obtained.

Cutting Force Coefficients

Milling Forces

The average milling forces during one tooth period in the *x* and *y*-directions are [92, 93],

$$\begin{aligned}\bar{F}_x &= \left\{\frac{Nbc}{8\pi}\left[K_t\cos(2\phi)-K_n\left[2\phi-\sin(2\phi)\right]\right]+\frac{Nb}{2\pi}\left[-K_{te}\sin(\phi)+K_{ne}\cos(\phi)\right]\right\}_{\phi_{st}}^{\phi_{ex}}\\ \bar{F}_y &= \left\{\frac{Nbc}{8\pi}\left[K_t\left[2\phi-\sin(2\phi)\right]+K_n\cos(2\phi)\right]-\frac{Nb}{2\pi}\left[K_{te}\cos(\phi)+K_{ne}\sin(\phi)\right]\right\}_{\phi_{st}}^{\phi_{ex}}\end{aligned} \quad (5.1)$$

where K_{te} and K_{ne} are the tangential and normal edge cutting force coefficients, respectively. In slotting tests (see Figure 7), the entry and exit angles of the cutter are

$\phi_{st} = 0$ and $\phi_{ex} = \pi$, respectively. The average forces per tooth period for this case are found to be:

$$\begin{aligned} \overline{F}_x &= -\frac{Nb}{4} K_n c - \frac{Nb}{\pi} K_{ne} \\ \overline{F}_y &= \frac{Nb}{4} K_t c + \frac{Nb}{\pi} K_{te} \end{aligned} . \tag{5.2}$$

Equation (5.2) can be written as a function of chip load (c) as:

$$\overline{F}_q = \overline{F}_{qc} c + \overline{F}_{qe} \quad (q = x, y, z). \tag{5.3}$$

The experimental procedure consists of completing multiple cutting tests at varying chip loads and recording the cutting forces. For each chip load increment, the average cutting forces in the x and y-directions are measured, and then a linear regression of the data points is made to extract the cutting coefficients using Eqs. (5.2) and (5.3):

$$\begin{aligned} K_t &= \frac{4\overline{F}_{yc}}{Nb}, \qquad K_{te} = \frac{\pi \overline{F}_{ye}}{Nb} \\ K_n &= -\frac{4\overline{F}_{xc}}{Nb}, \quad K_{ne} = -\frac{\pi \overline{F}_{xe}}{Nb}. \end{aligned} \tag{5.4}$$

For radial immersions less than the cutter diameter, the entry and exit angles differ from the slotting case. For up-milling (see Figure 7) the entry and exit angles of the cutter are $\phi_{st} = 0$ and $\phi_{ex} = \cos^{-1}\left(1 - \frac{a}{R}\right)$, where a is the radial depth of cut. Substituting in Eq. (5.1) gives:

$$\overline{F}_x = \frac{Nbc}{8\pi}\left[K_t\left[\cos\left(2\phi_{ex}\right) - 1\right] - K_n\left[2\phi_{ex} - \sin\left(2\phi_{ex}\right)\right]\right] + \frac{Nb}{2\pi}\left[-K_{te}\sin\left(\phi_{ex}\right) + K_{ne}\left[\cos\left(\phi_{ex}\right) - 1\right]\right] \tag{5.5}$$

Factoring Eq. (5.5) in terms of chip load c gives:

$$\bar{F}_x = \bar{F}_{xc} c + \bar{F}_{xe} \tag{5.6}$$

$$\begin{aligned} \bar{F}_{xc} &= \frac{Nb}{8\pi}\left[K_t\left[\cos\left(2\phi_{ex}\right)-1\right]-K_n\left[2\phi_{ex}-\sin\left(2\phi_{ex}\right)\right]\right] \\ \bar{F}_{xe} &= \frac{Nb}{2\pi}\left[-K_{te}\sin\left(\phi_{ex}\right)+K_{ne}\left[\cos\left(\phi_{ex}\right)-1\right]\right]. \end{aligned} \tag{5.7}$$

Similarly, the following equations are obtained for the *y*-direction.

$$\bar{F}_y = \frac{Nbc}{8\pi}\left[K_t\left[2\phi_{ex}-\sin\left(2\phi_{ex}\right)\right]+K_n\left[\cos\left(2\phi_{ex}\right)-1\right]\right]-\frac{Nb}{2\pi}\left[K_{te}\left[\cos\left(\phi_{ex}\right)-1\right]+K_{ne}\sin\left(\phi_{ex}\right)\right] \tag{5.8}$$

$$\bar{F}_y = \bar{F}_{yc} c + \bar{F}_{ye} \tag{5.9}$$

$$\begin{aligned} \bar{F}_{yc} &= \frac{Nb}{8\pi}\left[K_t\left[2\phi_{ex}-\sin\left(2\phi_{ex}\right)\right]+K_n\left[\cos\left(2\phi_{ex}\right)-1\right]\right] \\ \bar{F}_{ye} &= -\frac{Nb}{2\pi}\left[K_{te}\left[\cos\left(\phi_{ex}\right)-1\right]+K_{ne}\sin\left(\phi_{ex}\right)\right] \end{aligned} \tag{5.10}$$

Writing Eqs. (5.7) and (5.10) in matrix form to solve for the cutting coefficients the final equation can be expressed as shown in Eq. (5.11).

$$\begin{Bmatrix} \bar{F}_{xc} \\ \bar{F}_{yc} \\ \bar{F}_{xe} \\ \bar{F}_{ye} \end{Bmatrix} = \begin{bmatrix} \frac{Nb}{8\pi}\begin{bmatrix} \cos\left(2\phi_{ex}\right)-1 & -2\phi_{ex}+\sin\left(2\phi_{ex}\right) \\ 2\phi_{ex}-\sin\left(2\phi_{ex}\right) & \cos\left(2\phi_{ex}\right)-1 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \frac{Nb}{2\pi}\begin{bmatrix} -\sin\left(\phi_{ex}\right) & \cos\left(\phi_{ex}\right)-1 \\ 1-\cos\left(\phi_{ex}\right) & -\sin\left(\phi_{ex}\right) \end{bmatrix} \end{bmatrix} \begin{Bmatrix} K_t \\ K_n \\ K_{te} \\ K_{ne} \end{Bmatrix} \tag{5.11}$$

The same procedure can be used to solve for the cutting coefficients in the down-milling case (Figure 7).

Experimental Procedure

Proper selection of a suitable dynamometer to measure the dynamic cutting forces is important. Some of the factors that need to be addressed are the calibration range of the

dynamometer and its dynamic response. Simulation of the cutting forces helps in addressing the issue of cutting force magnitude range. Using time-domain simulation of the cutting forces and approximate cutting coefficient values, an estimate of the typical range of cutting forces can be found. Euler integration is used to solve for the tool displacement during the cut in the 2nd order differential equation (Eq. (3.2)) and find the corresponding cutting forces in the *x* and *y*-directions. An example is shown in Figure 39. It is seen that a dynamometer with the 0 kN to 5 kN range is acceptable, although the force levels are relatively small compared to the full scale value. A Kistler 9257A dynamometer with 5 kN range was available for these tests. One requirement for this dynamometer is that the cutting force is applied to the dynamometer not more than 25 mm above the top surface of the dynamometer.

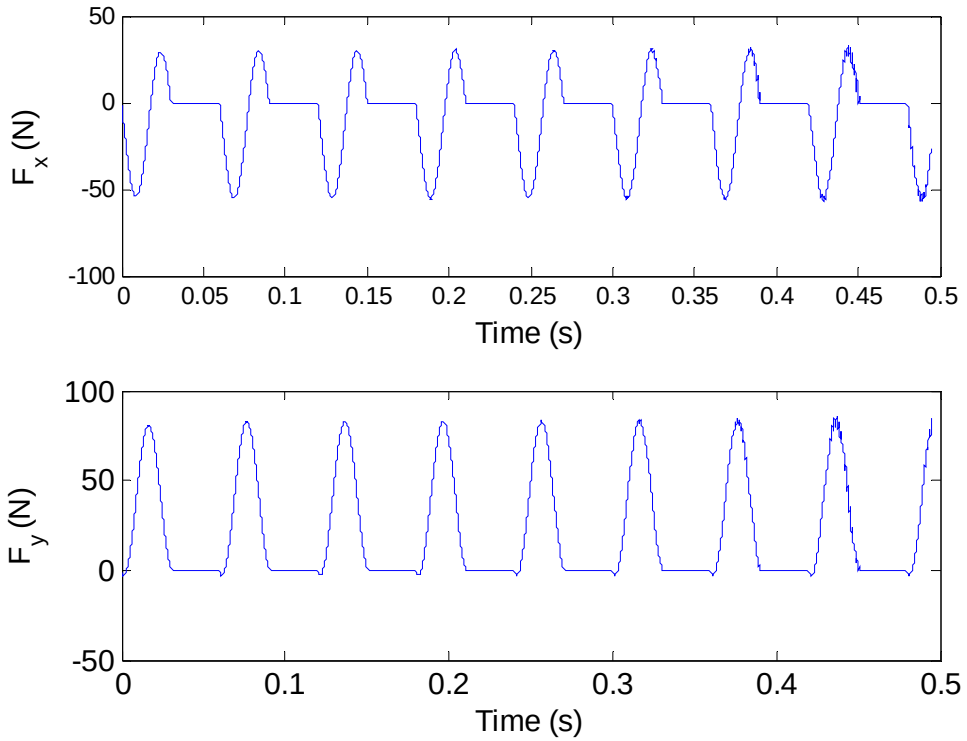

Figure 39. Example simulation of cutting forces to facilitate proper selection of dynamometer.

A 25 mm thick 6061-T6 aluminum work-piece was sized to 100 mm x 85 mm, then faced and drilled to fit the dynamometer hole pattern as shown in Figure 40. Slotting cutting tests were made for a 25.4 mm diameter end mill with a 145 mm overhang and a single 12° helix insert for chip load range of 0.1-0.24 mm/tooth in 0.02 mm/tooth steps. The cutting forces in *x* and *y*-directions were measured for each chip load using an axial depth of 0.4 mm. Two sets of measurements were made for a 1000 rpm spindle speed. To address the influence of spindle speed on cutting coefficients, the above two sets were repeated for {5000, 10000, 15000 and 20000} rpm. The average value of the measured cutting forces was inserted into Eq. (5.4) to solve for the cutting coefficients. Average cutting coefficients of the two sets of measurements at each spindle speed are listed in Table 8.

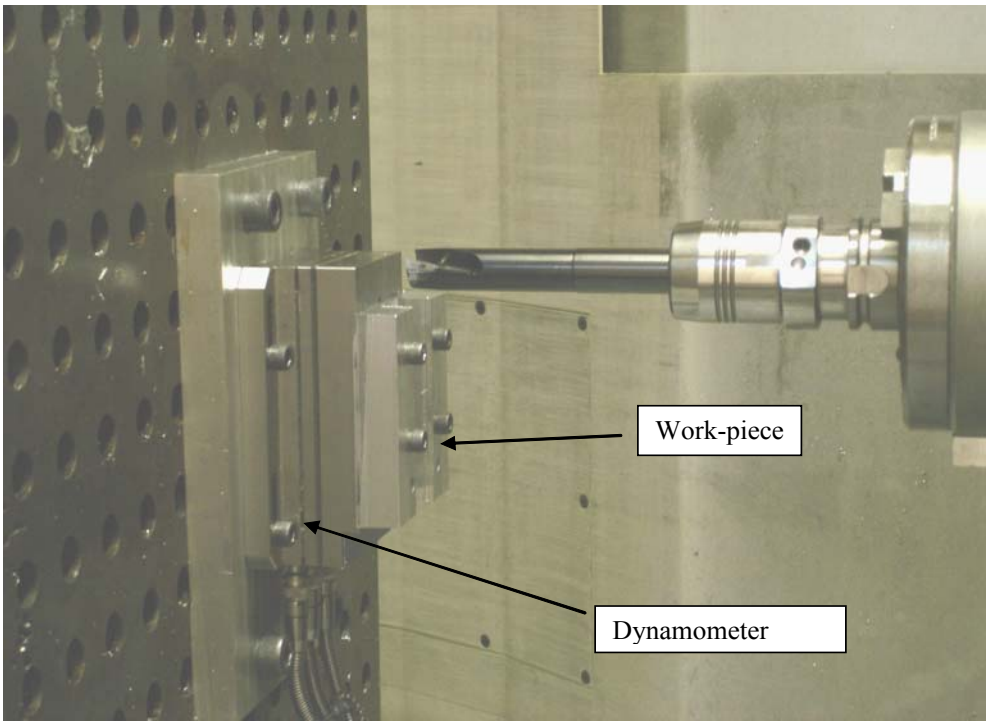

Figure 40. Work-piece, dynamometer and tool setup

A regression analysis of the cutting force coefficients as a function of spindle speed was carried out. For K_t and K_n, a linear regression with logarithmic transformation of spindle speed indicates a statistically significant relation with a P-Value of less than 0.007. Figure 41 and Figure 42 show the trend line for this regression for both K_t and K_n, respectively. For the edge cutting force coefficients K_{ne} and K_{te} the regression doesn't indicate a significant statistical relation between K_{ne} or K_{te} and spindle speed. The P-Value for the slope of the regression was 0.39 and 0.55, respectively.

Table 8. Cutting coefficients for 1 insert endmill for slotting cutting tests

Ω (krpm)	K_t (N/mm^2)	K_n (N/mm^2)	K_{te} (N/mm)	K_{ne} (N/mm)
1	1321	379	28	32
5	832	183	47	34
10	841	62	37	38
15	655	34	52	33
20	670	65	37	26

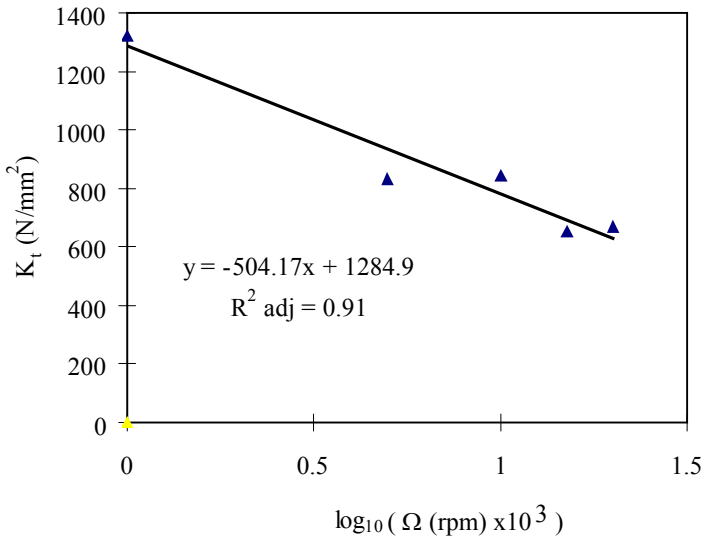

Figure 41. Cutting coefficient in tangential direction (K_t)

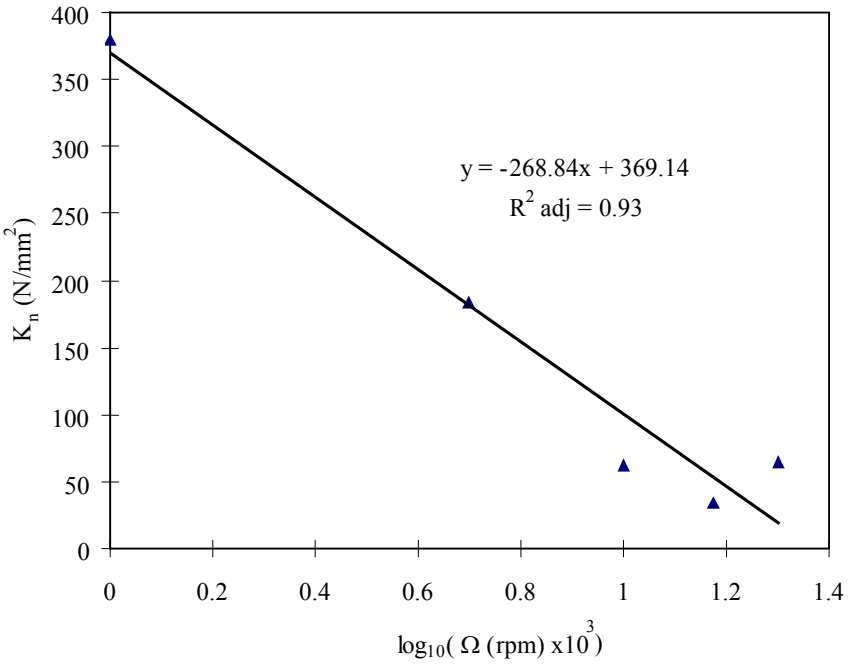

Figure 42. Cutting coefficient in normal direction (K_n)

A similar set of measurements were made using partial radial immersion (up milling) for a 15000 rpm spindle speed. Equation (5.11) was used to find the cutting coefficients in this case. The results are provided in Table 9.

Table 9. Up milling cutting coefficients for 12% radial immersion

K_t(N/mm^2)	K_n(N/mm^2)	K_{te}(N/mm)	K_{ne}(N/mm)
833	431	6	8

To verify the fit, the cutting coefficients obtained were used in a time-domain simulation of the cutting forces. The measured forces were then overlaid on the simulated forces. Figure 43 shows a case for 0.12 mm/tooth chip load and 1000 rpm. Also Figure 44 and Figure 45 show the fit for higher spindle speeds of 5000 and 20000 rpm, respectively.

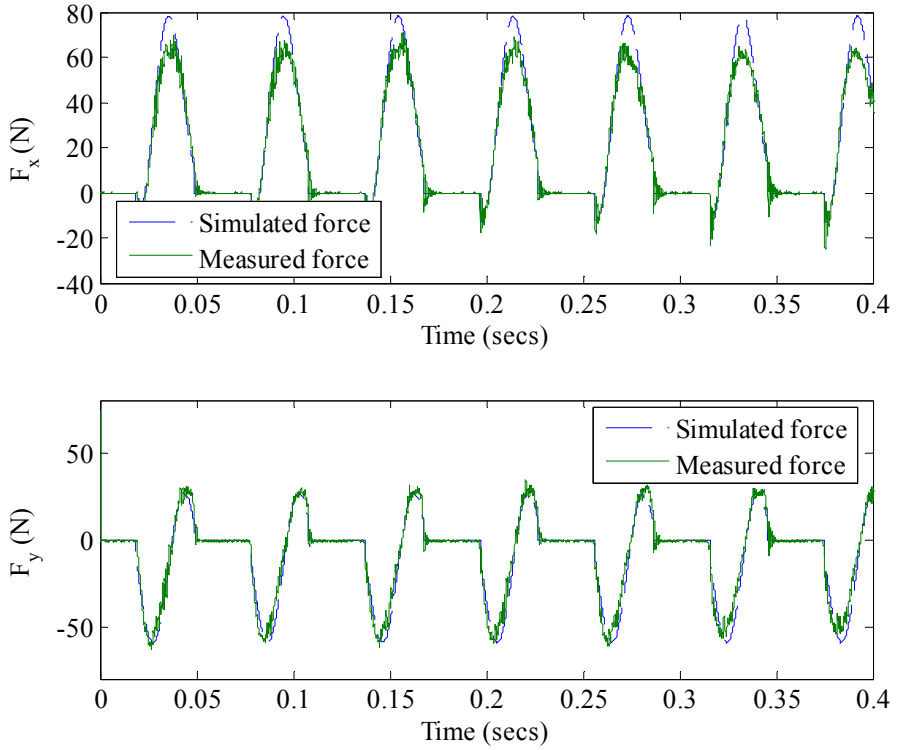

Figure 43. Simulated and measured forces for 0.12 mm/tooth chip load and 1000 rpm.

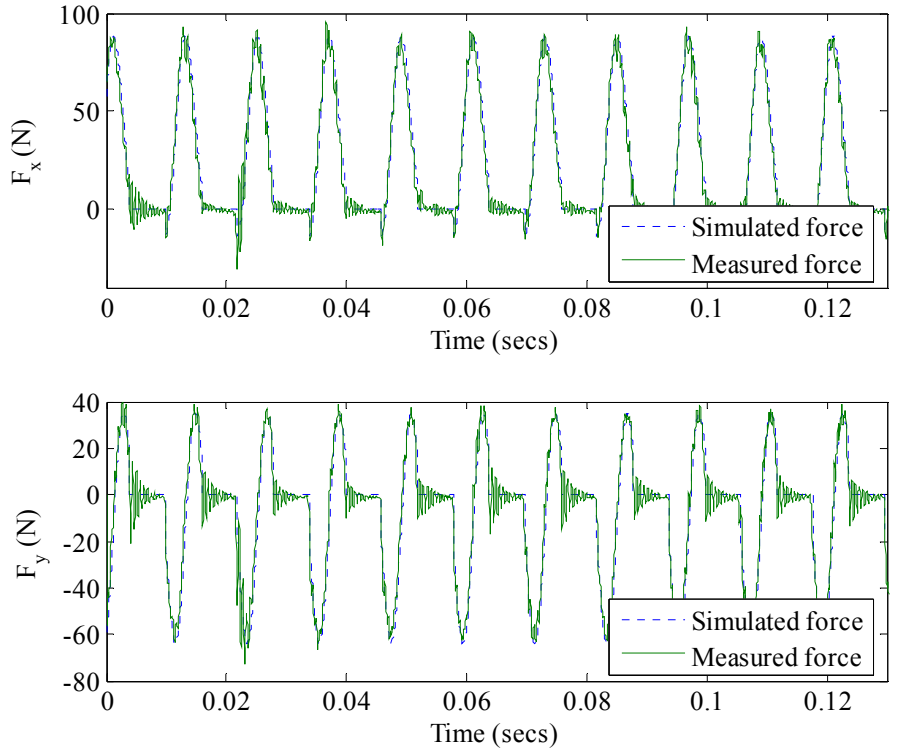

Figure 44. Simulated and measured cutting forces for 0.2mm/tooth chip load, *b*=0.4 mm and 5000 rpm.

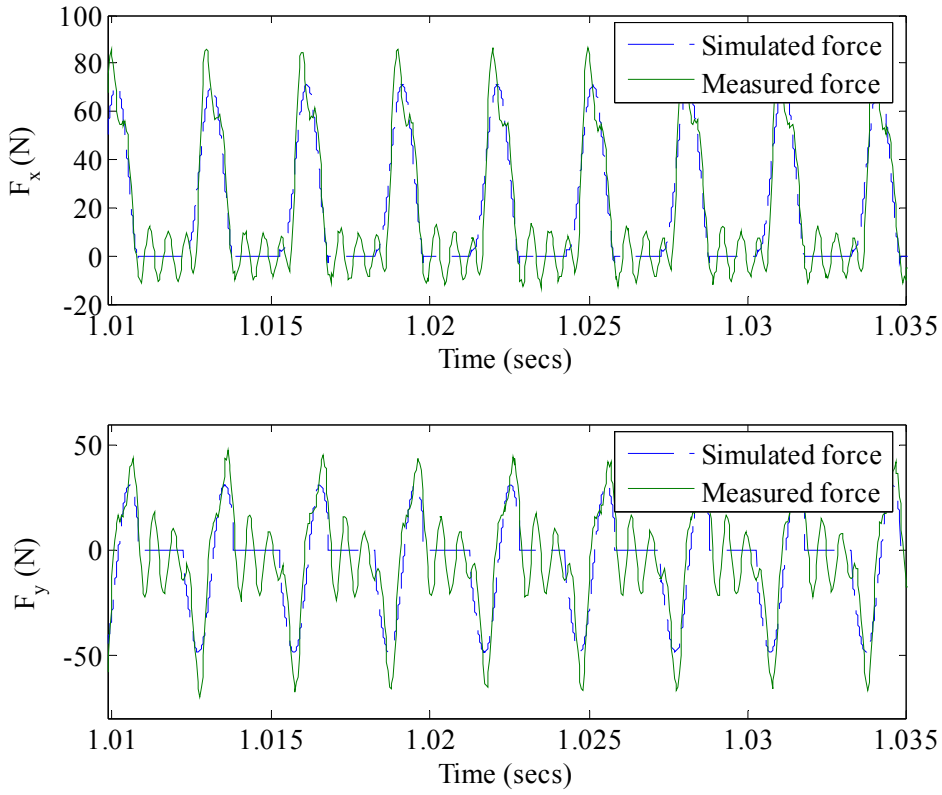

Figure 45. Simulated and measured forces at 20 krpm and b=0.4 mm for slotting.

Covariance Matrix (Linear Multi-Response Model)

The regression analysis performed in the previous section is a single response analysis. However, the measured responses are the forces in both the x and y-directions during a single measurement (dynamometer). Obviously this is a multi-response measurement. Therefore analysis of the data should take into consideration the multivariate nature of the data. The interrelationship existing between the variables could render univariate investigations meaningless. The development for a multi-response model follows the description in [94]. If we let Q be the number of experimental runs and r be the number of response variables measured for each setting (two in our case, i.e., F_x and F_y) of a group of variables (chip load only in our case). The i^{th} response model can be written in vector form as

$$Y_i = Z_i \beta_i + \varepsilon_i \qquad i = 1, 2, ..., r \tag{5.12}$$

where Y_i is an $Q \times 1$ vector of observations in the i^{th} response, Z_i is an $Q \times p_i$ matrix of rank p_i (for the simple linear model $p_i = 2$), $\boldsymbol{\beta}_i$ is a $p_i \times 1$ vector of unknown constant parameters, and ε_i is an $Q \times 1$ random error vector associated with i^{th} response. The assumption of simple linear regression apply here, that is $E(\varepsilon_i) = 0$ and $Var(\boldsymbol{\varepsilon}_i) = \boldsymbol{\sigma}_{ii} \boldsymbol{I}_Q$. However, the covariance matrix between the responses is not zero,

$$\begin{aligned} &Var(\boldsymbol{\varepsilon}_i) = \boldsymbol{\sigma}_{ii} \boldsymbol{I}_Q && i = 1, 2, ..., r \\ &Cov(\boldsymbol{\varepsilon}_i, \boldsymbol{\varepsilon}_j) = \boldsymbol{\sigma}_{ij} \boldsymbol{I}_Q && i, j = 1, 2, .., r; i \neq j \end{aligned} \tag{5.13}$$

We denote the $r \times r$ covariance matrix whose *(i,j)* th element is $\sigma_{ij}\ (i, j = 1, 2, ..., r)$ by Σ. For the case of two responses, Eq. (5.12) can be written in matrix form as:

$$\begin{bmatrix} \underset{Q\times 1}{Y_1} \\ \underset{Q\times 1}{Y_2} \end{bmatrix} = \begin{bmatrix} \underset{Q\times 2}{Z_1} & \mathbf{0} \\ \mathbf{0} & \underset{Q\times 2}{Z_2} \end{bmatrix} \begin{bmatrix} \beta_{01} \\ \beta_{11} \\ \beta_{02} \\ \beta_{12} \end{bmatrix} + \begin{bmatrix} \underset{Q\times 1}{\varepsilon_1} \\ \underset{Q\times 1}{\varepsilon_2} \end{bmatrix} \tag{5.14}$$

where

$$Z_1 = Z_2 = \begin{bmatrix} \underset{Q\times 1}{1} & \underset{Q\times 1}{c} \end{bmatrix} \tag{5.15}$$

where *c* represents the chip load vector (see Eq. (5.3)) and the left hand side vector of Eq. (5.14) represents the observed average cutting forces in the *x* and *y*-directions. From Eq. (5.13) it can be seen that $\boldsymbol{\varepsilon}$ has the following variance-covariance matrix,

$$\Delta = Var(\boldsymbol{\varepsilon}) = \boldsymbol{\Sigma} \otimes \boldsymbol{I}_{\boldsymbol{Q}} \tag{5.16}$$

where $\otimes$ is a symbol for the direct (or Kronecker) product of matrices. The direct product of two matrices $\boldsymbol{\Sigma}$ and $\boldsymbol{I}_{\boldsymbol{Q}}$ both of size $r \times r$ gives an $r^2 \times r^2$ matrix which is

partitioned as $\sigma_{ij}\boldsymbol{I}_{\boldsymbol{Q}}$ where σ_{ij} is the *(i,j)^{the}* element of matrix $\boldsymbol{\Sigma}$. The best linear unbiased estimate of $\boldsymbol{\beta}$ is given by [95]

$$\hat{\boldsymbol{\beta}} = \left(\boldsymbol{Z}'\Delta^{-1}\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}'\Delta^{-1}\tilde{\mathbf{Y}} \tag{5.17}$$

where $\tilde{Y}$ is the left hand side of Eq. (5.14). The variance-covariance matrix of the estimated vector $\hat{\boldsymbol{\beta}}$ is

$$Var\left(\hat{\boldsymbol{\beta}}\right) = \left(\boldsymbol{Z}'\Delta^{-1}\boldsymbol{Z}\right)^{-1} \tag{5.18}$$

Since $\boldsymbol{\Sigma}$ is usually unknown, it is estimated using the following equation [95]

$$\hat{\sigma}_{ij} = \frac{Y_i'\left[\boldsymbol{I}_N - \boldsymbol{Z}_i\left(\boldsymbol{Z}_i'\boldsymbol{Z}_i\right)^{-1}\boldsymbol{Z}_i'\right]\left[\boldsymbol{I}_N - \boldsymbol{Z}_j\left(\boldsymbol{Z}_j'\boldsymbol{Z}_j\right)^{-1}\boldsymbol{Z}_j'\right]Y_j}{Q} \tag{5.19}$$

$$i, j = 1, 2, ..., r$$

It should be noted that $\hat{\sigma}_{ij}$ is computed from the residual vectors which result from ordinary least-squares fit of the i^{th} and j^{th} single response models to their respective data sets. Using this estimate for $\boldsymbol{\Sigma}$ in Eq.(5.19), an estimate of the variance of $\hat{\boldsymbol{\beta}}$ can be obtained. The cutting force coefficients are determined using a linear transformation

$$\left[K\right] = \left[A\right]\left[\hat{\boldsymbol{\beta}}\right] \tag{5.20}$$

where the matrix A for slotting (see Eq. (5.4)) is

$$A = \begin{bmatrix} -\frac{\pi}{Nb} & 0 & 0 & 0 \\ 0 & -\frac{4}{Nb} & 0 & 0 \\ 0 & 0 & \frac{\pi}{Nb} & 0 \\ 0 & 0 & 0 & \frac{4}{Nb} \end{bmatrix} \tag{5.21}$$

Therefore the variance-covariance matrix of cutting force coefficients can be found as

$$Var\left(\{K\}\right) = [A]' Var\left(\hat{\boldsymbol{\beta}}\right)[A]. \tag{5.22}$$

Using the procedure outlined above, the cutting force coefficients and their corresponding correlation matrices are calculated and listed in Table 10 for cutting tests carried out according to the same procedure described earlier, noting that the correlation matrix is obtained directly from the covariance matrix (see Eq. (4.5)) As indicated in Table 10, a high correlation between K_t and K_{te} , and K_n and K_{ne} is found. This high correlation is justified since both of the corresponding cutting coefficients (K_t and K_{te} or K_n and K_{ne}) are obtained from the same regression fit and cutting force direction. However, a small correlation between the x and y-directions of the forces is found (between K_t and K_n or K_{ne}) which may be due to experimental error.

Table 10. Estimated cutting force coefficients and their correlation matrix for 7475 aluminum and a 12.7 mm diameter solid carbide endmill with 4 teeth and 30 degree helix angle.

	K_t (N/m^2)	K_n (N/m^2)	K_{te} (N/m)	K_{ne} (N/m)
Mean	8.41E+08	2.53E+08	1.27E+04	1.01E+04
Standard deviation	2.19E+07	2.66E+07	1.70E+03	2.07E+03
Coefficient of variation	0.03	0.11	0.13	0.20
P - Value	2.E-08	8.E-05	3.E-04	3.E-03
Correlation Coeff. Matrix	**Kne**	**Kn**	**Kte**	**Kt**
Kne	1.00			
Kn	-0.93	1.00		
Kte	-0.13	0.12	1.00	
Kt	0.12	-0.13	-0.93	1.00

Compliant Tool Modal Parameters

The cutting tests were conducted on a Makino V55 vertical milling machining center located at Techsolve, an Ohio-based not-for-profit manufacturing research organization. The cutting tool was a 12.7 mm diameter solid carbide end mill (CRHEC500S4R30-KC610M) with 100 mm overall length (*OAL*), 70 mm over-hang length, 30° helix angle, and 4 flutes. A relatively long tool over-hang length was used in order to obtain a compliant tool that could reasonably be modeled as single degree of freedom. Four measurements of the frequency response function of the tool (Figure 46) were made after running the spindle for 30 seconds at a specific spindle speed then completing a tap test in the *x*-direction, then running the spindle for another 30 seconds and taking a tap test in the *y*-direction, then removing the holder from the machine, and replacing and repeating the above procedure for a different speed. This measurement procedure enabled the estimation of the variation of the modal parameters due to the spindle thermal effect and holder replacement effect. Figure 47 and Figure 48 show the frequency response measurement of the tool in the *x* and *y*-directions respectively. Table 11 lists the fitted tool modal parameters obtained by the peak amplitude method (see Milling Model section in Chapter 4) with their average and standard deviation, and Table 12 lists the correlation coefficient matrix. The correlation coefficient between M_x and K_x, for example, is calculated according to

$$r_{M_x K_x} = \frac{\sum_{i=1}^{4}\left(M_{x_i} - \bar{M}_x\right)\left(K_{x_i} - \bar{K}_x\right)}{\sqrt{\sum_{i=1}^{4}\left(M_{x_i} - \bar{M}_x\right)^2 \sum_{i=1}^{4}\left(K_{x_i} - \bar{K}_x\right)^2}} \tag{5.23}$$

The values shown in Table 12 indicate strong correlation coefficients for $r_{M_xK_x}$ and $r_{M_yK_y}$. This is expected since the natural frequency of the tool is constant so that K_x or K_y depend entirely on M_x or M_y, respectively. Also, since the tool is symmetric the correlations for $r_{M_yM_x}$ and $r_{K_yK_x}$ are also expected to be high, which is the case for $r_{M_yM_x} = 0.86$. As for the correlation coefficient, $r_{C_yC_x} = 0.75$, some correlation is expected since the tool-holder interface damping is ideally symmetric for the round tool. The minimal correlation indicated between the damping and other modal parameters can be justified since there is no direct relationship between damping and mass or stiffness.

The mean, standard deviation and the correlation matrix are used to generate the random sample of input modal parameters in order to estimate uncertainty.

Table 11. Tool modal parameters in *x* and *y*-directions.

	x			***y***		
measurement state	**M (kg)**	**C (N.s/m)**	**K (N/m)**	**M (kg)**	**C (N.s/m)**	**K (N/m)**
static	0.03	24.34	4.8E+06	0.03	29.09	4.3E+06
5 krpm and replacement	0.03	22.05	4.4E+06	0.02	37.25	2.6E+06
10 krpm and replacement	0.03	22.66	4.3E+06	0.02	29.54	2.9E+06
20 krpm and replacement	0.02	24.18	3.9E+06	0.02	29.85	3.4E+06
mean	0.03	23.31	4.4E+06	0.02	31.43	3.3E+06
standard deviation	0.002	0.976	3.16E+05	0.004	3.368	6.60E+05
coefficient of variation (CV)	0.07	0.04	0.07	0.20	0.11	0.20

Table 12. Correlation coefficient matrix for modal parameters.

Correlation Coefficient	**Mx**	**Cx**	**Kx**	**My**	**Cy**	**Ky**
Mx	1.00					
Cx	0.23	1.00				
Kx	0.99	0.13	1.00			
My	0.86	0.69	0.80	1.00		
Cy	-0.09	-0.75	-0.05	-0.50	1.00	
Ky	0.66	0.88	0.58	0.95	-0.64	1.00

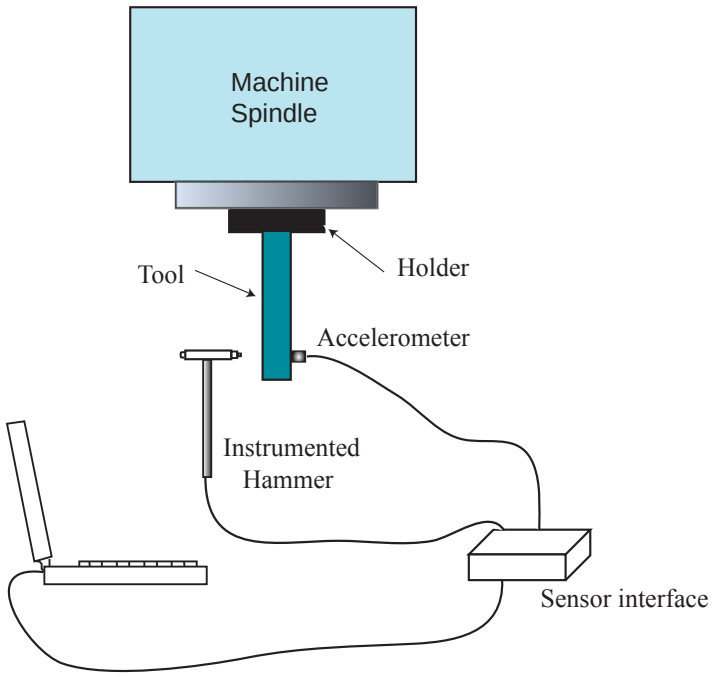

Figure 46. Modal analysis test equipment typically used in machine tool structures.

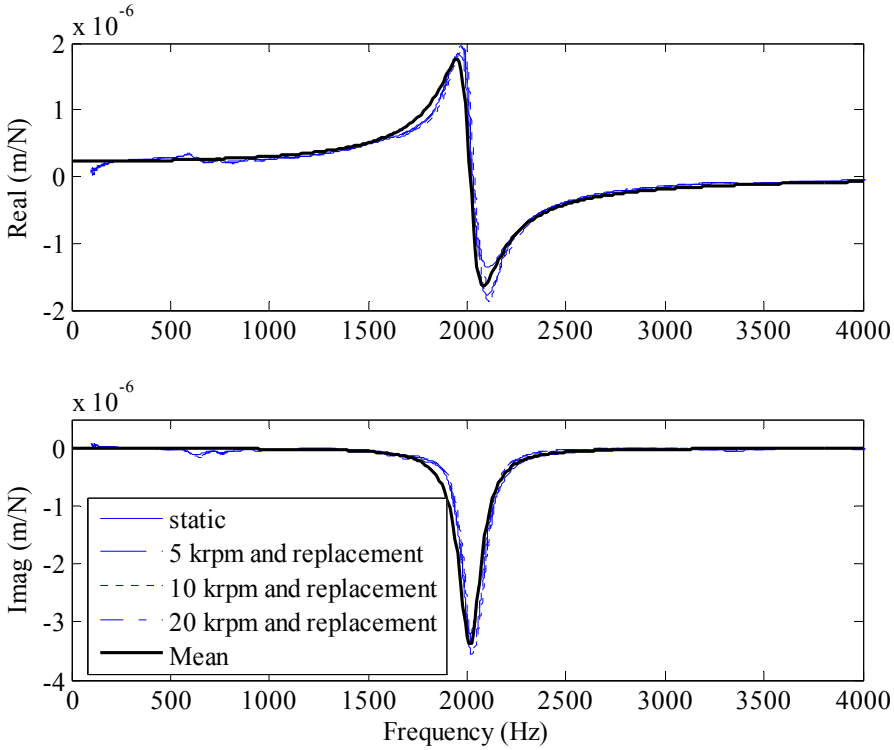

Figure 47. Frequency response function measurement of tool in *x*-direction. Four sets of measurements are made to estimate spindle thermal and holder replacement effects.

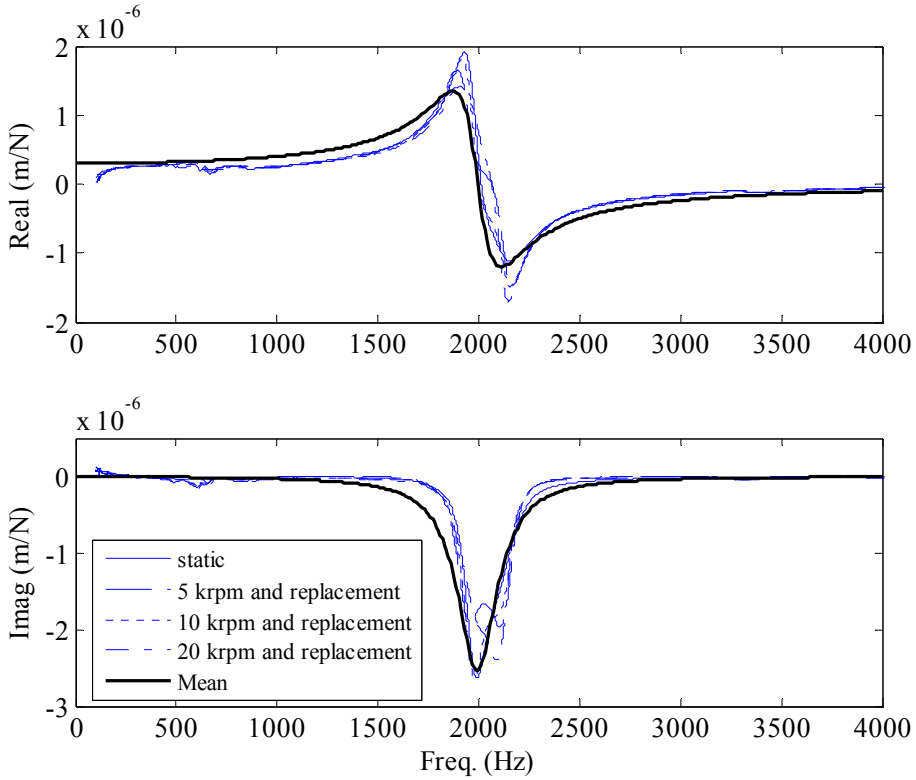

Figure 48. Frequency response function measurement of tool in *y*-direction. Four sets of measurements are made to estimate spindle thermal and holder replacement effects.

Stability Lobe Validation

In this section the stability lobe diagram for the 70 mm over-hang length compliant tool is verified. First the stability limit uncertainty is predicted using Latin Hyper-Cube and Monte Carlo sampling, then a description of the experimental procedure is provided and results are discussed.

Stability Lobe Uncertainty

Due to the relatively high uncertainty in input parameters (Table 11), the sensitivity method cannot be used because of the nonlinearity of the axial depth limit to the respective parameters. The alternative random sampling methods (Latin Hyper-Cube and Monte Carlo) are used instead. The stability lobes are generated using TFEA. A random sample of size *L=1000* is generated from the normal distribution for each input parameter

group (cutting force coefficients and modal parameters) using their corresponding standard deviation, mean values and covariance matrix. Latin Hyper-Cube sampling is used to generate the samples for the cutting force coefficients and modal parameters groups. Also, a random sample of the same size is generated for the radial immersion and spindle speed using Monte Carlo Simulation with no correlation assumed. This random sample of input parameters is used to generate the stability lobe diagram uncertainty intervals. Figure 49 shows the boxplot (a plot used to show variation and measures of central tendency for a sample) of axial depth as a function of spindle speed. The grey boxes indicate the range of minimum and maximum axial depths, the black boxes indicate the lower 2.5 percentile and upper 97.5 percentile (95 % confidence interval), while the two lines indicate the median and mean of the sample. It is seen that the median and mean lines do not match, which indicates the distribution is skewed. Examination of the histogram at selected spindle speeds (see Figure 50) validates this conclusion. At 10000 rpm the distribution appears close to normal, however, in checking the normality of the distribution at this speed (see Figure 51) we find that it is in fact not normal with a P-value of less than 0.005. This is illustrated by the deviation of the observations from the normal probability line.

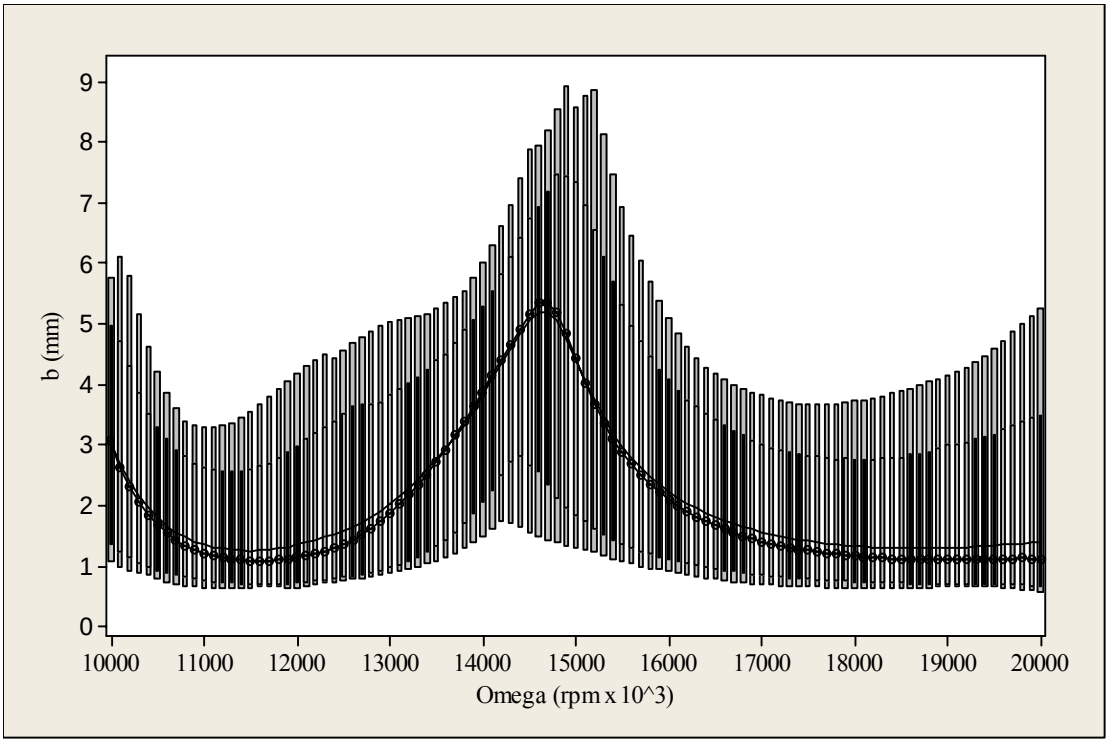

Figure 49. Boxplot of stability lobes boundary uncertainty. The minimum and maximum values are shown for each spindle speed (grey boxes), the mean and median of axial depth limit are indicated by the line and circles respectively, also shown is the 2.5 and 97.5 percentiles (black boxes).

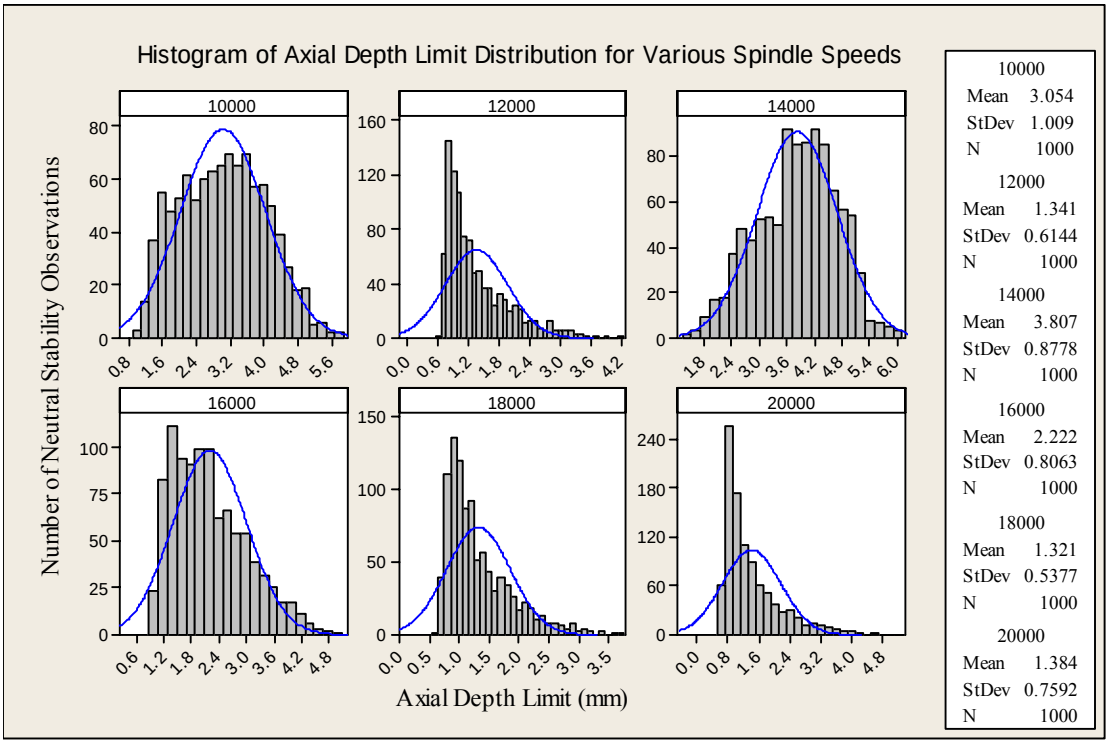

Figure 50. Histograms of axial depth limit distributions for various spindle speeds.

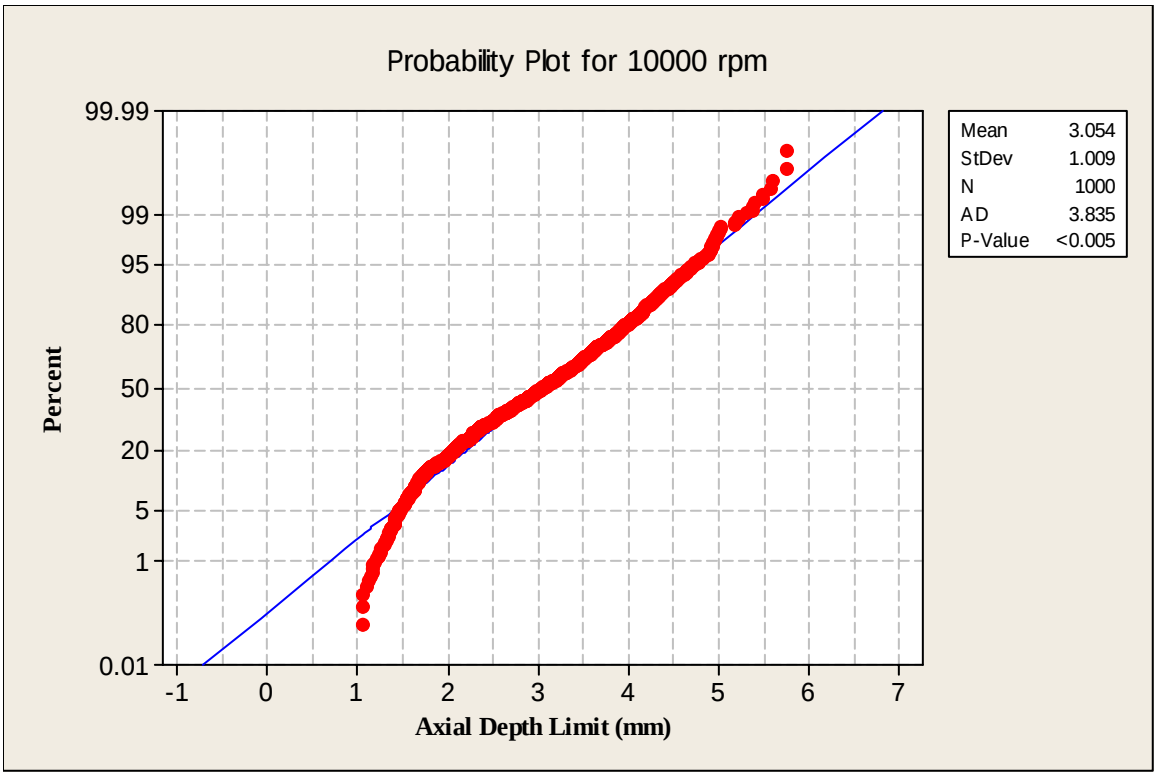

Figure 51. Probability plot of axial depth limit distribution at 10000 rpm spindle speed.

Experimental Procedure

The stability lobes were verified experimentally for 25% partial radial immersion down milling and 0.1 mm/tooth chip load. The same tool with modal tool parameters listed in Table 11 was used. A 7475 aluminum work-piece was mounted (see Figure 52) to a Makino V55 vertical machining center table. Cutting tests with different axial depths were conducted at a range of spindle speeds from 10000 rpm to 20000 rpm in 1000 rpm steps. The stability of each cutting operation was determined by recording the sound signal of the cut. The Fast Fourier Transform was used to transform the sound time-domain sound signal into the frequency domain. An analysis of the signal frequencies identified the chatter frequency, if one existed (i.e., significant content was seen at frequencies other than the runout and tooth passing frequencies). It was observed that the

chatter frequency when it existed was always slightly higher than the tool natural frequency, as expected.

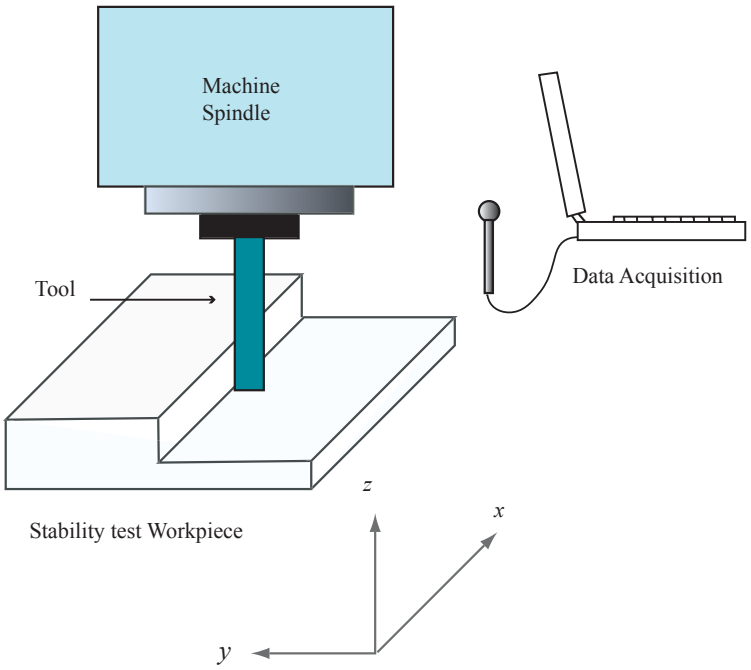

Figure 52. Schematic of stability tests for partial radial immersion cutting conditions.

Results

The cutting test conditions are shown in Figure 53 with the boxplot of axial depth limit uncertainty. Also, the stability lobe boundary is overlaid using mean values of input parameters. In order to identify the stability of each cut, as noted previously, the sound signal was analyzed. In Figure 54 we can see some of these signals in the frequency domain. As noted previously, the chatter frequency occurs near the natural frequency of the tool. The natural frequency of the tool is approximately 2000 Hz

($f_n = \frac{1}{2\pi}\sqrt{K_{y_{mean}} / M_{y_{mean}}}$, see Table 11). For 13000 rpm and a 1.52 mm axial depth, and 14000 rpm and 3.05 mm, for example, the chatter frequencies occur near 2100 and 2200 Hz, respectively. It should be noted that the chatter frequencies were difficult to

identify when the tooth passing frequency or one of its harmonics are near the tool natural frequency. This is evident from the cutting test at 10000 rpm and 16000 rpm where there is high amplitude near the tool natural frequency. In that case, examinations of the cut surface of the workpiece help in identifying chatter (due to the corresponding rough surface finish).

In Figure 53, the stability of the cutting conditions agreed well with the median of stability prediction almost everywhere along the spindle speed. However, near fractions of the tool natural frequency ($\Omega = 60 f_n / N = 15 f_n = 30000$ rpm), poor agreement between prediction and experimental result is observed. This may be attributed to confidence in the modal fitting near the natural frequency of the tool

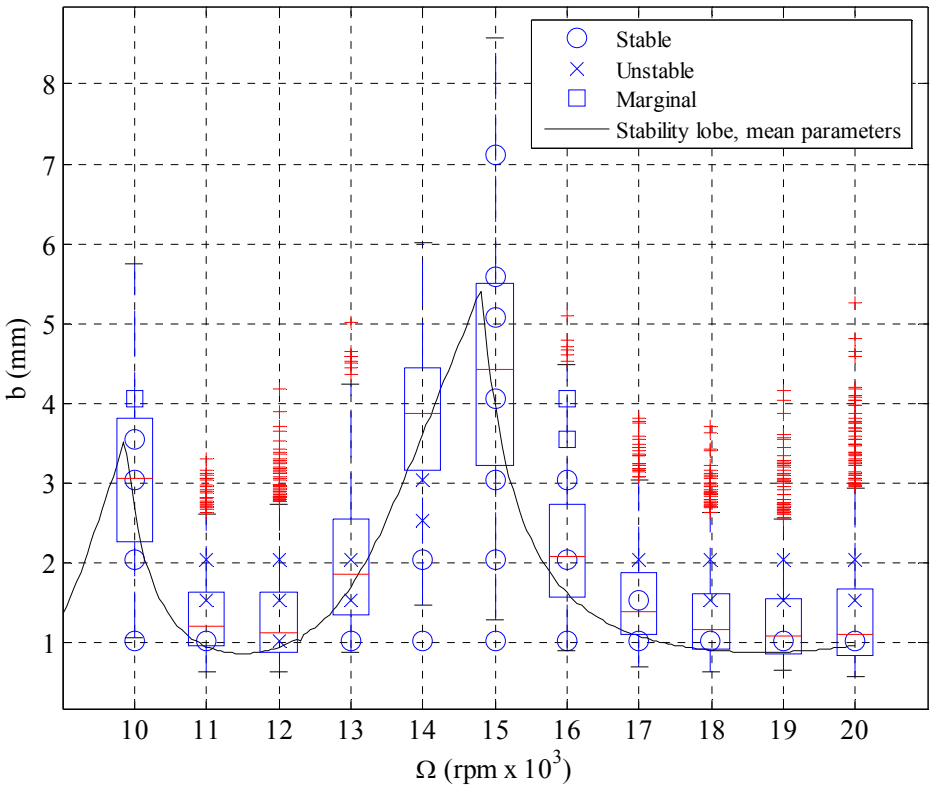

Figure 53. Stability lobe generated using mean values of input parameters with experimental results overlaid, also shown the boxplot corresponding to each spindle speed used in the measurements.

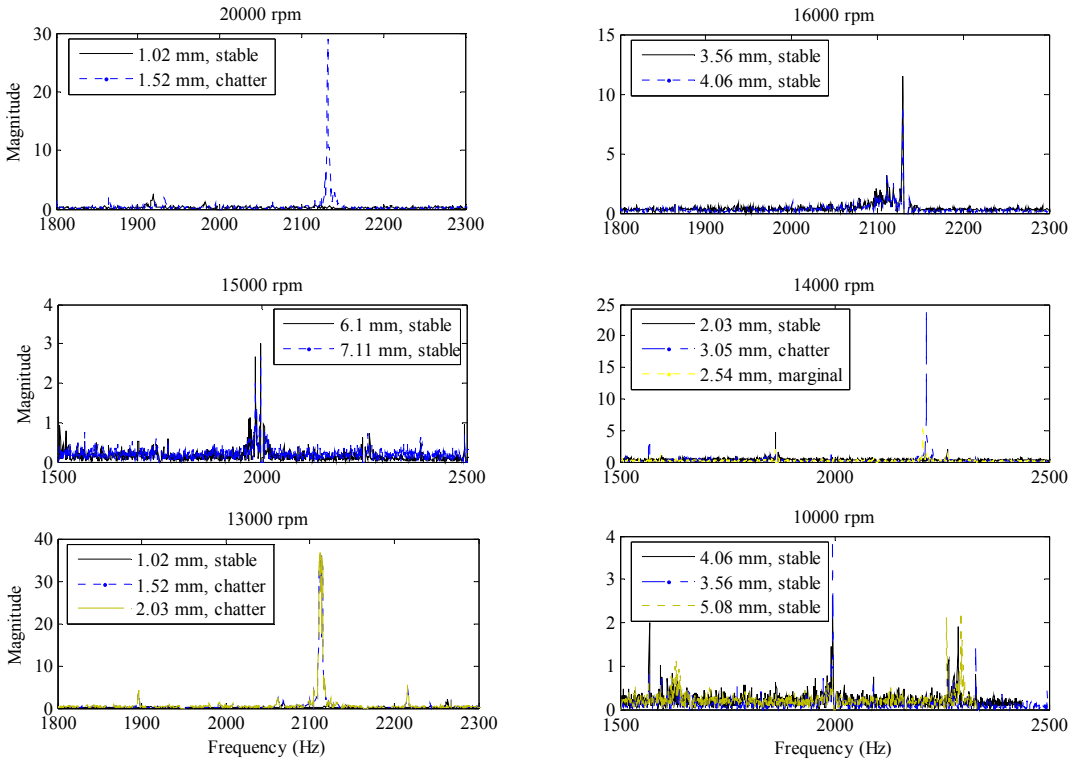

Figure 54. Fast Fourier Transform (FFT) of sound signals for selected stability tests.

Pareto Front Validation

This section begins with the calculation of the Pareto Front for a specific single degree of freedom tool considering confidence levels in the axial depth limit, b_{lim}, (see Robust Optimization section in Chapter 4), after which the experimental procedure of the tests is described, followed by the results.

Pareto Front Simulation Results

The Pareto front for *SLE* and *MRR* is generated for the same material (7475 aluminum) and tool used in the stability tests (Table 11 and Table 12) and the same cutting conditions of radial immersion and chip load. Two cases are considered: 1) no uncertainty in the input parameters; and 2) uncertainty in input parameters or axial depth limit, b_{lim}, where an uncertainty of $U = u_c\left(b_{\text{lim}}\right)$ is used. The robust optimization algorithm (Chapter 4) is used to generate the Pareto front for the input parameters

uncertainty case. Figure 55 and Figure 56 illustrate the Pareto front for the aforementioned cases. It can be seen that the uncertainty designs predict higher *SLE* for the same *MRR* compared to the mean value one. Also, in Figure 56 the knee in both Pareto Fronts indicate the design beyond which the *SLE* increases at a higher rate, which makes that knee a preferred design point. In considering Figure 56, the *SLE* difference between uncertainty and without uncertainty cases is larger at higher *MRR* than at lower *MRR*. This is attributed to the fact that as higher *MRR* is required, the axial depth, *b,* approaches b_{lim}. Here, the predicted uncertainty in b_{lim} changes the design variables (b or Ω) substantially to account for the uncertainty. This penalizes the *SLE* for the uncertainty case and makes it significantly larger than the no uncertainty case at higher *MRR*. However, at lower *MRR* the axial depth b is far from b_{lim} and is therefore less affected by $u_c(b_{lim})$.

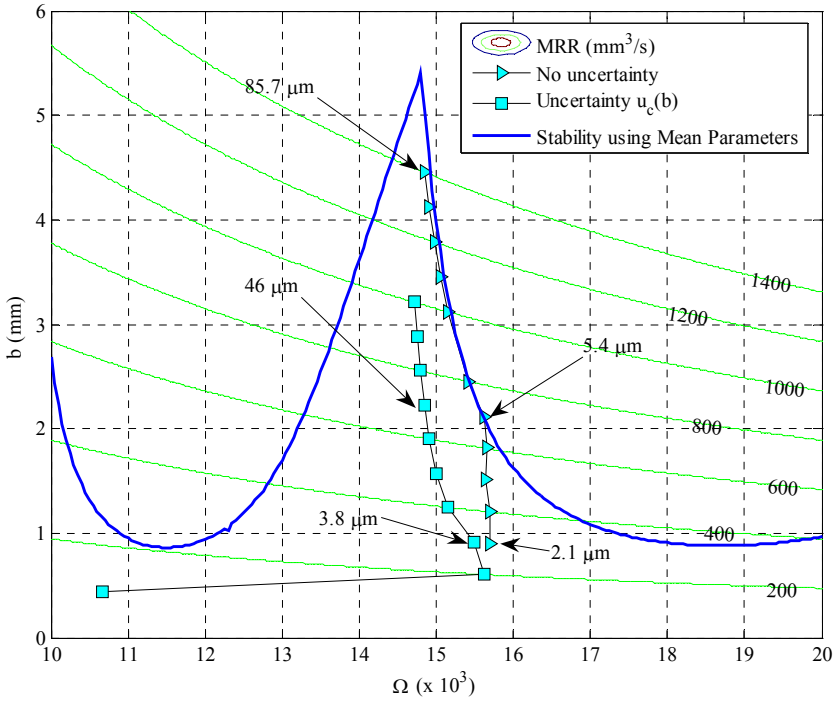

Figure 55. Stability boundary using mean values in the input parameters Pareto optimal designs are overlaid for two cases: mean values and uncertain input parameters.

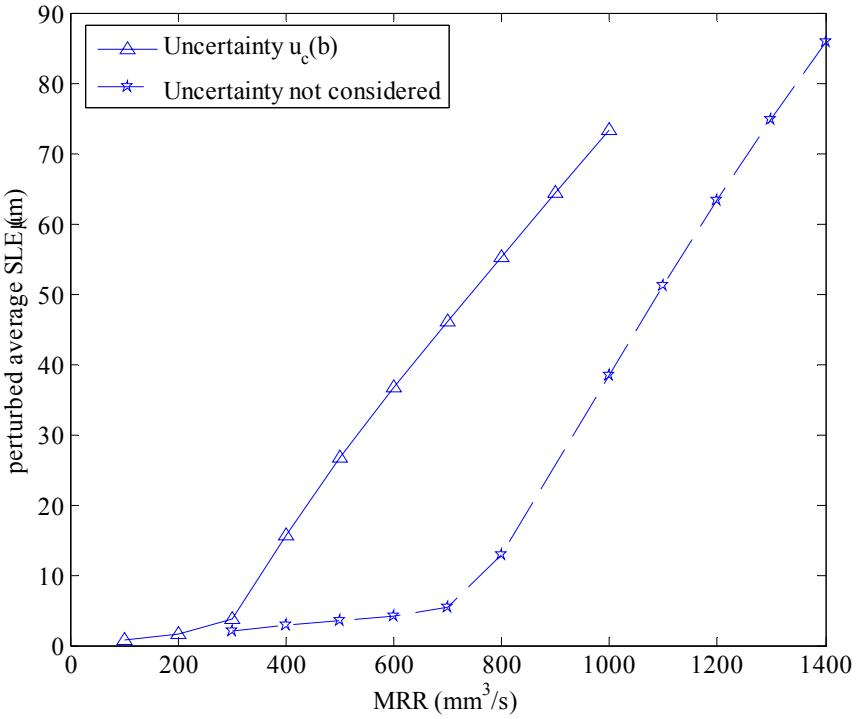

Figure 56. Pareto Front of perturbed average *SLE* and *MRR*. The Pareto Front with uncertainty in axial depth is compared to the one with no uncertainty.

Experimental Procedure and Results

As a first step in conducting the surface location error tests, the work-piece was machined to a specified dimension (nominally 40 mm width) using shallow axial depth slotting cuts (see Figure 57). Careful attention was paid to minimizing positioning errors of the machine by feeding from the same direction prior to cutting (i.e., minimize the influence of reversal errors). The cutting conditions of two mean value Pareto optimal designs were selected from Figure 56 for the case of no uncertainty. The first point corresponds to the knee design point in the figure and the second point corresponds to the maximum *MRR* for the case that uncertainty was not considered. For these two design conditions of axial depth and spindle speed, four additional cuts were made for each case by varying the spindle speed around the selected design (see Table 13). The purpose of these extra cuts was to check the sensitivity of the stability and surface location error to

spindle speed. The work-piece webs, shown in Figure 57, were milled from both sides. A coordinate measuring machine (CMM) was used to measure the base of the web (dimension *a*) and the top portion (dimension *b*). The measured surface location error was then taken to be $SLE = \left(\frac{a-b}{2}\right) - 3.175 \text{ mm}$, where the commanded radial depth was 3.175 mm. Each dimension (*a* and *b*) was measured 15 times in order to evaluate the CMM machine measurement repeatability. The 15 measurements had a maximum standard deviation of $2\ \mu m$ (the machine accuracy is estimated at $< 5\ \mu m$ based on recent calibration tests). The measured *SLE* for the set of cutting tests is shown in Figure 58 and Figure 59. It should be noted here that all cuts were stable. Therefore, all *SLE* results are shown in the figures. The error of the reference dimension (*a=40 mm)* is also shown in these figures. This would identify if there is a trend in the measured *SLE* due to the errors in the reference dimension. The standard deviation of the reference dimension is $4\ \mu m$ and $8\ \mu m$ for 4.45 mm and 2.12 mm axial depth cuts, respectively.

To illustrate the effect of the helix angle of the tool (30°) on the *SLE*, the CMM measurement was repeated for distances of {1, 2, 3, and 3.4} mm from the top surface of the work-piece web along the tool axis. Figure 61 shows that the *SLE* varies along the axial depth of the cut. This variation corresponds with previous *SLE* studies [40] where similar variation of *SLE* was observed.

Although the measured *SLE* does not agree well with the mean predicted value (Figure 62), there is some agreement in the trend of median of *SLE* (Figure 60) and the measured *SLE* (Figure 59). Also the measured *SLE* is within the uncertainty bounds on *SLE* (Figure 60). The disagreement between the measured and predicted *SLE* can be attributed to: 1) the milling model used in the prediction assumed straight cutter teeth (the

actual tool had a 30° helix angle) which would yield higher *SLE*; 2) the cutting force coefficients used in the prediction were measured for 8900 rpm, while the *SLE* cuts were made for around 15000 rpm. At higher spindle speeds the cutting force coefficients (cutting forces) tend toward lower values. These two factors explain the high prediction of *SLE* for the 1400 mm^3/s case relative to the measured one. However, for 700 mm^3/s case they fail to explain the difference. This may highlight model weaknesses at this level of axial depth (2.12 mm).

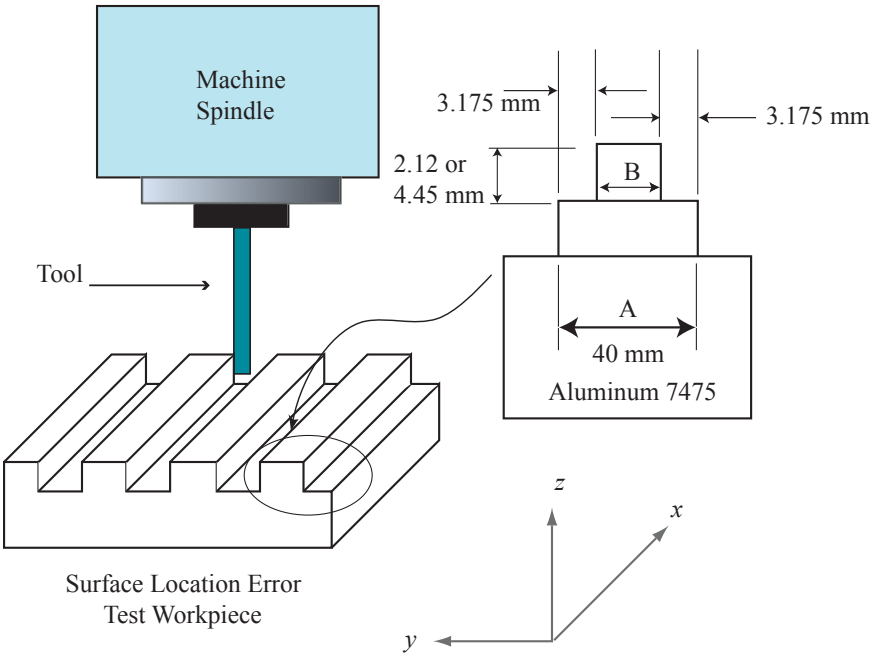

Figure 57. Surface location error experiment schematic.

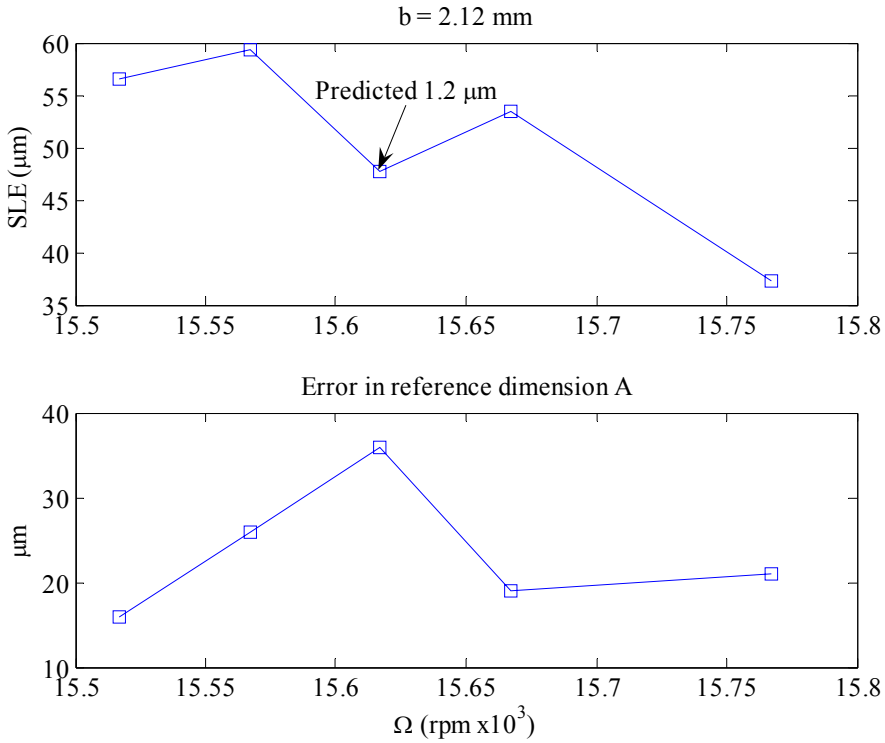

Figure 58. Measured surface location error of b=2.12 mm and the reference dimension (A) error.

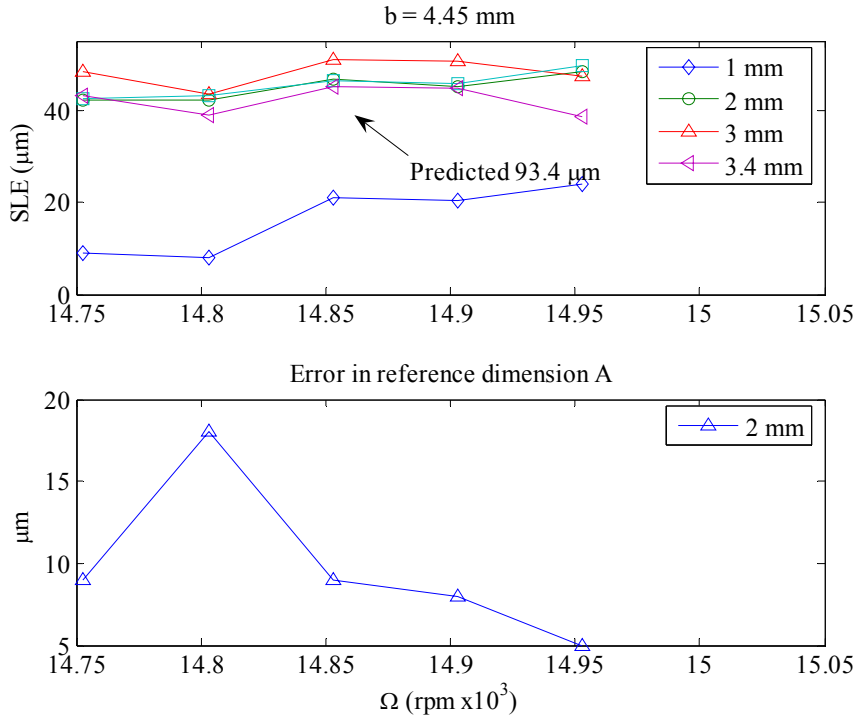

Figure 59. Measured surface location error of b=4.45 mm and the reference dimension (A) error.

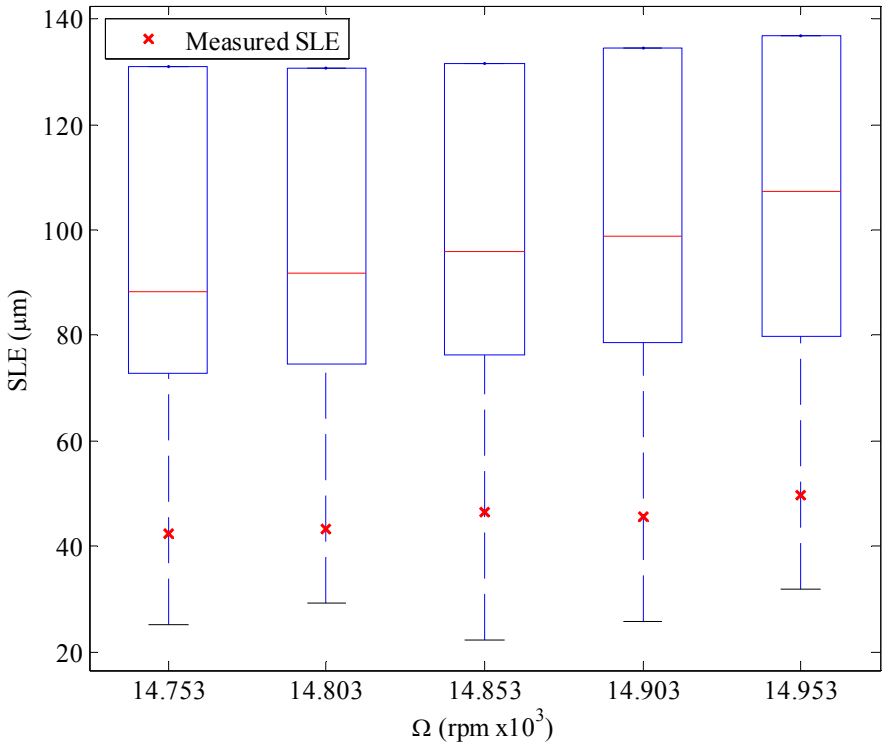

Figure 60. Boxplot of *SLE* uncertainty at spindle speeds for 4.45 mm axial depth case. The upper tail of the *SLE* uncertainty is not present due to the undefined *SLE* in the unstable region.

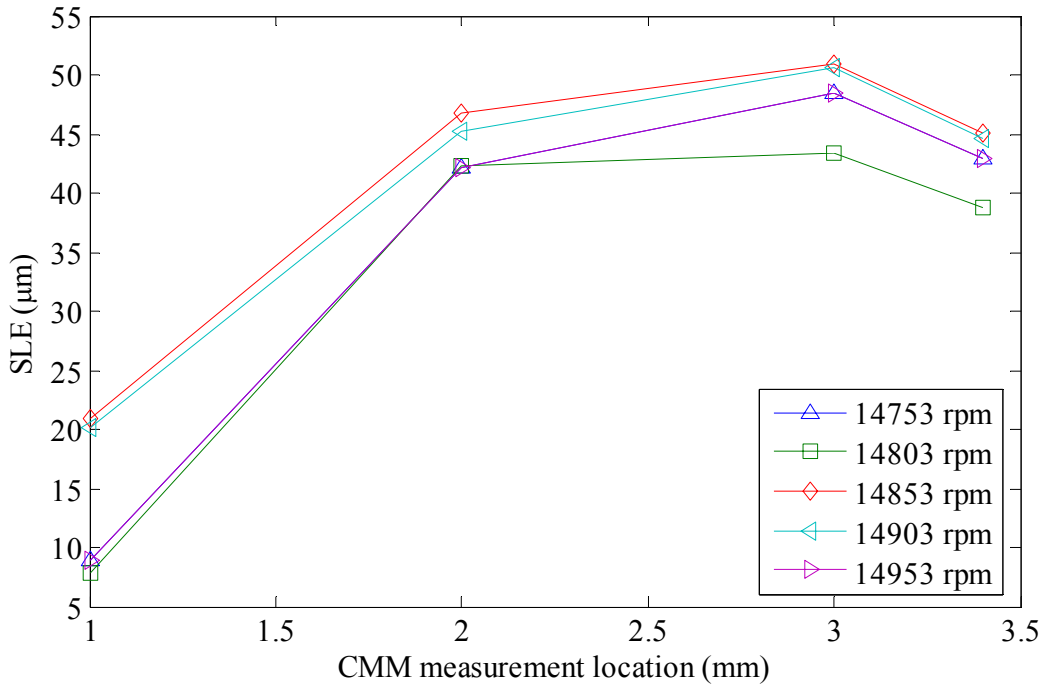

Figure 61. Measured surface location error of b=4.45 mm case. The CMM probe measurement is repeated along the axial depth of the tool with {1, 2, 3, and 3.4} mm from the top surface of the work-piece web.

Table 13. Surface location error cutting conditions for two Pareto optimal designs with no uncertainty considered.

Tool # (100 mm OAL)	CRHEC500S4R30-KC610M			
Cut No.	**b (mm)**	**Ω (rpm)**	**MRR (mm^3/s)**	**SLE (μm)**
1	2.12	15617	700	1.2
2	2.12	15667		
3	2.12	15767		
4	2.12	15567		
5	2.12	15517		
6	4.45	14853	1400	93.4
7	4.45	14803		
8	4.45	14753		
9	4.45	14903		
10	4.45	14953		

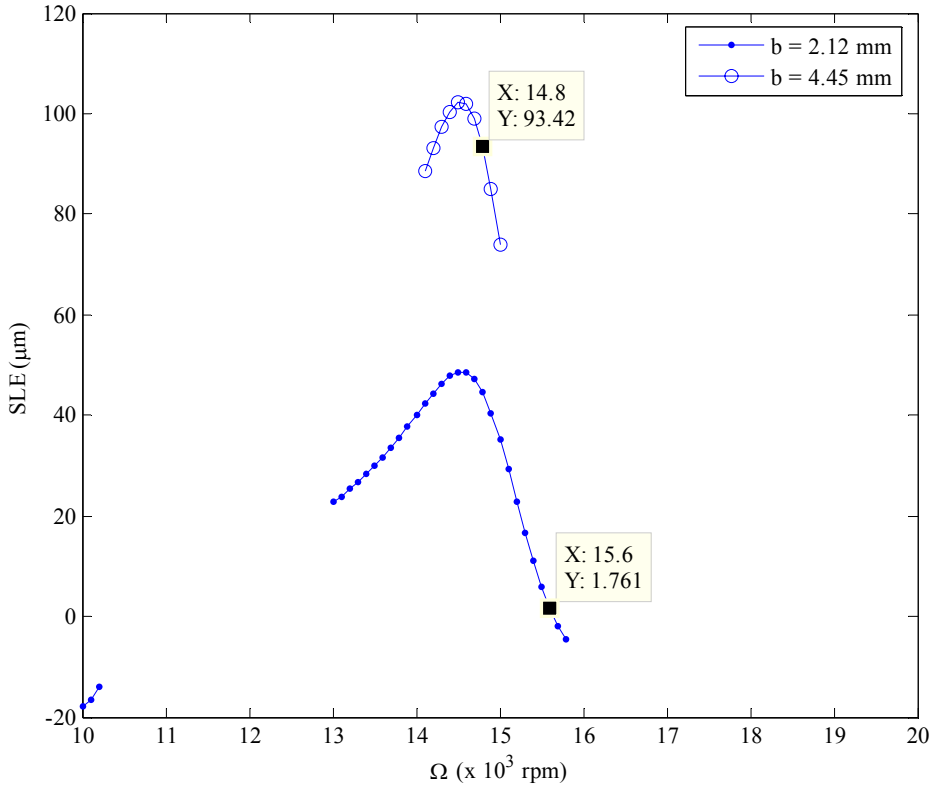

Figure 62. Surface location error of preferred design conditions with no uncertainty considered in the optimization. Optimum spindle speeds are indicated in the figure.

Conclusions

The uncertainty in axial depth limit, b_{lim}, and *SLE* indicates a non-normal distribution, although for convenience a normal distribution was used to estimate the confidence levels of b_{lim} and *SLE*. This non-normality predicates the use of different measures of uncertainty bounds in order to account for a specific confidence interval.

There is good agreement between the prediction of stability and the experimental results. It is shown that there is a distinct 'grey region' of neutral stability boundary (marginal stability) rather than a 'black and white' step change between stable and unstable zones as suggested by the single stability boundary typically indicated in stability lobe diagrams. Also, the uncertainty identified in the b_{lim} boxplot indicates that the distribution is skewed to higher b_{lim} values near the tool system natural frequency. This was also confirmed by the experimental results where higher axial depths were generally feasible.

The measured surface location error of the Pareto design points didn't show high sensitivity to spindle speed variation. This shows the validity of the optimization algorithm selection of a design that mitigates the effect of spindle speed on *SLE*.

In this worst case scenario of uncertainty in modal parameters (thermal effects and dynamic variations due to tool removal and replacement), there was substantial uncertainty seen in b_{lim} and *SLE*. Reduction of uncertainty in the input parameters may be necessary to fully realize the benefits of high-speed milling. This may be done by conducting more tests to lessen the uncertainty bounds and/or completing modal tests of the tool-holder assembly each time it is removed and replaced.

Although no experimental verification of robust optimum designs was done, it is interesting to note that the predicted robust optimum design corresponding to 700 mm^3/s

had approximately the same value of measured *SLE* (Figure 55 and Figure 56). This is due to the fact that the robust design selected a design with spindle speed that was 1000 rpm lower than the measured one (with no uncertainty considered). Also, in considering uncertainty (see Figure 55), the 1400 mm^3/s was not realizable. This may be due to an overestimation of the uncertainty in the input parameters. As mentioned previously, this requires more testing in order to better estimate the uncertainty of input parameters.

CHAPTER 6
SUMMARY

This chapter provides a summary of the work completed in this dissertation with a detailed procedure on how to implement the robust optimization algorithm. Then the limitations of the milling model are addressed with suggestions for future research.

Robust Optimization Algorithm

In order to implement the robust optimization algorithm for a specific compliant tool/work-piece system, the following steps should be taken:

1. Measure the tool/work-piece frequency response and complete a modal fit to the measured response. The confidence levels in the fitted modal parameters are estimated by repeating the measurement at different thermal states of the machining spindle. In case this measurement cannot be repeated each time the tool/holder assembly is removed from the spindle, then several measurements should be made wherein the tool/holder assembly is removed from the spindle and replaced. This will account for the dynamic non-repeatability due to tool/holder replacement. The mean, standard deviation, and correlation between the modal parameters are calculated.

2. Measure the cutting force coefficients for the tool/workpiece material. Chapter 5 gives the procedure used in estimating the mean values of these coefficients and details the regression analysis needed to estimate the mean values, standard deviation, and correlation between these coefficients.

3. The confidence levels in the spindle speed and radial depth can be either estimated from experience or machine manufacturer data.

4. These steps enable estimation of the mean values, confidence levels (standard deviation), and correlation in the input parameters to the milling model. Depending on the confidence levels in these parameters, an uncertainty prediction method can be used to estimate the confidence levels in the stability boundary and *SLE*. Two methods can be used: 1) the sensitivity method; 2) the sampling methods (Monte Carlo and Latin Hyper-Cube). If the coefficient of variation in the input parameters (especially K or M) is larger than 1%, then the sensitivity method cannot be used due to the non-linearity of axial depth limit to these parameters. Chapter 5 gives an

example on how to use the sampling methods to assign confidence levels on the stability boundary and *SLE*.

5. The stability boundary confidence level, *U*, obtained in step 4 is used in the robust optimization algorithm. The algorithm formulation is repeated here

$$\begin{aligned} & min\left[\frac{\left|f_{SLE}\left(b,\Omega+\delta\right)\right|+\left|f_{SLE}\left(b,\Omega\right)\right|+\left|f_{SLE}\left(b,\Omega-\delta\right)\right|}{3}\right], \\ subject\ to: \quad & -f_{MRR}\left(b,\Omega\right)\le e_i, \qquad for\ i\ =\ 1\ldots k \\ & \left\{g_\lambda\left(b+U,\Omega-\delta\right)\cap g_\lambda\left(b+U,\Omega\right)\cap g_\lambda\left(b+U,\Omega+\delta\right)\right\}\le 1, \\ & for\ a\ series\ of\ selected\ limits\ (e)\ on\ f_{MRR}, \end{aligned} \tag{6.1}$$

The spindle speed perturbed *SLE* average is used to account for *SLE* sensitivity to spindle speed. A typical value for the perturbation is $\delta = 50$ rpm to 100 rpm . In order to calculate the trade-off curve between *SLE* and *MRR*, the optimization algorithm is run for a series of limits on the *MRR* objective.

6. The trade-off curve is used to select optimum cutting conditions that match the designer preferences. Typically a knee in the curve would indicate a preferential design where the highest *MRR* can be achieved for a moderate *SLE*.

This completes the description of the selection of robust cutting conditions.

Limitations and Future Research

In this section, the limitations of this research are discussed as well as the recommendations for future research. The limitations and recommendations are as follows:

1. Further efforts should account for the potential variation of the cutting force coefficients as a function of spindle speed. This entails a significant amount of experimental testing.

2. The peak amplitude method used to obtain the fitted modal parameters of the tool/workpiece system does not perform well near the system natural frequency. This makes the model predictions poor at regions where more accuracy is actually needed.

3. A weakness in the solution method (*TFEA*) was observed at shallow axial depths of cut (2 mm in our tests). Further testing at this condition is needed to verify this discrepancy and account for it.

4. The solution method (*TFEA*) assumes straight cutter teeth while most cutters have a helix angle. This makes the model predictions more conservative and can over predict the *SLE*.

APPENDIX A
TIME FINITE ELEMENT ANALYSIS

Mechanical Model

A schematic diagram of two degree of freedom milling process is shown in Figure 1 (repeated here). With the assumption of either a compliant structure or tool, a summation of forces gives the following equation of motion:

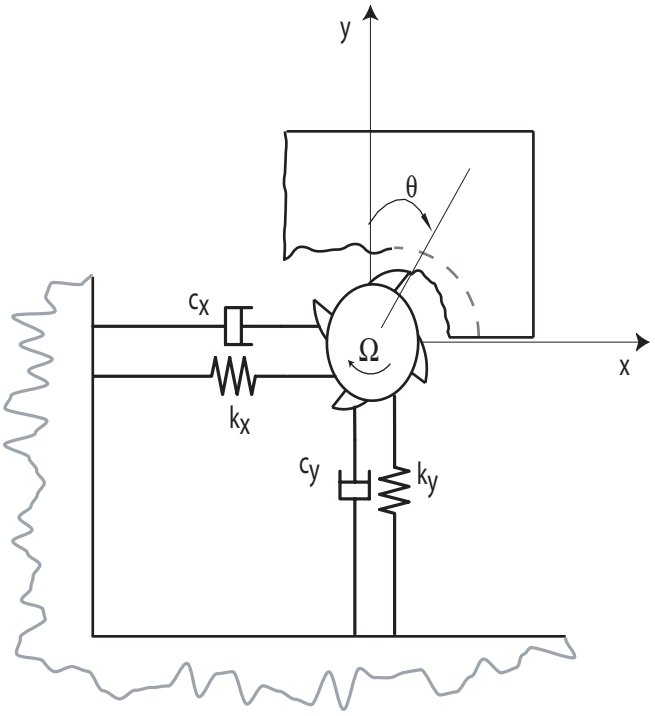

Figure 1. Schematic of 2-DOF milling tool

$$\begin{bmatrix} m_x & 0 \\ 0 & m_y \end{bmatrix} \begin{Bmatrix} \ddot{x}(t) \\ \ddot{y}(t) \end{Bmatrix} + \begin{bmatrix} c_x & 0 \\ 0 & c_y \end{bmatrix} \begin{Bmatrix} \dot{x}(t) \\ \dot{y}(t) \end{Bmatrix} + \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix} \begin{Bmatrix} x(t) \\ y(t) \end{Bmatrix} = \begin{Bmatrix} F_x(t) \\ F_y(t) \end{Bmatrix}, \tag{A.1}$$

where the terms $m_{x,y}$, $c_{x,y}$, $k_{x,y}$ and $F_{x,y}$ are the modal mass, damping, spring stiffness, and cutting forces in the flexible directions of the system. The x and y cutting force components on the p^{th} tooth are given by:

$$\begin{Bmatrix} F_{xp}(t) \\ F_{yp}(t) \end{Bmatrix} = g_p(t) \begin{bmatrix} -\cos\theta_p(t) & -\sin\theta_p(t) \\ \sin\theta_p(t) & -\cos\theta_p(t) \end{bmatrix} \begin{Bmatrix} F_{tp}(t) \\ F_{np}(t) \end{Bmatrix}, \tag{A.2}$$

where $g_p(t)$ acts as a switching function It is equal to one if the p^{th} tooth is active and zero if it is not cutting [54]. The tangential and normal cutting force components, $F_{tp}(t)$ and $F_{np}(t)$, respectively, are considered to be the product of linearized cutting coefficients K_t and K_n, the nominal depth of cut b, and the instantaneous chip thickness $w_p(t)$:

$$\begin{Bmatrix} F_{tp}(t) \\ F_{np}(t) \end{Bmatrix} = b w_p(t) \begin{Bmatrix} K_t \\ K_n \end{Bmatrix} + b \begin{Bmatrix} K_{te} \\ K_{ne} \end{Bmatrix}, \tag{A.3}$$

where $w_p(t)$ depends upon the feed per tooth, h, the cutter rotation angle θ_p, and regeneration in the compliant structure directions:

$$w_p(t) = h\sin\theta_p(t) + \left[x(t) - x(t-\tau)\right]\sin\theta_p(t) + \left[y(t) - y(t-\tau)\right]\cos\theta_p(t). \tag{A.4}$$

Here $\tau = 60/N\Omega$ is the tooth passing period, Ω is the spindle speed given in rpm, h is chip load (used instead of c to differentiate it from $\cos\theta_p$ defined later) and N is the total number of cutting teeth. The angular position of the p^{th} tooth for a cutter with evenly spaced teeth is $\theta_p(t)=(2\Pi\Omega/60)t+p2\Pi/N$.

The total cutting force equations are found by summing the forces on each cutting tooth in Eq. (A.2) and substituting Eqs. (A.3) and (A.4) into Eq. (A.2):

$$\begin{Bmatrix} F_x(t) \\ F_y(t) \end{Bmatrix} = \sum_{p=1}^{N} g_p(t) b \left(h \begin{Bmatrix} -K_t sc - K_n s^2 \\ K_t s^2 - K_n sc \end{Bmatrix} + \begin{Bmatrix} -K_{te} c - K_{ne} s \\ K_{te} s - K_{ne} c \end{Bmatrix} + \begin{bmatrix} -K_t sc - K_n s^2 & -K_t c^2 - K_n sc \\ K_t s^2 - K_n sc & K_t sc - K_n c^2 \end{bmatrix} \begin{Bmatrix} x(t) - x(t-\tau) \\ y(t) - y(t-\tau) \end{Bmatrix} \right), \tag{A.5}$$

where $s = \sin \theta_p(t)$ and $c = \cos \theta_p(t)$. A more compact form for the equation of motion is realized by making the following substitutions:

$$\boldsymbol{K}_c(t) = \sum_{p=1}^{N} g_p(t) \begin{bmatrix} -K_t sc - K_n s^2 & -K_t c^2 - K_n sc \\ K_t s^2 - K_n sc & K_t sc - K_n c^2 \end{bmatrix}, \tag{A.6}$$

$$\vec{f}_O(t) = \sum_{p=1}^{N} g_p(t) \left(h \begin{Bmatrix} -K_t sc - K_n s^2 \\ K_t s^2 - K_n sc \end{Bmatrix} + \begin{Bmatrix} -K_{te} c - K_{ne} s \\ K_{te} s - K_{ne} c \end{Bmatrix} \right). \tag{A.7}$$

Using Eqs. (A.5), (A.6) , and (A.7), Eq.(A.1) can be written as:

$$\boldsymbol{M}\ddot{\vec{X}}(t) + \boldsymbol{C}\dot{\vec{X}}(t) + \boldsymbol{K}\vec{X}(t) = \boldsymbol{K_C}(t) b \left(\vec{X}(t) - \vec{X}(t-\tau) \right) + \vec{f}_O(t) b \ , \tag{A.8}$$

where $\vec{X}(t) = \begin{bmatrix} x(t) & y(t) \end{bmatrix}^T$ is the two-element position vector and $\boldsymbol{M}$, $\boldsymbol{C}$, and $\boldsymbol{K}$ are the $2x2$ mass, damping, and stiffness matrices of Eq.(A.1).

Time Finite Element Analysis (TFEA)

The dynamic behavior of the milling process is governed by Eq.(A.8). Since this equation does not have a closed form solution, an approximate solution is sought to understand the behavior of the system. One such approximation technique used for dynamic systems is TFEA [54]. An approximate discrete linear map is constructed using time finite elements in the cut to exact mapping of free vibration out of the cut, where mapping is performed on displacement and velocity components of vibration [54-57]. The formulation of the dynamic map for the multiple degree of freedom systems closely

follows the discretization procedure outlined in references [54], but has been presented here for completeness.

Free Vibration

When the tool is not in contact with the work-piece, the system is governed by the equation for free vibration. The cutting forces then become zero:

$$\boldsymbol{M}\ddot{\vec{X}}(t)+\boldsymbol{C}\dot{\vec{X}}(t)+\boldsymbol{K}\vec{X}(t)=\boldsymbol{0}, \tag{A.9}$$

and the exact solution for the free vibration can be written with a state transition matrix $\Phi\left(t_f+t_c,t_c\right)$ where t_c is that the time the tool leaves the material and t_f is the duration of free vibration. Exact mapping of displacement and velocity components can be written in terms of state transition matrix as:

$$\begin{Bmatrix}\vec{X}\left(t_f+t_c\right)\\ \dot{\vec{X}}\left(t_f+t_c\right)\end{Bmatrix}=\Phi\left(t_f+t_c,t_c\right)\begin{Bmatrix}\vec{X}\left(t_c\right)\\ \dot{\vec{X}}\left(t_c\right)\end{Bmatrix}\quad . \tag{A.10}$$

Vibration during Cutting

When the tool is in the cut, its motion is governed by a time delayed-differential equation. Since this equation does not have a closed form solution, an approximate solution for the tool displacement is assumed for the j^{th} element of the n^{th} tooth passage as a linear combination of polynomials [54]:

$$\vec{X}(t)=\sum_{i=1}^{4}\vec{a}_{ji}^{n}\phi_i\left(\sigma_j\left(t\right)\right). \tag{A.11}$$

Here $\sigma_j\left(t\right)=t-n\tau-\sum_{k=1}^{j-1}t_k$ is the "local" time within the j^{th} element of the n^{th} period, the length of the k^{th} element is t_k and the trial functions $\Phi_i(\sigma_j(t))$ are cubic Hermite polynomials defined in Eq. (A.12),

$$\begin{aligned}
\phi_1(\sigma_j) &= 1-3\left(\frac{\sigma_j}{t_j}\right)^2+2\left(\frac{\sigma_j}{t_j}\right)^3 . \\
\phi_2(\sigma_j) &= t_j\left[\left(\frac{\sigma_j}{t_j}\right)-2\left(\frac{\sigma_j}{t_j}\right)^2+\left(\frac{\sigma_j}{t_j}\right)^3\right] . \\
\phi_3(\sigma_j) &= 3\left(\frac{\sigma_j}{t_j}\right)^2-2\left(\frac{\sigma_j}{t_j}\right)^3 . \\
\phi_4(\sigma_j) &= t_j\left[-2\left(\frac{\sigma_j}{t_j}\right)^2+\left(\frac{\sigma_j}{t_j}\right)^3\right] .
\end{aligned} \tag{A.12}$$

Substitution of the assumed solution of Eq. (A.11) into the equation of motion (Eq. (A.8)) leads to a non-zero error. The error from the assumed solution is "weighted" by multiplying by a set of test functions and setting the integral of the weighted error to zero. Two test functions are chosen to be a constant $\psi_1(\sigma_j)=1$ and $\psi_2(\sigma_j)= \sigma_j/t_j-1/2$ (linear). The integral is taken over the time for each element, $t_j=t_c/E$, thereby dividing the time in the cut t_c into E elements. The resulting two equations are

$$\int_0^{t_j}\left[\begin{array}{l}
\boldsymbol{M}\left(\sum_{i=1}^{4} a_{ji}^{n}\ddot{\phi}_i(\sigma_j)\psi_p(\sigma_j)\right)+\boldsymbol{C}\left(\sum_{i=1}^{4} a_{ji}^{n}\dot{\phi}_i(\sigma_j)\psi_p(\sigma_j)\right)+ \\
\left(\boldsymbol{K}-b\boldsymbol{K}_C(\sigma_j)\right)\left(\sum_{i=1}^{4} a_{ji}^{n}\phi_i(\sigma_j)\psi_p(\sigma_j)\right)+ \\
b\boldsymbol{K}_C(\sigma_j)\left(\sum_{i=1}^{4} a_{ji}^{n-1}\phi_i(\sigma_j)\psi_p(\sigma_j)\right)- \\
b\vec{f}_O(\sigma_j)\psi_p(\sigma_j)
\end{array}\right]_{p=1,2} d\sigma_j = 0 , \tag{A.13}$$

where $\boldsymbol{K}_c(\sigma_j)$ and $\vec{f}_o\left(\sigma_j\right)$ have been used in place of previously defined $\boldsymbol{K}_c(t)$ and $\vec{f}_o\left(t\right)$ to explicitly show the dependence on local time.

In Eq. **(A.13)**, the index *j* refers to the number of elements and the index *i* refers to the corresponding Hermite polynomial. The displacement and velocity at tool entry into the cut are specified by the coefficients of the first two basis functions on the first element: $\bar{a}_{11}^{n}$ and $\bar{a}_{12}^{n}$. The relationship between the initial and final conditions during free vibration can be mapped through the state transition matrix as

$$\begin{Bmatrix} a_{11} \\ a_{12} \end{Bmatrix}^{n} = \Phi \begin{Bmatrix} a_{E3} \\ a_{E4} \end{Bmatrix}^{n-1}, \tag{A.14}$$

where *E* is the total number of finite elements in the cut and the last term in Eq. (A.14) is the displacement and velocity of the element as it leaves the cut. For the remainder of the elements in the cut, a continuity constraint is imposed to set the position and velocity at the end of one element ($\bar{a}_{13}$ and $\bar{a}_{14}$ for the 1st element) equal to the position and velocity at the beginning of the next element ($\bar{a}_{21}$ and $\bar{a}_{23}$ for the 2nd element), see Figure 63.

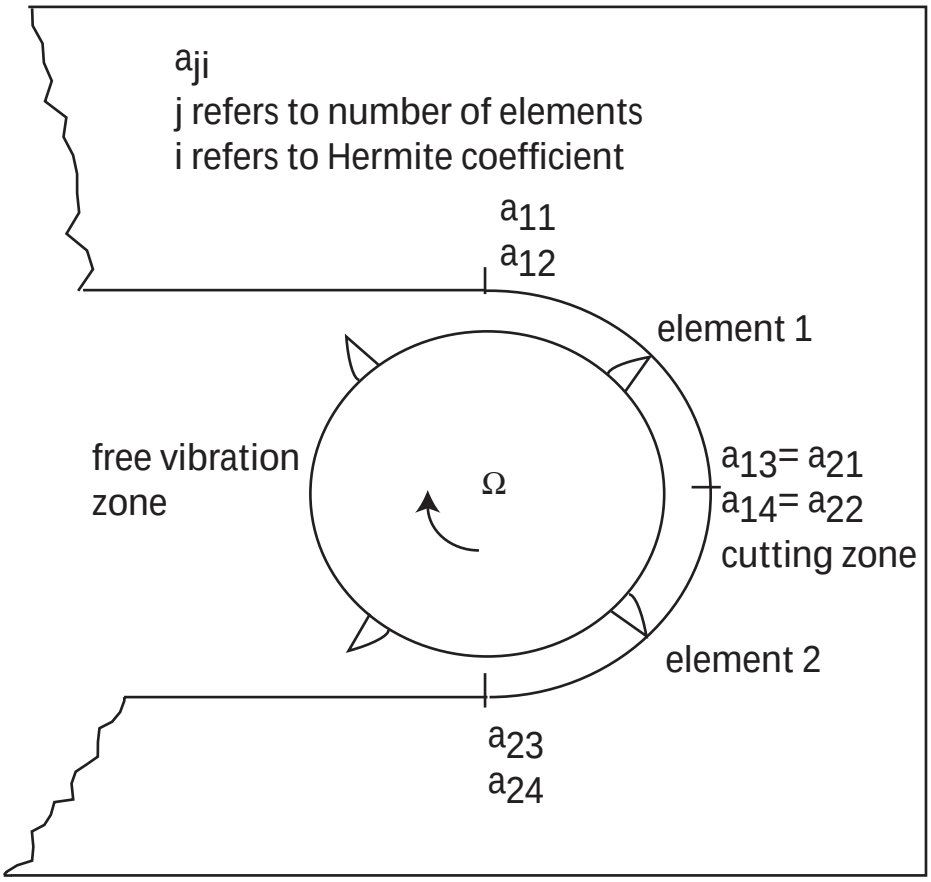

Figure 63. Slotting cut with time in the cut divided into two elements. Transition matrix maps the position and velocity exactly in free vibration zone, while elements map them in cutting zone.

Equations (A.13) and (A.14) can be arranged into a global matrix mapping the position and velocity of each tooth passage in terms of the previous one. Equation (A.13) maps the cutting zone approximately, while Eq. (A.14) maps the free vibration zone exactly. The following expression is for the case when number of elements $E = 3$

$$\begin{bmatrix} [I] & [0] & [0] & [0] \\ \left[N_1^1\right] & \left[N_2^1\right] & [0] & [0] \\ [0] & \left[N_1^2\right] & \left[N_2^2\right] & [0] \\ [0] & [0] & \left[N_1^3\right] & \left[N_2^3\right] \end{bmatrix} \begin{Bmatrix} \vec{a}_{11} \\ \vec{a}_{12} \\ \vec{a}_{21} \\ \vec{a}_{22} \\ \vec{a}_{31} \\ \vec{a}_{32} \\ \vec{a}_{33} \\ \vec{a}_{34} \end{Bmatrix}^n = \begin{bmatrix} [I] & [0] & [0] & [0] \\ \left[P_1^1\right] & \left[P_2^1\right] & [0] & [0] \\ [0] & \left[P_1^2\right] & \left[P_2^2\right] & [0] \\ [0] & [0] & \left[P_1^3\right] & \left[P_2^3\right] \end{bmatrix} \begin{Bmatrix} \vec{a}_{11} \\ \vec{a}_{12} \\ \vec{a}_{21} \\ \vec{a}_{22} \\ \vec{a}_{31} \\ \vec{a}_{32} \\ \vec{a}_{33} \\ \vec{a}_{34} \end{Bmatrix}^{n-1} + \begin{Bmatrix} \vec{0} \\ \vec{0} \\ \vec{C}_1 \\ \vec{C}_2 \\ \vec{C}_1 \\ \vec{C}_2 \\ \vec{C}_1 \\ \vec{C}_2 \end{Bmatrix}, \tag{A.15}$$

where the sub-matrices and elements of the sub-matrices for the j^{th} element are

$$\begin{aligned} \left[\mathrm{N}_1\right]^{\mathrm{j}} &= \begin{bmatrix} \mathrm{N}_{11} & \mathrm{N}_{12} \\ \mathrm{N}_{21} & \mathrm{N}_{22} \end{bmatrix}^{\mathrm{j}}, \left[\mathrm{N}_2\right]^{\mathrm{j}} = \begin{bmatrix} \mathrm{N}_{13} & \mathrm{N}_{14} \\ \mathrm{N}_{23} & \mathrm{N}_{24} \end{bmatrix}^{\mathrm{j}}, \\ \left[P_1\right]^{\mathrm{j}} &= \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}^{\mathrm{j}}, \left[P_2\right]^{\mathrm{j}} = \begin{bmatrix} P_{13} & P_{14} \\ P_{23} & P_{24} \end{bmatrix}^{\mathrm{j}}, \end{aligned} \tag{A.16}$$

$$\begin{aligned} N_{pi}^j &= \int_0^{t_j} \left[\begin{aligned} &\mathbf{M}\ddot{\phi}_i\left(\sigma_j\right) + \mathbf{C}\dot{\phi}_i\left(\sigma_j\right) + \\ &\left(\mathbf{K} - \mathrm{b}\mathbf{K}_{\mathrm{c}}\left(\sigma_j\right)\right)\phi_i\left(\sigma_j\right) \end{aligned} \right] \Psi_p\left(\sigma_j\right) d\sigma_j, \\ P_{pi}^j &= \int_0^{t_j} -\mathrm{b}\mathbf{K}_{\mathrm{c}}\left(\sigma_j\right)\phi_i\left(\sigma_j\right)\Psi_p\left(\sigma_j\right) d\sigma_j, \\ C_p^j &= \int_0^{t_j} b\vec{f}_o\left(\sigma_j\right)\Psi_p\left(\sigma_j\right) d\sigma_j. \end{aligned} \tag{A.17}$$

Equation (A.15) describes a discrete dynamical system, or map, that can be written as

$$\begin{aligned} &\left[\mathrm{A}\right]\vec{a}_n = \left[\mathrm{B}\right]\vec{a}_{n\text{-}1} + \vec{C}, \\ &\text{or} \\ &\vec{a}_n = \left[\mathrm{Q}\right]\vec{a}_{n\text{-}1} + \vec{D}. \end{aligned} \tag{A.18}$$

Stability Prediction

The eigenvalues of the transition matrix $\mathbf{Q}=\mathbf{A}^{-1}\mathbf{B}$ are called characteristic multipliers (CMs) and take on a discrete mapping analogy to the characteristic exponents that govern stability for continuous systems. The condition for stability is that the magnitudes of the CMs must be in a modulus of less than one for a given spindle speed (Ω) and depth of cut (b) for the milling process to be asymptotically stable.

Surface Location Error

Surface location error is defined as the error in the placement of the milling cutter teeth when the surface is generated. When the milling process is stable, the surface location error can be obtained by extracting the position of the tool when the surface is generated. At steady state, the displacement and velocity coefficients are constant and are found from fixed points ($\vec{a}_n^*$) of the dynamic map:

$$\vec{a}_n = \vec{a}_{n-1} = \vec{a}_n^* \quad . \tag{A.19}$$

Substitution of Eq. (A.19) into Eq. (A.18) gives the fixed point map solution or steady state coefficient vector:

$$\vec{a}_n^* = (\mathbf{I}-\mathbf{Q})^{-1}\,\vec{\mathrm{D}} \quad . \tag{A.20}$$

Since $\boldsymbol{Q}$ and $\vec{D}$ can be computed for the milling parameters, the fixed point displacement solution can be found and used to specify surface location error as a function of machining process parameters.

APPENDIX B
MATLAB CODE

Robust Optimization Code

Main program

```
% M. Kurdi (12/1/2004)
% surface location error and MRR robust optimization
clc; clear all; close all; pack;
global Min_speed Max_speed Min_depth Max_depth MRR_c band;
warning off all;
band =500;
Min_speed = 10e3; Max_speed = 20e3;
Min_depth = 1e-6; Max_depth = 18e-3;
nteeth = 1;
tic
%
%        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        % Finding the initial guess
%        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        % MRR_c     MRR constraint in mm^3/s
%
Max_MRR = 400;
hand = waitbar(0,'Please wait');
for MRR_c = 100:100:Max_MRR
   delete('MULTIPOINT OPTIMUMS.m')
   fid2 = fopen('MULTIPOINT OPTIMUMS.m','a');
   speed_vec = 0:.1:1.0; % spindle speed
   radial_vec = linspace(.03,1,11); % radial depth
   for i=1:length(speed_vec)  % loop for spindle speed
      x0(2) = speed_vec(i);
      for j=1:length(radial_vec) % loop for radial depth
         x0(3) =  radial_vec(j);
         % figure
         %   contour_plot;
         % obj_mrr;
         % hold on;
         a = x0(3) * 25.4e-3; % radial depth of cut
         h = 0.1e-3; % feed per tooth
         % calculate initial depth, using MRR constraint and initial speed
         rpm = x0(2) * (Max_speed - Min_speed) + Min_speed;
```

```
        b = MRR_c / (a * rpm * h * nteeth / 60 * 1e9);
        disp('axial depth(m)    speed (rpm)        radial immersion (m)');
        design_point = [num2str(b),'              ',num2str(rpm),'              ',num2str(a)];
        disp(design_point);
        x0(1) = (b - Min_depth) / (Max_depth - Min_depth);
        lb = [0 0 0.01];
        ub = [1 1 1];
        options = optimset('LargeScale','off','MaxFunEvals',50);%,'Display','iter');
        %fprintf(fid,'SLE_depth      depth       speed\n');
        [x,fval,EXITFLAG] = fmincon(@obj,x0,[],[],[],[],lb,ub,@confun,options);
        [c,ceq] = confun(x);
        if EXITFLAG > 0  % solution found
           depth = x(1) * (Max_depth - Min_depth) + Min_depth;
           rpm = x(2) * (Max_speed - Min_speed) + Min_speed;
           a = x(3);
           sle_exact = sle([rpm depth a]);
           % multipoint optimum file
           fprintf(fid2,'%e   %e     %e     %e     %e      %e      %e      %e
                %e\n',depth,rpm,x(3),fval,sle_exact,c(1),c(2),c(3),c(4));
        end % end if loop
      end % end radial loop
   end % end spindle speed loop
   % finding minimum of all solutions found
   fclose(fid2);
   fid2 = fopen('MULTIPOINT OPTIMUMS.m','r');
   xx = fscanf(fid2,'%e %e %e %e %e %e %e  %e  %e\n',[9 inf]);
   xx = xx';
   fclose(fid2);
   %        find minimum value of sle_depth for the range of speed initial
   %        guesses and a particular MRR
   [minimum_sle,index]=min(xx(:,4));
   fid3 = fopen('OPTIMUM POINTS.m','a');
   %        Optimum points file
   fprintf(fid3,'%e       %e      %e      %e      %e      %e      %e      %e      %e
        %e\n',MRR_c,xx(index,1),xx(index,2),xx(index,3),xx(index,4),xx(index,5),xx(ind
        ex,6),xx(index,7),xx(index,8),xx(index,9));
   fclose(fid3);
   waitbar(MRR_c/Max_MRR,hand);

end % end of MRR loop
fclose(hand)
```

Constraint Function

```
% objective function for SLE / depth of cut
function [c, ceq] = confun(x)
global Min_speed Max_speed Min_depth Max_depth MRR_c band;
```

```
x1 = x(1) * (Max_depth - Min_depth) + Min_depth;
x2 = x(2) * (Max_speed - Min_speed) + Min_speed;
% MRR constraint on the first perturbed point
c1 = confun1([x1 x2-band x(3)]);
c2 = confun1([x1 x2 x(3)]);
c3 = confun1([x1 x2+band x(3)]);
% MRR constraint
h = 0.1e-3;   % m/tooth
% b = x(1) ;
% x(2) rpm
% x(3) radial step
a          = x(3)*.0254; % radial depth in m
nteeth     = 1;
MRR = a .* x1 * h * nteeth .* x2 / 60 * 1e9;
c4 = MRR_c - MRR;
c = [c1 c2 c3 c4];
ceq = [];

% objective function for SLE / depth of cut
function c = confun1(x)
% Input:
%   rpm      speed (rpm)
%   E        number of elements
% Output:
%   CM       eigen value for rpm and doc
%   b        transition depth of cut (m)
%
b = x(1);
rpm = x(2);
E       = 25;              % number of elements
Kt      = 1295.9e6*(rpm/1000)^-0.2285; % N/m2
Kn      = ((rpm/1000)^2*1.8485-54.604*(rpm/1000)+423.77)*1e6;
Kte = ((rpm/1000)^2*-0.1335+3.2431*(rpm/1000)+27.216)*1e3;       % N/m
Kne = ((rpm/1000)^2*-0.0821+1.4447*(rpm/1000)+30.202)*1e3;
%%%
```

```
rho         = acos(1-rstep/(Diam/2))/(2*pi);        % fraction of time in cut
IMMERSION   = rstep/Diam;
opt         = 'up';
if TRAVang>LAGang   % MULTIPLE TEETH ARE IN CONTACT
   teethNcontact = floor(TRAVang/LAGang) +1;
else                % SINGLE TOOTH IN CONTACT
   teethNcontact = 1;
end
%%%
```

```
tf     = T-TC;                    % time for free vibs
tj     = TC/E;                    % time for each element
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      SET CUTTER ROTATION ANGLE FOR UP/DOWN-MILLING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
switch opt
  case 'up'
    t0mat   = [0 tj*(1:(E-1))];           % upmilling
    locat   = 2*DOF+lmx+1:3*DOF;
  case 'down'
    tex     = pi/omega; tent=tex-TC;         % downmilling
    t0mat   = [tent tent+tj*(1:(E-1))];     % downmilling
    locat   = (E+1)*2*DOF-DOF-lmy+1:(E+1)*2*DOF-DOF;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      STATE TRANSITION MATRIX
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
G1  = [zeros(size(M)) M; eye(size(M)) zeros(size(M))];
G2  = [K  C; zeros(size(M)) -eye(size(M))];
G   = -G1\G2;
PHI = expm(G*tf);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      N & P are used to create A & B which then become Q in..... a_n = Q a_n-1 + D
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for e=1:E,
  t0 = t0mat(e);
  C1 = V'*[ -1/4*b*(-h*Kt*cos(2*t0*omega+2*omega*tj)+2*h*Kn*omega*tj-
h*Kn*sin(2*t0*omega+2*omega*tj)+4*Kte*sin(t0*omega+omega*tj)-
4*Kne*cos(t0*omega+omega*tj)+h*Kt*cos(2*t0*omega)+h*Kn*sin(2*t0*omega)-
4*Kte*sin(t0*omega)+4*Kne*cos(t0*omega))/omega;
    1/4*b*(2*h*Kt*omega*tj-
h*Kt*sin(2*t0*omega+2*omega*tj)+h*Kn*cos(2*t0*omega+2*omega*tj)-
4*Kte*sin(t0*omega+omega*tj)+4*Kne*cos(t0*omega+omega*tj)+h*Kt*sin(2*t0*ome
ga)-h*Kn*cos(2*t0*omega)+4*Kte*sin(t0*omega)-4*Kne*cos(t0*omega))/omega];
  C2 = V'*[ 1/8*b*(-
h*Kt*sin(2*t0*omega+2*omega*tj)+h*Kt*cos(2*t0*omega+2*omega*tj)*omega*tj+h*
Kn*cos(2*t0*omega+2*omega*tj)+h*Kn*sin(2*t0*omega+2*omega*tj)*omega*tj+8*K
te*sin(t0*omega+omega*tj)-4*Kte*cos(t0*omega+omega*tj)*omega*tj-
```

```
8*Kne*cos(t0*omega+omega*tj)-
4*Kne*sin(t0*omega+omega*tj)*omega*tj+h*Kt*sin(2*t0*omega)-
h*Kn*cos(2*t0*omega)-
8*Kte*sin(t0*omega)+8*Kne*cos(t0*omega)+h*Kt*tj*cos(2*t0*omega)*omega+h*Kn*
tj*sin(2*t0*omega)*omega-4*Kte*tj*cos(t0*omega)*omega-
4*Kne*tj*sin(t0*omega)*omega)/tj/omega^2;
      -
1/8*b*(h*Kt*cos(2*t0*omega+2*omega*tj)+h*Kt*sin(2*t0*omega+2*omega*tj)*omeg
a*tj+h*Kn*sin(2*t0*omega+2*omega*tj)-
h*Kn*cos(2*t0*omega+2*omega*tj)*omega*tj-
8*Kte*sin(t0*omega+omega*tj)+4*Kte*cos(t0*omega+omega*tj)*omega*tj+8*Kne*co
s(t0*omega+omega*tj)+4*Kne*sin(t0*omega+omega*tj)*omega*tj-
h*Kt*cos(2*t0*omega)-h*Kn*sin(2*t0*omega)+8*Kte*sin(t0*omega)-
8*Kne*cos(t0*omega)+h*Kt*tj*sin(2*t0*omega)*omega-
h*Kn*tj*cos(2*t0*omega)*omega+4*Kte*tj*cos(t0*omega)*omega+4*Kne*tj*sin(t0*o
mega)*omega)/tj/omega^2];
   P11 = [  1/8*b*(-
3*Kt*sin(2*t0*omega+2*omega*tj)+3*Kn*cos(2*t0*omega+2*omega*tj)+2*Kn*omeg
a^4*tj^4+3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*sin(2*t0*ome
ga+2*omega*tj)+3*Kt*sin(2*t0*omega)-
3*Kn*cos(2*t0*omega)+2*Kt*cos(2*t0*omega)*omega^3*tj^3+2*Kn*sin(2*t0*omega)
*omega^3*tj^3+3*Kt*omega*tj*cos(2*t0*omega)+3*Kn*omega*tj*sin(2*t0*omega))/o
mega^4/tj^3,  1/8*b*(-3*Kt*cos(2*t0*omega+2*omega*tj)-
3*Kn*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^4*tj^4-
3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*cos(2*t0*omega+2*om
ega*tj)+3*Kt*cos(2*t0*omega)+3*Kn*sin(2*t0*omega)-
2*Kt*sin(2*t0*omega)*omega^3*tj^3+2*Kn*cos(2*t0*omega)*omega^3*tj^3-
3*Kt*omega*tj*sin(2*t0*omega)+3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3;
      1/8*b*(-3*Kt*cos(2*t0*omega+2*omega*tj)-3*Kn*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^4*tj^4-
3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*cos(2*t0*omega+2*om
ega*tj)+3*Kt*cos(2*t0*omega)+3*Kn*sin(2*t0*omega)-
2*Kt*sin(2*t0*omega)*omega^3*tj^3+2*Kn*cos(2*t0*omega)*omega^3*tj^3-
3*Kt*omega*tj*sin(2*t0*omega)+3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3, -
1/8*b*(-
2*Kn*omega^4*tj^4+3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*s
in(2*t0*omega+2*omega*tj)-
3*Kt*sin(2*t0*omega+2*omega*tj)+3*Kn*cos(2*t0*omega+2*omega*tj)+2*Kt*cos(2*
t0*omega)*omega^3*tj^3+2*Kn*sin(2*t0*omega)*omega^3*tj^3+3*Kt*omega*tj*cos(
2*t0*omega)+3*Kn*omega*tj*sin(2*t0*omega)+3*Kt*sin(2*t0*omega)-
3*Kn*cos(2*t0*omega))/omega^4/tj^3];
   P12 =[
1/48*b*(6*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+6*Kt*omega*tj*cos(2*t0*ome
ga+2*omega*tj)+2*Kn*omega^4*tj^4-
9*Kt*sin(2*t0*omega+2*omega*tj)+9*Kn*cos(2*t0*omega+2*omega*tj)+12*Kn*ome
ga*tj*sin(2*t0*omega)+12*Kt*omega*tj*cos(2*t0*omega)-
```

```
6*Kt*omega^2*tj^2*sin(2*t0*omega)+6*Kn*omega^2*tj^2*cos(2*t0*omega)+9*Kt*si
n(2*t0*omega)-9*Kn*cos(2*t0*omega))/tj^2/omega^4, -
1/48*b*(9*Kt*cos(2*t0*omega+2*omega*tj)+9*Kn*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^4*tj^4+6*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)+12*Kt*omega*tj*sin(2*t0*omega)+6*
Kn*omega^2*tj^2*sin(2*t0*omega)+6*Kt*omega^2*tj^2*cos(2*t0*omega)-
9*Kt*cos(2*t0*omega)-12*Kn*omega*tj*cos(2*t0*omega)-
9*Kn*sin(2*t0*omega))/tj^2/omega^4;
-
1/48*b*(2*Kt*omega^4*tj^4+6*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+9*Kt*cos(
2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)+9*Kn*sin(2*t0*omega+2*omega*tj)+1
2*Kt*omega*tj*sin(2*t0*omega)+6*Kn*omega^2*tj^2*sin(2*t0*omega)+6*Kt*omega^
2*tj^2*cos(2*t0*omega)-9*Kt*cos(2*t0*omega)-12*Kn*omega*tj*cos(2*t0*omega)-
9*Kn*sin(2*t0*omega))/tj^2/omega^4, 1/48*b*(-
6*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+2*Kn*omega^4*tj^4-
9*Kn*cos(2*t0*omega+2*omega*tj)+9*Kt*sin(2*t0*omega+2*omega*tj)-
12*Kt*omega*tj*cos(2*t0*omega)-
12*Kn*omega*tj*sin(2*t0*omega)+6*Kt*omega^2*tj^2*sin(2*t0*omega)-
6*Kn*omega^2*tj^2*cos(2*t0*omega)+9*Kn*cos(2*t0*omega)-
9*Kt*sin(2*t0*omega))/tj^2/omega^4];
   P13 =[  1/8*b*(2*Kn*omega^4*tj^4-
3*Kn*cos(2*t0*omega+2*omega*tj)+3*Kt*sin(2*t0*omega+2*omega*tj)-
3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
2*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
3*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+3*Kn*cos(2*t0*omega)-
3*Kt*sin(2*t0*omega)-3*Kt*omega*tj*cos(2*t0*omega)-
3*Kn*omega*tj*sin(2*t0*omega))/omega^4/tj^3,
1/8*b*(2*Kt*omega^4*tj^4+2*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
2*Kn*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+3*Kt*cos(2*t0*omega+2*omega*t
j)+3*Kn*sin(2*t0*omega+2*omega*tj)+3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
3*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)-3*Kt*cos(2*t0*omega)-
3*Kn*sin(2*t0*omega)+3*Kt*omega*tj*sin(2*t0*omega)-
3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3;
      -
1/8*b*(2*Kt*omega^4*tj^4+2*Kn*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+3*Kn
*omega*tj*cos(2*t0*omega+2*omega*tj)-3*Kn*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
3*Kt*cos(2*t0*omega+2*omega*tj)+3*Kn*sin(2*t0*omega)+3*Kt*cos(2*t0*omega)-
3*Kt*omega*tj*sin(2*t0*omega)+3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3,
1/8*b*(2*Kn*omega^4*tj^4+3*Kn*cos(2*t0*omega+2*omega*tj)+2*Kn*omega^3*tj^3
*sin(2*t0*omega+2*omega*tj)-
3*Kt*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj
```

```
)+3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*sin(2*t0*omega+2*o
mega*tj)-
3*Kn*cos(2*t0*omega)+3*Kt*sin(2*t0*omega)+3*Kt*omega*tj*cos(2*t0*omega)+3*
Kn*omega*tj*sin(2*t0*omega))/omega^4/tj^3];
   P14 =[ -1/48*b*(6*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
12*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-
6*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
12*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+2*Kn*omega^4*tj^4-
9*Kn*cos(2*t0*omega+2*omega*tj)+9*Kt*sin(2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*sin(2*t0*omega)-
6*Kt*omega*tj*cos(2*t0*omega)+9*Kn*cos(2*t0*omega)-
9*Kt*sin(2*t0*omega))/tj^2/omega^4, 1/48*b*(-2*Kt*omega^4*tj^4-
12*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+12*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+6*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+6*Kn*sin(2*t0*omega+2
*omega*tj)*omega^2*tj^2-9*Kt*cos(2*t0*omega+2*omega*tj)-
9*Kn*sin(2*t0*omega+2*omega*tj)-
6*Kt*omega*tj*sin(2*t0*omega)+6*Kn*omega*tj*cos(2*t0*omega)+9*Kt*cos(2*t0*o
mega)+9*Kn*sin(2*t0*omega))/tj^2/omega^4;
       1/48*b*(2*Kt*omega^4*tj^4-
12*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+12*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+6*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+6*Kn*sin(2*t0*omega+2
*omega*tj)*omega^2*tj^2-9*Kt*cos(2*t0*omega+2*omega*tj)-
9*Kn*sin(2*t0*omega+2*omega*tj)-
6*Kt*omega*tj*sin(2*t0*omega)+6*Kn*omega*tj*cos(2*t0*omega)+9*Kt*cos(2*t0*o
mega)+9*Kn*sin(2*t0*omega))/tj^2/omega^4, 1/48*b*(-
12*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-9*Kn*cos(2*t0*omega+2*omega*tj)-
2*Kn*omega^4*tj^4-6*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
12*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+6*Kn*cos(2*t0*omega+2*omega*tj)*o
mega^2*tj^2+9*Kt*sin(2*t0*omega+2*omega*tj)+9*Kn*cos(2*t0*omega)-
9*Kt*sin(2*t0*omega)-6*Kn*omega*tj*sin(2*t0*omega)-
6*Kt*omega*tj*cos(2*t0*omega))/tj^2/omega^4];
   P21 =[ -1/80*b*(-15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
15*Kn*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+60*Kt*cos(2*t0*omega+2*omega
*tj)+60*Kn*sin(2*t0*omega+2*omega*tj)+4*Kn*omega^5*tj^5+60*Kt*omega*tj*sin(2
*t0*omega)-
10*Kn*omega^3*tj^3*cos(2*t0*omega)+15*Kn*omega^2*tj^2*sin(2*t0*omega)+15*K
t*omega^2*tj^2*cos(2*t0*omega)+10*Kt*omega^3*tj^3*sin(2*t0*omega)-
60*Kn*omega*tj*cos(2*t0*omega)+10*Kn*sin(2*t0*omega)*omega^4*tj^4+10*Kt*co
s(2*t0*omega)*omega^4*tj^4-60*Kn*sin(2*t0*omega)-
60*Kt*cos(2*t0*omega)+60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
60*Kn*omega*tj*cos(2*t0*omega+2*omega*tj))/omega^5/tj^4, -1/80*b*(-
60*Kn*cos(2*t0*omega)+60*Kt*sin(2*t0*omega)-
15*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kt*sin(2*t0*omega+2*omega
*tj)*omega^2*tj^2+60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-
60*Kt*sin(2*t0*omega+2*omega*tj)+60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+
60*Kn*cos(2*t0*omega+2*omega*tj)+10*Kt*omega^3*tj^3*cos(2*t0*omega)-
```

```
15*Kt*omega^2*tj^2*sin(2*t0*omega)-
10*Kt*sin(2*t0*omega)*omega^4*tj^4+10*Kn*cos(2*t0*omega)*omega^4*tj^4+15*K
n*omega^2*tj^2*cos(2*t0*omega)+60*Kt*omega*tj*cos(2*t0*omega)+10*Kn*omega^
3*tj^3*sin(2*t0*omega)+60*Kn*omega*tj*sin(2*t0*omega)+4*Kt*omega^5*tj^5)/ome
ga^5/tj^4;
      1/80*b*(60*Kn*cos(2*t0*omega)-
60*Kt*sin(2*t0*omega)+15*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
15*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kt*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-60*Kn*cos(2*t0*omega+2*omega*tj)-
10*Kt*omega^3*tj^3*cos(2*t0*omega)+15*Kt*omega^2*tj^2*sin(2*t0*omega)+10*Kt
*sin(2*t0*omega)*omega^4*tj^4-10*Kn*cos(2*t0*omega)*omega^4*tj^4-
15*Kn*omega^2*tj^2*cos(2*t0*omega)-60*Kt*omega*tj*cos(2*t0*omega)-
10*Kn*omega^3*tj^3*sin(2*t0*omega)-
60*Kn*omega*tj*sin(2*t0*omega)+4*Kt*omega^5*tj^5)/omega^5/tj^4, -1/80*b*(-
60*Kt*omega*tj*sin(2*t0*omega)+60*Kn*omega*tj*cos(2*t0*omega)-
10*Kn*sin(2*t0*omega)*omega^4*tj^4-10*Kt*cos(2*t0*omega)*omega^4*tj^4-
15*Kt*omega^2*tj^2*cos(2*t0*omega)-
10*Kt*omega^3*tj^3*sin(2*t0*omega)+10*Kn*omega^3*tj^3*cos(2*t0*omega)-
15*Kn*omega^2*tj^2*sin(2*t0*omega)+60*Kn*sin(2*t0*omega)+60*Kt*cos(2*t0*ome
ga)-
60*Kt*cos(2*t0*omega+2*omega*tj)+4*Kn*omega^5*tj^5+60*Kn*omega*tj*cos(2*t0
*omega+2*omega*tj)+15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*sin
(2*t0*omega+2*omega*tj)*omega^2*tj^2-60*Kn*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj))/omega^5/tj^4];
   P22 = [ 1/480*b*(-2*Kn*omega^5*tj^5-180*Kt*cos(2*t0*omega+2*omega*tj)-
180*Kn*sin(2*t0*omega+2*omega*tj)-
135*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*cos(2*t0*omega+2
*omega*tj)+30*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+30*Kn*sin(2*t0*ome
ga+2*omega*tj)*omega^2*tj^2-
225*Kt*omega*tj*sin(2*t0*omega)+180*Kn*sin(2*t0*omega)+180*Kt*cos(2*t0*omeg
a)+225*Kn*omega*tj*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*cos(2*t0*omega)+30*Kt*omega^3*tj^3*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*cos(2*t0*omega)-
120*Kn*omega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5, -
1/480*b*(135*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*sin(2*t0
*omega+2*omega*tj)-
180*Kt*sin(2*t0*omega+2*omega*tj)+180*Kn*cos(2*t0*omega+2*omega*tj)+2*Kt*o
mega^5*tj^5+30*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kt*omega^3*tj^3*cos(2*t0*omega)+120*Kn*omega^2*tj^2*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*sin(2*t0*omega)+225*Kt*omega*tj*cos(2*t0*omega)+225*Kn*
omega*tj*sin(2*t0*omega)+180*Kt*sin(2*t0*omega)-
180*Kn*cos(2*t0*omega))/tj^3/omega^5;
```

```
      -
1/480*b*(135*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*sin(2*t0
*omega+2*omega*tj)-
180*Kt*sin(2*t0*omega+2*omega*tj)+180*Kn*cos(2*t0*omega+2*omega*tj)-
2*Kt*omega^5*tj^5+30*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kt*omega^3*tj^3*cos(2*t0*omega)+120*Kn*omega^2*tj^2*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*sin(2*t0*omega)+225*Kt*omega*tj*cos(2*t0*omega)+225*Kn*
omega*tj*sin(2*t0*omega)+180*Kt*sin(2*t0*omega)-
180*Kn*cos(2*t0*omega))/tj^3/omega^5, -1/480*b*(2*Kn*omega^5*tj^5-
180*Kt*cos(2*t0*omega+2*omega*tj)-180*Kn*sin(2*t0*omega+2*omega*tj)-
135*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*cos(2*t0*omega+2
*omega*tj)+30*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+30*Kn*sin(2*t0*ome
ga+2*omega*tj)*omega^2*tj^2-
225*Kt*omega*tj*sin(2*t0*omega)+180*Kn*sin(2*t0*omega)+180*Kt*cos(2*t0*omeg
a)+225*Kn*omega*tj*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*cos(2*t0*omega)+30*Kt*omega^3*tj^3*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*cos(2*t0*omega)-
120*Kn*omega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5];
   P23 = [ -1/80*b*(-4*Kn*omega^5*tj^5-60*Kt*cos(2*t0*omega+2*omega*tj)-
60*Kn*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*sin(2*t0*omega
+2*omega*tj)*omega^2*tj^2-
10*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*sin(2*t0*o
mega+2*omega*tj)+10*Kt*omega^4*tj^4*cos(2*t0*omega+2*omega*tj)+10*Kn*omega
a^3*tj^3*cos(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega)+60*Kn*sin(2*t0*omega)+60*Kt*cos(2*t0*omega)+
60*Kn*omega*tj*cos(2*t0*omega)-15*Kt*omega^2*tj^2*cos(2*t0*omega)-
15*Kn*omega^2*tj^2*sin(2*t0*omega))/omega^5/tj^4, -1/80*b*(-
60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kt*sin(2*t0*omega+2*omega*tj)-
60*Kn*cos(2*t0*omega+2*omega*tj)-
10*Kt*omega^4*tj^4*sin(2*t0*omega+2*omega*tj)-
10*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*cos(2*t0*o
mega+2*omega*tj)-10*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
4*Kt*omega^5*tj^5-
15*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*cos(2*t0*omega+2*omega
*tj)*omega^2*tj^2-
15*Kn*omega^2*tj^2*cos(2*t0*omega)+15*Kt*omega^2*tj^2*sin(2*t0*omega)-
60*Kt*omega*tj*cos(2*t0*omega)-60*Kn*omega*tj*sin(2*t0*omega)-
60*Kt*sin(2*t0*omega)+60*Kn*cos(2*t0*omega))/omega^5/tj^4;
      -1/80*b*(-60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kt*sin(2*t0*omega+2*omega*tj)-
60*Kn*cos(2*t0*omega+2*omega*tj)-
```

```
10*Kt*omega^4*tj^4*sin(2*t0*omega+2*omega*tj)-
10*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*cos(2*t0*o
mega+2*omega*tj)-
10*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+4*Kt*omega^5*tj^5-
15*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*cos(2*t0*omega+2*omega
*tj)*omega^2*tj^2-
15*Kn*omega^2*tj^2*cos(2*t0*omega)+15*Kt*omega^2*tj^2*sin(2*t0*omega)-
60*Kt*omega*tj*cos(2*t0*omega)-60*Kn*omega*tj*sin(2*t0*omega)-
60*Kt*sin(2*t0*omega)+60*Kn*cos(2*t0*omega))/omega^5/tj^4,
1/80*b*(4*Kn*omega^5*tj^5-60*Kt*cos(2*t0*omega+2*omega*tj)-
60*Kn*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*sin(2*t0*omega
+2*omega*tj)*omega^2*tj^2-
10*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*sin(2*t0*o
mega+2*omega*tj)+10*Kt*omega^4*tj^4*cos(2*t0*omega+2*omega*tj)+10*Kn*omeg
a^3*tj^3*cos(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega)+60*Kn*sin(2*t0*omega)+60*Kt*cos(2*t0*omega)+
60*Kn*omega*tj*cos(2*t0*omega)-15*Kt*omega^2*tj^2*cos(2*t0*omega)-
15*Kn*omega^2*tj^2*sin(2*t0*omega))/omega^5/tj^4];
   P24 = [ -
1/480*b*(2*Kn*omega^5*tj^5+180*Kt*cos(2*t0*omega+2*omega*tj)+180*Kn*sin(2*t
0*omega+2*omega*tj)+225*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
225*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)-
120*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
120*Kn*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+30*Kn*omega^3*tj^3*cos(2*t0*o
mega+2*omega*tj)+135*Kt*omega*tj*sin(2*t0*omega)-180*Kn*sin(2*t0*omega)-
180*Kt*cos(2*t0*omega)-
135*Kn*omega*tj*cos(2*t0*omega)+30*Kt*omega^2*tj^2*cos(2*t0*omega)+30*Kn*o
mega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5, -
1/480*b*(225*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+225*Kn*omega*tj*sin(2*t0
*omega+2*omega*tj)-
180*Kt*sin(2*t0*omega+2*omega*tj)+180*Kn*cos(2*t0*omega+2*omega*tj)-
30*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)-
30*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^5*tj^5+120*Kt*sin(
2*t0*omega+2*omega*tj)*omega^2*tj^2-
120*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+30*Kn*omega^2*tj^2*cos(2*t0*
omega)-
30*Kt*omega^2*tj^2*sin(2*t0*omega)+135*Kt*omega*tj*cos(2*t0*omega)+135*Kn*o
mega*tj*sin(2*t0*omega)+180*Kt*sin(2*t0*omega)-
180*Kn*cos(2*t0*omega))/tj^3/omega^5;
      1/480*b*(-225*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
225*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+180*Kt*sin(2*t0*omega+2*omega*tj
)-
180*Kn*cos(2*t0*omega+2*omega*tj)+30*Kt*omega^3*tj^3*cos(2*t0*omega+2*ome
```

```
ga*tj)+30*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^5*tj^5-
120*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+120*Kn*cos(2*t0*omega+2*ome
ga*tj)*omega^2*tj^2-
30*Kn*omega^2*tj^2*cos(2*t0*omega)+30*Kt*omega^2*tj^2*sin(2*t0*omega)-
135*Kt*omega*tj*cos(2*t0*omega)-135*Kn*omega*tj*sin(2*t0*omega)-
180*Kt*sin(2*t0*omega)+180*Kn*cos(2*t0*omega))/tj^3/omega^5, -
1/480*b*(2*Kn*omega^5*tj^5-180*Kt*cos(2*t0*omega+2*omega*tj)-
180*Kn*sin(2*t0*omega+2*omega*tj)-
225*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+225*Kn*omega*tj*cos(2*t0*omega+2
*omega*tj)+120*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+120*Kn*sin(2*t0*o
mega+2*omega*tj)*omega^2*tj^2+30*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)
-30*Kn*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)-
135*Kt*omega*tj*sin(2*t0*omega)+180*Kn*sin(2*t0*omega)+180*Kt*cos(2*t0*omeg
a)+135*Kn*omega*tj*cos(2*t0*omega)-30*Kt*omega^2*tj^2*cos(2*t0*omega)-
30*Kn*omega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5];
   P11 = [P11(1,1)*ones(lmx,1) P11(1,2)*ones(lmx,1); P11(2,1)*ones(lmx,1)
P11(2,2)*ones(lmx,1)]*V;
   P12 = [P12(1,1)*ones(lmx,1) P12(1,2)*ones(lmx,1); P12(2,1)*ones(lmx,1)
P12(2,2)*ones(lmx,1)]*V;
   P13 = [P13(1,1)*ones(lmx,1) P13(1,2)*ones(lmx,1); P13(2,1)*ones(lmx,1)
P13(2,2)*ones(lmx,1)]*V;
   P14 = [P14(1,1)*ones(lmx,1) P14(1,2)*ones(lmx,1); P14(2,1)*ones(lmx,1)
P14(2,2)*ones(lmx,1)]*V;
   P21 = [P21(1,1)*ones(lmx,1) P21(1,2)*ones(lmx,1); P21(2,1)*ones(lmx,1)
P21(2,2)*ones(lmx,1)]*V;
   P22 = [P22(1,1)*ones(lmx,1) P22(1,2)*ones(lmx,1); P22(2,1)*ones(lmx,1)
P22(2,2)*ones(lmx,1)]*V;
   P23 = [P23(1,1)*ones(lmx,1) P23(1,2)*ones(lmx,1); P23(2,1)*ones(lmx,1)
P23(2,2)*ones(lmx,1)]*V;
   P24 = [P24(1,1)*ones(lmx,1) P24(1,2)*ones(lmx,1); P24(2,1)*ones(lmx,1)
P24(2,2)*ones(lmx,1)]*V;
   N11 = -C+1/2*K*tj+P11;
   N12 = -M+1/12*K*tj^2+P12;
   N13 = C+1/2*K*tj+P13;
   N14 = M-1/12*K*tj^2+P14;
   N21 = M/tj-1/10*K*tj+P21;
   N22 = 1/2*M-1/12*C*tj-1/120*K*tj^2+P22;
   N23 = -M/tj+1/10*K*tj+P23;
   N24 = 1/2*M+1/12*C*tj-1/120*K*tj^2+P24;
   N1  = [N11 N12; N21 N22];
   N2  = [N13 N14; N23 N24];
   P1  = [P11 P12; P21 P22];
   P2  = [P13 P14; P23 P24];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % BUILD GLOBAL MATRICES
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
   A(1:2*DOF,1:2*DOF) = eye(2*DOF);
   A(2*DOF*e+1:2*DOF*e+2*DOF,2*DOF*(e-1)+1:2*DOF*(e-1)+2*DOF) = N1;
   A(2*DOF*e+1:2*DOF*e+2*DOF,2*DOF*(e-1)+2*DOF+1:2*DOF*(e-
1)+2*DOF+2*DOF) = N2;
   B(2*DOF*e+1:2*DOF*e+2*DOF,2*DOF*(e-1)+1:2*DOF*(e-1)+2*DOF) = P1;
   B(2*DOF*e+1:2*DOF*e+2*DOF,2*DOF*(e-1)+2*DOF+1:2*DOF*(e-
1)+2*DOF+2*DOF) = P2;
   B(1:2*(DOF),E*2*(DOF)+1:(E+1)*2*(DOF)) = PHI;
   Cvec(1:2*DOF,1) = zeros(2*DOF,1);
   Cvec(2*DOF*e+1:2*DOF*e+DOF,1) = C1;
   Cvec(2*DOF*e+DOF+1:2*DOF*e+2*DOF,1) = C2;
end; % end # of elements loop
Q          = A\B;
[vec,lam]   = eig(Q);
CM         = max(abs(diag(lam)));
D          = A\Cvec;
N1 = zeros(2*DOF,2*DOF); N2 = N1;
P1 = N1; P2 = P1;
c = CM - 1;
return
%save TFEA_STABSLE_LOW ss zz CM ee IMMERSION SLE
% NOTES
% ss - spindle speeds
% zz - depth of cut
% CM - charistic multipliers or eigenvalues
% SLE- surface location error
```

Objective Function

```
%
% average surface location error
% % Input:
% x(1) depth
% x(2) speed
% x(3) radial
function SLE_AVER = obj(x)
global Min_speed Max_speed Min_depth Max_depth MRR_c band;
x1 = x(1) * (Max_depth - Min_depth) + Min_depth;
x2 = x(2) * (Max_speed - Min_speed) + Min_speed;
% MRR constraint on the first perturbed point
sle1 = obj1([x1, x2-band ,x(3)]);
sle2 = obj1([x1, x2, x(3)]);
sle3 = obj1([x1, x2+band ,x(3)]);
SLE_AVER = (sle1+sle2+sle3)/3;
```

```
% objective function for SLE
function SLE = obj1(x)
% Input:
% x(1) depth
% x(2) speed
% x(3) radial
        b = x(1);
        rpm = x(2);
        rstep      = x(3);                        % radial immersion (inches)
        E      = 25;              % number of elements
         % adjust cutting coefficients to spindle speed
   Kt      = 1295.9e6*(rpm/1000)^-0.2285; % N/m2
        Kn      = ((rpm/1000)^2*1.8485-54.604*(rpm/1000)+423.77)*1e6;
        Kte = ((rpm/1000)^2*-0.1335+3.2431*(rpm/1000)+27.216)*1e3;       % N/m
        Kne = ((rpm/1000)^2*-0.0821+1.4447*(rpm/1000)+30.202)*1e3;

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %       CUT PROCESS DESCRIPTION GEOMETRY/IMMERSION/PROCESS
PARAMETERS
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        h          = 0.1e-3;                     % feed per tooth
        nteeth      = 1;                           % number of teeth
        Diam        = 1;                          % inches
        TRAVang     = acos(1-rstep/(Diam/2));              % angular travel during cutting
        LAGang      = 2*pi/nteeth;                     % separation angle for teeth
        rho         = acos(1-rstep/(Diam/2))/(2*pi);        % fraction of time in cut
        IMMERSION   = rstep/Diam;
        opt         = 'up';
        if TRAVang>LAGang   % MULTIPLE TEETH ARE IN CONTACT
      teethNcontact = floor(TRAVang/LAGang) +1;
        else               % SINGLE TOOTH IN CONTACT
      teethNcontact = 1;
        end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %      LOAD SYSTEM IDENTIFICATION MATRICES
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       load SYSTEM_ID_1mode
   Kx =  4.4528e+006;
   Ky =  3.5542e+006;
   Mx =   0.4362;
   My =    0.3471;
   Cx =   82.5955;
```

```
   Cy =   89.8606;
% zeta_x = .02996; zeta_y = 0.02576;
% freq_x = 362.75; freq_y = 362.71;
% Kx = 1.308e6; Ky = 1.194e6;
% Mx = Kx/(freq_x*2*pi)^2;
% My = Ky/(freq_y*2*pi)^2;
% Cx = zeta_x * 2 * Mx * 2*pi * freq_x;
% Cy = zeta_y * 2 * My * 2*pi * freq_y;
M   =[Mx zeros(size(Mx)); zeros(size(Mx)) My];
C   =[Cx zeros(size(Mx)); zeros(size(Mx)) Cy];
K   =[Kx zeros(size(Mx)); zeros(size(Mx)) Ky];
lmx = length(Mx(1,:));
lmy = length(My(1,:));
lmx=1; lmy=1; DOF=2; Mx=M(1,1); My=M(2,2);
DOF = lmx+lmy;
        V       = [ones(1,lmx) zeros(1,lmy); zeros(1,lmx) ones(1,lmy)];
        A       = zeros((E+1)*2*DOF,(E+1)*2*DOF);
        B       = A;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %       BEGIN LOOP CALCULATIONS OVER RPM vs DOC FIELD
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        speed   = rpm;
        omega   = speed/60*(2*pi);              % radians per second
        T       = (2*pi)/omega/nteeth;          % tooth pass period
        TC      = rho*T*nteeth;                 % time a single tooth spends in the cut
        tf      = T-TC;                         % time for free vibs
        tj      = TC/E;                         % time for each element

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %       SET CUTTER ROTATION ANGLE FOR UP/DOWN-MILLING
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        switch opt
        case 'up'
        t0mat   = [0 tj*(1:(E-1))];             % upmilling
        locat   = 2*DOF+lmx+1:3*DOF;
        case 'down'
        tex     = pi/omega; tent=tex-TC;        % downmilling
        t0mat   = [tent tent+tj*(1:(E-1))];     % downmilling
        locat   = (E+1)*2*DOF-DOF-lmy+1:(E+1)*2*DOF-DOF;
        end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
        %       STATE TRANSITION MATRIX
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        G1  = [zeros(size(M)) M; eye(size(M)) zeros(size(M))];
        G2  = [K  C; zeros(size(M)) -eye(size(M))];
        G   = -G1\G2;
        PHI = expm(G*tf);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %       N & P are used to create A & B which then become Q in..... a_n = Q a_n-1
+ D
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for e=1:E,
t0 = t0mat(e);
C1 = V'*[ -1/4*b*(-h*Kt*cos(2*t0*omega+2*omega*tj)+2*h*Kn*omega*tj-
h*Kn*sin(2*t0*omega+2*omega*tj)+4*Kte*sin(t0*omega+omega*tj)-
4*Kne*cos(t0*omega+omega*tj)+h*Kt*cos(2*t0*omega)+h*Kn*sin(2*t0*omega)-
4*Kte*sin(t0*omega)+4*Kne*cos(t0*omega))/omega;
  1/4*b*(2*h*Kt*omega*tj-
h*Kt*sin(2*t0*omega+2*omega*tj)+h*Kn*cos(2*t0*omega+2*omega*tj)-
4*Kte*sin(t0*omega+omega*tj)+4*Kne*cos(t0*omega+omega*tj)+h*Kt*sin(2*t0*ome
ga)-h*Kn*cos(2*t0*omega)+4*Kte*sin(t0*omega)-4*Kne*cos(t0*omega))/omega];
C2 = V'*[ 1/8*b*(-
h*Kt*sin(2*t0*omega+2*omega*tj)+h*Kt*cos(2*t0*omega+2*omega*tj)*omega*tj+h*
Kn*cos(2*t0*omega+2*omega*tj)+h*Kn*sin(2*t0*omega+2*omega*tj)*omega*tj+8*K
te*sin(t0*omega+omega*tj)-4*Kte*cos(t0*omega+omega*tj)*omega*tj-
8*Kne*cos(t0*omega+omega*tj)-
4*Kne*sin(t0*omega+omega*tj)*omega*tj+h*Kt*sin(2*t0*omega)-
h*Kn*cos(2*t0*omega)-
8*Kte*sin(t0*omega)+8*Kne*cos(t0*omega)+h*Kt*tj*cos(2*t0*omega)*omega+h*Kn*
tj*sin(2*t0*omega)*omega-4*Kte*tj*cos(t0*omega)*omega-
4*Kne*tj*sin(t0*omega)*omega)/tj/omega^2;
 -
1/8*b*(h*Kt*cos(2*t0*omega+2*omega*tj)+h*Kt*sin(2*t0*omega+2*omega*tj)*omeg
a*tj+h*Kn*sin(2*t0*omega+2*omega*tj)-
h*Kn*cos(2*t0*omega+2*omega*tj)*omega*tj-
8*Kte*sin(t0*omega+omega*tj)+4*Kte*cos(t0*omega+omega*tj)*omega*tj+8*Kne*co
s(t0*omega+omega*tj)+4*Kne*sin(t0*omega+omega*tj)*omega*tj-
h*Kt*cos(2*t0*omega)-h*Kn*sin(2*t0*omega)+8*Kte*sin(t0*omega)-
8*Kne*cos(t0*omega)+h*Kt*tj*sin(2*t0*omega)*omega-
h*Kn*tj*cos(2*t0*omega)*omega+4*Kte*tj*cos(t0*omega)*omega+4*Kne*tj*sin(t0*o
mega)*omega)/tj/omega^2];
```

```
P11 = [  1/8*b*(-
3*Kt*sin(2*t0*omega+2*omega*tj)+3*Kn*cos(2*t0*omega+2*omega*tj)+2*Kn*omeg
a^4*tj^4+3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*sin(2*t0*ome
ga+2*omega*tj)+3*Kt*sin(2*t0*omega)-
3*Kn*cos(2*t0*omega)+2*Kt*cos(2*t0*omega)*omega^3*tj^3+2*Kn*sin(2*t0*omega)
*omega^3*tj^3+3*Kt*omega*tj*cos(2*t0*omega)+3*Kn*omega*tj*sin(2*t0*omega))/o
mega^4/tj^3,  1/8*b*(-3*Kt*cos(2*t0*omega+2*omega*tj)-
3*Kn*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^4*tj^4-
3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*cos(2*t0*omega+2*om
ega*tj)+3*Kt*cos(2*t0*omega)+3*Kn*sin(2*t0*omega)-
2*Kt*sin(2*t0*omega)*omega^3*tj^3+2*Kn*cos(2*t0*omega)*omega^3*tj^3-
3*Kt*omega*tj*sin(2*t0*omega)+3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3;
  1/8*b*(-3*Kt*cos(2*t0*omega+2*omega*tj)-3*Kn*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^4*tj^4-
3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*cos(2*t0*omega+2*om
ega*tj)+3*Kt*cos(2*t0*omega)+3*Kn*sin(2*t0*omega)-
2*Kt*sin(2*t0*omega)*omega^3*tj^3+2*Kn*cos(2*t0*omega)*omega^3*tj^3-
3*Kt*omega*tj*sin(2*t0*omega)+3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3, -
1/8*b*(-
2*Kn*omega^4*tj^4+3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*s
in(2*t0*omega+2*omega*tj)-
3*Kt*sin(2*t0*omega+2*omega*tj)+3*Kn*cos(2*t0*omega+2*omega*tj)+2*Kt*cos(2*
t0*omega)*omega^3*tj^3+2*Kn*sin(2*t0*omega)*omega^3*tj^3+3*Kt*omega*tj*cos(
2*t0*omega)+3*Kn*omega*tj*sin(2*t0*omega)+3*Kt*sin(2*t0*omega)-
3*Kn*cos(2*t0*omega))/omega^4/tj^3];
P12 =[
1/48*b*(6*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+6*Kt*omega*tj*cos(2*t0*ome
ga+2*omega*tj)+2*Kn*omega^4*tj^4-
9*Kt*sin(2*t0*omega+2*omega*tj)+9*Kn*cos(2*t0*omega+2*omega*tj)+12*Kn*ome
ga*tj*sin(2*t0*omega)+12*Kt*omega*tj*cos(2*t0*omega)-
6*Kt*omega^2*tj^2*sin(2*t0*omega)+6*Kn*omega^2*tj^2*cos(2*t0*omega)+9*Kt*si
n(2*t0*omega)-9*Kn*cos(2*t0*omega))/tj^2/omega^4, -
1/48*b*(9*Kt*cos(2*t0*omega+2*omega*tj)+9*Kn*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^4*tj^4+6*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)+12*Kt*omega*tj*sin(2*t0*omega)+6*
Kn*omega^2*tj^2*sin(2*t0*omega)+6*Kt*omega^2*tj^2*cos(2*t0*omega)-
9*Kt*cos(2*t0*omega)-12*Kn*omega*tj*cos(2*t0*omega)-
9*Kn*sin(2*t0*omega))/tj^2/omega^4;
 -
1/48*b*(2*Kt*omega^4*tj^4+6*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+9*Kt*cos(
2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)+9*Kn*sin(2*t0*omega+2*omega*tj)+1
2*Kt*omega*tj*sin(2*t0*omega)+6*Kn*omega^2*tj^2*sin(2*t0*omega)+6*Kt*omega^
2*tj^2*cos(2*t0*omega)-9*Kt*cos(2*t0*omega)-12*Kn*omega*tj*cos(2*t0*omega)-
9*Kn*sin(2*t0*omega))/tj^2/omega^4, 1/48*b*(-
6*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
```

```
6*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+2*Kn*omega^4*tj^4-
9*Kn*cos(2*t0*omega+2*omega*tj)+9*Kt*sin(2*t0*omega+2*omega*tj)-
12*Kt*omega*tj*cos(2*t0*omega)-
12*Kn*omega*tj*sin(2*t0*omega)+6*Kt*omega^2*tj^2*sin(2*t0*omega)-
6*Kn*omega^2*tj^2*cos(2*t0*omega)+9*Kn*cos(2*t0*omega)-
9*Kt*sin(2*t0*omega))/tj^2/omega^4];
P13 =[  1/8*b*(2*Kn*omega^4*tj^4-
3*Kn*cos(2*t0*omega+2*omega*tj)+3*Kt*sin(2*t0*omega+2*omega*tj)-
3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
2*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
3*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+3*Kn*cos(2*t0*omega)-
3*Kt*sin(2*t0*omega)-3*Kt*omega*tj*cos(2*t0*omega)-
3*Kn*omega*tj*sin(2*t0*omega))/omega^4/tj^3,
1/8*b*(2*Kt*omega^4*tj^4+2*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
2*Kn*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+3*Kt*cos(2*t0*omega+2*omega*t
j)+3*Kn*sin(2*t0*omega+2*omega*tj)+3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
3*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)-3*Kt*cos(2*t0*omega)-
3*Kn*sin(2*t0*omega)+3*Kt*omega*tj*sin(2*t0*omega)-
3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3;
 -
1/8*b*(2*Kt*omega^4*tj^4+2*Kn*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+3*Kn
*omega*tj*cos(2*t0*omega+2*omega*tj)-3*Kn*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
3*Kt*cos(2*t0*omega+2*omega*tj)+3*Kn*sin(2*t0*omega)+3*Kt*cos(2*t0*omega)-
3*Kt*omega*tj*sin(2*t0*omega)+3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3,
1/8*b*(2*Kn*omega^4*tj^4+3*Kn*cos(2*t0*omega+2*omega*tj)+2*Kn*omega^3*tj^3
*sin(2*t0*omega+2*omega*tj)-
3*Kt*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj
)+3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*sin(2*t0*omega+2*o
mega*tj)-
3*Kn*cos(2*t0*omega)+3*Kt*sin(2*t0*omega)+3*Kt*omega*tj*cos(2*t0*omega)+3*
Kn*omega*tj*sin(2*t0*omega))/omega^4/tj^3];
P14 =[ -1/48*b*(6*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
12*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-
6*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
12*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+2*Kn*omega^4*tj^4-
9*Kn*cos(2*t0*omega+2*omega*tj)+9*Kt*sin(2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*sin(2*t0*omega)-
6*Kt*omega*tj*cos(2*t0*omega)+9*Kn*cos(2*t0*omega)-
9*Kt*sin(2*t0*omega))/tj^2/omega^4, 1/48*b*(-2*Kt*omega^4*tj^4-
12*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+12*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+6*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+6*Kn*sin(2*t0*omega+2
*omega*tj)*omega^2*tj^2-9*Kt*cos(2*t0*omega+2*omega*tj)-
9*Kn*sin(2*t0*omega+2*omega*tj)-
```

```
6*Kt*omega*tj*sin(2*t0*omega)+6*Kn*omega*tj*cos(2*t0*omega)+9*Kt*cos(2*t0*o
mega)+9*Kn*sin(2*t0*omega))/tj^2/omega^4;
  1/48*b*(2*Kt*omega^4*tj^4-
12*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+12*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+6*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+6*Kn*sin(2*t0*omega+2
*omega*tj)*omega^2*tj^2-9*Kt*cos(2*t0*omega+2*omega*tj)-
9*Kn*sin(2*t0*omega+2*omega*tj)-
6*Kt*omega*tj*sin(2*t0*omega)+6*Kn*omega*tj*cos(2*t0*omega)+9*Kt*cos(2*t0*o
mega)+9*Kn*sin(2*t0*omega))/tj^2/omega^4, 1/48*b*(-
12*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-9*Kn*cos(2*t0*omega+2*omega*tj)-
2*Kn*omega^4*tj^4-6*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
12*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+6*Kn*cos(2*t0*omega+2*omega*tj)*o
mega^2*tj^2+9*Kt*sin(2*t0*omega+2*omega*tj)+9*Kn*cos(2*t0*omega)-
9*Kt*sin(2*t0*omega)-6*Kn*omega*tj*sin(2*t0*omega)-
6*Kt*omega*tj*cos(2*t0*omega))/tj^2/omega^4];
P21 =[ -1/80*b*(-15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
15*Kn*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+60*Kt*cos(2*t0*omega+2*omega
*tj)+60*Kn*sin(2*t0*omega+2*omega*tj)+4*Kn*omega^5*tj^5+60*Kt*omega*tj*sin(2
*t0*omega)-
10*Kn*omega^3*tj^3*cos(2*t0*omega)+15*Kn*omega^2*tj^2*sin(2*t0*omega)+15*K
t*omega^2*tj^2*cos(2*t0*omega)+10*Kt*omega^3*tj^3*sin(2*t0*omega)-
60*Kn*omega*tj*cos(2*t0*omega)+10*Kn*sin(2*t0*omega)*omega^4*tj^4+10*Kt*co
s(2*t0*omega)*omega^4*tj^4-60*Kn*sin(2*t0*omega)-
60*Kt*cos(2*t0*omega)+60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
60*Kn*omega*tj*cos(2*t0*omega+2*omega*tj))/omega^5/tj^4, -1/80*b*(-
60*Kn*cos(2*t0*omega)+60*Kt*sin(2*t0*omega)-
15*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kt*sin(2*t0*omega+2*omega
*tj)*omega^2*tj^2+60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-
60*Kt*sin(2*t0*omega+2*omega*tj)+60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+
60*Kn*cos(2*t0*omega+2*omega*tj)+10*Kt*omega^3*tj^3*cos(2*t0*omega)-
15*Kt*omega^2*tj^2*sin(2*t0*omega)-
10*Kt*sin(2*t0*omega)*omega^4*tj^4+10*Kn*cos(2*t0*omega)*omega^4*tj^4+15*K
n*omega^2*tj^2*cos(2*t0*omega)+60*Kt*omega*tj*cos(2*t0*omega)+10*Kn*omega^
3*tj^3*sin(2*t0*omega)+60*Kn*omega*tj*sin(2*t0*omega)+4*Kt*omega^5*tj^5)/ome
ga^5/tj^4;
  1/80*b*(60*Kn*cos(2*t0*omega)-
60*Kt*sin(2*t0*omega)+15*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
15*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kt*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-60*Kn*cos(2*t0*omega+2*omega*tj)-
10*Kt*omega^3*tj^3*cos(2*t0*omega)+15*Kt*omega^2*tj^2*sin(2*t0*omega)+10*Kt
*sin(2*t0*omega)*omega^4*tj^4-10*Kn*cos(2*t0*omega)*omega^4*tj^4-
15*Kn*omega^2*tj^2*cos(2*t0*omega)-60*Kt*omega*tj*cos(2*t0*omega)-
10*Kn*omega^3*tj^3*sin(2*t0*omega)-
60*Kn*omega*tj*sin(2*t0*omega)+4*Kt*omega^5*tj^5)/omega^5/tj^4, -1/80*b*(-
60*Kt*omega*tj*sin(2*t0*omega)+60*Kn*omega*tj*cos(2*t0*omega)-
```

```
10*Kn*sin(2*t0*omega)*omega^4*tj^4-10*Kt*cos(2*t0*omega)*omega^4*tj^4-
15*Kt*omega^2*tj^2*cos(2*t0*omega)-
10*Kt*omega^3*tj^3*sin(2*t0*omega)+10*Kn*omega^3*tj^3*cos(2*t0*omega)-
15*Kn*omega^2*tj^2*sin(2*t0*omega)+60*Kn*sin(2*t0*omega)+60*Kt*cos(2*t0*ome
ga)-
60*Kt*cos(2*t0*omega+2*omega*tj)+4*Kn*omega^5*tj^5+60*Kn*omega*tj*cos(2*t0
*omega+2*omega*tj)+15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*sin
(2*t0*omega+2*omega*tj)*omega^2*tj^2-60*Kn*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj))/omega^5/tj^4];
P22 = [ 1/480*b*(-2*Kn*omega^5*tj^5-180*Kt*cos(2*t0*omega+2*omega*tj)-
180*Kn*sin(2*t0*omega+2*omega*tj)-
135*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*cos(2*t0*omega+2
*omega*tj)+30*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+30*Kn*sin(2*t0*ome
ga+2*omega*tj)*omega^2*tj^2-
225*Kt*omega*tj*sin(2*t0*omega)+180*Kn*sin(2*t0*omega)+180*Kt*cos(2*t0*omeg
a)+225*Kn*omega*tj*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*cos(2*t0*omega)+30*Kt*omega^3*tj^3*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*cos(2*t0*omega)-
120*Kn*omega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5, -
1/480*b*(135*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*sin(2*t0
*omega+2*omega*tj)-
180*Kt*sin(2*t0*omega+2*omega*tj)+180*Kn*cos(2*t0*omega+2*omega*tj)+2*Kt*o
mega^5*tj^5+30*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kt*omega^3*tj^3*cos(2*t0*omega)+120*Kn*omega^2*tj^2*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*sin(2*t0*omega)+225*Kt*omega*tj*cos(2*t0*omega)+225*Kn*
omega*tj*sin(2*t0*omega)+180*Kt*sin(2*t0*omega)-
180*Kn*cos(2*t0*omega))/tj^3/omega^5;
 -
1/480*b*(135*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*sin(2*t0
*omega+2*omega*tj)-
180*Kt*sin(2*t0*omega+2*omega*tj)+180*Kn*cos(2*t0*omega+2*omega*tj)-
2*Kt*omega^5*tj^5+30*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kt*omega^3*tj^3*cos(2*t0*omega)+120*Kn*omega^2*tj^2*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*sin(2*t0*omega)+225*Kt*omega*tj*cos(2*t0*omega)+225*Kn*
omega*tj*sin(2*t0*omega)+180*Kt*sin(2*t0*omega)-
180*Kn*cos(2*t0*omega))/tj^3/omega^5, -1/480*b*(2*Kn*omega^5*tj^5-
180*Kt*cos(2*t0*omega+2*omega*tj)-180*Kn*sin(2*t0*omega+2*omega*tj)-
135*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*cos(2*t0*omega+2
*omega*tj)+30*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+30*Kn*sin(2*t0*ome
ga+2*omega*tj)*omega^2*tj^2-
225*Kt*omega*tj*sin(2*t0*omega)+180*Kn*sin(2*t0*omega)+180*Kt*cos(2*t0*omeg
a)+225*Kn*omega*tj*cos(2*t0*omega)-
```

```
120*Kt*omega^2*tj^2*cos(2*t0*omega)+30*Kt*omega^3*tj^3*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*cos(2*t0*omega)-
120*Kn*omega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5];
P23 = [ -1/80*b*(-4*Kn*omega^5*tj^5-60*Kt*cos(2*t0*omega+2*omega*tj)-
60*Kn*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*sin(2*t0*omega
+2*omega*tj)*omega^2*tj^2-
10*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*sin(2*t0*o
mega+2*omega*tj)+10*Kt*omega^4*tj^4*cos(2*t0*omega+2*omega*tj)+10*Kn*omeg
a^3*tj^3*cos(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega)+60*Kn*sin(2*t0*omega)+60*Kt*cos(2*t0*omega)+
60*Kn*omega*tj*cos(2*t0*omega)-15*Kt*omega^2*tj^2*cos(2*t0*omega)-
15*Kn*omega^2*tj^2*sin(2*t0*omega))/omega^5/tj^4, -1/80*b*(-
60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kt*sin(2*t0*omega+2*omega*tj)-
60*Kn*cos(2*t0*omega+2*omega*tj)-
10*Kt*omega^4*tj^4*sin(2*t0*omega+2*omega*tj)-
10*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*cos(2*t0*o
mega+2*omega*tj)-10*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
4*Kt*omega^5*tj^5-
15*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*cos(2*t0*omega+2*omega
*tj)*omega^2*tj^2-
15*Kn*omega^2*tj^2*cos(2*t0*omega)+15*Kt*omega^2*tj^2*sin(2*t0*omega)-
60*Kt*omega*tj*cos(2*t0*omega)-60*Kn*omega*tj*sin(2*t0*omega)-
60*Kt*sin(2*t0*omega)+60*Kn*cos(2*t0*omega))/omega^5/tj^4;
 -1/80*b*(-60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kt*sin(2*t0*omega+2*omega*tj)-
60*Kn*cos(2*t0*omega+2*omega*tj)-
10*Kt*omega^4*tj^4*sin(2*t0*omega+2*omega*tj)-
10*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*cos(2*t0*o
mega+2*omega*tj)-
10*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+4*Kt*omega^5*tj^5-
15*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*cos(2*t0*omega+2*omega
*tj)*omega^2*tj^2-
15*Kn*omega^2*tj^2*cos(2*t0*omega)+15*Kt*omega^2*tj^2*sin(2*t0*omega)-
60*Kt*omega*tj*cos(2*t0*omega)-60*Kn*omega*tj*sin(2*t0*omega)-
60*Kt*sin(2*t0*omega)+60*Kn*cos(2*t0*omega))/omega^5/tj^4,
1/80*b*(4*Kn*omega^5*tj^5-60*Kt*cos(2*t0*omega+2*omega*tj)-
60*Kn*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*sin(2*t0*omega
+2*omega*tj)*omega^2*tj^2-
10*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*sin(2*t0*o
mega+2*omega*tj)+10*Kt*omega^4*tj^4*cos(2*t0*omega+2*omega*tj)+10*Kn*omeg
a^3*tj^3*cos(2*t0*omega+2*omega*tj)-
```

```
60*Kt*omega*tj*sin(2*t0*omega)+60*Kn*sin(2*t0*omega)+60*Kt*cos(2*t0*omega)+
60*Kn*omega*tj*cos(2*t0*omega)-15*Kt*omega^2*tj^2*cos(2*t0*omega)-
15*Kn*omega^2*tj^2*sin(2*t0*omega))/omega^5/tj^4];
P24 = [ -
1/480*b*(2*Kn*omega^5*tj^5+180*Kt*cos(2*t0*omega+2*omega*tj)+180*Kn*sin(2*t
0*omega+2*omega*tj)+225*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
225*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)-
120*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
120*Kn*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+30*Kn*omega^3*tj^3*cos(2*t0*o
mega+2*omega*tj)+135*Kt*omega*tj*sin(2*t0*omega)-180*Kn*sin(2*t0*omega)-
180*Kt*cos(2*t0*omega)-
135*Kn*omega*tj*cos(2*t0*omega)+30*Kt*omega^2*tj^2*cos(2*t0*omega)+30*Kn*o
mega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5, -
1/480*b*(225*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+225*Kn*omega*tj*sin(2*t0
*omega+2*omega*tj)-
180*Kt*sin(2*t0*omega+2*omega*tj)+180*Kn*cos(2*t0*omega+2*omega*tj)-
30*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)-
30*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^5*tj^5+120*Kt*sin(
2*t0*omega+2*omega*tj)*omega^2*tj^2-
120*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+30*Kn*omega^2*tj^2*cos(2*t0*
omega)-
30*Kt*omega^2*tj^2*sin(2*t0*omega)+135*Kt*omega*tj*cos(2*t0*omega)+135*Kn*o
mega*tj*sin(2*t0*omega)+180*Kt*sin(2*t0*omega)-
180*Kn*cos(2*t0*omega))/tj^3/omega^5;
 1/480*b*(-225*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
225*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+180*Kt*sin(2*t0*omega+2*omega*tj
)-
180*Kn*cos(2*t0*omega+2*omega*tj)+30*Kt*omega^3*tj^3*cos(2*t0*omega+2*ome
ga*tj)+30*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^5*tj^5-
120*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+120*Kn*cos(2*t0*omega+2*ome
ga*tj)*omega^2*tj^2-
30*Kn*omega^2*tj^2*cos(2*t0*omega)+30*Kt*omega^2*tj^2*sin(2*t0*omega)-
135*Kt*omega*tj*cos(2*t0*omega)-135*Kn*omega*tj*sin(2*t0*omega)-
180*Kt*sin(2*t0*omega)+180*Kn*cos(2*t0*omega))/tj^3/omega^5, -
1/480*b*(2*Kn*omega^5*tj^5-180*Kt*cos(2*t0*omega+2*omega*tj)-
180*Kn*sin(2*t0*omega+2*omega*tj)-
225*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+225*Kn*omega*tj*cos(2*t0*omega+2
*omega*tj)+120*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+120*Kn*sin(2*t0*o
mega+2*omega*tj)*omega^2*tj^2+30*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)
-30*Kn*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)-
135*Kt*omega*tj*sin(2*t0*omega)+180*Kn*sin(2*t0*omega)+180*Kt*cos(2*t0*omeg
a)+135*Kn*omega*tj*cos(2*t0*omega)-30*Kt*omega^2*tj^2*cos(2*t0*omega)-
30*Kn*omega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5];
```

```
P11 = [P11(1,1)*ones(lmx,1) P11(1,2)*ones(lmx,1); P11(2,1)*ones(lmx,1)
P11(2,2)*ones(lmx,1)]*V;
P12 = [P12(1,1)*ones(lmx,1) P12(1,2)*ones(lmx,1); P12(2,1)*ones(lmx,1)
P12(2,2)*ones(lmx,1)]*V;
P13 = [P13(1,1)*ones(lmx,1) P13(1,2)*ones(lmx,1); P13(2,1)*ones(lmx,1)
P13(2,2)*ones(lmx,1)]*V;
P14 = [P14(1,1)*ones(lmx,1) P14(1,2)*ones(lmx,1); P14(2,1)*ones(lmx,1)
P14(2,2)*ones(lmx,1)]*V;
P21 = [P21(1,1)*ones(lmx,1) P21(1,2)*ones(lmx,1); P21(2,1)*ones(lmx,1)
P21(2,2)*ones(lmx,1)]*V;
P22 = [P22(1,1)*ones(lmx,1) P22(1,2)*ones(lmx,1); P22(2,1)*ones(lmx,1)
P22(2,2)*ones(lmx,1)]*V;
P23 = [P23(1,1)*ones(lmx,1) P23(1,2)*ones(lmx,1); P23(2,1)*ones(lmx,1)
P23(2,2)*ones(lmx,1)]*V;
P24 = [P24(1,1)*ones(lmx,1) P24(1,2)*ones(lmx,1); P24(2,1)*ones(lmx,1)
P24(2,2)*ones(lmx,1)]*V;
N11 = -C+1/2*K*tj+P11;
N12 = -M+1/12*K*tj^2+P12;
N13 = C+1/2*K*tj+P13;
N14 = M-1/12*K*tj^2+P14;
N21 = M/tj-1/10*K*tj+P21;
N22 = 1/2*M-1/12*C*tj-1/120*K*tj^2+P22;
N23 = -M/tj+1/10*K*tj+P23;
N24 = 1/2*M+1/12*C*tj-1/120*K*tj^2+P24;
   N1  = [N11 N12; N21 N22];
   N2  = [N13 N14; N23 N24];
   P1  = [P11 P12; P21 P22];
   P2  = [P13 P14; P23 P24];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
   % BUILD GLOBAL MATRICES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
   A(1:2*DOF,1:2*DOF) = eye(2*DOF);
   A(2*DOF*e+1:2*DOF*e+2*DOF,2*DOF*(e-1)+1:2*DOF*(e-1)+2*DOF) = N1;
   A(2*DOF*e+1:2*DOF*e+2*DOF,2*DOF*(e-1)+2*DOF+1:2*DOF*(e-
1)+2*DOF+2*DOF) = N2;
   B(2*DOF*e+1:2*DOF*e+2*DOF,2*DOF*(e-1)+1:2*DOF*(e-1)+2*DOF) = P1;
   B(2*DOF*e+1:2*DOF*e+2*DOF,2*DOF*(e-1)+2*DOF+1:2*DOF*(e-
1)+2*DOF+2*DOF) = P2;
   B(1:2*(DOF),E*2*(DOF)+1:(E+1)*2*(DOF)) = PHI;
   Cvec(1:2*DOF,1) = zeros(2*DOF,1);
   Cvec(2*DOF*e+1:2*DOF*e+DOF,1) = C1;
   Cvec(2*DOF*e+DOF+1:2*DOF*e+2*DOF,1) = C2;
end; % end # of elements loop
         Q          = A\B;
```

```
        [vec,lam]   = eig(Q);
                        CM          = max(abs(diag(lam)));
                        D           = A\Cvec;
                        % Extract SLE coefficients

                                    if CM<1
                    SLE_vec     = inv((eye(size(Q))-Q))*D;
                    SLE         = abs(sum(SLE_vec(locat)));
                                    else
                    SLE         = 100;
                                    end
        N1 = zeros(2*DOF,2*DOF); N2 = N1;
                        P1 = N1; P2 = P1;
return;
```

Uncertainty Analysis Code

Stability Uncertainty, Sensitivity Method

```
% M. Kurdi (1/26/2005)
% Function to find uncertainty in axial depth to change in cutting
% coefficients, dynamic parameters, and cutting process variables
% Input:
%       b:      depth of cut (m)
%       rpm:    spindle speed
%       rstep:  radial step (inches)
%       Kt
%       Kn
%       Kte
%       Kre
%       DELTA_Kt    finite change in Kt
%       DELTA_b     finite change in b
%       system_ID:  Modal parameters
% The derivative of Max eigen value is found for a miniscule perturbation
% in input parameters, then its effect on the change of axial depth is
% found.
clear all; close all; clc;tic;
% function uncer
% percentage of uncertainty in cutting coefficients, dynamic parameters
% and process parameters
% tic;
percent_Kcut = 0.05; % cutting coefficents uncertainty
percent_Dyn = 0.005; % modal parameters uncertainty
percent_rstep = 0.0001; % radial step uncertainty
percent_rpm = 0.005; % spindle speed uncertainty
% nominal values of process parameters and their calculated uncertainty
rstep = 0.2;
```

```
rpm_vec = 10000:200:30000;
DELTA_rstep = percent_rstep*rstep;
% cutting coefficient uncertainty
Kt = 6e8;
DELTA_Kt = percent_Kcut*Kt;
Kn = .3*Kt;
DELTA_Kn = percent_Kcut*Kn;
Kte=0;
DELTA_Kte = percent_Kcut*Kte;
Kne=0;
DELTA_Kne = percent_Kcut*Kne;
% nominal values of dynamic parameters and their calculated uncertainty
Kx =  4.4528e+006;
Mx = 0.4362;
Cx = 83;
% Y direction parameters
Ky = 3.5542e+006;
My = 0.4362;
Cy = 89.9;
DELTA_Mx = Mx*percent_Dyn;
DELTA_My = My*percent_Dyn;
DELTA_Kx = Kx*percent_Dyn;
DELTA_Ky = Ky*percent_Dyn;
DELTA_Cx = Cx*percent_Dyn;
DELTA_Cy = Cy*percent_Dyn;
% to calculate the numerical derivative with respect to each input
% variable set the miniscule change in each input
% set miniscule change in input parameters to estimate the derivative
step_percent = 0.002;
dKt = step_percent*Kt; % N/m2
dKn = step_percent*Kn; % N/m2
dKte = step_percent*30; % N/m
dKne = step_percent*30; % N/m
drstep = step_percent*rstep; % inch
dKx = step_percent*Kx; % N/m
dKy = step_percent*Ky; % N/m
dCx = step_percent*Cx; %
dCy = step_percent*Cy;
dMx = step_percent*Mx; % Kg
dMy = step_percent*My; % Kg
h = waitbar(0,'Please wait...');
% computation here %
for i=1:length(rpm_vec)
   waitbar(i/length(rpm_vec),h);
   rpm = rpm_vec(i);
   drpm = step_percent*rpm; % rpm
```

```
    DELTA_rpm = percent_rpm * rpm;
    % Find depth of cut corresponding to stability boundary using nominal
    % settings of input parameters
    [b(i)] = bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy); % depth at
          boundary
    % Find numerical derivative of maximum eigenvalue with respect to input
    % parameters
    % perturb cutting coefficient Kt by dKt

    [b1] = bisection(rpm,rstep,Kt-dKt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
    [b2] = bisection(rpm,rstep,Kt+dKt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
    % derivative of eigen matrix w.r.t cutting coefficient Kt
    d_b_Kt(i) = (b2-b1)/dKt/2;
    b1 =[]; b2 =[];
    % perturb cutting coefficient Kn by dKn
    [b1] = bisection(rpm,rstep,Kt,Kn-dKn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
    [b2] = bisection(rpm,rstep,Kt,Kn+dKn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
%
%     % derivative of eigen matrix w.r.t cutting coefficient Kt
    d_b_Kn(i) = (b2-b1)/dKn/2;
    b1 =[]; b2 =[];
    % perturb cutting coefficient Kte by dKte
    [b1] = bisection(rpm,rstep,Kt,Kn,Kte-dKte,Kne,Mx,Kx,Cx,My,Ky,Cy);
    [b2] = bisection(rpm,rstep,Kt,Kn,Kte+dKte,Kne,Mx,Kx,Cx,My,Ky,Cy);
    % derivative of eigen matrix w.r.t cutting coefficient Kt
    d_b_Kte(i) = (b2-b1)/dKte/2;
    b1 =[]; b2 =[];
    % perturb cutting coefficient Kne by dKne
    [b1] = bisection(rpm,rstep,Kt,Kn,Kte,Kne-dKne,Mx,Kx,Cx,My,Ky,Cy);
    [b2] = bisection(rpm,rstep,Kt,Kn,Kte,Kne+dKne,Mx,Kx,Cx,My,Ky,Cy);
   % derivative of eigen matrix w.r.t cutting coefficient Kne
    d_b_Kne(i) = (b2-b1)/dKne/2;
    b1 =[]; b2 =[];
%
    % perturb depth of cut rstep by drstep
    [b1] = bisection(rpm,rstep-drstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
    [b2] = bisection(rpm,rstep+drstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
    % derivative of eigen value w.r.t rstep of cut
    d_b_rstep(i) = (b2-b1)/drstep/2;
    b1 =[]; b2 =[];
    % perturb spindle speed by drpm
    [b1] = bisection(rpm-drpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
    [b2] = bisection(rpm+drpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
    % derivative of eigen value w.r.t rpm
    d_b_rpm(i) = (b2-b1)/drpm/2;
    b1 =[]; b2 =[];
```

```
%      perturb Kx by dKx
   [b1] = bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx-dKx,Cx,My,Ky,Cy);
   [b2] = bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx+dKx,Cx,My,Ky,Cy);
%     derivative of eigen value w.r.t Kx
   d_b_Kx(i) = (b2-b1)/dKx/2;
   b1 =[]; b2 =[];
    % perturb Ky by dKy
   [b1] = bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky-dKy,Cy);
   [b2] = bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky+dKy,Cy);
%
   % derivative of eigen value w.r.t Ky
   d_b_Ky(i) = (b2-b1)/dKy/2;
   b1 =[]; b2 =[];
    % perturb Cx by dCx
   [b1] = bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx-dCx,My,Ky,Cy);
   [b2] = bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx+dCx,My,Ky,Cy);
   % derivative of eigen value w.r.t Cx
   d_b_Cx(i) = (b2-b1)/dCx/2;
   b1 =[]; b2 =[];
   % perturb Cy by dCy
   [b1] = bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy-dCy);
   [b2] = bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy+dCy);
   % derivative of eigen value w.r.t Cy
   d_b_Cy(i) = (b2-b1)/dCy/2;
   b1 =[]; b2 =[];
     % perturb Mx by dMx
   [b1] = bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx-dMx,Kx,Cx,My,Ky,Cy);
   [b2] = bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx+dMx,Kx,Cx,My,Ky,Cy);
   % derivative of eigen value w.r.t Mx
   d_b_Mx(i) = (b2-b1)/dMx/2;
   b1 =[]; b2 =[];
    % perturb My by dMy
   [b1] = bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My-dMy,Ky,Cy);
   [b2] = bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My+dMy,Ky,Cy);
%
%     % derivative of eigen value w.r.t My
   d_b_My(i) = (b2-b1)/dMy/2;
   b1 =[]; b2 =[];
%
DELTA_b(i) =( (DELTA_Kt * d_b_Kt(i))^2 + (DELTA_Kn * d_b_Kn(i))^2 + ...
         (DELTA_Kne * d_b_Kne(i))^2 + (DELTA_Kte * d_b_Kte(i))^2 +...
         (DELTA_Kx * d_b_Kx(i))^2 + (DELTA_Mx * d_b_Mx(i))^2 + ...
          (DELTA_Cx * d_b_Cx(i))^2 + (DELTA_Ky * d_b_Ky(i))^2 +...
           (DELTA_My * d_b_My(i))^2 + (DELTA_Cy * d_b_Cy(i))^2+...
           (DELTA_rstep * d_b_rstep(i))^2 + (DELTA_rpm* d_b_rpm(i))^2)^0.5;
end
```

```
close(h);
% d_b_rpm
% % Find the uncertainty in depth of cut for a corresponding uncertainty in
% % input paramters
%
% save uncer_march_10_smallconverror
figure
plot(rpm_vec*1/60/(sqrt(Ky/My)/2/pi),b*1000,'-g')
%
% set(gca,'fontname','times','fontsize',16);
% xlabel('\Omega (x10^3 rpm)','fontsize',14)
% ylabel('b (mm)','fontsize',14)
% legend('Stability boundary, nominal input','Stability boundary uncertainty');
% axis([5 20 0 15])
```

Stability Uncertainty, Monte Carlo and Latin Hyper-Cube

```
%
% M. Kurdi (6/17/05)
% 4 OAL TOOL
% Program to complete LatinHyper and Monte simulation for TFEA stability lobes
% clear all; close all;
% function LatinHyper
% tic;
% chip_load=0.1e-3;% chip load
% nteeth = 4;
% Diam =0.5;
% E=15;
% N = 1000; % number of iterations
%
%
% % AL 6061
% % percent_Kt = 7.13/100; % cutting coefficents uncertainty
% % percent_Kn = 8.09/100;
% % percent_Kte = 30.3/100;
% % percent_Kne = 23.9/100;
%
% % 5 OAL TOOL UNCERTAINTIES
% % percent_KX = 0.054; % modal parameters uncertainty
% % percent_CX = .286;
% % percent_MX =.045;
% %
% % percent_KY = 0.054; % modal parameters uncertainty
% % percent_CY = .173;
% % percent_MY =.055;
% % 4 OAL TOOL UNCERTAINTIES due to thermal effect only
```

```
%  percent_MX = 0.074;
%  percent_CX = 0.042;
%  percent_KX = 0.073 ;
%  percent_MY = 0.2;
%  percent_CY = 0.107;
%  percent_KY = 0.2 ;
% percent_rstep = 0.0005; % radial step uncertainty
% percent_rpm = 0.005; % spindle speed uncertainty
% % speed_min = str2num(input('Min_speed = ','s'));
% % speed_max = str2num(input('Max_speed = ','s'));
% % speed = speed_min:200:speed_max;
% speed = 10000:100:20000;
% h = waitbar(0,'Please wait...');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%5
% % Cutting Coefficients %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
% % AL 6061
% % mean_Kt =7.06E+08; % N/m2
% % mean_Kn = 2.50E+08;
% % mean_Kte = 1.29E+04; % N/m;
% % mean_Kne = 6.57E+03;
% % AL 7475
% mean_Kt =  690480868.527357;
% mean_Kte = 12022.3004909002;
% mean_Kn = 142535991.092323;
% mean_Kne =11281.4601645315;
% std_Kn=4009843*4.45; % N
% std_Kne=310.909*4.45;
% std_Kte=200.731*4.45;
% std_Kt=2588583*4.45;
% % std_Kt = percent_Kt*mean_Kt;
% % std_Kn = percent_Kn*mean_Kn;
% % std_Kte = percent_Kte*mean_Kte;
% % std_Kne = percent_Kne*mean_Kne;
% % Kne Kn Kte Kt
% % AL  6061
% % SIGMA_K = [1.480E+07         -1.778E+11     -8.216E+06     9.871E+10;
% %           -1.778E+11        2.458E+15      9.871E+10      -1.365E+15;
% %           -8.216E+06        9.871E+10      9.163E+07      -1.101E+12;
% %           9.871E+10         -1.365E+15     -1.101E+12     1.522E+16
% %           ];
%
% % AL 7475
```

```
%   SIGMA_K = [ 42157610.7365206        -506483170409.775         -
3598978.12573119        43238262783.6325;
%        -506483170409.775    7.00379474676691e+015        43238262783.6325
-597911116174549;
%        -3598978.12573128        43238262783.6335        17574719.1179838         -
211143357093.21;
%         43238262783.6335        -597911116174562        -211143357093.21
2.91975098408051e+015];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
% % Modal Parameters
% % X %%%%%%%%%%%%%%%%
% % 5 OAL TOOL
% % mean_Kx =  2.64E+06;
% % mean_Mx = 0.049;
% % mean_Cx = 8.972;
%  % dynamic parameters for 4 OAL tool
%         mean_Mx = 0.027 ;
%         mean_Cx=  23.309;
%         mean_Kx= 4359275.000 ;
% std_Cx = percent_CX*mean_Cx;
% std_Kx = percent_KX*mean_Kx;
% std_Mx = percent_MX*mean_Mx;
% % Mx Cx Kx My Cy Ky 5 OAL
% % SIGMA = [3.85E-06      4.03E-03        2.48E+02        2.40E-06        -3.18E-03
        1.31E+02;
% %        4.03E-03    5.27E+00        2.69E+05        4.08E-03        -3.32E+00
        2.19E+05;
% %        2.48E+02    2.69E+05        1.61E+10        1.67E+02        -2.07E+05
        9.15E+09;
% %        2.40E-06    4.08E-03        1.67E+02        4.23E-06        -1.88E-03
        2.24E+02;
% %        -3.18E-03   -3.32E+00       -2.07E+05       -1.88E-03        2.71E+00        -
1.04E+05;
% %        1.31E+02    2.19E+05        9.15E+09        2.24E+02        -1.04E+05
        1.19E+10
% %           ];
% % Mx Cx Kx My Cy Ky 4 OAL
% SIGMA = [4.04188E-06    0.000450265   631.110625    7.25563E-06   -0.000584252
        878.998125;
% 0.000450265         0.953490935   38828.325      0.00283473    -2.467636648
        567721.5525;
% 631.110625 38828.325       1.00042E+11   1068.011875   -51720.94813  1.21332E+11;
% 7.25563E-06         0.00283473     1068.011875   1.76519E-05   -0.007067488
        2638.261875;
```

```
% -0.000584252         -2.467636648  -51720.94813  -0.007067488  11.34481701   -
1426512.396;
% 878.998125 567721.5525   1.21332E+11   2638.261875   -1426512.396
          4.36003E+11];
% % % Y %%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % 5 OAL TOOL
% % mean_Ky = 2.26e+006;
% % mean_Cy = 10.651;
% % mean_My = 0.042;
% % Y direction parameters 4 OAL TOOL
% mean_Ky = 3301775.000;
% mean_My =  0.021;
% mean_Cy = 31.432;
% std_My = percent_MY*mean_My;
% std_Ky = percent_KY*mean_Ky;
% std_Cy = percent_CY*mean_Cy;
% % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % % Radial step inches
% % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mean_rstep = 0.25*.5;
% std_rstep = percent_rstep*mean_rstep;
% % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% randn('state',0)
% Mode = lhsnorm([mean_Mx mean_Cx mean_Kx mean_My mean_Cy
mean_Ky],SIGMA,N);
% % Mode(:,1) is Mx random vector
% % Mode(:,2) is Cx random vector
% % Mode(:,3) is Kx random vector
% % Mode(:,4) is My random vector
% % Mode(:,5) is Cy random vector
% % Mode(:,6) is Ky random vector
% Cut_Coeff = lhsnorm([mean_Kne mean_Kn mean_Kte mean_Kt],SIGMA_K,N);
% % Cut_Coeff(:,1) Kne
% % Cut_Coeff(:,2) Kn
% % Cut_Coeff(:,3) Kte
% % Cut_Coeff(:,4) Kt
% sample = randn(N, 2);
% for j=1:length(speed)
% waitbar(j/length(speed),h)
% for i=1:N
% % Unless otherwise specified, all dimensions in m
% % Define input parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Cutting coefficients
% Kt = Cut_Coeff(i,4);
```

```
% Kn = Cut_Coeff(i,2);
% Kte = Cut_Coeff(i,3);
% Kne = Cut_Coeff(i,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % milling parameters
% % Spindle speed
% mean_rpm = speed(j);
% std_rpm = percent_rpm*mean_rpm;
% rpm = mean_rpm + std_rpm*sample(i,1);
% % rstep
% rstep = mean_rstep + std_rstep*sample(i,2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Dynamic parameters
% % X direction is feed direction
% Kx =Mode(i,3);
% Mx = Mode(i,1);
% Cx = Mode(i,2);
% % Y direction parameters
% Ky = Mode(i,6);
% My = Mode(i,4);
% Cy = Mode(i,5);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Calculate axial depth corresponding to input paramters
% % that is on the stability boundaries
% b(i,j) =
bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx,Cx,Kx,My,Cy,Ky,chip_load,nteeth,Diam,E);
% end % i end monte loop for one spindle speed
% end % j end spindle speed range
% std_dev = std(b)
% b_mean = mean(b)
% time=toc;
% save Latin_AL7475 std_dev speed b_mean b time
% close(h);
% hold on;
% h1 = plot(speed/1000,(mean(b)-2*std(b))*1000,'-r')
% hold on;
% h2 = plot(speed/1000,mean(b)*1000,'g-');
% hold on;
% h3 = plot(speed/1000,(2*std(b)+mean(b))*1000,'-r');
% legend([h1,h2,h3],'lower boundary','mean','upper boundary')
figure
plot(speed/1000,std(b)*2*1000)
%
```

```
% Function to stability lobe using bisection method.
% Input:
%       rpm ;
%       rstep:      radial immersion (inches)
%   Output:
%   b   depth of cut (m)
function [b] =
bisection(rpm,rstep,Kt,Kn,Kte,Kne,Mx,Cx,Kx,My,Cy,Ky,h,nteeth,Diam,E)
% E
%    h                          % feed per tooth
%       nteeth                              % number of teeth
%       Diam                                % inches
      TRAVang     = acos(1-rstep/(Diam/2));             % angular travel during cutting
      LAGang      = 2*pi/nteeth;                        % separation angle for teeth
      rho         = acos(1-rstep/(Diam/2))/(2*pi);      % fraction of time in cut
      IMMERSION   = rstep/Diam;
      opt         = 'down';
      if TRAVang>LAGang   % MULTIPLE TEETH ARE IN CONTACT
    teethNcontact = floor(TRAVang/LAGang) +1;
      else              % SINGLE TOOTH IN CONTACT
    teethNcontact = 1;
      end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %       LOAD SYSTEM IDENTIFICATION MATRICES
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      M   =[Mx zeros(size(Mx)); zeros(size(Mx)) My];
      C   =[Cx zeros(size(Mx)); zeros(size(Mx)) Cy];
      K   =[Kx zeros(size(Mx)); zeros(size(Mx)) Ky];
      lmx = length(Mx(1,:));
      lmy = length(My(1,:));
      DOF = lmx+lmy;
      V       = [ones(1,lmx) zeros(1,lmy); zeros(1,lmx) ones(1,lmy)];
      A       = zeros((E+1)*2*DOF,(E+1)*2*DOF);
      B       = A;
      b_r = [1e-10 100e-2];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %       BEGIN LOOP CALCULATIONS OVER RPM vs DOC FIELD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            while (abs(b_r(1) - b_r(2)) / b_r(1) > 1e-6)
    warning off MATLAB:singularMatrix;
    warning off MATLAB:nearlySingularMatrix;
```

```
                b = (b_r(1) + b_r(2)) / 2;
                speed   = rpm;
                omega   = speed/60*(2*pi);            % radians per second
                T       = (2*pi)/omega/nteeth;        % tooth pass period
                TC      = rho*T*nteeth;               % time a single tooth spends in the cut
                tf      = T-TC;                       % time for free vibs
                tj      = TC/E;                       % time for each element
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %       SET CUTTER ROTATION ANGLE FOR UP/DOWN-MILLING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                switch opt
                case 'up'
                t0mat   = [0 tj*(1:(E-1))];           % upmilling
                locat   = 2*DOF+lmx+1:3*DOF;
                case 'down'
                tex     = pi/omega; tent=tex-TC;      % downmilling
                t0mat   = [tent tent+tj*(1:(E-1))];   % downmilling
                locat   = (E+1)*2*DOF-DOF-lmy+1:(E+1)*2*DOF-DOF;
                end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %       STATE TRANSITION MATRIX
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                G1  = [zeros(size(M)) M; eye(size(M)) zeros(size(M))];
                G2  = [K  C; zeros(size(M)) -eye(size(M))];
                G   = -G1\G2;
                PHI = expm(G*tf);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %       N & P are used to create A & B which then become Q in..... a_n =
Q a_n-1 + D
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        for e=1:E,
 t0 = t0mat(e);
C1 = V'*[ -1/4*b*(-h*Kt*cos(2*t0*omega+2*omega*tj)+2*h*Kn*omega*tj-
h*Kn*sin(2*t0*omega+2*omega*tj)+4*Kte*sin(t0*omega+omega*tj)-
4*Kne*cos(t0*omega+omega*tj)+h*Kt*cos(2*t0*omega)+h*Kn*sin(2*t0*omega)-
4*Kte*sin(t0*omega)+4*Kne*cos(t0*omega))/omega;
  1/4*b*(2*h*Kt*omega*tj-
h*Kt*sin(2*t0*omega+2*omega*tj)+h*Kn*cos(2*t0*omega+2*omega*tj)-
```

```
4*Kte*sin(t0*omega+omega*tj)+4*Kne*cos(t0*omega+omega*tj)+h*Kt*sin(2*t0*ome
ga)-h*Kn*cos(2*t0*omega)+4*Kte*sin(t0*omega)-4*Kne*cos(t0*omega))/omega];
 C2 = V'*[ 1/8*b*(-
h*Kt*sin(2*t0*omega+2*omega*tj)+h*Kt*cos(2*t0*omega+2*omega*tj)*omega*tj+h*
Kn*cos(2*t0*omega+2*omega*tj)+h*Kn*sin(2*t0*omega+2*omega*tj)*omega*tj+8*K
te*sin(t0*omega+omega*tj)-4*Kte*cos(t0*omega+omega*tj)*omega*tj-
8*Kne*cos(t0*omega+omega*tj)-
4*Kne*sin(t0*omega+omega*tj)*omega*tj+h*Kt*sin(2*t0*omega)-
h*Kn*cos(2*t0*omega)-
8*Kte*sin(t0*omega)+8*Kne*cos(t0*omega)+h*Kt*tj*cos(2*t0*omega)*omega+h*Kn*
tj*sin(2*t0*omega)*omega-4*Kte*tj*cos(t0*omega)*omega-
4*Kne*tj*sin(t0*omega)*omega)/tj/omega^2;
 -
1/8*b*(h*Kt*cos(2*t0*omega+2*omega*tj)+h*Kt*sin(2*t0*omega+2*omega*tj)*omeg
a*tj+h*Kn*sin(2*t0*omega+2*omega*tj)-
h*Kn*cos(2*t0*omega+2*omega*tj)*omega*tj-
8*Kte*sin(t0*omega+omega*tj)+4*Kte*cos(t0*omega+omega*tj)*omega*tj+8*Kne*co
s(t0*omega+omega*tj)+4*Kne*sin(t0*omega+omega*tj)*omega*tj-
h*Kt*cos(2*t0*omega)-h*Kn*sin(2*t0*omega)+8*Kte*sin(t0*omega)-
8*Kne*cos(t0*omega)+h*Kt*tj*sin(2*t0*omega)*omega-
h*Kn*tj*cos(2*t0*omega)*omega+4*Kte*tj*cos(t0*omega)*omega+4*Kne*tj*sin(t0*o
mega)*omega)/tj/omega^2];
 P11 = [  1/8*b*(-
3*Kt*sin(2*t0*omega+2*omega*tj)+3*Kn*cos(2*t0*omega+2*omega*tj)+2*Kn*omeg
a^4*tj^4+3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*sin(2*t0*ome
ga+2*omega*tj)+3*Kt*sin(2*t0*omega)-
3*Kn*cos(2*t0*omega)+2*Kt*cos(2*t0*omega)*omega^3*tj^3+2*Kn*sin(2*t0*omega)
*omega^3*tj^3+3*Kt*omega*tj*cos(2*t0*omega)+3*Kn*omega*tj*sin(2*t0*omega))/o
mega^4/tj^3,  1/8*b*(-3*Kt*cos(2*t0*omega+2*omega*tj)-
3*Kn*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^4*tj^4-
3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*cos(2*t0*omega+2*om
ega*tj)+3*Kt*cos(2*t0*omega)+3*Kn*sin(2*t0*omega)-
2*Kt*sin(2*t0*omega)*omega^3*tj^3+2*Kn*cos(2*t0*omega)*omega^3*tj^3-
3*Kt*omega*tj*sin(2*t0*omega)+3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3;
  1/8*b*(-3*Kt*cos(2*t0*omega+2*omega*tj)-3*Kn*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^4*tj^4-
3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*cos(2*t0*omega+2*om
ega*tj)+3*Kt*cos(2*t0*omega)+3*Kn*sin(2*t0*omega)-
2*Kt*sin(2*t0*omega)*omega^3*tj^3+2*Kn*cos(2*t0*omega)*omega^3*tj^3-
3*Kt*omega*tj*sin(2*t0*omega)+3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3, -
1/8*b*(-
2*Kn*omega^4*tj^4+3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*s
in(2*t0*omega+2*omega*tj)-
3*Kt*sin(2*t0*omega+2*omega*tj)+3*Kn*cos(2*t0*omega+2*omega*tj)+2*Kt*cos(2*
t0*omega)*omega^3*tj^3+2*Kn*sin(2*t0*omega)*omega^3*tj^3+3*Kt*omega*tj*cos(
```

```
2*t0*omega)+3*Kn*omega*tj*sin(2*t0*omega)+3*Kt*sin(2*t0*omega)-
3*Kn*cos(2*t0*omega))/omega^4/tj^3];
 P12 =[
1/48*b*(6*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+6*Kt*omega*tj*cos(2*t0*ome
ga+2*omega*tj)+2*Kn*omega^4*tj^4-
9*Kt*sin(2*t0*omega+2*omega*tj)+9*Kn*cos(2*t0*omega+2*omega*tj)+12*Kn*ome
ga*tj*sin(2*t0*omega)+12*Kt*omega*tj*cos(2*t0*omega)-
6*Kt*omega^2*tj^2*sin(2*t0*omega)+6*Kn*omega^2*tj^2*cos(2*t0*omega)+9*Kt*si
n(2*t0*omega)-9*Kn*cos(2*t0*omega))/tj^2/omega^4, -
1/48*b*(9*Kt*cos(2*t0*omega+2*omega*tj)+9*Kn*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^4*tj^4+6*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)+12*Kt*omega*tj*sin(2*t0*omega)+6*
Kn*omega^2*tj^2*sin(2*t0*omega)+6*Kt*omega^2*tj^2*cos(2*t0*omega)-
9*Kt*cos(2*t0*omega)-12*Kn*omega*tj*cos(2*t0*omega)-
9*Kn*sin(2*t0*omega))/tj^2/omega^4;
 -
1/48*b*(2*Kt*omega^4*tj^4+6*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+9*Kt*cos(
2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)+9*Kn*sin(2*t0*omega+2*omega*tj)+1
2*Kt*omega*tj*sin(2*t0*omega)+6*Kn*omega^2*tj^2*sin(2*t0*omega)+6*Kt*omega^
2*tj^2*cos(2*t0*omega)-9*Kt*cos(2*t0*omega)-12*Kn*omega*tj*cos(2*t0*omega)-
9*Kn*sin(2*t0*omega))/tj^2/omega^4, 1/48*b*(-
6*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+2*Kn*omega^4*tj^4-
9*Kn*cos(2*t0*omega+2*omega*tj)+9*Kt*sin(2*t0*omega+2*omega*tj)-
12*Kt*omega*tj*cos(2*t0*omega)-
12*Kn*omega*tj*sin(2*t0*omega)+6*Kt*omega^2*tj^2*sin(2*t0*omega)-
6*Kn*omega^2*tj^2*cos(2*t0*omega)+9*Kn*cos(2*t0*omega)-
9*Kt*sin(2*t0*omega))/tj^2/omega^4];
P13 =[  1/8*b*(2*Kn*omega^4*tj^4-
3*Kn*cos(2*t0*omega+2*omega*tj)+3*Kt*sin(2*t0*omega+2*omega*tj)-
3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
2*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
3*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+3*Kn*cos(2*t0*omega)-
3*Kt*sin(2*t0*omega)-3*Kt*omega*tj*cos(2*t0*omega)-
3*Kn*omega*tj*sin(2*t0*omega))/omega^4/tj^3,
1/8*b*(2*Kt*omega^4*tj^4+2*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
2*Kn*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+3*Kt*cos(2*t0*omega+2*omega*t
j)+3*Kn*sin(2*t0*omega+2*omega*tj)+3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
3*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)-3*Kt*cos(2*t0*omega)-
3*Kn*sin(2*t0*omega)+3*Kt*omega*tj*sin(2*t0*omega)-
3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3;
 -
1/8*b*(2*Kt*omega^4*tj^4+2*Kn*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+3*Kn
*omega*tj*cos(2*t0*omega+2*omega*tj)-3*Kn*sin(2*t0*omega+2*omega*tj)-
```

```
2*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
3*Kt*cos(2*t0*omega+2*omega*tj)+3*Kn*sin(2*t0*omega)+3*Kt*cos(2*t0*omega)-
3*Kt*omega*tj*sin(2*t0*omega)+3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3,
1/8*b*(2*Kn*omega^4*tj^4+3*Kn*cos(2*t0*omega+2*omega*tj)+2*Kn*omega^3*tj^3
*sin(2*t0*omega+2*omega*tj)-
3*Kt*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj
)+3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*sin(2*t0*omega+2*o
mega*tj)-
3*Kn*cos(2*t0*omega)+3*Kt*sin(2*t0*omega)+3*Kt*omega*tj*cos(2*t0*omega)+3*
Kn*omega*tj*sin(2*t0*omega))/omega^4/tj^3];


P14 =[ -1/48*b*(6*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
12*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-
6*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
12*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+2*Kn*omega^4*tj^4-
9*Kn*cos(2*t0*omega+2*omega*tj)+9*Kt*sin(2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*sin(2*t0*omega)-
6*Kt*omega*tj*cos(2*t0*omega)+9*Kn*cos(2*t0*omega)-
9*Kt*sin(2*t0*omega))/tj^2/omega^4, 1/48*b*(-2*Kt*omega^4*tj^4-
12*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+12*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+6*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+6*Kn*sin(2*t0*omega+2
*omega*tj)*omega^2*tj^2-9*Kt*cos(2*t0*omega+2*omega*tj)-
9*Kn*sin(2*t0*omega+2*omega*tj)-
6*Kt*omega*tj*sin(2*t0*omega)+6*Kn*omega*tj*cos(2*t0*omega)+9*Kt*cos(2*t0*o
mega)+9*Kn*sin(2*t0*omega))/tj^2/omega^4;
  1/48*b*(2*Kt*omega^4*tj^4-
12*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+12*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+6*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+6*Kn*sin(2*t0*omega+2
*omega*tj)*omega^2*tj^2-9*Kt*cos(2*t0*omega+2*omega*tj)-
9*Kn*sin(2*t0*omega+2*omega*tj)-
6*Kt*omega*tj*sin(2*t0*omega)+6*Kn*omega*tj*cos(2*t0*omega)+9*Kt*cos(2*t0*o
mega)+9*Kn*sin(2*t0*omega))/tj^2/omega^4, 1/48*b*(-
12*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-9*Kn*cos(2*t0*omega+2*omega*tj)-
2*Kn*omega^4*tj^4-6*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
12*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+6*Kn*cos(2*t0*omega+2*omega*tj)*o
mega^2*tj^2+9*Kt*sin(2*t0*omega+2*omega*tj)+9*Kn*cos(2*t0*omega)-
9*Kt*sin(2*t0*omega)-6*Kn*omega*tj*sin(2*t0*omega)-
6*Kt*omega*tj*cos(2*t0*omega))/tj^2/omega^4];
 P21 =[ -1/80*b*(-15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
15*Kn*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+60*Kt*cos(2*t0*omega+2*omega
*tj)+60*Kn*sin(2*t0*omega+2*omega*tj)+4*Kn*omega^5*tj^5+60*Kt*omega*tj*sin(2
*t0*omega)-
10*Kn*omega^3*tj^3*cos(2*t0*omega)+15*Kn*omega^2*tj^2*sin(2*t0*omega)+15*K
t*omega^2*tj^2*cos(2*t0*omega)+10*Kt*omega^3*tj^3*sin(2*t0*omega)-
```

```
60*Kn*omega*tj*cos(2*t0*omega)+10*Kn*sin(2*t0*omega)*omega^4*tj^4+10*Kt*co
s(2*t0*omega)*omega^4*tj^4-60*Kn*sin(2*t0*omega)-
60*Kt*cos(2*t0*omega)+60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
60*Kn*omega*tj*cos(2*t0*omega+2*omega*tj))/omega^5/tj^4, -1/80*b*(-
60*Kn*cos(2*t0*omega)+60*Kt*sin(2*t0*omega)-
15*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kt*sin(2*t0*omega+2*omega
*tj)*omega^2*tj^2+60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-
60*Kt*sin(2*t0*omega+2*omega*tj)+60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+
60*Kn*cos(2*t0*omega+2*omega*tj)+10*Kt*omega^3*tj^3*cos(2*t0*omega)-
15*Kt*omega^2*tj^2*sin(2*t0*omega)-
10*Kt*sin(2*t0*omega)*omega^4*tj^4+10*Kn*cos(2*t0*omega)*omega^4*tj^4+15*K
n*omega^2*tj^2*cos(2*t0*omega)+60*Kt*omega*tj*cos(2*t0*omega)+10*Kn*omega^
3*tj^3*sin(2*t0*omega)+60*Kn*omega*tj*sin(2*t0*omega)+4*Kt*omega^5*tj^5)/ome
ga^5/tj^4;
   1/80*b*(60*Kn*cos(2*t0*omega)-
60*Kt*sin(2*t0*omega)+15*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
15*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kt*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-60*Kn*cos(2*t0*omega+2*omega*tj)-
10*Kt*omega^3*tj^3*cos(2*t0*omega)+15*Kt*omega^2*tj^2*sin(2*t0*omega)+10*Kt
*sin(2*t0*omega)*omega^4*tj^4-10*Kn*cos(2*t0*omega)*omega^4*tj^4-
15*Kn*omega^2*tj^2*cos(2*t0*omega)-60*Kt*omega*tj*cos(2*t0*omega)-
10*Kn*omega^3*tj^3*sin(2*t0*omega)-
60*Kn*omega*tj*sin(2*t0*omega)+4*Kt*omega^5*tj^5)/omega^5/tj^4, -1/80*b*(-
60*Kt*omega*tj*sin(2*t0*omega)+60*Kn*omega*tj*cos(2*t0*omega)-
10*Kn*sin(2*t0*omega)*omega^4*tj^4-10*Kt*cos(2*t0*omega)*omega^4*tj^4-
15*Kt*omega^2*tj^2*cos(2*t0*omega)-
10*Kt*omega^3*tj^3*sin(2*t0*omega)+10*Kn*omega^3*tj^3*cos(2*t0*omega)-
15*Kn*omega^2*tj^2*sin(2*t0*omega)+60*Kn*sin(2*t0*omega)+60*Kt*cos(2*t0*ome
ga)-
60*Kt*cos(2*t0*omega+2*omega*tj)+4*Kn*omega^5*tj^5+60*Kn*omega*tj*cos(2*t0
*omega+2*omega*tj)+15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*sin
(2*t0*omega+2*omega*tj)*omega^2*tj^2-60*Kn*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj))/omega^5/tj^4];
 P22 = [ 1/480*b*(-2*Kn*omega^5*tj^5-180*Kt*cos(2*t0*omega+2*omega*tj)-
180*Kn*sin(2*t0*omega+2*omega*tj)-
135*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*cos(2*t0*omega+2
*omega*tj)+30*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+30*Kn*sin(2*t0*ome
ga+2*omega*tj)*omega^2*tj^2-
225*Kt*omega*tj*sin(2*t0*omega)+180*Kn*sin(2*t0*omega)+180*Kt*cos(2*t0*omeg
a)+225*Kn*omega*tj*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*cos(2*t0*omega)+30*Kt*omega^3*tj^3*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*cos(2*t0*omega)-
120*Kn*omega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5, -
1/480*b*(135*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*sin(2*t0
*omega+2*omega*tj)-
```

```
180*Kt*sin(2*t0*omega+2*omega*tj)+180*Kn*cos(2*t0*omega+2*omega*tj)+2*Kt*o
mega^5*tj^5+30*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kt*omega^3*tj^3*cos(2*t0*omega)+120*Kn*omega^2*tj^2*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*sin(2*t0*omega)+225*Kt*omega*tj*cos(2*t0*omega)+225*Kn*
omega*tj*sin(2*t0*omega)+180*Kt*sin(2*t0*omega)-
180*Kn*cos(2*t0*omega))/tj^3/omega^5;
 -
1/480*b*(135*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*sin(2*t0
*omega+2*omega*tj)-
180*Kt*sin(2*t0*omega+2*omega*tj)+180*Kn*cos(2*t0*omega+2*omega*tj)-
2*Kt*omega^5*tj^5+30*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kt*omega^3*tj^3*cos(2*t0*omega)+120*Kn*omega^2*tj^2*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*sin(2*t0*omega)+225*Kt*omega*tj*cos(2*t0*omega)+225*Kn*
omega*tj*sin(2*t0*omega)+180*Kt*sin(2*t0*omega)-
180*Kn*cos(2*t0*omega))/tj^3/omega^5, -1/480*b*(2*Kn*omega^5*tj^5-
180*Kt*cos(2*t0*omega+2*omega*tj)-180*Kn*sin(2*t0*omega+2*omega*tj)-
135*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*cos(2*t0*omega+2
*omega*tj)+30*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+30*Kn*sin(2*t0*ome
ga+2*omega*tj)*omega^2*tj^2-
225*Kt*omega*tj*sin(2*t0*omega)+180*Kn*sin(2*t0*omega)+180*Kt*cos(2*t0*omeg
a)+225*Kn*omega*tj*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*cos(2*t0*omega)+30*Kt*omega^3*tj^3*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*cos(2*t0*omega)-
120*Kn*omega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5];
 P23 = [ -1/80*b*(-4*Kn*omega^5*tj^5-60*Kt*cos(2*t0*omega+2*omega*tj)-
60*Kn*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*sin(2*t0*omega
+2*omega*tj)*omega^2*tj^2-
10*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*sin(2*t0*o
mega+2*omega*tj)+10*Kt*omega^4*tj^4*cos(2*t0*omega+2*omega*tj)+10*Kn*omeg
a^3*tj^3*cos(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega)+60*Kn*sin(2*t0*omega)+60*Kt*cos(2*t0*omega)+
60*Kn*omega*tj*cos(2*t0*omega)-15*Kt*omega^2*tj^2*cos(2*t0*omega)-
15*Kn*omega^2*tj^2*sin(2*t0*omega))/omega^5/tj^4, -1/80*b*(-
60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kt*sin(2*t0*omega+2*omega*tj)-
60*Kn*cos(2*t0*omega+2*omega*tj)-
10*Kt*omega^4*tj^4*sin(2*t0*omega+2*omega*tj)-
10*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*cos(2*t0*o
mega+2*omega*tj)-10*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
4*Kt*omega^5*tj^5-
```

```
15*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*cos(2*t0*omega+2*omega
*tj)*omega^2*tj^2-
15*Kn*omega^2*tj^2*cos(2*t0*omega)+15*Kt*omega^2*tj^2*sin(2*t0*omega)-
60*Kt*omega*tj*cos(2*t0*omega)-60*Kn*omega*tj*sin(2*t0*omega)-
60*Kt*sin(2*t0*omega)+60*Kn*cos(2*t0*omega))/omega^5/tj^4;
 -1/80*b*(-60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kt*sin(2*t0*omega+2*omega*tj)-
60*Kn*cos(2*t0*omega+2*omega*tj)-
10*Kt*omega^4*tj^4*sin(2*t0*omega+2*omega*tj)-
10*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*cos(2*t0*o
mega+2*omega*tj)-
10*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+4*Kt*omega^5*tj^5-
15*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*cos(2*t0*omega+2*omega
*tj)*omega^2*tj^2-
15*Kn*omega^2*tj^2*cos(2*t0*omega)+15*Kt*omega^2*tj^2*sin(2*t0*omega)-
60*Kt*omega*tj*cos(2*t0*omega)-60*Kn*omega*tj*sin(2*t0*omega)-
60*Kt*sin(2*t0*omega)+60*Kn*cos(2*t0*omega))/omega^5/tj^4,
1/80*b*(4*Kn*omega^5*tj^5-60*Kt*cos(2*t0*omega+2*omega*tj)-
60*Kn*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*sin(2*t0*omega
+2*omega*tj)*omega^2*tj^2-
10*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*sin(2*t0*o
mega+2*omega*tj)+10*Kt*omega^4*tj^4*cos(2*t0*omega+2*omega*tj)+10*Kn*omeg
a^3*tj^3*cos(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega)+60*Kn*sin(2*t0*omega)+60*Kt*cos(2*t0*omega)+
60*Kn*omega*tj*cos(2*t0*omega)-15*Kt*omega^2*tj^2*cos(2*t0*omega)-
15*Kn*omega^2*tj^2*sin(2*t0*omega))/omega^5/tj^4];
 P24 = [ -
1/480*b*(2*Kn*omega^5*tj^5+180*Kt*cos(2*t0*omega+2*omega*tj)+180*Kn*sin(2*t
0*omega+2*omega*tj)+225*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
225*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)-
120*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
120*Kn*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+30*Kn*omega^3*tj^3*cos(2*t0*o
mega+2*omega*tj)+135*Kt*omega*tj*sin(2*t0*omega)-180*Kn*sin(2*t0*omega)-
180*Kt*cos(2*t0*omega)-
135*Kn*omega*tj*cos(2*t0*omega)+30*Kt*omega^2*tj^2*cos(2*t0*omega)+30*Kn*o
mega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5, -
1/480*b*(225*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+225*Kn*omega*tj*sin(2*t0
*omega+2*omega*tj)-
180*Kt*sin(2*t0*omega+2*omega*tj)+180*Kn*cos(2*t0*omega+2*omega*tj)-
30*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)-
30*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^5*tj^5+120*Kt*sin(
2*t0*omega+2*omega*tj)*omega^2*tj^2-
120*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+30*Kn*omega^2*tj^2*cos(2*t0*
```

```
omega)-
30*Kt*omega^2*tj^2*sin(2*t0*omega)+135*Kt*omega*tj*cos(2*t0*omega)+135*Kn*o
mega*tj*sin(2*t0*omega)+180*Kt*sin(2*t0*omega)-
180*Kn*cos(2*t0*omega))/tj^3/omega^5;
 1/480*b*(-225*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
225*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+180*Kt*sin(2*t0*omega+2*omega*tj
)-
180*Kn*cos(2*t0*omega+2*omega*tj)+30*Kt*omega^3*tj^3*cos(2*t0*omega+2*ome
ga*tj)+30*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^5*tj^5-
120*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+120*Kn*cos(2*t0*omega+2*ome
ga*tj)*omega^2*tj^2-
30*Kn*omega^2*tj^2*cos(2*t0*omega)+30*Kt*omega^2*tj^2*sin(2*t0*omega)-
135*Kt*omega*tj*cos(2*t0*omega)-135*Kn*omega*tj*sin(2*t0*omega)-
180*Kt*sin(2*t0*omega)+180*Kn*cos(2*t0*omega))/tj^3/omega^5, -
1/480*b*(2*Kn*omega^5*tj^5-180*Kt*cos(2*t0*omega+2*omega*tj)-
180*Kn*sin(2*t0*omega+2*omega*tj)-
225*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+225*Kn*omega*tj*cos(2*t0*omega+2
*omega*tj)+120*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+120*Kn*sin(2*t0*o
mega+2*omega*tj)*omega^2*tj^2+30*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)
-30*Kn*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)-
135*Kt*omega*tj*sin(2*t0*omega)+180*Kn*sin(2*t0*omega)+180*Kt*cos(2*t0*omeg
a)+135*Kn*omega*tj*cos(2*t0*omega)-30*Kt*omega^2*tj^2*cos(2*t0*omega)-
30*Kn*omega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5];
 P11 = [P11(1,1)*ones(lmx,1) P11(1,2)*ones(lmx,1); P11(2,1)*ones(lmx,1)
P11(2,2)*ones(lmx,1)]*V;
P12 = [P12(1,1)*ones(lmx,1) P12(1,2)*ones(lmx,1); P12(2,1)*ones(lmx,1)
P12(2,2)*ones(lmx,1)]*V;
P13 = [P13(1,1)*ones(lmx,1) P13(1,2)*ones(lmx,1); P13(2,1)*ones(lmx,1)
P13(2,2)*ones(lmx,1)]*V;
P14 = [P14(1,1)*ones(lmx,1) P14(1,2)*ones(lmx,1); P14(2,1)*ones(lmx,1)
P14(2,2)*ones(lmx,1)]*V;
P21 = [P21(1,1)*ones(lmx,1) P21(1,2)*ones(lmx,1); P21(2,1)*ones(lmx,1)
P21(2,2)*ones(lmx,1)]*V;
P22 = [P22(1,1)*ones(lmx,1) P22(1,2)*ones(lmx,1); P22(2,1)*ones(lmx,1)
P22(2,2)*ones(lmx,1)]*V;
P23 = [P23(1,1)*ones(lmx,1) P23(1,2)*ones(lmx,1); P23(2,1)*ones(lmx,1)
P23(2,2)*ones(lmx,1)]*V;
P24 = [P24(1,1)*ones(lmx,1) P24(1,2)*ones(lmx,1); P24(2,1)*ones(lmx,1)
P24(2,2)*ones(lmx,1)]*V;
N11 = -C+1/2*K*tj+P11;
N12 = -M+1/12*K*tj^2+P12;
N13 = C+1/2*K*tj+P13;
N14 = M-1/12*K*tj^2+P14;
N21 = M/tj-1/10*K*tj+P21;
N22 = 1/2*M-1/12*C*tj-1/120*K*tj^2+P22;
N23 = -M/tj+1/10*K*tj+P23;
```

```
N24 = 1/2*M+1/12*C*tj-1/120*K*tj^2+P24;

   N1  = [N11 N12; N21 N22];
   N2  = [N13 N14; N23 N24];

   P1  = [P11 P12; P21 P22];
   P2  = [P13 P14; P23 P24];

%%%
```

```
                end % while loop depth of cut
```

Uncertainty *SLE*, Sensitivity Method

```
% M. Kurdi (3/28/2005)
% Function to find uncertainty in SLE to change in cutting
% coefficients, dynamic parameters, and cutting process variables
% Input:
%       b:      depth of cut (m)
%       rpm:    spindle speed
%       rstep:  radial step (inches)
%       Kt
%       Kn
%       Kte
%       Kre
%       DELTA_Kt    finite change in Kt
%       DELTA_b     finite change in b
%       system_ID:  Modal parameters
% The derivative of Max eigen value is found for a miniscule perturbation
% in input parameters, then its effect on the change of axial depth is
% found.
clear all; close all; clc;tic;
% function uncer
% percentage of uncertainty in cutting coefficients, dynamic parameters
% and process parameters
% tic;
percent_Kcut = 0.05; % cutting coefficents uncertainty
percent_Dyn = 0.005; % modal parameters uncertainty
percent_rstep = 0.0001; % radial step uncertainty
percent_rpm = 0.005; % spindle speed uncertainty
% nominal values of process parameters and their calculated uncertainty
rstep = 0.2;
b=1e-3;
rpm_vec = 5500:50:5600;
DELTA_rstep = percent_rstep*rstep;
% cutting coefficient uncertainty
Kt = 6e8;
DELTA_Kt = percent_Kcut*Kt;
Kn = .3*Kt;
DELTA_Kn = percent_Kcut*Kn;
Kte=0;
DELTA_Kte = percent_Kcut*Kte;
Kne=0;
DELTA_Kne = percent_Kcut*Kne;
% nominal values of dynamic parameters and their calculated uncertainty
Kx =  4.4528e+006;
Mx = 0.4362;
Cx = 83;
```

```
% Y direction parameters
Ky = 3.5542e+006;
My = 0.4362;
Cy = 89.9;
DELTA_Mx = Mx*percent_Dyn;
DELTA_My = My*percent_Dyn;
DELTA_Kx = Kx*percent_Dyn;
DELTA_Ky = Ky*percent_Dyn;
DELTA_Cx = Cx*percent_Dyn;
DELTA_Cy = Cy*percent_Dyn;
% to calculate the numerical derivative with respect to each input
% variable set the miniscule change in each input
% set miniscule change in input parameters to estimate the derivative
step_percent = 0.002;
dKt = step_percent*Kt; % N/m2
dKn = step_percent*Kn; % N/m2
dKte = step_percent*30; % N/m
dKne = step_percent*30; % N/m
drstep = step_percent*rstep; % inch
dKx = step_percent*Kx; % N/m
dKy = step_percent*Ky; % N/m
dCx = step_percent*Cx; %
dCy = step_percent*Cy;
dMx = step_percent*Mx; % Kg
dMy = step_percent*My; % Kg
h = waitbar(0,'Please wait...');
% computation here %
for i=1:length(rpm_vec)
   waitbar(i/length(rpm_vec),h);
   rpm = rpm_vec(i);
   drpm = step_percent*rpm; % rpm
   DELTA_rpm = percent_rpm * rpm;
   % Find depth of cut corresponding to stability boundary using nominal
   % settings of input parameters
   [sle(i)] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy); % depth at
boundary
   % Find numerical derivative of maximum eigenvalue with respect to input
   % parameters
   % perturb cutting coefficient Kt by dKt
   [sle1] = sle_f(b,rpm,rstep,Kt-dKt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
   [sle2] = sle_f(b,rpm,rstep,Kt+dKt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
   % derivative of eigen matrix w.r.t cutting coefficient Kt
   d_sle_Kt(i) = (sle2-sle1)/dKt/2;
   dsleKt_log(i) = d_sle_Kt(i)*Kt/(sle(i))*2;
   sle1 =[]; sle2 =[];
```

```
   % perturb cutting coefficient Kn by dKn
   [sle1] = sle_f(b,rpm,rstep,Kt,Kn-dKn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
   [sle2] = sle_f(b,rpm,rstep,Kt,Kn+dKn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
%
%    % derivative of eigen matrix w.r.t cutting coefficient Kt
   d_sle_Kn(i) = (sle2-sle1)/dKn/2;
   dsleKn_log(i) = d_sle_Kn(i)*Kn/(sle(i))*2;
   sle1 =[]; sle2 =[];
   % perturb cutting coefficient Kte by dKte
    [sle1] = sle_f(b,rpm,rstep,Kt,Kn,Kte-dKte,Kne,Mx,Kx,Cx,My,Ky,Cy);
   [sle2] = sle_f(b,rpm,rstep,Kt,Kn,Kte+dKte,Kne,Mx,Kx,Cx,My,Ky,Cy);
   % derivative of eigen matrix w.r.t cutting coefficient Kt
   d_sle_Kte(i) = (sle2-sle1)/dKte/2;
   dsleKte_log(i) = d_sle_Kte(i)*Kte/(sle(i))*2;
   sle1 =[]; sle2 =[];
   % perturb cutting coefficient Kne by dKne
   [sle1] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne-dKne,Mx,Kx,Cx,My,Ky,Cy);
   [sle2] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne+dKne,Mx,Kx,Cx,My,Ky,Cy);
   % derivative of eigen matrix w.r.t cutting coefficient Kne
   d_sle_Kne(i) = (sle2-sle1)/dKne/2;
   dsleKne_log(i) = d_sle_Kne(i)*Kne/(sle(i))*2;
   sle1 =[]; sle2 =[];
%
   % perturb depth of cut rstep by drstep
   [sle1] = sle_f(b,rpm,rstep-drstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
   [sle2] = sle_f(b,rpm,rstep+drstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
   % derivative of eigen value w.r.t rstep of cut
   d_sle_rstep(i) = (sle2-sle1)/drstep/2;
   dslerstep_log(i) = d_sle_rstep(i)*rstep/(sle(i))*2;
   sle1 =[]; sle2 =[];
   % perturb spindle speed by drpm
   [sle1] = sle_f(b,rpm-drpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
   [sle2] = sle_f(b,rpm+drpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy);
   % derivative of eigen value w.r.t rpm
   d_sle_rpm(i) = (sle2-sle1)/drpm/2;
   dslerpm_log(i) = d_sle_rpm(i)*rpm/(sle(i))*2;
   sle1 =[]; sle2 =[];
%     perturb Kx by dKx
   [sle1] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx-dKx,Cx,My,Ky,Cy);
   [sle2] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx+dKx,Cx,My,Ky,Cy);
%    derivative of eigen value w.r.t Kx
   d_sle_Kx(i) = (sle2-sle1)/dKx/2;
   dsleKx_log(i) = d_sle_Kx(i)*Kx/(sle(i))*2;
   sle1 =[]; sle2 =[];
    % perturb Ky by dKy
   [sle1] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky-dKy,Cy);
```

```
   [sle2] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky+dKy,Cy);
   % derivative of eigen value w.r.t Ky
   d_sle_Ky(i) = (sle2-sle1)/dKy/2;
   dsleKy_log(i) = d_sle_Ky(i)*Ky/(sle(i))*2;
   sle1 =[]; sle2 =[];
    % perturb Cx by dCx
   [sle1] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx-dCx,My,Ky,Cy);
   [sle2] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx+dCx,My,Ky,Cy);
   % derivative of eigen value w.r.t Cx
   d_sle_Cx(i) = (sle2-sle1)/dCx/2;
   dsleCx_log(i) = d_sle_Cx(i)*Cx/(sle(i))*2;
   sle1 =[]; sle2 =[];
   % perturb Cy by dCy
   [sle1] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy-dCy);
   [sle2] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy+dCy);
   % derivative of eigen value w.r.t Cy
   d_sle_Cy(i) = (sle2-sle1)/dCy/2;
   dsleCy_log(i) = d_sle_Cy(i)*Cy/(sle(i))*2;
   sle1 =[]; sle2 =[];
     % perturb Mx by dMx
   [sle1] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne,Mx-dMx,Kx,Cx,My,Ky,Cy);
   [sle2] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne,Mx+dMx,Kx,Cx,My,Ky,Cy);
   % derivative of eigen value w.r.t Mx
   d_sle_Mx(i) = (sle2-sle1)/dMx/2;
   dsleMx_log(i) = d_sle_Mx(i)*Mx/(sle(i))*2;
   sle1 =[]; sle2 =[];
    % perturb My by dMy
   [sle1] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My-dMy,Ky,Cy);
   [sle2] = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My+dMy,Ky,Cy);
%     % derivative of eigen value w.r.t My
   d_sle_My(i) = (sle2-sle1)/dMy/2;
   dsleMy_log(i) = d_sle_My(i)*My/(sle(i))*2;
   sle1 =[]; sle2 =[];
DELTA_sle(i) =( (DELTA_Kt * d_sle_Kt(i))^2 + (DELTA_Kn * d_sle_Kn(i))^2 + ...
        (DELTA_Kne * d_sle_Kne(i))^2 + (DELTA_Kte * d_sle_Kte(i))^2 +...
        (DELTA_Kx * d_sle_Kx(i))^2 + (DELTA_Mx * d_sle_Mx(i))^2 + ...
         (DELTA_Cx * d_sle_Cx(i))^2 + (DELTA_Ky * d_sle_Ky(i))^2 +...
          (DELTA_My * d_sle_My(i))^2 + (DELTA_Cy * d_sle_Cy(i))^2+...
          (DELTA_rstep * d_sle_rstep(i))^2 + (DELTA_rpm* d_sle_rpm(i))^2)^0.5
end
close(h);
% % Find the uncertainty in depth of cut for a corresponding uncertainty in
% % input paramters
time_total=toc;
% save uncer_march_30_sle
% subplot(2,1,1)
```

```
% plot(rpm_vec/1000,sle*1e6,'-g',rpm_vec/1000,(sle+2*DELTA_sle)*1e6,'-
k',rpm_vec/1000,(sle-2*DELTA_sle)*1e6,'-k')
% set(gca,'fontname','times','fontsize',16);
% xlabel('\Omega (x10^3 rpm)','fontsize',14)
% ylabel('SLE (\mum)','fontsize',14)
% legend('Stability boundary, nominal input','\pm2u_c(SLE)');
% axis([5 20 -12 28])
% subplot(2,1,2)
% plot(rpm_vec/1000,d_sle_Ky*Ky*1e6,'<',rpm_vec/1000,d_sle_My*My*1e6,'>',...
%     rpm_vec/1000,d_sle_Cy*Cy*1e6,'o',rpm_vec/1000,d_sle_rpm.*rpm_vec*1e6,'+',...
%     rpm_vec/1000,d_sle_rstep*rstep*1e6,'^',rpm_vec/1000,d_sle_Kt*Kt*1e6,'s',...
%     rpm_vec/1000,d_sle_Kn*Kn*1e6,'*');
% legend('K_y','M_y','C_y','\Omega','r_{step}','K_t','K_n');
% xlabel('\Omega (x10^3 rpm)','fontsize',14)
% ylabel('x_i \partial(SLE)/\partial(x_i)');
% figure;
% plot(rpm_vec/1000,dsleKy_log,'<',rpm_vec/1000,dsleMy_log,'>',...
%     rpm_vec/1000,dsleCy_log,'o',rpm_vec/1000,dslerpm_log,'+',...
%     rpm_vec/1000,dslerstep_log,'^',rpm_vec/1000,dsleKt_log,'s',...
%     rpm_vec/1000,dsleKn_log,'*');
% legend('K_y','M_y','C_y','\Omega','r_{step}','K_t','K_n');
% xlabel('\Omega (x10^3 rpm)','fontsize',14)
% ylabel('\partial(SLE)/\partial(x_i)x_i/SLE');
% figure;
plot(rpm_vec/1000,abs(d_sle_Ky)*Ky*1e6,'.',rpm_vec/1000,abs(d_sle_My)*My*1e6,':',.
..
%     rpm_vec/1000,abs(d_sle_Cy)*Cy*1e6,'--
',rpm_vec/1000,abs(d_sle_rpm).*rpm_vec*1e6,'-',...
%
rpm_vec/1000,abs(d_sle_rstep)*rstep*1e6,'^',rpm_vec/1000,abs(d_sle_Kt)*Kt*1e6,'s',...
%     rpm_vec/1000,abs(d_sle_Kn)*Kn*1e6);
% legend('K_y','M_y','C_y','\Omega','r_{step}','K_t','K_n');
% figure;
% plot(rpm_vec/1000,abs(d_sle_Ky)*Ky./abs(d_sle_My)/My)
% legend('K_y/M_y');
figure
plot(rpm_vec/1000,DELTA_sle*1e6)
xlabel('\Omega (x10^3 rpm)','fontsize',14)
ylabel('u_c(SLE) (\mum)')
figure
plot(rpm_vec/1000,DELTA_sle*1e6,'-<k')
figure
plot(rpm_vec/1000,sle*1e6,'-g',rpm_vec/1000,(sle+2*DELTA_sle)*1e6,'-
k',rpm_vec/1000,(sle-2*DELTA_sle)*1e6,'-k')
```

Uncertainty *SLE,* Monte Carlo and Latin Hype-Cube Sampling Methods

```
%
% M. Kurdi (6/17/05)
% 4 OAL TOOL
% Program to complete LatinHyper and Monte simulation for SLE
clear all;
% function LatinHyper
tic;
chip_load=0.1e-3;% chip load
nteeth = 4;
Diam =0.5;
E=20;
N = 1000; % number of iterations
baxial=4.45e-3;
% AL 6061
% percent_Kt = 7.13/100; % cutting coefficents uncertainty
% percent_Kn = 8.09/100;
% percent_Kte = 30.3/100;
% percent_Kne = 23.9/100;
% 5 OAL TOOL UNCERTAINTIES
% percent_KX = 0.054; % modal parameters uncertainty
% percent_CX = .286;
% percent_MX =.045;
% percent_KY = 0.054; % modal parameters uncertainty
% percent_CY = .173;
% percent_MY =.055;
% 4 OAL TOOL UNCERTAINTIES due to thermal effect only
 percent_MX = 0.074;
 percent_CX = 0.042;
 percent_KX = 0.073 ;
 percent_MY = 0.2;
 percent_CY = 0.107;
 percent_KY = 0.2 ;
 percent_rstep = 0.0005; % radial step uncertainty
percent_rpm = 0.005; % spindle speed uncertainty
% speed_min = str2num(input('Min_speed = ','s'));
% speed_max = str2num(input('Max_speed = ','s'));
% speed = speed_min:200:speed_max;
% speed = [   14753 14803 14853 14903 14953]; 4.45 mm
speed = [15517 15567 15617 15667  15767]; % 2.12 mm
h = waitbar(0,'Please wait...');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%5
% Cutting Coefficients %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
```

```
% AL 6061
% mean_Kt =7.06E+08; % N/m2
% mean_Kn = 2.50E+08;
% mean_Kte = 1.29E+04; % N/m;
% mean_Kne = 6.57E+03;
% AL 7475
mean_Kt =  690480868.527357;
mean_Kte = 12022.3004909002;
mean_Kn = 142535991.092323;
mean_Kne =11281.4601645315;
std_Kn=4009843*4.45; % N
std_Kne=310.909*4.45;
std_Kte=200.731*4.45;
std_Kt=2588583*4.45;
% std_Kt = percent_Kt*mean_Kt;
% std_Kn = percent_Kn*mean_Kn;
% std_Kte = percent_Kte*mean_Kte;
% std_Kne = percent_Kne*mean_Kne;
% Kne Kn Kte Kt
% AL  6061
% SIGMA_K = [1.480E+07  -1.778E+11    -8.216E+06    9.871E+10;
%           -1.778E+11 2.458E+15     9.871E+10     -1.365E+15;
%           -8.216E+06 9.871E+10     9.163E+07     -1.101E+12;
%           9.871E+10  -1.365E+15    -1.101E+12    1.522E+16
%           ];
% AL 7475
  SIGMA_K = [ 42157610.7365206        -506483170409.775        -3598978.12573119
43238262783.6325;
       -506483170409.775     7.00379474676691e+015         43238262783.6325         -
597911116174549;
       -3598978.12573128         43238262783.6335         17574719.1179838         -
211143357093.21;
        43238262783.6335         -597911116174562         -211143357093.21
2.91975098408051e+015];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Modal Parameters
% X %%%%%%%%%%%%%%%%
% 5 OAL TOOL
% mean_Kx =  2.64E+06;
% mean_Mx = 0.049;
% mean_Cx = 8.972;
 % dynamic parameters for 4 OAL tool
       mean_Mx = 0.027 ;
       mean_Cx=  23.309;
       mean_Kx= 4359275.000 ;
```

```
std_Cx = percent_CX*mean_Cx;
std_Kx = percent_KX*mean_Kx;
std_Mx = percent_MX*mean_Mx;
% Mx Cx Kx My Cy Ky 5 OAL
% SIGMA = [3.85E-06        4.03E-03        2.48E+02        2.40E-06        -3.18E-03
        1.31E+02;
%        4.03E-03        5.27E+00        2.69E+05        4.08E-03        -3.32E+00
        2.19E+05;
%        2.48E+02        2.69E+05        1.61E+10        1.67E+02        -2.07E+05
        9.15E+09;
%        2.40E-06        4.08E-03        1.67E+02        4.23E-06        -1.88E-03
        2.24E+02;
%        -3.18E-03       -3.32E+00       -2.07E+05       -1.88E-03       2.71E+00        -
1.04E+05;
%        1.31E+02        2.19E+05        9.15E+09        2.24E+02        -1.04E+05
        1.19E+10
%
%           ];
% Mx Cx Kx My Cy Ky 4 OAL
SIGMA = [4.04188E-06        0.000450265   631.110625    7.25563E-06   -0.000584252
        878.998125;
0.000450265   0.953490935   38828.325      0.00283473    -2.467636648  567721.5525;
631.110625    38828.325     1.00042E+11    1068.011875   -51720.94813  1.21332E+11;
7.25563E-06   0.00283473    1068.011875    1.76519E-05   -0.007067488  2638.261875;
-0.000584252  -2.467636648  -51720.94813   -0.007067488  11.34481701   -1426512.396;
878.998125    567721.5525   1.21332E+11    2638.261875   -1426512.396
        4.36003E+11];
% % Y %%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 5 OAL TOOL
% mean_Ky = 2.26e+006;
% mean_Cy = 10.651;
% mean_My = 0.042;
% Y direction parameters 4 OAL TOOL
mean_Ky = 3301775.000;
mean_My =  0.021;
mean_Cy = 31.432;
std_My = percent_MY*mean_My;
std_Ky = percent_KY*mean_Ky;
std_Cy = percent_CY*mean_Cy;
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Radial step inches
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mean_rstep = 0.25*.5;
std_rstep = percent_rstep*mean_rstep;
```

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
randn('state',0)
Mode = lhsnorm([mean_Mx mean_Cx mean_Kx mean_My mean_Cy
mean_Ky],SIGMA,N);
% Mode(:,1) is Mx random vector
% Mode(:,2) is Cx random vector
% Mode(:,3) is Kx random vector
% Mode(:,4) is My random vector
% Mode(:,5) is Cy random vector
% Mode(:,6) is Ky random vector
Cut_Coeff = lhsnorm([mean_Kne mean_Kn mean_Kte mean_Kt],SIGMA_K,N);
% Cut_Coeff(:,1) Kne
% Cut_Coeff(:,2) Kn
% Cut_Coeff(:,3) Kte
% Cut_Coeff(:,4) Kt
sample = randn(N, 2);
for j=1:length(speed)
waitbar(j/length(speed),h)
for i=1:N
% Unless otherwise specified, all dimensions in m
% Define input parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Cutting coefficients
Kt = Cut_Coeff(i,4);
Kn = Cut_Coeff(i,2);
Kte = Cut_Coeff(i,3);
Kne = Cut_Coeff(i,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% milling parameters
% Spindle speed
mean_rpm = speed(j);
std_rpm = percent_rpm*mean_rpm;
rpm = mean_rpm + std_rpm*sample(i,1);
% rstep
rstep = mean_rstep + std_rstep*sample(i,2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dynamic parameters
% X direction is feed direction
Kx =Mode(i,3);
Mx = Mode(i,1);
Cx = Mode(i,2);
% Y direction parameters
```

```
Ky = Mode(i,6);
My = Mode(i,4);
Cy = Mode(i,5);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate axial depth corresponding to input paramters
% that is on the stability boundaries
sle(i,j) =
sle_f(baxial,rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy,chip_load,nteeth,Diam,E);
end % i end monte loop for one spindle speed
end % j end spindle speed range
for i=1:length(speed)
index = find(isnan(sle(:,i))==0);
sle_mean(i) = mean(sle(index,i));
std_dev(i) = std(sle(index,i));
end
time=toc;
save Latin_AL7475SLE2p12 std_dev speed sle_mean sle time
close(h);
% hold on;
% h1 = plot(speed/1000,(sle_mean-2*std_dev)*1e6,'-r')
% hold on;
% h2 = plot(speed/1000,sle_mean*1e6,'g-');
% hold on;
% h3 = plot(speed/1000,(2*std_dev+sle_mean)*1e6,'-r');
% legend([h1,h2,h3],'lower boundary','mean','upper boundary')
% hold on;
% for i=1:1000
%
%     plot(speed/1000,sle(i,:)*1e6,'.');
% end
% % figure
% % plot(speed/1000,std(sle)*2*1000)
% %

% % Input:
%        rpm ;
%        rstep:      radial immersion (inches)
%   Output:
%   b   depth of cut (m)

function SLE = sle_f(b,rpm,rstep,Kt,Kn,Kte,Kne,Mx,Kx,Cx,My,Ky,Cy,h,nteeth,Diam,E)
%     E=30;
%
%     h          = 0.1e-3;                       % feed per tooth
%        nteeth      = 1;                              % number of teeth
```

```
%   Diam       = 1;                          % inches
    TRAVang    = acos(1-rstep/(Diam/2));            % angular travel during cutting
    LAGang     = 2*pi/nteeth;                    % separation angle for teeth
    rho        = acos(1-rstep/(Diam/2))/(2*pi);       % fraction of time in cut
    IMMERSION  = rstep/Diam;
    opt        = 'down';
    if TRAVang>LAGang  % MULTIPLE TEETH ARE IN CONTACT
   teethNcontact = floor(TRAVang/LAGang) +1;
    else            % SINGLE TOOTH IN CONTACT
   teethNcontact = 1;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %       SYSTEM IDENTIFICATION MATRICES
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    M   =[Mx zeros(size(Mx)); zeros(size(Mx)) My];
    C   =[Cx zeros(size(Mx)); zeros(size(Mx)) Cy];
    K   =[Kx zeros(size(Mx)); zeros(size(Mx)) Ky];
    lmx = length(Mx(1,:));
    lmy = length(My(1,:));
    DOF = lmx+lmy;
    V     = [ones(1,lmx) zeros(1,lmy); zeros(1,lmx) ones(1,lmy)];
    A     = zeros((E+1)*2*DOF,(E+1)*2*DOF);
    B     = A;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %      BEGIN LOOP CALCULATIONS OVER RPM vs DOC FIELD
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        speed   = rpm;
        omega   = speed/60*(2*pi);            % radians per second
        T       = (2*pi)/omega/nteeth;         % tooth pass period
        TC      = rho*T*nteeth;              % time a single tooth spends in the cut
        tf      = T-TC;                    % time for free vibs
        tj      = TC/E;                    % time for each element
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %      SET CUTTER ROTATION ANGLE FOR UP/DOWN-MILLING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        switch opt
        case 'up'
        t0mat   = [0 tj*(1:(E-1))];           % upmilling
        locat   = 2*DOF+lmx+1:3*DOF;
        case 'down'
```

```
        tex     = pi/omega; tent=tex-TC;          % downmilling
        t0mat   = [tent tent+tj*(1:(E-1))];     % downmilling
        locat   = (E+1)*2*DOF-DOF-lmy+1:(E+1)*2*DOF-DOF;
        end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %       STATE TRANSITION MATRIX
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        G1  = [zeros(size(M)) M; eye(size(M)) zeros(size(M))];
        G2  = [K  C; zeros(size(M)) -eye(size(M))];
        G   = -G1\G2;
        PHI = expm(G*tf);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %       N & P are used to create A & B which then become Q in..... a_n =
Q a_n-1 + D
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for e=1:E,
   t0 = t0mat(e);
  C1 = V'*[ -1/4*b*(-h*Kt*cos(2*t0*omega+2*omega*tj)+2*h*Kn*omega*tj-
h*Kn*sin(2*t0*omega+2*omega*tj)+4*Kte*sin(t0*omega+omega*tj)-
4*Kne*cos(t0*omega+omega*tj)+h*Kt*cos(2*t0*omega)+h*Kn*sin(2*t0*omega)-
4*Kte*sin(t0*omega)+4*Kne*cos(t0*omega))/omega;
   1/4*b*(2*h*Kt*omega*tj-
h*Kt*sin(2*t0*omega+2*omega*tj)+h*Kn*cos(2*t0*omega+2*omega*tj)-
4*Kte*sin(t0*omega+omega*tj)+4*Kne*cos(t0*omega+omega*tj)+h*Kt*sin(2*t0*ome
ga)-h*Kn*cos(2*t0*omega)+4*Kte*sin(t0*omega)-4*Kne*cos(t0*omega))/omega];
  C2 = V'*[ 1/8*b*(-
h*Kt*sin(2*t0*omega+2*omega*tj)+h*Kt*cos(2*t0*omega+2*omega*tj)*omega*tj+h*
Kn*cos(2*t0*omega+2*omega*tj)+h*Kn*sin(2*t0*omega+2*omega*tj)*omega*tj+8*K
te*sin(t0*omega+omega*tj)-4*Kte*cos(t0*omega+omega*tj)*omega*tj-
8*Kne*cos(t0*omega+omega*tj)-
4*Kne*sin(t0*omega+omega*tj)*omega*tj+h*Kt*sin(2*t0*omega)-
h*Kn*cos(2*t0*omega)-
8*Kte*sin(t0*omega)+8*Kne*cos(t0*omega)+h*Kt*tj*cos(2*t0*omega)*omega+h*Kn*
tj*sin(2*t0*omega)*omega-4*Kte*tj*cos(t0*omega)*omega-
4*Kne*tj*sin(t0*omega)*omega)/tj/omega^2;
   -
1/8*b*(h*Kt*cos(2*t0*omega+2*omega*tj)+h*Kt*sin(2*t0*omega+2*omega*tj)*omeg
a*tj+h*Kn*sin(2*t0*omega+2*omega*tj)-
h*Kn*cos(2*t0*omega+2*omega*tj)*omega*tj-
```

```
8*Kte*sin(t0*omega+omega*tj)+4*Kte*cos(t0*omega+omega*tj)*omega*tj+8*Kne*co
s(t0*omega+omega*tj)+4*Kne*sin(t0*omega+omega*tj)*omega*tj-
h*Kt*cos(2*t0*omega)-h*Kn*sin(2*t0*omega)+8*Kte*sin(t0*omega)-
8*Kne*cos(t0*omega)+h*Kt*tj*sin(2*t0*omega)*omega-
h*Kn*tj*cos(2*t0*omega)*omega+4*Kte*tj*cos(t0*omega)*omega+4*Kne*tj*sin(t0*o
mega)*omega)/tj/omega^2];
P11 = [  1/8*b*(-
3*Kt*sin(2*t0*omega+2*omega*tj)+3*Kn*cos(2*t0*omega+2*omega*tj)+2*Kn*omeg
a^4*tj^4+3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*sin(2*t0*ome
ga+2*omega*tj)+3*Kt*sin(2*t0*omega)-
3*Kn*cos(2*t0*omega)+2*Kt*cos(2*t0*omega)*omega^3*tj^3+2*Kn*sin(2*t0*omega)
*omega^3*tj^3+3*Kt*omega*tj*cos(2*t0*omega)+3*Kn*omega*tj*sin(2*t0*omega))/o
mega^4/tj^3,  1/8*b*(-3*Kt*cos(2*t0*omega+2*omega*tj)-
3*Kn*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^4*tj^4-
3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*cos(2*t0*omega+2*om
ega*tj)+3*Kt*cos(2*t0*omega)+3*Kn*sin(2*t0*omega)-
2*Kt*sin(2*t0*omega)*omega^3*tj^3+2*Kn*cos(2*t0*omega)*omega^3*tj^3-
3*Kt*omega*tj*sin(2*t0*omega)+3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3;
     1/8*b*(-3*Kt*cos(2*t0*omega+2*omega*tj)-3*Kn*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^4*tj^4-
3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*cos(2*t0*omega+2*om
ega*tj)+3*Kt*cos(2*t0*omega)+3*Kn*sin(2*t0*omega)-
2*Kt*sin(2*t0*omega)*omega^3*tj^3+2*Kn*cos(2*t0*omega)*omega^3*tj^3-
3*Kt*omega*tj*sin(2*t0*omega)+3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3, -
1/8*b*(-
2*Kn*omega^4*tj^4+3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*s
in(2*t0*omega+2*omega*tj)-
3*Kt*sin(2*t0*omega+2*omega*tj)+3*Kn*cos(2*t0*omega+2*omega*tj)+2*Kt*cos(2*
t0*omega)*omega^3*tj^3+2*Kn*sin(2*t0*omega)*omega^3*tj^3+3*Kt*omega*tj*cos(
2*t0*omega)+3*Kn*omega*tj*sin(2*t0*omega)+3*Kt*sin(2*t0*omega)-
3*Kn*cos(2*t0*omega))/omega^4/tj^3];
P12 =[
1/48*b*(6*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+6*Kt*omega*tj*cos(2*t0*ome
ga+2*omega*tj)+2*Kn*omega^4*tj^4-
9*Kt*sin(2*t0*omega+2*omega*tj)+9*Kn*cos(2*t0*omega+2*omega*tj)+12*Kn*ome
ga*tj*sin(2*t0*omega)+12*Kt*omega*tj*cos(2*t0*omega)-
6*Kt*omega^2*tj^2*sin(2*t0*omega)+6*Kn*omega^2*tj^2*cos(2*t0*omega)+9*Kt*si
n(2*t0*omega)-9*Kn*cos(2*t0*omega))/tj^2/omega^4, -
1/48*b*(9*Kt*cos(2*t0*omega+2*omega*tj)+9*Kn*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^4*tj^4+6*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)+12*Kt*omega*tj*sin(2*t0*omega)+6*
Kn*omega^2*tj^2*sin(2*t0*omega)+6*Kt*omega^2*tj^2*cos(2*t0*omega)-
9*Kt*cos(2*t0*omega)-12*Kn*omega*tj*cos(2*t0*omega)-
9*Kn*sin(2*t0*omega))/tj^2/omega^4;
    -
1/48*b*(2*Kt*omega^4*tj^4+6*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+9*Kt*cos(
```

```
2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)+9*Kn*sin(2*t0*omega+2*omega*tj)+1
2*Kt*omega*tj*sin(2*t0*omega)+6*Kn*omega^2*tj^2*sin(2*t0*omega)+6*Kt*omega^
2*tj^2*cos(2*t0*omega)-9*Kt*cos(2*t0*omega)-12*Kn*omega*tj*cos(2*t0*omega)-
9*Kn*sin(2*t0*omega))/tj^2/omega^4, 1/48*b*(-
6*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+2*Kn*omega^4*tj^4-
9*Kn*cos(2*t0*omega+2*omega*tj)+9*Kt*sin(2*t0*omega+2*omega*tj)-
12*Kt*omega*tj*cos(2*t0*omega)-
12*Kn*omega*tj*sin(2*t0*omega)+6*Kt*omega^2*tj^2*sin(2*t0*omega)-
6*Kn*omega^2*tj^2*cos(2*t0*omega)+9*Kn*cos(2*t0*omega)-
9*Kt*sin(2*t0*omega))/tj^2/omega^4];
P13 =[  1/8*b*(2*Kn*omega^4*tj^4-
3*Kn*cos(2*t0*omega+2*omega*tj)+3*Kt*sin(2*t0*omega+2*omega*tj)-
3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
2*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
3*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+3*Kn*cos(2*t0*omega)-
3*Kt*sin(2*t0*omega)-3*Kt*omega*tj*cos(2*t0*omega)-
3*Kn*omega*tj*sin(2*t0*omega))/omega^4/tj^3,
1/8*b*(2*Kt*omega^4*tj^4+2*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
2*Kn*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+3*Kt*cos(2*t0*omega+2*omega*t
j)+3*Kn*sin(2*t0*omega+2*omega*tj)+3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
3*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)-3*Kt*cos(2*t0*omega)-
3*Kn*sin(2*t0*omega)+3*Kt*omega*tj*sin(2*t0*omega)-
3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3;
    -
1/8*b*(2*Kt*omega^4*tj^4+2*Kn*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+3*Kn
*omega*tj*cos(2*t0*omega+2*omega*tj)-3*Kn*sin(2*t0*omega+2*omega*tj)-
2*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
3*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
3*Kt*cos(2*t0*omega+2*omega*tj)+3*Kn*sin(2*t0*omega)+3*Kt*cos(2*t0*omega)-
3*Kt*omega*tj*sin(2*t0*omega)+3*Kn*omega*tj*cos(2*t0*omega))/omega^4/tj^3,
1/8*b*(2*Kn*omega^4*tj^4+3*Kn*cos(2*t0*omega+2*omega*tj)+2*Kn*omega^3*tj^3
*sin(2*t0*omega+2*omega*tj)-
3*Kt*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj
)+3*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+3*Kn*omega*tj*sin(2*t0*omega+2*o
mega*tj)-
3*Kn*cos(2*t0*omega)+3*Kt*sin(2*t0*omega)+3*Kt*omega*tj*cos(2*t0*omega)+3*
Kn*omega*tj*sin(2*t0*omega))/omega^4/tj^3];
P14 =[ -1/48*b*(6*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
12*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-
6*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
12*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+2*Kn*omega^4*tj^4-
9*Kn*cos(2*t0*omega+2*omega*tj)+9*Kt*sin(2*t0*omega+2*omega*tj)-
6*Kn*omega*tj*sin(2*t0*omega)-
```

```
6*Kt*omega*tj*cos(2*t0*omega)+9*Kn*cos(2*t0*omega)-
9*Kt*sin(2*t0*omega))/tj^2/omega^4, 1/48*b*(-2*Kt*omega^4*tj^4-
12*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+12*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+6*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+6*Kn*sin(2*t0*omega+2
*omega*tj)*omega^2*tj^2-9*Kt*cos(2*t0*omega+2*omega*tj)-
9*Kn*sin(2*t0*omega+2*omega*tj)-
6*Kt*omega*tj*sin(2*t0*omega)+6*Kn*omega*tj*cos(2*t0*omega)+9*Kt*cos(2*t0*o
mega)+9*Kn*sin(2*t0*omega))/tj^2/omega^4;
    1/48*b*(2*Kt*omega^4*tj^4-
12*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+12*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+6*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+6*Kn*sin(2*t0*omega+2
*omega*tj)*omega^2*tj^2-9*Kt*cos(2*t0*omega+2*omega*tj)-
9*Kn*sin(2*t0*omega+2*omega*tj)-
6*Kt*omega*tj*sin(2*t0*omega)+6*Kn*omega*tj*cos(2*t0*omega)+9*Kt*cos(2*t0*o
mega)+9*Kn*sin(2*t0*omega))/tj^2/omega^4, 1/48*b*(-
12*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-9*Kn*cos(2*t0*omega+2*omega*tj)-
2*Kn*omega^4*tj^4-6*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
12*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+6*Kn*cos(2*t0*omega+2*omega*tj)*o
mega^2*tj^2+9*Kt*sin(2*t0*omega+2*omega*tj)+9*Kn*cos(2*t0*omega)-
9*Kt*sin(2*t0*omega)-6*Kn*omega*tj*sin(2*t0*omega)-
6*Kt*omega*tj*cos(2*t0*omega))/tj^2/omega^4];
P21 =[ -1/80*b*(-15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
15*Kn*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+60*Kt*cos(2*t0*omega+2*omega
*tj)+60*Kn*sin(2*t0*omega+2*omega*tj)+4*Kn*omega^5*tj^5+60*Kt*omega*tj*sin(2
*t0*omega)-
10*Kn*omega^3*tj^3*cos(2*t0*omega)+15*Kn*omega^2*tj^2*sin(2*t0*omega)+15*K
t*omega^2*tj^2*cos(2*t0*omega)+10*Kt*omega^3*tj^3*sin(2*t0*omega)-
60*Kn*omega*tj*cos(2*t0*omega)+10*Kn*sin(2*t0*omega)*omega^4*tj^4+10*Kt*co
s(2*t0*omega)*omega^4*tj^4-60*Kn*sin(2*t0*omega)-
60*Kt*cos(2*t0*omega)+60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
60*Kn*omega*tj*cos(2*t0*omega+2*omega*tj))/omega^5/tj^4, -1/80*b*(-
60*Kn*cos(2*t0*omega)+60*Kt*sin(2*t0*omega)-
15*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kt*sin(2*t0*omega+2*omega
*tj)*omega^2*tj^2+60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)-
60*Kt*sin(2*t0*omega+2*omega*tj)+60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+
60*Kn*cos(2*t0*omega+2*omega*tj)+10*Kt*omega^3*tj^3*cos(2*t0*omega)-
15*Kt*omega^2*tj^2*sin(2*t0*omega)-
10*Kt*sin(2*t0*omega)*omega^4*tj^4+10*Kn*cos(2*t0*omega)*omega^4*tj^4+15*K
n*omega^2*tj^2*cos(2*t0*omega)+60*Kt*omega*tj*cos(2*t0*omega)+10*Kn*omega^
3*tj^3*sin(2*t0*omega)+60*Kn*omega*tj*sin(2*t0*omega)+4*Kt*omega^5*tj^5)/ome
ga^5/tj^4;
    1/80*b*(60*Kn*cos(2*t0*omega)-
60*Kt*sin(2*t0*omega)+15*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
15*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kt*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-60*Kn*cos(2*t0*omega+2*omega*tj)-
```

```
10*Kt*omega^3*tj^3*cos(2*t0*omega)+15*Kt*omega^2*tj^2*sin(2*t0*omega)+10*Kt
*sin(2*t0*omega)*omega^4*tj^4-10*Kn*cos(2*t0*omega)*omega^4*tj^4-
15*Kn*omega^2*tj^2*cos(2*t0*omega)-60*Kt*omega*tj*cos(2*t0*omega)-
10*Kn*omega^3*tj^3*sin(2*t0*omega)-
60*Kn*omega*tj*sin(2*t0*omega)+4*Kt*omega^5*tj^5)/omega^5/tj^4, -1/80*b*(-
60*Kt*omega*tj*sin(2*t0*omega)+60*Kn*omega*tj*cos(2*t0*omega)-
10*Kn*sin(2*t0*omega)*omega^4*tj^4-10*Kt*cos(2*t0*omega)*omega^4*tj^4-
15*Kt*omega^2*tj^2*cos(2*t0*omega)-
10*Kt*omega^3*tj^3*sin(2*t0*omega)+10*Kn*omega^3*tj^3*cos(2*t0*omega)-
15*Kn*omega^2*tj^2*sin(2*t0*omega)+60*Kn*sin(2*t0*omega)+60*Kt*cos(2*t0*ome
ga)-
60*Kt*cos(2*t0*omega+2*omega*tj)+4*Kn*omega^5*tj^5+60*Kn*omega*tj*cos(2*t0
*omega+2*omega*tj)+15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*sin
(2*t0*omega+2*omega*tj)*omega^2*tj^2-60*Kn*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj))/omega^5/tj^4];
P22 = [ 1/480*b*(-2*Kn*omega^5*tj^5-180*Kt*cos(2*t0*omega+2*omega*tj)-
180*Kn*sin(2*t0*omega+2*omega*tj)-
135*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*cos(2*t0*omega+2
*omega*tj)+30*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+30*Kn*sin(2*t0*ome
ga+2*omega*tj)*omega^2*tj^2-
225*Kt*omega*tj*sin(2*t0*omega)+180*Kn*sin(2*t0*omega)+180*Kt*cos(2*t0*omeg
a)+225*Kn*omega*tj*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*cos(2*t0*omega)+30*Kt*omega^3*tj^3*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*cos(2*t0*omega)-
120*Kn*omega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5, -
1/480*b*(135*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*sin(2*t0
*omega+2*omega*tj)-
180*Kt*sin(2*t0*omega+2*omega*tj)+180*Kn*cos(2*t0*omega+2*omega*tj)+2*Kt*o
mega^5*tj^5+30*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kt*omega^3*tj^3*cos(2*t0*omega)+120*Kn*omega^2*tj^2*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*sin(2*t0*omega)+225*Kt*omega*tj*cos(2*t0*omega)+225*Kn*
omega*tj*sin(2*t0*omega)+180*Kt*sin(2*t0*omega)-
180*Kn*cos(2*t0*omega))/tj^3/omega^5;
    -
1/480*b*(135*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*sin(2*t0
*omega+2*omega*tj)-
180*Kt*sin(2*t0*omega+2*omega*tj)+180*Kn*cos(2*t0*omega+2*omega*tj)-
2*Kt*omega^5*tj^5+30*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kt*omega^3*tj^3*cos(2*t0*omega)+120*Kn*omega^2*tj^2*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*sin(2*t0*omega)+225*Kt*omega*tj*cos(2*t0*omega)+225*Kn*
omega*tj*sin(2*t0*omega)+180*Kt*sin(2*t0*omega)-
180*Kn*cos(2*t0*omega))/tj^3/omega^5, -1/480*b*(2*Kn*omega^5*tj^5-
```

```
180*Kt*cos(2*t0*omega+2*omega*tj)-180*Kn*sin(2*t0*omega+2*omega*tj)-
135*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+135*Kn*omega*tj*cos(2*t0*omega+2
*omega*tj)+30*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+30*Kn*sin(2*t0*ome
ga+2*omega*tj)*omega^2*tj^2-
225*Kt*omega*tj*sin(2*t0*omega)+180*Kn*sin(2*t0*omega)+180*Kt*cos(2*t0*omeg
a)+225*Kn*omega*tj*cos(2*t0*omega)-
120*Kt*omega^2*tj^2*cos(2*t0*omega)+30*Kt*omega^3*tj^3*sin(2*t0*omega)-
30*Kn*omega^3*tj^3*cos(2*t0*omega)-
120*Kn*omega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5];
P23 = [ -1/80*b*(-4*Kn*omega^5*tj^5-60*Kt*cos(2*t0*omega+2*omega*tj)-
60*Kn*sin(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*sin(2*t0*omega
+2*omega*tj)*omega^2*tj^2-
10*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*sin(2*t0*o
mega+2*omega*tj)+10*Kt*omega^4*tj^4*cos(2*t0*omega+2*omega*tj)+10*Kn*omeg
a^3*tj^3*cos(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega)+60*Kn*sin(2*t0*omega)+60*Kt*cos(2*t0*omega)+
60*Kn*omega*tj*cos(2*t0*omega)-15*Kt*omega^2*tj^2*cos(2*t0*omega)-
15*Kn*omega^2*tj^2*sin(2*t0*omega))/omega^5/tj^4, -1/80*b*(-
60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kt*sin(2*t0*omega+2*omega*tj)-
60*Kn*cos(2*t0*omega+2*omega*tj)-
10*Kt*omega^4*tj^4*sin(2*t0*omega+2*omega*tj)-
10*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*cos(2*t0*o
mega+2*omega*tj)-10*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)-
4*Kt*omega^5*tj^5-
15*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*cos(2*t0*omega+2*omega
*tj)*omega^2*tj^2-
15*Kn*omega^2*tj^2*cos(2*t0*omega)+15*Kt*omega^2*tj^2*sin(2*t0*omega)-
60*Kt*omega*tj*cos(2*t0*omega)-60*Kn*omega*tj*sin(2*t0*omega)-
60*Kt*sin(2*t0*omega)+60*Kn*cos(2*t0*omega))/omega^5/tj^4;
    -1/80*b*(-60*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
60*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kt*sin(2*t0*omega+2*omega*tj)-
60*Kn*cos(2*t0*omega+2*omega*tj)-
10*Kt*omega^4*tj^4*sin(2*t0*omega+2*omega*tj)-
10*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*cos(2*t0*o
mega+2*omega*tj)-
10*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+4*Kt*omega^5*tj^5-
15*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*cos(2*t0*omega+2*omega
*tj)*omega^2*tj^2-
15*Kn*omega^2*tj^2*cos(2*t0*omega)+15*Kt*omega^2*tj^2*sin(2*t0*omega)-
60*Kt*omega*tj*cos(2*t0*omega)-60*Kn*omega*tj*sin(2*t0*omega)-
60*Kt*sin(2*t0*omega)+60*Kn*cos(2*t0*omega))/omega^5/tj^4,
1/80*b*(4*Kn*omega^5*tj^5-60*Kt*cos(2*t0*omega+2*omega*tj)-
60*Kn*sin(2*t0*omega+2*omega*tj)-
```

```
60*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+60*Kn*omega*tj*cos(2*t0*omega+2*o
mega*tj)+15*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+15*Kn*sin(2*t0*omega
+2*omega*tj)*omega^2*tj^2-
10*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+10*Kn*omega^4*tj^4*sin(2*t0*o
mega+2*omega*tj)+10*Kt*omega^4*tj^4*cos(2*t0*omega+2*omega*tj)+10*Kn*omeg
a^3*tj^3*cos(2*t0*omega+2*omega*tj)-
60*Kt*omega*tj*sin(2*t0*omega)+60*Kn*sin(2*t0*omega)+60*Kt*cos(2*t0*omega)+
60*Kn*omega*tj*cos(2*t0*omega)-15*Kt*omega^2*tj^2*cos(2*t0*omega)-
15*Kn*omega^2*tj^2*sin(2*t0*omega))/omega^5/tj^4];
P24 = [ -
1/480*b*(2*Kn*omega^5*tj^5+180*Kt*cos(2*t0*omega+2*omega*tj)+180*Kn*sin(2*t
0*omega+2*omega*tj)+225*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)-
225*Kn*omega*tj*cos(2*t0*omega+2*omega*tj)-
120*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2-
120*Kn*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2-
30*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+30*Kn*omega^3*tj^3*cos(2*t0*o
mega+2*omega*tj)+135*Kt*omega*tj*sin(2*t0*omega)-180*Kn*sin(2*t0*omega)-
180*Kt*cos(2*t0*omega)-
135*Kn*omega*tj*cos(2*t0*omega)+30*Kt*omega^2*tj^2*cos(2*t0*omega)+30*Kn*o
mega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5, -
1/480*b*(225*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)+225*Kn*omega*tj*sin(2*t0
*omega+2*omega*tj)-
180*Kt*sin(2*t0*omega+2*omega*tj)+180*Kn*cos(2*t0*omega+2*omega*tj)-
30*Kt*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)-
30*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^5*tj^5+120*Kt*sin(
2*t0*omega+2*omega*tj)*omega^2*tj^2-
120*Kn*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+30*Kn*omega^2*tj^2*cos(2*t0*
omega)-
30*Kt*omega^2*tj^2*sin(2*t0*omega)+135*Kt*omega*tj*cos(2*t0*omega)+135*Kn*o
mega*tj*sin(2*t0*omega)+180*Kt*sin(2*t0*omega)-
180*Kn*cos(2*t0*omega))/tj^3/omega^5;
    1/480*b*(-225*Kt*omega*tj*cos(2*t0*omega+2*omega*tj)-
225*Kn*omega*tj*sin(2*t0*omega+2*omega*tj)+180*Kt*sin(2*t0*omega+2*omega*tj
)-
180*Kn*cos(2*t0*omega+2*omega*tj)+30*Kt*omega^3*tj^3*cos(2*t0*omega+2*ome
ga*tj)+30*Kn*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)+2*Kt*omega^5*tj^5-
120*Kt*sin(2*t0*omega+2*omega*tj)*omega^2*tj^2+120*Kn*cos(2*t0*omega+2*ome
ga*tj)*omega^2*tj^2-
30*Kn*omega^2*tj^2*cos(2*t0*omega)+30*Kt*omega^2*tj^2*sin(2*t0*omega)-
135*Kt*omega*tj*cos(2*t0*omega)-135*Kn*omega*tj*sin(2*t0*omega)-
180*Kt*sin(2*t0*omega)+180*Kn*cos(2*t0*omega))/tj^3/omega^5, -
1/480*b*(2*Kn*omega^5*tj^5-180*Kt*cos(2*t0*omega+2*omega*tj)-
180*Kn*sin(2*t0*omega+2*omega*tj)-
225*Kt*omega*tj*sin(2*t0*omega+2*omega*tj)+225*Kn*omega*tj*cos(2*t0*omega+2
*omega*tj)+120*Kt*cos(2*t0*omega+2*omega*tj)*omega^2*tj^2+120*Kn*sin(2*t0*o
mega+2*omega*tj)*omega^2*tj^2+30*Kt*omega^3*tj^3*sin(2*t0*omega+2*omega*tj)
```

```
-30*Kn*omega^3*tj^3*cos(2*t0*omega+2*omega*tj)-
135*Kt*omega*tj*sin(2*t0*omega)+180*Kn*sin(2*t0*omega)+180*Kt*cos(2*t0*omeg
a)+135*Kn*omega*tj*cos(2*t0*omega)-30*Kt*omega^2*tj^2*cos(2*t0*omega)-
30*Kn*omega^2*tj^2*sin(2*t0*omega))/tj^3/omega^5];
P11 = [P11(1,1)*ones(lmx,1) P11(1,2)*ones(lmx,1); P11(2,1)*ones(lmx,1)
P11(2,2)*ones(lmx,1)]*V;
   P12 = [P12(1,1)*ones(lmx,1) P12(1,2)*ones(lmx,1); P12(2,1)*ones(lmx,1)
P12(2,2)*ones(lmx,1)]*V;
   P13 = [P13(1,1)*ones(lmx,1) P13(1,2)*ones(lmx,1); P13(2,1)*ones(lmx,1)
P13(2,2)*ones(lmx,1)]*V;
   P14 = [P14(1,1)*ones(lmx,1) P14(1,2)*ones(lmx,1); P14(2,1)*ones(lmx,1)
P14(2,2)*ones(lmx,1)]*V;
   P21 = [P21(1,1)*ones(lmx,1) P21(1,2)*ones(lmx,1); P21(2,1)*ones(lmx,1)
P21(2,2)*ones(lmx,1)]*V;
   P22 = [P22(1,1)*ones(lmx,1) P22(1,2)*ones(lmx,1); P22(2,1)*ones(lmx,1)
P22(2,2)*ones(lmx,1)]*V;
   P23 = [P23(1,1)*ones(lmx,1) P23(1,2)*ones(lmx,1); P23(2,1)*ones(lmx,1)
P23(2,2)*ones(lmx,1)]*V;
   P24 = [P24(1,1)*ones(lmx,1) P24(1,2)*ones(lmx,1); P24(2,1)*ones(lmx,1)
P24(2,2)*ones(lmx,1)]*V;
   N11 = -C+1/2*K*tj+P11;
   N12 = -M+1/12*K*tj^2+P12;
   N13 = C+1/2*K*tj+P13;
   N14 = M-1/12*K*tj^2+P14;
   N21 = M/tj-1/10*K*tj+P21;
   N22 = 1/2*M-1/12*C*tj-1/120*K*tj^2+P22;
   N23 = -M/tj+1/10*K*tj+P23;
   N24 = 1/2*M+1/12*C*tj-1/120*K*tj^2+P24;
   N1  = [N11 N12; N21 N22];
   N2  = [N13 N14; N23 N24];
   P1  = [P11 P12; P21 P22];
   P2  = [P13 P14; P23 P24];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
      % BUILD GLOBAL MATRICES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
      A(1:2*DOF,1:2*DOF) = eye(2*DOF);
      A(2*DOF*e+1:2*DOF*e+2*DOF,2*DOF*(e-1)+1:2*DOF*(e-1)+2*DOF) = N1;
      A(2*DOF*e+1:2*DOF*e+2*DOF,2*DOF*(e-1)+2*DOF+1:2*DOF*(e-
1)+2*DOF+2*DOF) = N2;
      B(2*DOF*e+1:2*DOF*e+2*DOF,2*DOF*(e-1)+1:2*DOF*(e-1)+2*DOF) = P1;
      B(2*DOF*e+1:2*DOF*e+2*DOF,2*DOF*(e-1)+2*DOF+1:2*DOF*(e-
1)+2*DOF+2*DOF) = P2;
      B(1:2*(DOF),E*2*(DOF)+1:(E+1)*2*(DOF)) = PHI;
      Cvec(1:2*DOF,1) = zeros(2*DOF,1);
```

```
    Cvec(2*DOF*e+1:2*DOF*e+DOF,1) = C1;
    Cvec(2*DOF*e+DOF+1:2*DOF*e+2*DOF,1) = C2;
  end; % end # of elements loop
`               Q          = A\B;
                [vec,lam]   = eig(Q);
                CM          = max(abs(diag(lam)));
                D          = A\Cvec;
                % Extract SLE coefficients
                if CM<1
       SLE_vec     = inv((eye(size(Q))-Q))*D;
       SLE         = (sum(SLE_vec(locat)));
                else
      SLE          = nan;
                end
                N1 = zeros(2*DOF,2*DOF); N2 = N1;
                P1 = N1; P2 = P1;
```

LIST OF REFERENCES

1. Plaskos, C., Hodgson, A.J., and Cinquin, P., Modelling and Optimization of Bone-Cutting Forces in Orthopaedic Surgery. Medical Image Computing and Computer-Assisted Intervention - Miccai 2003, Pt 1, 2003. **2878**: p. 254.

2. De Lacalle, L.N.L., Lamikiz, A., Sanchez, J.A., and Cabanes, I., Cutting Conditions and Tool Optimization in the High-Speed Milling of Aluminium Alloys. Proceedings of the Institution of Mechanical Engineers Part B-Journal of Engineering Manufacture, 2001. **215**(9): p. 1257.

3. De Lacalle, L.N.L., Lamikiz, A., Salgado, M.A., Herranz, S., and Rivero, A., Process Planning for Reliable High-Speed Machining of Moulds. International Journal of Production Research, 2002. **40**(12): p. 2789.

4. Fallbohmer, P., Rodriguez, C.A., Ozel, T., and Altan, T., High-Speed Machining of Cast Iron and Alloy Steels for Die and Mold Manufacturing. Journal of Materials Processing Technology, 2000. **98**(1): p. 104.

5. Ermer, D.S., A Century of Optimizing Machining Operations. Journal of Manufacturing Science and Engineering-Transactions of the ASME, 1997. **119**(4B): p. 817.

6. Sonmez, A.I., Baykasoglu, A., Turkay, D., and Filiz, I.H., Dynamic Optimization of Multi-Pass Milling Operations Via Geometric Programming. International Journal of Machine Tools & Manufacture, 1999. **39**(2): p. 297.

7. Wang, J. and Armarego, E.J.A., Computer-Aided Optimization of Multiple Constraint Single Pass Face Milling Operations. Machining Science and Technology, 2001. **5**(1): p. 77.

8. Armarego, E.J.A., Smith, A.J.R., and Wang, J., Computer-Aided Constrained Optimization Analyses and Strategies for Multi-Pass Helical Tooth Milling Operations. CIRP Annals, 1994. **43**(1): p. 437.

9. Ermer, D.S. and Patel, D.C. Maximization of the Production Rate with Constraints by Linear Programming and Sensitivity Analysis. In *Proceedings of NAMRC*. 1974.

10. Ermer, D.S., Optimization of Constrained Machining Economics Problem by Geometric Programming. ASME Journal of Engineering for Industry, 1971. **93**: p. 1067.

11. Hati, S.K. and Rao, S.S., Determination of Optimum Machining Conditions - Deterministic and Probabilistic Approaches. ASME Journal of Engineering for Industry, 1975. **98**: p. 354.

12. Hitomi, K., *Analysis* of Optimal Machining Speeds for Automatic Manufacturing. International Journal of Production Research, 1989. **27**: p. 1685.

13. Shalaby, M.A. and Riad, M.S. A Linear Optimization Model for Single Pass Turning Operations. in *Proc. 27th Int. MATADOR Conf*. 1988.

14. Armarego, E.J.A., Smith, A.J.R., and Wang, J., Constrained Optimization Strategies and Cam Software for Single-Pass Peripheral Milling. International Journal of Production Research, 1993. **31**(9): p. 2139.

15. Challa, K. and Berra, P.B., Automated Planning and Optimization of Machining Processes: A Systems Approach. Computers and Industrial Engineering, 1976. **1**: p. 35.

16. Juan, H., Yu, S.F., and Lee, B.Y., The Optimal Cutting-Parameter Selection of Production Cost in HSM for Skd61 Tool Steels. International Journal of Machine Tools & Manufacture, 2003. **43**(7): p. 679.

17. Iwata, K., Murostu, Y., T., I., and Fujii, S., A Probabilistic Approach to the Determination of the Optimum Cutting. ASME Journal of Engineering for Industry, 1972. **94**: p. 1099.

18. Beightler, C.S. and Philips, D.T. Optimization in Tool Engineering Using Geometric Programming. In *AIIE Trans*. 1970.

19. Gopalakrishnan, B. and Al-Khayyal, F., Machine Parameter Selection for Turning with Constraints: An Analytical Approach Based on Geometric Programming. International Journal of Production Research, 1991. **29**: p. 1897.

20. Walvekar, A.G. and Lambert, B.K., An Application of Geometric Programming to Machining Variable Selection. International Journal of Production Research, 1970. **8**: p. 241.

21. Tandon, V., El-Mounayri, H., and Kishawy, H., NC End Milling Optimization Using Evolutionary Computation. International Journal of Machine Tools and Manufacture, 2002. **42**(5): p. 595.

22. Boothroyd, G. and Rusek, P., Maximum Rate of Profit Criteria in Machining. ASME Journal of Engineering for Industry, 1976. **98**: p. 217.

23. Agapiou, J.S., The Optimization of Machining Operations Based on a Combined Criterion - Part I: The Use of Combined Objectives in Single-Pass Operations. ASME Journal of Engineering for Industry, 1992. **114**: p. 500.

24. Jha, N.K., A Discrete Data Base Multiple Objective Optimization of Milling Operation through Geometric Programming. ASME Journal of Engineering For Industry, 1990. **112**: p. 368.

25. Abuelnaga, A.M. and Eldardiry, M.A., Optimization Methods for Metal-Cutting. International Journal of Machine Tools & Manufacture, 1984. **24**(1): p. 11.

26. Milner, D.A., Use of Linear-Programming for Machinability Data Optimization. Mechanical Engineering, 1977. **99**(7): p. 96.

27. Tae, J.K. and Dong, W.C., Adaptive Optimization of Face Milling Operations Using Neural Networks. Journal of Manufacturing Science and Engineering, 1998. **120**: p. 443.

28. Koulams, C.P., Simultaneous Determination of Optimal Machining Conditions and Tool Replacement Policies in Constrained Machining Economics Problem by Geometric Programming. International Journal of Production Research, 1991. **29**(12): p. 2407.

29. Eskicioglu, H., Nisli, M.S., and Kilic, S.E. An Application of Geometric Programming to Single-Pass Turning Operations. in *Proceedings of International MTDR Conference*. 1985. Birmingham.

30. Petropoulos, P.G., Optimal Selection of Machining Rate Variables by Geometric Programming. International Journal of Production Research, 1975. **13**: p. 390.

31. Hough, C.L. and Goforth, R.E. Quadratic Posylognomials: An Extension of Posynomial Geometric Programming. In *AIIE Transactions*. 1981.

32. Philipson, R.H. and Ravindran, A., Application of Goal Programming to Machinability Data Optimization. ASME Journal of Mechanical Design, 1978. **100**: p. 286.

33. Sundaram, R.M., An Application of Goal Programming Technique in Metal Cutting. International Journal of Production Research, 1978. **16**: p. 375.

34. Subbarao, P.C. and Jacobs, C.H. Application of Nonlinear Goal Programming to Machining Variable Optimization. In *Proceedings of NAMRC*. 1978.

35. Fang, X.D. and Jawahir, I.S., Predicting Total Machining Performance in Finish Turning Using Integrated Fuzzy-Set Models of the Machinability Parameters. International Journal of Production Research., 1994. **32**: p. 833.

36. Edgeworth, F.Y., Mathematical Psychics, ed. P. Keagan. 1881, London; C.K. Paul.

37. Schwier, A.S., *Manual* of Political Economy, Translation of the French Edition *(1927)*. 1971, London-Basingslohe: Macmillan Press Ltd.

38. Kalyanmoy, D., Multi-Objective Optimization Using Evolutionary Algorithms. 2001, West Sussex, UK: John Wiley.

39. Taylor, F.W., On the Art of Cutting Metals. Transactions of ASME, 1907. **28**: p. 31.

40. Tlusty, J., Manufacturing Processes and Equipment. 2000, Upper Saddle River, NJ; Prentice Hall.

41. Tlusty, J. and Polacek, M. The Stability of the Machine Tool against Self Excited Vibration in Machining. In *Production Engineering Research Conference*. 1963. Pittsburgh.

42. Altintas, Y. and Budak, E., Analytical Prediction of Stability Lobes in Milling. CIRP Annals, 1995. **44**(1): p. 357.

43. Tlusty, J., Zaton, W., and Ismail, F., Stability Lobes in Milling. CIRP Annals, 1983. **32**(1): p. 309.

44. Tlusty, J., Basic Non-Linearity in Machining Chatter. CIRP Annals, 1981. **30**(1): p. 299.

45. DeVor, R.E., Kline, W.A., and Zdeblick, W.J. A Mechanistic Model for the Force System in End Milling with Application to Machining Airframe Structures. In *North American Manufacturing Research Conference*. 1980.

46. Tlusty, J. and MacNeil, P., Dynamics of Cutting Forces in End Milling. CIRP Annals, 1975. **24**.

47. Kline, W.A., DeVor, R.E., and Shareef, J.R., The Prediction of Surface Accuracy in End Milling. ASME Journal of Engineering for Industry, 1982. **104**: p. 272.

48. Kline, W.A. and DeVor, R.E., The Effect of Runout on Cutting Geometry and Forces in End Milling. International Journal of Machine Tool Design and Research, 1983. **23**(2/3): p. 123.

49. Sutherland, J.W. and Devor, R.E., An Improved Method for Cutting Force and Surface Error Prediction in Flexible End Milling Systems. Journal of Engineering for Industry-Transactions of the ASME, 1986. **108**(4): p. 269.

50. Tlusty, J. and Ismail, F., Special Aspects of Chatter in Milling. Journal of Vibration Acoustics Stress and Reliability in Design-Transactions of the ASME, 1983. **105**(1): p. 24.

51. Merdol, S.D. and Altintas, Y., Multi Frequency Solution of Chatter Stability for Low Immersion Milling. Journal of Manufacturing Science and Engineering-Transactions of the ASME, 2004. **126**(3): p. 459.

52. Budak, E. and Altintas, Y., Analytical Prediction of Chatter Stability in Milling - Part 1: General Formulation. Journal of Dynamic Systems Measurement and Control-Transactions of the ASME, 1998. **120**(1): p. 22.

53. Budak, E. and Altintas, Y., Analytical Prediction of Chatter Stability in Milling - Part Ii: Application of the General Formulation to Common Milling Systems. Journal of Dynamic Systems Measurement and Control-Transactions of the ASME, 1998. **120**(1): p. 31.

54. Mann, B.P., Dynamics of Milling Process. 2003, Ph.D dissertation, Saint Louis, MO: Washington University.

55. Mann, B.P., Bayly, P.V., Davies, M.A., and Halley, J.E., Limit Cycles, Bifurcations, and Accuracy of the Milling Process. Journal of Sound and Vibration, 2004. **277**(1-2): p. 31.

56. Mann, B.P., Insperger, T., Bayly, P.V., and Stepan, G., Stability of up-Milling and Down-Milling, Part 2: Experimental Verification. International Journal of Machine Tools and Manufacture, 2003. **43**: p. 35.

57. Insperger, T., Mann, B.P., Stepan, G., and Bayly, P.V., Stability of up-Milling and Down-Milling, Part1: Alternative Analytical Methods. International Journal of Machine Tools and Manufacture, 2003. **43**: p. 25.

58. Halley, J.E., Stability of Low Radial Immersion Milling. 1999, M.Sc. thesis, St. Louis, Mo: Washington University.

59. Eschenauer, H.A., Koski, J., and Osyczka, A., Multicriteria Design Optimization: Procedures and Applications. 1986, New York: Springer-Verlag.

60. Collette, Y. and Siarry, P., Multiobjective Optimization,.ed. D.R. Roy. 2003, Berlin: Springer-Verlag.

61. Deb, K., Multi-Objective Optimization Using Evolutionary Algorithms. 2001, West Sussex, UK: John Wiley.

62. Rakowska, J., Haftka, R.T., and Watson, L.T., Tracing the Efficient Curve for Multi-Objective Control-Structure Optimization. Computing Systems in Engineering, 1991. **2**(5/6): p. 461.

63. Deb, K., Multi-objective genetic algorithms: Problem difficulties and construction of test problems. Evolutionary Computational Journal, 1999. **7**(3): p. 205.

64. Marler, R.T. and Arora, J.S., Survey of Multi-Objective Optimization Methods for Engineering. Structural Multidisciplinary Optimization, 2004. **26**: p. 369.

65. Keeney, R.L. and Raiffa, H., Decisions with Multiple Objectives: Preferences and Value Tradeoff. 1993, New York: Cambridge University Press Publishing.

66. Rangaswamy, A. and Shell, G.R., Using Computers to Realize Joint Gains in Negotiations: Toward an "Electronic Bargaining Table. Management Science, 1997. **43**(8): p. 1147.

67. Rangaswamy, A. and Lilien, G.L., Software Tools for New Product Development. Journal of Marketing Research, 1997. **34**(1): p. 177.

68. Coello Coello, C.A., An Updated Survey of G. A. Based Multi-Objective Optimization Techniques. In *Technical Report Lania-RD-98-08*. 1998, Xalapa, Veracruz, Mexico.

69. Miettinen, K.M., Nonlinear Multiobjective Optimization.1999, Boston: Kluwer Academic Publisher.

70. Hwang, C.-L., Masud, A.S., Paidy, S.R., and Yoon, K., Multiple Objective Decision Making, Methods and Applications: A State-of-the-Art Survey. Lecture Notes in Economics and Mathematical Systems, No. 164, ed. M. Beckmann and H.P. Kunzi. 1979, Berlin: Springer Verlag.

71. Haimes, Y.Y., Lasdon, L.S., and Wismer, D.A. On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization. In *IEEE Transactions Syst. Man Cybern*. 1971.

72. Das, I. An Improved Technique for Choosing Parameters for Pareto Surface Generation Using Normal-Boundary Intersection. In *ISSMO/UBCAD/AIASA, Third World Congress of Structural and Multidisciplinary Optimization*. 1999. Buffalo: University of Buffalo, Center for Advanced Design.

73. Das, I. and Dennis, J.E., Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. SIAM Journal of Optimization, 1998. **8**(631-657).

74. Messac, A., Ismail-Yahaya, A., and Mattson, C.A., The Normalized Normal Constraint Method for Generating the Pareto Frontier. Structural Multidisciplinary Optimization, 2003. **25**: p. 86.

75. Salama, M. and Garba, J., Simultaneous Optimization of Controlled Structures. Computational Mechanics, 1988. **3**: p. 275.

76. Shin, Y.S., Haftka, R.T., Watson, L.T., and Plaut, R.H., Tracing Structural Optima as a Function of Available Resources by a Homotopy Method. Computational Methods in Applied Mechanical Engineering, 1988. **70**: p. 151.

77. Rakowska, J., Haftka, R.T., and Watson, L.T., An Active Set Algorithm for Tracing Parametrized Optima. Structural Optimization, 1991. **3**: p. 29.

78. Deb, K., Optimization for Engineering Design: Algorithms and Examples. 1995, New Delhi: Prentice Hall.

79. Hu, X. and Eberhart, R. *Solving* Constrained Nonlinear Optimization Problems with Particle Swarm Optimization. In *6th World Multiconference on Systemics, Cybernetics and Informatics*. 2002. Orlando, USA.

80. Kennedy, J., Minds and Cultures: Particle Swarm Implications for Beings in Sociocognitive Space. Adaptive Behavior, 1999. **7**(3-4): p. 269.

81. Schmitz, T. and Ziegert, J., Examination of Surface Location Error Due to Phasing of Cutter Vibrations. Precision Engineering-Journal of the American Society for Precision Engineering, 1999. **23**(1): p. 51.

82. Smith, S. and Tlusty, J., An Overview of Modeling and Simulation of the Milling Process. Journal of Engineering for Industry-Transactions of the ASME, 1991. **113**(2): p. 169.

83. Guerra, R.E.H., Schmitt-Braess, G., Haber, R.H., Alique, A., and Alique, J.R., Using Circle Criteria for Verifying Asymptotic Stability in Pl-Like Fuzzy Control Systems: Application to the Milling Process. Iee Proceedings-Control Theory and Applications, 2003. **150**(6): p. 619.

84. Kim, S.I., Landers, R.G., and Ulsoy, A.G., Robust Machining Force Control with Process Compensation. Journal of Manufacturing Science and Engineering-Transactions of the ASME, 2003. **125**(3): p. 423.

85. Rober, S.J., Shin, Y.C., and Nwokah, O.D.I., A Digital Robust Controller for Cutting Force Control in the End Milling Process. Journal of Dynamic Systems Measurement and Control-Transactions of the ASME, 1997. **119**(2): p. 146.

86. Pandit, S.M., Modal and Spectrum Analysis*: Data Dependent Systems in State Space*. 1991: John Wiley and Sons Inc.

87. Ewins, D.J., Modal Testing: Theory and Practice. 1984, New York: John Wiley and Sons Inc.

88. Guide to the Expression of Uncertainty in Measurement, in *ISO*. 1995.

89. McKay, M.D. and Beckmann, R.J., A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. Technometrics, 1979. **21**(2): p. 239.

90. Stein, M., Large Sample Properties of Simulation Using Latin Hypercube Sampling. Technometrics, 1987. **29**(2): p. 143.

91. Cheng, J. and Druzdzel, M.J. Latin Hypercube Sampling in Bayesian Networks. In *Thirteenth International Florida Artificial Intelligence Research Society*. 2000. Menlo Park, CA: AAAI Press.

92. Budak, E., Altintas, Y., and Armarego, E.J.A., Prediction of Milling Force Coefficients from Orthogonal Cutting Data. Journal of Manufacturing Science and Engineering-Transactions of the ASME, 1996. **118**(2): p. 216.

93. Altintas, Y., Manufacturing Automation : Metal Cutting Mechanics*, Machine Tool Vibrations, and CNC Design*. 2000, Cambridge: Cambridge University Press. 286.

94. Khuri, A.I. and Cornell, J.A., Response Surfaces: Design and Analysis. 1996, New York: Marcel Dekker Inc.

95. Zellner, A., An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests for Aggregation Bias. Journal of American Statistic Association, 1962. **57**: p. 348.

BIOGRAPHICAL SKETCH

Mohammad Kurdi was born and raised in the suburbs of Amman, Jordan. He finished his B.Sc degree in mechanical engineering in 1995 from the University of Jordan. He worked in his dad's family optometric practice while attending graduate school at the University of Jordan, where he obtained an M.Sc. in mechanical engineering in 1999. After graduate school he joined Royal Jordanian Airlines as an aircraft maintenance engineer where he worked for one year then he moved to Jordan Petroleum Refinery Co. and worked for 2 years as a development engineer. In August 2002 he enrolled in the mechanical engineering graduate program at the University of Florida where he obtained a Master of Science in December, 2003 and a Ph.D degree in August, 2005.