Milling stability map identification and machining parameter optimization using Bayesian inference

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Aaron William Cornelius

May 2024

© by Aaron William Cornelius, 2024
All Rights Reserved.

Acknowledgments

First, thank you to my friends and family. You have stood by me every step of the way and supported me through good and bad. Next, to my committee members, Drs. Bradley Jared, Tony Shi, and Jaydeep Karandikar, for all their help and feedback. This dissertation would not have been the same without their contributions. To all of my fellow students for helping me: you've been a constant sounding board over the last four years and I couldn't have finished without you.

Lastly, and most importantly, I would like to thank everyone who has provided mentorship and guidance for me: George, Sarah, Sebastian, Toby, Stan, Andrew, Jaydeep, and, of course, Dr. Schmitz. You have all helped me pursue my interests, opened countless doors for me, and encouraged me to grow both as a researcher and as a person. I wouldn't be where I am without your help. Thank you.

Abstract

This dissertation describes a physics-guided Bayesian learning approach for statistically modelling and optimizing machining processes under a state of uncertainty. This approach uses a series of automatically-selected cutting tests to refine uncertainties about the machining system's dynamics and cutting force and identify higher productivity cutting parameters. The algorithm is evaluated experimentally and compared to the cutting tool manufacturer's recommendations, both in laboratory conditions and in an industrial setting to optimize the machining process for a large aluminum component. These results show that the proposed Bayesian model can quickly identify both highly-productive machining parameters and accurate information about the underlying system in a small number of cutting tests, providing a robust solution for optimizing machining parameters without requiring any specialized equipment or measurement tools.

Table of Contents

1 Introduction **1**

1.1 Broader impacts . . . 2

1.2 Dissertation organization . . . 3

2 Background and Literature Review **4**

2.1 Introduction to machining . . . 4

2.1.1 Cutting force models . . . 9

2.1.2 Frequency response function . . . 11

2.1.3 Alternatives for cutting parameter selection . . . 14

3 Bayesian inference and prior work **18**

3.1 Bayesian inference . . . 18

3.2 Bayesian machining modelling . . . 27

4 Bayesian machining modelling **30**

4.1 Prior . . . 32

4.1.1 Cutting force model . . . 35

4.1.2 Frequency response model . . . 35

4.2 Physics models . . . 41

4.2.1 Stability map algorithms . . . 42

4.2.2 Cutting power . . . 53

4.2.3 Spindle bending moment . . . 55

4.2.4 Tool bending stress . . . 57

4.3 Test selection . . . 62
4.3.1 MRR maximization with risk tolerances . . . 64
4.3.2 Expected reduction in uncertainty . . . 67
4.4 Defining the posterior distribution . . . 72
4.4.1 Stability classification likelihood . . . 73
4.4.2 Chatter frequency likelihood . . . 77
4.4.3 Spindle power likelihood . . . 78
4.4.4 Bending stress likelihood . . . 80
4.5 Sampling the posterior distribution . . . 81
4.5.1 Rejection sampling . . . 81
4.5.2 Updating sample weights . . . 82
4.5.3 Laplace's approximation . . . 83
4.5.4 Markov Chain Monte Carlo (MCMC) sampling . . . 84

5 Results and Discussion: Laboratory Testing 97
5.1 Case study 1: Risk tolerance and chatter frequency . . . 98
5.2 Case study 2: Process damping identification . . . 103
5.3 Case study 3: Beam priors and uncertainty reduction test selection . . . 109
5.4 Case study 4: Spindle power and uniform priors . . . 114

6 Results and Discussion: Industrial Testing 129

7 Conclusion and Recommendations 140
7.1 Summary . . . 140
7.2 Future work . . . 141

Bibliography 143

Appendices 151
A Bayesian coin flip modelling code . . . 151
B MCMC demonstration code . . . 154
C Bayesian framework code guide . . . 160

C.1 SimulationSettings.m 160
C.2 StartSimulation.m 167
C.3 ResumeSimulation.m 169

Vita **172**

List of Tables

2.1 List of machining parameters 5

5.1 Testing parameters for the MRR-maximization case study 99
5.2 Testing parameters for laboratory case study 2, from [51] 104
5.3 Comparison of Bayesian inference approaches for learning process damping [40]. 108
5.4 Testing parameters for the laboratory case study 3 110
5.5 Prior definition for the 6.35 mm endmill demonstration 110
5.6 List of cutting tests for test set 3. The f_n column gives the uncertainty for the natural frequency after each test cut, expressed as the mean μ plus or minus one standard deviation σ. 112
5.7 Testing parameters for laboratory case study 4 116
5.8 Prior for laboratory test set 4 116
5.9 Test cuts for laboratory case study 4 120
5.10 Comparison of the prior and posterior distributions for each system variable. 120

6.1 Bayesian model parameters for laboratory case study 130
6.2 Tool geometry for the industrial case study 130
6.3 Roughing baseline for all tools based on the cutting tool manufacturer's recommendations 132
6.4 Finishing baseline for all tools based on the cutting tool manufacturer's recommendations 132
6.5 Comparison of predicted and actual RCSA parameters for the cutting tools . 133
6.6 Final optimized roughing parameters for the industrial case study 138
6.7 Final optimized finishing parameters for the industrial case study 138

List of Figures

2.1 Comparison of stable (a) and unstable (b) surface finishes 5

2.2 (a) Regenerative effect for milling stability, Figure 4.20 from [3]. (b) Comparison of forces in stable and unstable cuts 5

2.3 Example stability map, calculated using the Altintas zero-order approximation frequency domain algorithm [4]. The optimal machining parameters (i.e. with the highest MRR) are shown by the star. 7

2.4 The same stability map shown in Figure 2.3 with an uncertainty boundary of $\sigma = 0.01$ mm. The gray colors indicate the probability that the true stability boundary lies within that region. 7

2.5 Normal and tangential force directions on the cutting tool 10

2.6 Process for fitting cutting forces to a series of cutting tests. (a) Piezoelectric cutting force dynamometer. (b) Fitting coefficients to a series of tests. 12

2.7 (a) Measuring an FRF using a modal hammer. (b) Measured FRF. 13

2.8 A sequence of cutting tests selected using the adaptive spindle speed method proposed by Smith [15]. 15

2.9 A sequence of cutting tests selected using the adaptive parameter method proposed by Bergmann. Figure 4(a) from [18]. 15

3.1 Prior distribution for the coin's fairness 20

3.2 Posterior distribution after experiencing a single heads (a) or tails (b). 22

3.3 Posterior distribution for the coin's fairness after (a) 10, (b) 100, and (c) 500 flips using the Gaussian prior distribution 22

3.4 Posterior distribution after 1072 flips. The stepping is due to accuracy limitations with double-precision floating point numbers. 24
3.5 Posterior distribution for the coin's fairness, calculated using the logarithmic equation after (a) 1072, (b) 10000, and (c) 22000 flips using the Gaussian prior distribution 24
3.6 Posterior distribution for the coin's fairness after (a) 10, (b) 100, and (c) 1000 flips using the uniform prior distribution 26
3.7 Prior (a) and posterior (b) probabilistic stability map, Figures 1 and 4 from [33]. The color indicates the probability of stability: darker colors indicate that a cut is more likely to be stable, while lighter colors indicate a cut is more likely to chatter. 29
3.8 Tests and posterior probabilistic stability map, Figure 18 from [36]. The blue solid line represents the 50% probability of stability boundary from the prior, while the solid red line is the 50% probability of stability in the posterior. . . 29

4.1 Flowchart outlining the steps in the Bayesian inference framework 31
4.2 Example of a 3-dimensional prior distribution. The on-diagonal plots show the marginal distributions for each element, while the off-diagonal plots show the conditional distributions to help identify correlation between variables. . 34
4.3 Modelling the tool FRF using Euler-Bernoulli beam theory. (a) A 6.35mm endmill with a 10:1 L/D ratio. (b) Modelling the tool as a fixed-free beam with an end load. (c) Comparison between the measured and predicted FRFs. 37
4.4 (a) Forces and moments applied to fixed-free and free-free beams in the X direction, Figure 7.15 from [3]. (b) Those components coupled together into a single body, Figure 7.14 from [3]. 38
4.5 Receptance coupling substructure analysis for predicting the tool tip FRF, Figure 1 from [49]. 40

4.6 Demonstration of how the prior is propagated through the stability algorithm, using the zero-order approximation frequency-domain algorithm. (a) Probabilistic stability map. The color of each point indicates the probability that a cut would be stable, with darker colors indicating a lower probability of stability. (b) Chatter frequency distribution. Darker colors indicate a higher likelihood of seeing specific chatter frequencies. 43
4.7 Bifurcations in the stability lobe, not accurately modelled by the ZOA algorithm. Figure 4 from [55]. 45
4.8 Bayesian stability updating using a surrogate model trained on a discrete stability algorithm, Figure 4.7 from [37]. The black curve is the prior stability map, while the red curve represents the posterior isocurve showing 50% of stability. 47
4.9 (Top) Stability map calculated using the ZOA algorithm. Dashed lines are possible stability limits given by the $\tilde{b}_{lim}$ values. The solid line is the b_{lim} value. (Bottom) Chatter frequencies for the stability map. Dashed lines are possible chatter frequencies, while the solid line is the dominant chatter frequency associated with b_{lim}. 50
4.10 Cutting tool divided into discrete elements for time domain simulation, Figure 4.55 from [3]. 52
4.11 TDS-calculated stability map, compared to the ZOA algorithm. 54
4.12 Spindle power curve for a Haas VF4SS machining center [57]. 54
4.13 (a) Long tools will experience high bending moments due to cutting forces [58]. (b) CAT40 toolholder exhibiting fretting damage at the top and bottom of the taper [59]. 56
4.14 (a) A long endmill which is susceptible to breaking due to excessive deflection. (b) That same tool modeled as a clamped-free beam. 58
4.15 Predicted bending stress and acceptable cutting parameter limits for the two flute endmill shown in Figure 4.14 61

4.16 Expected value test selection, Figure 20 from [35]. The left image shows how the second test was selected at the point with the highest value, and the right image shows the posterior stability map after running that test. 63
4.17 Stability uncertainty test selection from [34]. (a) Prior, Figure 7. (b) Posterior, Figure 8. The green line is the nominal stability boundary calculated using the ZOA. 65
4.18 Entropy-based test selection, Figure 12 from [39]. The blue diamond is the point with highest possible information about the stability map, while the red star is the lowest axial depth point with at least $\lambda\mathcal{I}_{max}$ informativeness. . . . 65
4.19 Probabilistic stability map for demonstrating uncertainty reduction test selection 69
4.20 Expected reduction in system uncertainty, considering only binary stable/unstable test results, for (a) K_s, (b) β, (c) f_n, (d) k, and (e) ζ. 70
4.21 Expected reduction in system uncertainty considering both stability and chatter frequency likelihood for (a) K_s, (b) β, (c) f_n, (d) k, and (e) ζ. 71
4.22 Calculating Δ_b, Δ_{f_c} for a given sample, used for calculating the stability likelihood using the Gaussian dropoff method. (Top) Δ_b is the difference between the predicted limiting depth of cut, b_{lim} (line), and the unstable axial depth ($\times$). (Bottom) Δ_{f_c} is the difference between the actual ($\times$) and predicted (line) chatter frequencies. 74
4.23 Comparison between different stability likelihood functions for a stable cut (Gaussian dropoff, sigmoid, and CDF). The Gaussian dropoff and CDFs both use $\sigma_S = 1$ mm, while the sigmoid uses $k = 0.6$ mm so it closely matches the Gaussian CDF. 76
4.24 Spindle current drive example, Figure 10(a) from[67] 79
4.25 Demonstration of the Metropolis MCMC algorithm. (a) Initial starting point and proposal distribution. (b) Evaluating the first candidate point. (c) Evaluating the second candidate point. (d) The chain after rejecting the last move. 87

4.26 Demonstration of the Metropolis MCMC algorithm showing how it approaches the target marginal distribution over time after 25 (a), 100 (b), 1000 (c), and 10000 (d) samples. 88
4.27 Demonstration of how a poor starting point can bias the output chain 89
4.28 Comparison of MCMC proposal distributions for exploring a unit normal distribution. (a) $\sigma = 0.25$, (b) $\sigma = 1$, (c) $\sigma = 4$ 91

5.1 Machining setup for the MRR-maximization case study 99
5.2 Cutting test series with risk tolerance of 50% and no chatter frequency updating 101
5.3 Cutting test series with risk tolerance of 50% and chatter frequency updating using $\sigma_{f_c} = 50$ Hz 101
5.4 Comparison of the posterior stability maps after observing an unstable cut, based on whether the chatter frequency is not (left) or is (right) included in the Bayesian updating 102
5.5 Cutting test series with risk tolerance of 75% and chatter frequency updating using $\sigma_{f_c} = 50$ Hz 102
5.6 Process damping mechanic at the cutting edge, from [82] 104
5.7 Process damping tests from [51]. (a) Test setup with workpiece flexure (Figure 11 in source). (b) High spindle speed test results (Figure 12 in source). (c) Low spindle speed cutting test results showing the effect of process damping (Figure 17 in source). 105
5.8 Stability maps showing the separated learning process for learning cutting tests. (a) and (d) show the prior distribution at high and low speeds respectively. (b) and (e) show the posterior after the cutting force learning. (c) and (f) show the posterior after the process damping learning. 108
5.9 Setup for lab test 3 111
5.10 Prior probabilistic stability map for laboratory test case 3 111
5.11 Cutting test series for laboratory case study 3, selected using the MRR-maximizing strategy. 113

5.12 Cutting test series for laboratory case study 3, selected using the hybrid exploration/exploitation testing series. 115
5.13 Setup for lab test 4. (a) Full setup. (b) Closeup of the tool and workpiece. . 115
5.14 Rogowski coil setup in the machine tool control cabinet for measuring spindle power. 117
5.15 Prior probabilistic stability map for case study 4. 119
5.16 Posterior after the first unstable cut. (a) Posterior probabilistic stability map. The point with both an unstable and stable cut was stable at 0.06 mm feed and unstable at 0.03 mm feed. (b) Posterior distributions for the natural frequencies. 119
5.17 Posterior stability map and cutting tests, (a) with and (b) without using the measured spindle power in the Basyesian updating. Cases where both a stable and unstable test are overlapping indicate that the 0.03 mm feedrate test chattered while the 0.06 mm feedrate test was stable. 121
5.18 Distribution of samples drawn from the prior distribution for case study 4 . . 123
5.19 Posterior distribution for case study 4, without spindle power in the Bayesian updating 124
5.20 Posterior distribution for case study 4, with spindle power in the Bayesian updating 125
5.21 Frequency-domain microphone data collected from test cut 5 at 0.03 mm feed per flute. While the peak at 1632 Hz has the highest magnitude, the 1215 Hz peak is actually the dominant chatter frequency. 127

6.1 Shift in the stability map due to switching between two nominally identical machine tools. 135
6.2 Engagement between the tool and workpiece as the tool moved into the corner of a pocket. 137

Chapter 1

Introduction

While physics-based models are capable of accurately simulating machining processes and predicting their outcomes, it is generally difficult to measure the inputs required by these simulations. For example, existing methods to predict machining stability require expensive instruments to measure the stability map inputs (i.e., the machining system structural dynamics and workpiece/tool cutting force model) or many time-consuming cutting tests to experimentally determine the stability map. Therefore, even when a machining operation has been simulated to predict stability or some other factor, in practice there is considerable uncertainty about the outcome.

This uncertainty takes two forms. First, there is epistemic uncertainty in system parameters like the dynamics, occurring either because the system has not been measured directly or because of the uncertainty in those measurements[1]. Second, there is aleatoric uncertainty which captures errors in simulated predictions that would occur even if the system parameters were perfectly known, for example due to approximation errors in the simulation itself. Since this error cannot be captured by the simulated model, it must be treated as essentially random.

This work describes a physics-informed Bayesian framework for modelling and simulating machining processes under a state of uncertainty. Initial probabilistic guesses about the system, such as the cutting forces and the frequency responses, are propagated through physics-based machining simulations like stability or bending stress models. Those guesses

[1]Note that there are different ways of defining and dividing epistemic and aleatoric uncertainty [1].

are then updated based on the results of automatically-selected cutting tests. As more test cuts are performed the model inputs converge towards their true values, ultimately reducing uncertainty about both the machining simulations and the underlying system parameters. This model is flexible and can be tailored for different applications by selecting different physics algorithms, test selection methods, and likelihood functions.

After the general Bayesian framework is established, it is applied to optimize machining parameters and select highly productive cutting parameters, improving on existing methods for cutting parameters by reducing the number of test cuts and incorporating additional constraints on machining parameters. This is accomplished by establishing more accurate priors for the model inputs using physics-based methods, adding physics-based models to identify machining parameters that may exceed machine tool capabilities (e.g. the available spindle torque), selecting tests to maximize information gain and incorporate user risk preference, and incorporating the chatter frequency and cutting torque into the Bayesian updating.

The research outcomes were evaluated in two primary ways. First, a combination of simulations and test cuts were used to develop and evaluate the methods in a laboratory setting. Test cuts were performed on computer numerically controlled (CNC) machining centers, and the results were compared to direct measurements. Second, the methods was deployed in a production setting and used to optimize an example machining operation for an industrial partner. This application-focused testing evaluated the number of cutting tests required to select optimal parameters and the productivity of those parameters compared to the tool manufacturer's recommendations.

1.1 Broader impacts

Currently, there are great concerns about the capacity of the US manufacturing base, particularly regarding defense and healthcare. For example, the United States Industrial Base and Sustainment Program highlighted the lack of domestic machine tool production as a critical security risk [2]. By optimizing machining parameters and reducing part cycle times, the Bayesian inference parameter identification method described here can increase

the capacity of existing manufacturing equipment and reduce the need for additional machine tools to meet current demand. The reduced part cost and lead time will add value to the US manufacturing industry and economy. This could also lead to job creation based on the new positions required to support the increased manufacturing capacity.

1.2 Dissertation organization

The remainder of the dissertation is organized as follows. Chapter 2 provides an introduction to machining dynamics and stability and reviews the current state-of-the-art for stability modelling and cutting parameter selection. Chapter 3 provides an introduction to Bayesian inference and probabilistic modelling and reviews the prior literature for how Bayesian modelling has been applied to machining. Chapter 4 then defines the proposed Bayesian stability model, breaking it down into individual components and discussing how those components interact. Chapters 5 and 6 show specific examples of how the Bayesian stability model can be applied, showing testing results in a laboratory setting and at an industrial partner respectively. Chapter 7 summarizes the results, presents conclusions, and discusses opportunities for future research and work. Appendices A and B contain code for specific examples demonstrating Bayesian inference and Markov Chain Monte Carlo sampling. Finally, Appendix C contains a brief introduction to the demonstration codebase that is published alongside this dissertation.

Chapter 2

Background and Literature Review

2.1 Introduction to machining

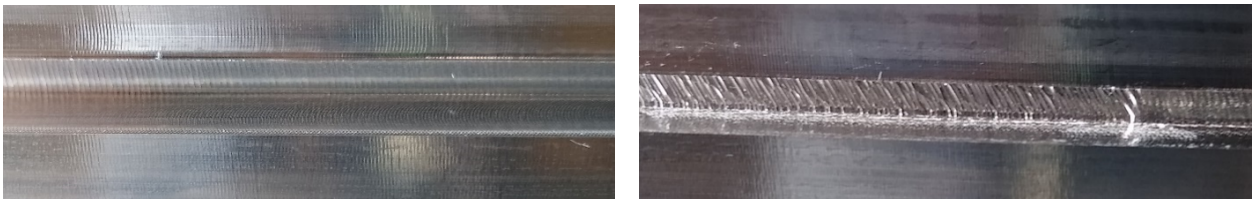
A major challenge in milling is selecting cutting parameters, such as spindle speed and depth of cut [3]. Productive machining parameters enable parts to be machined more quickly and economically. One common way to quantify this productivity is the material removal rate (MRR) defined in Equation (2.1), which is the volume of material machined away per unit time for some set of cutting parameters ω, as defined in Table 2.1:

$$MRR(\omega) = \omega_a \omega_b \omega_n \omega_{f_z} N_t \tag{2.1}$$

Productivity can be improved by increasing the radial width, axial depth, feed, or spindle speed; however, these parameters are subject to constraints. One of the most important constraints is chatter, a self-excited vibration that occurs at some machining parameters and can result in scrapped workpieces and damage to the tool and machine. Figure 2.1 compares the workpiece surface finish resulting from stable and unstable/chatter combinations of machining parameters. Chatter occurs due to the relative vibration of the cutting tool, machine structure, and workpiece. This vibration leaves varying amounts of stock on the machined surface, which is encountered by the next flute of the tool (Figure 2.2a). The higher chip thickness results in higher cutting force and thus larger vibrations. This feedback loop is referred to as the regenerative effect. In a stable cut, the cutting force and deflection reach

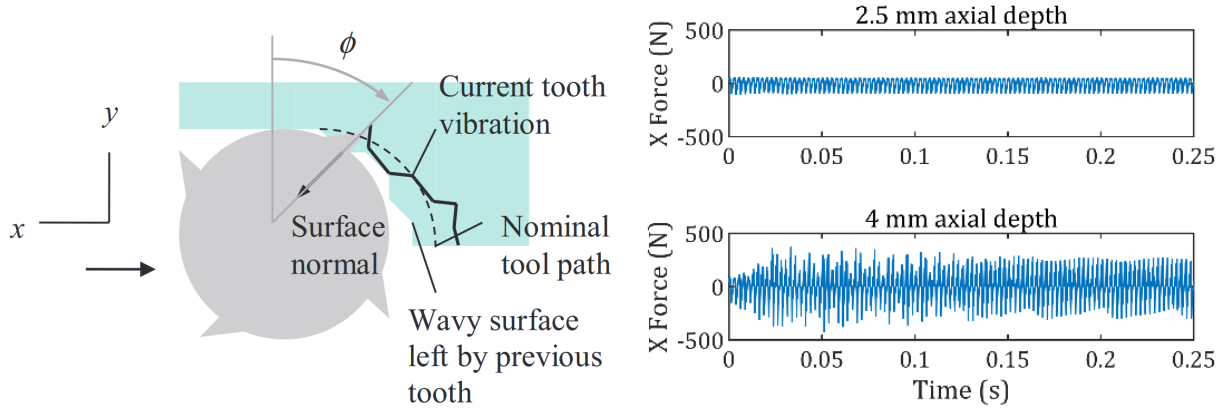
Table 2.1: List of machining parameters

Variable	Value	Units
ω_n	Spindle speed	rpm
ω_a	Radial depth of cut	mm
ω_b	Axial depth of cut	mm
ω_{f_z}	Feed per flute	mm/flute
ω_ϕ	Feed direction	degrees
ω_θ	Cut direction (climb vs conventional)	
N_t	Number of flutes	



(a) **(b)**

Figure 2.1: Comparison of stable (a) and unstable (b) surface finishes



(a) **(b)**

Figure 2.2: (a) Regenerative effect for milling stability, Figure 4.20 from [3]. (b) Comparison of forces in stable and unstable cuts

a stationary periodic equilibrium which repeats for every passing tooth. In a chattering (or unstable) cut, the cutting force and deflection increase exponentially and do not repeat with every flute pass, until the cutting edge of the tool becomes disengaged from the cut: see Figure 2.2b. This effect is governed by three primary variables: the dynamic stiffness of the tool/holder/spindle/machine/workpiece assembly, the force required for the specific tool to cut the work material, and the selected cutting parameters.

For a given tool assembly and workpiece, the combinations of cutting parameters that lead to chatter can be visualized using the stability map. Figure 2.3 displays an example stability map, showing how stability depends on axial depth of cut and spindle speed for a selected radial width of cut. Cuts using spindle speed and axial depth combinations which lie below the blue stability boundaries will be stable, while cuts above the boundary will chatter. The stability map can be used to select the most productive machining parameters ω_{opt} for that application using Equation (2.2), where $b_{lim}(n)$ is the limiting axial depth from the stability map at the spindle speed n and Ω is the set of all possible cutting parameters for that cutting tool and machine combination (for example, considering the machine's maximum spindle speed).

$$\omega_{opt} = \arg\max_{\omega \in \Omega}(MRR(\omega)) \text{ s. t. } \omega_b < b_{lim}(\omega_n) \tag{2.2}$$

However, even if the system parameters have been measured it is generally not ideal to jump straight to the peak of the stability lobe due to uncertainty. While the stability map shown in Figure 2.3 treats the stability map as purely deterministic, the predicted stability map will generally not be perfectly accurate and it is possible to experience chatter below that boundary or stable cutting above it. Figure 2.4 applies an uncertainty of 0.01 mm to the predicted stability boundary. Instead of determining whether a point will be stable or unstable based on whether it is above b_{lim}, the probability of stability $P_S(\omega)$ is 50% exactly at that line, increases at lower axial depths, and decreases as higher axial depths (see Section 4.4.1 for more detailed calculations.) Given this uncertainty, selecting the best cutting parameters can be seen as a stochastic optimization problem. Equation (2.3) shows a simple version which tries to maximize the expected MRR $E(MRR(\omega))$. Note that this

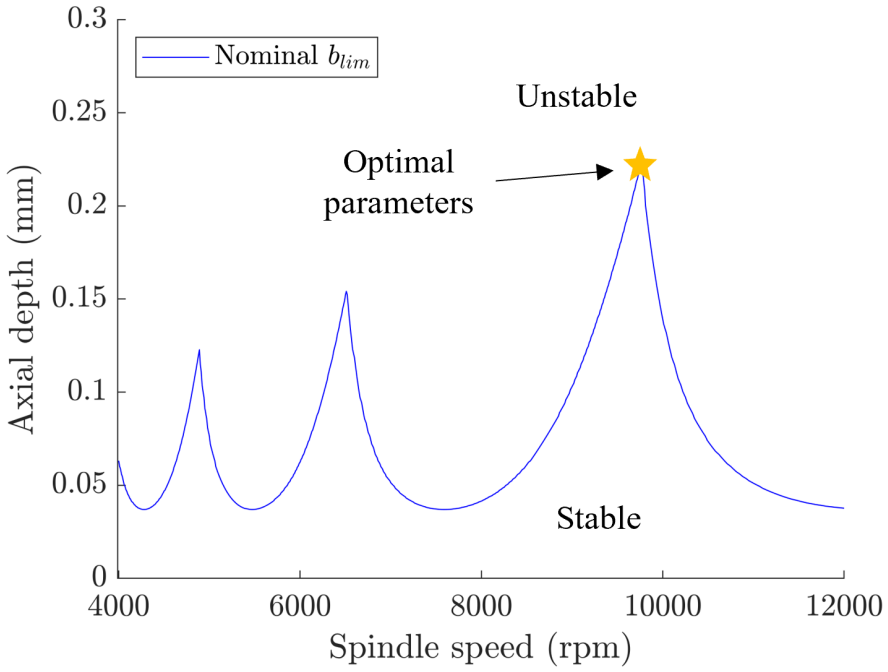


Figure 2.3: Example stability map, calculated using the Altintas zero-order approximation frequency domain algorithm [4]. The optimal machining parameters (i.e. with the highest MRR) are shown by the star.

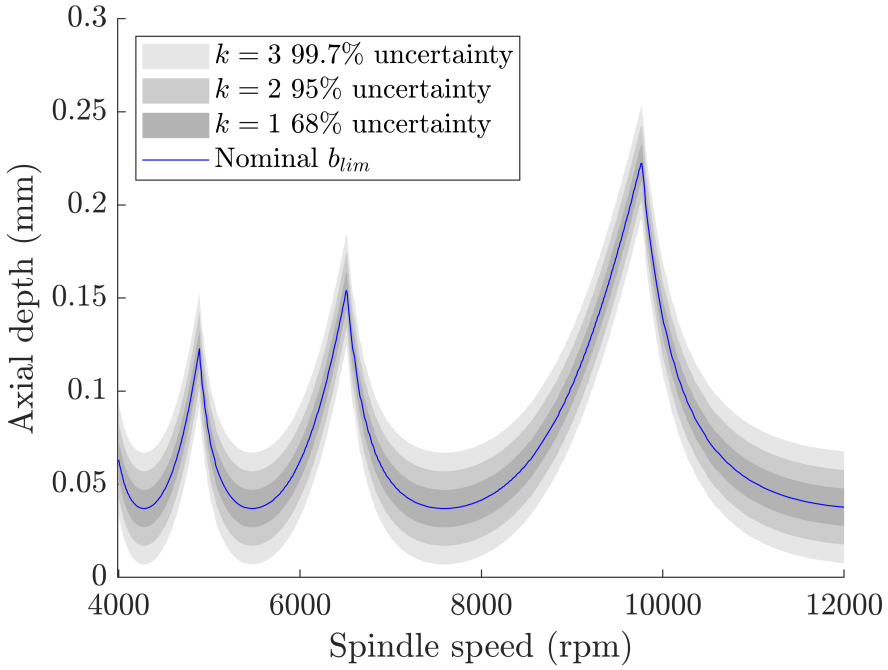


Figure 2.4: The same stability map shown in Figure 2.3 with an uncertainty boundary of $\sigma = 0.01$ mm. The gray colors indicate the probability that the true stability boundary lies within that region.

uncertainty is associated with the user's ability to predict the outcome of the cutting test, not the outcome itself. If a test at point ω is stable, then barring any changes in the system any future cutting tests at the point will be stable.

$$\omega_{opt} = \arg\max_{\omega \in \Omega}(E(MRR(\omega))) = \arg\max_{\omega \in \Omega}(MRR(\omega) \cdot P_S(\omega)) \tag{2.3}$$

Why does this uncertainty occur? One major cause is the difficulty of predicting whether a cut will be stable. The tool's vibration is governed by the equation of motion shown in Equation (2.4), where x is a vector with the current system position in both the X and Y axes, $[m]$, $[c]$ and $[k]$ are the mass, damping, and stiffness matrices, and F is the instantaneous cutting force[1]. Determining whether a set of cutting parameters will be stable or not requires solving this equation of motion, but this is non-trivial since the driving force F is not only time-dependent but also depends on the position of the system at time $t - \tau$, where τ is the time interval between flute passes. There is no complete closed-form analytical solution to solve this second-order delayed differential equation with periodically varying coefficients and determine stability.

$$[m]\ddot{\mathbf{x}} + [c]\dot{\mathbf{x}} + [k]\mathbf{x} = F\left(t, \mathbf{x}(t), \mathbf{x}(t-\tau)\right) \tag{2.4}$$

However, there are a variety of approximate solutions, such as the frequency domain algorithm used to generate Figure 2.3. These algorithms use the system dynamics and tool/workpiece cutting force model to predict which cutting parameters will be stable or unstable[2]. These inputs depend not just on the cutter being used, but also on the tool holder, machine tool, and tool/workpiece material combination, and must therefore be measured for each individual tool. There are commercially available kits for performing these measurements and calculating stability maps (e.g. [7]). Machine shops who have applied these methods have seen dramatic improvements in productivity; however, the cost, expertise required to perform the measurements, and various approximations in the physics models

[1]Note that this equation still contains some approximations, assuming constant flute spacing on the tool and no process damping.

[2]For descriptions of specific algorithms, see Section 4.2.1, or consult [5, 6], which review many common stability algorithms.

have prevented widespread adoption. The following subsections provide a brief introduction to cutting force models and the frequency response and how they can be directly measured.

2.1.1 Cutting force models

The cutting force model describes how much force is required to shear away a chip from the workpiece. Cutting force is most commonly modeled as a mechanistic simulation which assumes that the force on a flute varies linearly with the thickness of the chip that is currently being formed. There are several different forms for this equation. The two-element force model uses Equation (2.5), where F_n and F_t are the cutting force components normal and tangential to the cutting edge in Newtons respectively (see Figure 2.5), β is the cutting force angle in degrees, K_s is the specific cutting force coefficient in $\mathrm{N/m^2}$ and b and h are the width and thickness of the chip being removed in meters [3]. Alternatively, the cutting force for this form can be expressed in terms of the tangential and normal cutting force coefficients k_{tc} and k_{nc}, as shown in Equation (2.6). These forms are equivalent, and it is easy to translate back and forth between the two different expressions. This approach assumes that the cutting force is directly proportional to the chip thickness.

$$F_n = \cos(\beta) K_s b h,\ F_t = \sin(\beta) K_s b h \tag{2.5}$$

$$F_n = k_{nc} b h,\ F_t = k_{tc} b h \tag{2.6}$$

The four-element force model extends this by adding tangential and normal edge coefficients k_{te} and k_{ne}, defining additional forces which are not proportional to chip thickness as shown in Equation (2.7). This helps model the effect of the tool's finite edge radius and more accurately models cutting forces at small chip thicknesses.

$$F_n = k_{nc} b h + k_{ne} b,\ F_t = k_{tc} b h + k_{te} b \tag{2.7}$$

Finally, the six-element adds axial cutting and edge coefficients k_{ac} and k_{ae} which model the force along the tool axis F_a, as shown in Equation (2.8). These axial forces usually do not affect stability since most tools are much stiffer in the Z direction than the XY axes,

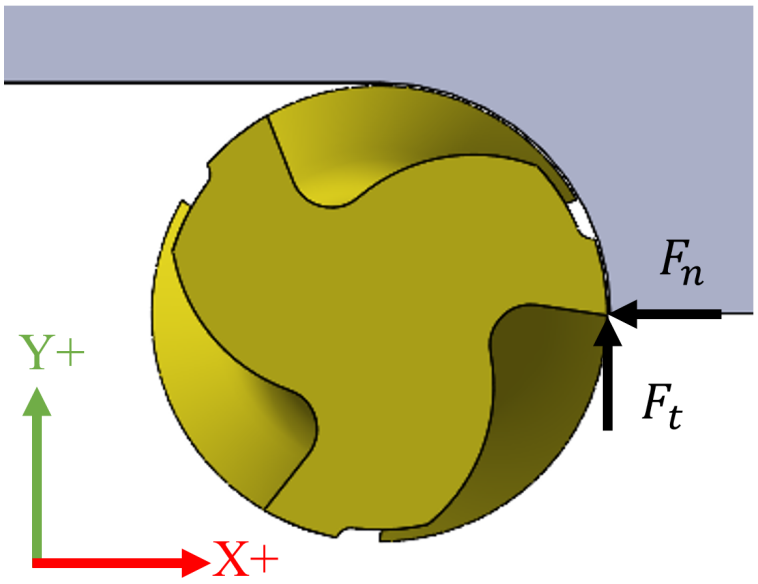


Figure 2.5: Normal and tangential force directions on the cutting tool

but can be important under some circumstances, for example when machining thin-floored components where excessive axial force could damage the part.

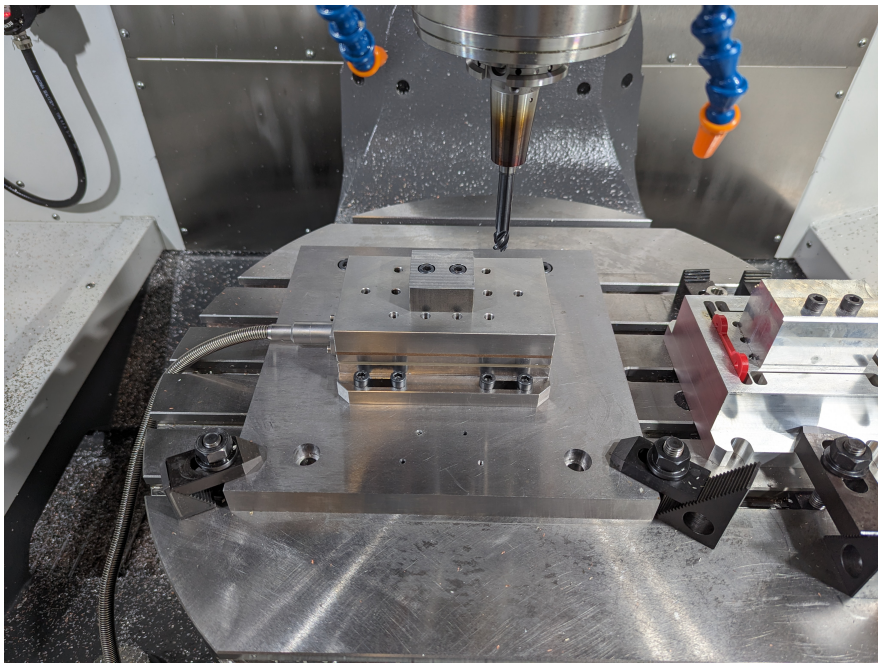
$$F_n = k_{nc}bh + k_{ne}b,\ F_t = k_{tc}bh + k_{te}b,\ F_a = k_{ac}bh + k_{ae}b \tag{2.8}$$

The cutting force model can be measured by running a series of cutting tests at different feed per flute values and measuring the cutting forces. The forces from each cut are averaged to remove the time-dependent variation in the force, and the cutting force coefficients are fit to minimize the root mean square error between the measured and modeled forces. This process is illustrated in Figure 2.6. There are also nonlinear optimization techniques which fit the instantaneous forces as the tool rotates through the cut [8]. These approaches are more flexible, allowing the user to fit alternative nonlinear force models. However, all these approaches require a cutting force dynamometer to accurately measure the machining forces. These devices can be extremely expensive, easily costing upwards of $50000, and are thus not feasible to purchase for most machine shops. Additionally, dynamometers have their own dynamics, requiring filtering to correct the measurements when the dynamometer's natural frequency is excited [8].

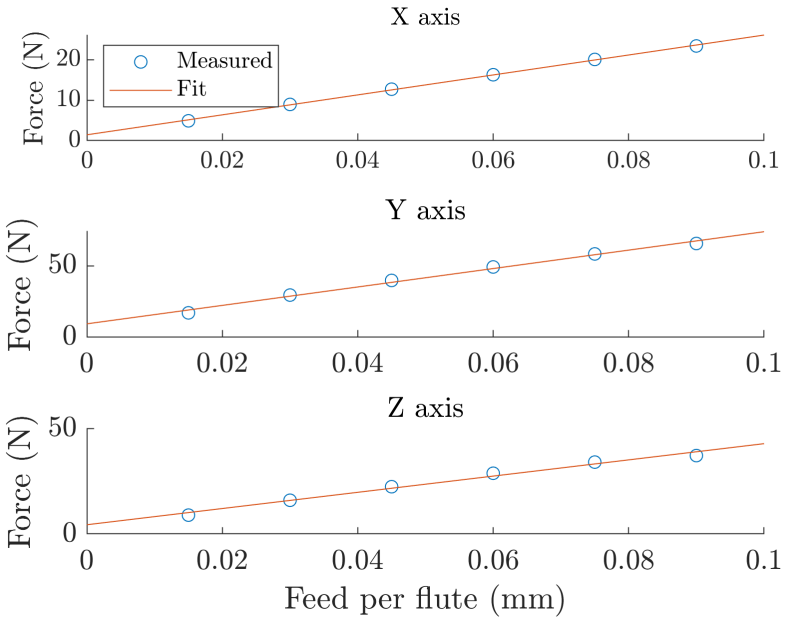
One alternative is using the spindle power to estimate the cutting forces: however, this can only measure the tangential component of the machining force since radial forces do not contribute to the spindle torque [9]. Other authors have used the axis motor current to estimate the cutting forces, but this requires significant filtering to compensate for the motor and control dynamics and can have lower accuracy when the machine is performing complex multi-axis contouring motions [10].

2.1.2 Frequency response function

The frequency response function (FRF) describes the vibration of an object in response to oscillating forces. Figure 2.7 shows an example tool tip FRF measured by modal tap testing, where a modal testing hammer is used to apply an impulse to the tool and the resulting tool vibration is recorded using a piezoelectric accelerometer [3]. However, this equipment is again expensive and few machine shops have access to tap testing equipment

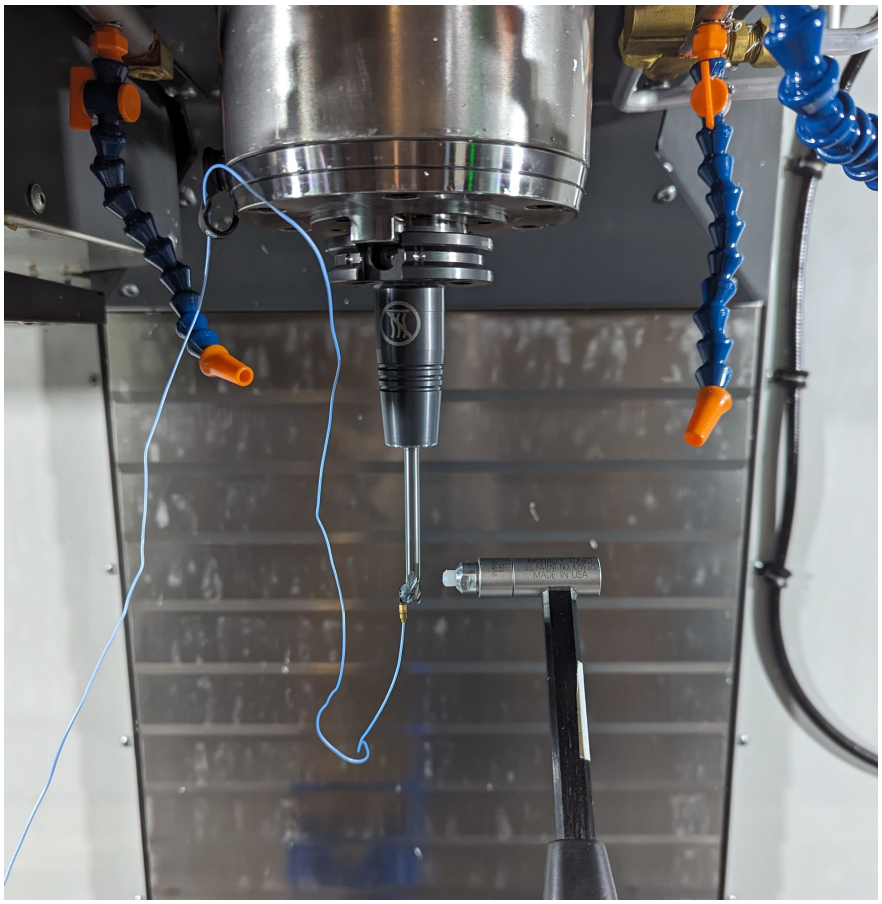


(a)

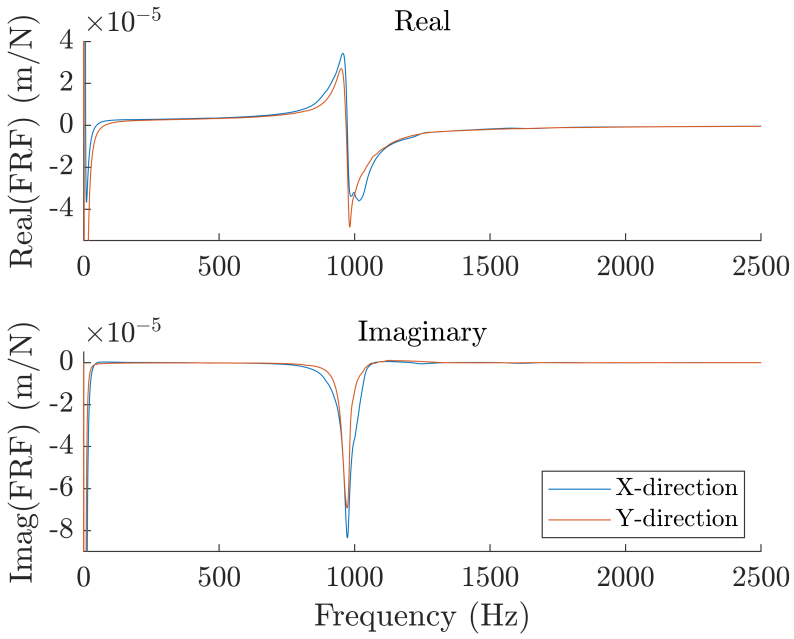


(b)

Figure 2.6: Process for fitting cutting forces to a series of cutting tests. (a) Piezoelectric cutting force dynamometer. (b) Fitting coefficients to a series of tests.



(a)



(b)

Figure 2.7: (a) Measuring an FRF using a modal hammer. (b) Measured FRF.

or the expertise to use it. Furthermore, tap tested dynamics are not perfectly accurate: mounting an accelerometer to the cutting tool can alter the natural frequency due to the additional mass loading [11], and spindle dynamics can shift when the bearings are rotating [12]. Therefore, even if the tool has been tap tested there may be some errors in the measured FRF which can affect the calculated stability map. An alternative way to measure the FRF is operational modal analysis, which analyzes the vibration of the tool or workpiece during machining to calculate the frequency response [13, 14]. However, this requires measurements using capacitance probes or laser vibrometers. Due to these restrictions and the high cost of measurement equipment, tap testing has not achieved widespread use in industry.

2.1.3 Alternatives for cutting parameter selection

Several algorithms have been proposed for on-machine adaptation of machining parameters which use basic knowledge of the machining process to recommend new cutting parameters when chatter is encountered. Smith and Delio presented the initial method, which uses the chatter frequency obtained using a microphone to calculate the new spindle speed and iterates until stable cutting parameters are encountered [15]. A similar method was described by Tarng and Li [16]. Yamato *et al.* extended this method to multi-spindle machining of thin workpieces [17]. However, these approaches are limited since they only adapt the spindle speed and not the axial depth. If the pre-selected axial depth is too low, then the approach will not identify the best possible MRR. If the axial depth is too high, on the other hand, then there may not be any spindle speed which will be stable. Figure 2.8 demonstrates this approach. These approaches also only identify a single set of stable cutting parameters, not the full stability map. They are also not well-suited to complex machining systems with multiple intersecting modes, since that can result in multiple conflicting chatter frequencies. Bergmann *et al.* proposed a method for online adaptation of both spindle speed and axial depth using a machine learning surrogate stability map [18]. However, this approach does not incorporate any physical knowledge of the chatter mechanism and requires many test cuts to converge close to the optimal machining parameters (17 tests in their example, shown in Figure 2.9).

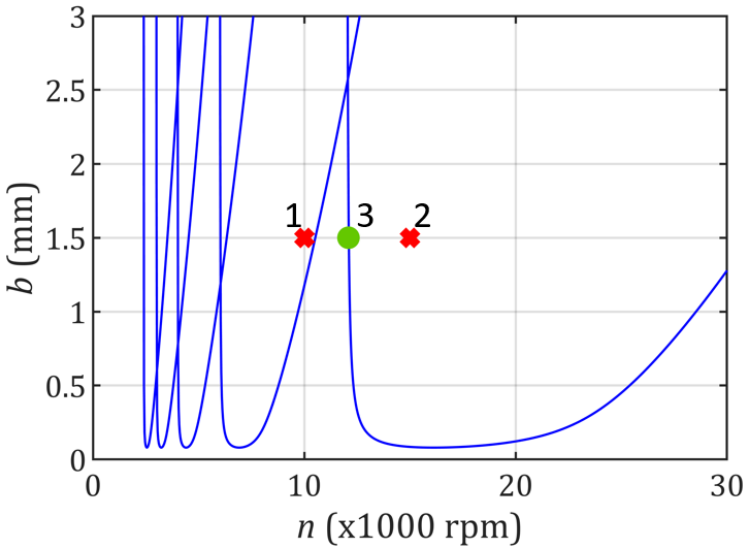


Figure 2.8: A sequence of cutting tests selected using the adaptive spindle speed method proposed by Smith [15].

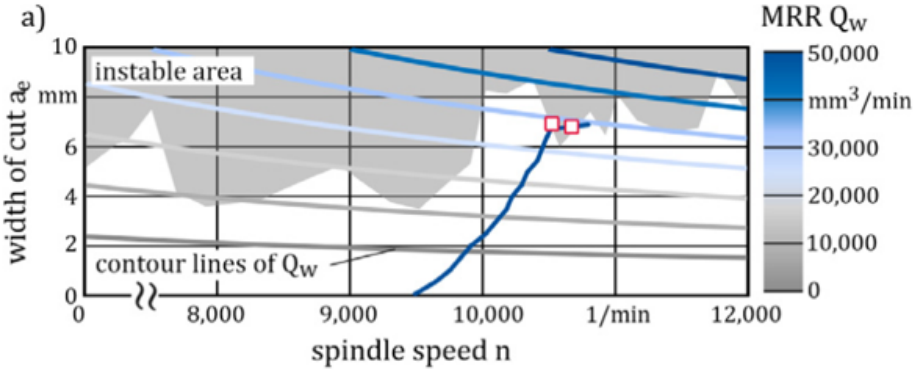


Figure 2.9: A sequence of cutting tests selected using the adaptive parameter method proposed by Bergmann. Figure 4(a) from [18].

Machine learning models have been proposed as an alternative method for identifying the stability map and have been a popular research topic in recent years. These methods avoid the need to directly measure the tool. Instead, they are trained on points on the stability map. As more points are added, the machine learning model converges towards the true stability map. Several approaches have been tested, ranging from simple approaches like k-nearest neighbors [19] to Gaussian processes [20]. Unfortunately, these methods generally require too many training points to be practical. The hybrid reinforcement-learning/nearest-neighbor algorithm presented by Freidrich *et al.* required 300000 simulated test points to achieve 98% accuracy [21]. Denkena *et al.* compared different machine learning algorithms to see which could most accurately encode stability maps, achieving 94% accuracy when trained on 130 test cuts [22]. Greis *et al.* compared support vector machines, k-nearest-neighbors, and neural network models, using 2020 initial training points and then 130 new test points to update the stability map [19, 23]. Oleaga *et al.* applied random-forest classifiers which required 18 separate tap-testing measurements and stability maps to train their model [24]. Kawai *et al.* used more than 100 samples to train their Gaussian process (estimated from a scatter plot of their training data) [20].

The most successful machine learning models are those which incorporate physics knowledge to train and update the network. Postel *et al.* used a physics-based model to train an ensemble of deep neural networks, and then retrained the ensemble based on a series of cutting tests [25]. Compared to machine learning approaches which were not trained on physics models, this approach had higher accuracy with fewer test points. However, this method still requires many tests: 90 test cuts were required to achieve a prediction accuracy of 84%. Postel *et al.* also proposed a genetic algorithm neural network method for inverse parameter identification [26]. This method leveraged the chatter frequency to accurately identify the stability map with only 20 test cuts; however, it required manual parameter tuning and did not provide any guidance for how to pick the most informative test cuts, so it is not practical for use in production environments. Rezaei *et al.* trained a generative adversarial network to approximate stability maps based on the results of cutting tests [27]. This approach could often identify the stability map in 5-10 test cuts, but again did not assist in selecting cutting parameters to test, did not provide information on the underlying

system parameters, and required a large training corpus which includes the true stability map. Deng *et al.* reduced the number of cutting tests by using the limiting axial depth rather than binary stable/unstable data to train [28]. In practice, the limiting axial depth is difficult to measure and is usually inferred by running multiple test cuts at different depths, so it is not significantly more practical for production use.

Chapter 3

Bayesian inference and prior work

3.1 Bayesian inference

Bayesian inference is a method for updating initial assumptions about a probability as new observations are taken. Rather than coming up with a single prediction, Bayesian approaches construct a distribution of outputs based on the uncertainty in the model. Those distributions are then updated using Bayes' rule, shown in Equation (3.1). $P(A)$ is the initial probability assigned to A (referred to as the *prior*) and $P(A|B)$ is the new probability assigned to A after observing event B (referred to as the *posterior*). $P(B|A)$ is the probability of observing event B for a given set of values for A, and $P(B)$ is a normalization factor to ensure that the posterior is a valid probability distribution: i.e, that integrating $P(A|B)$ over the entire probability space gives a total probability of 100%. In practice, it is difficult to directly calculate $P(B)$ for most complex problems, so the non-normalized version $p(A|B)$ is used and normalized numerically.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \propto p(A|B) = P(B|A) \cdot P(A) \tag{3.1}$$

Physics-guided Bayesian inference is well-suited for discovering hidden information about a system which is difficult to measure directly. The true value of A may be unknown, but by making observations of B and connecting them to the probabilities of A using some

algorithm based on understanding of the system, it is possible to infer information that cannot be directly measured and provide robust error-bounds on that knowledge.

As a concrete example, imagine that you are flipping a coin of unknown provenance. You are concerned that this coin may be weighted to give unfair results. A standard coin would have 50% probability of getting heads and 50% probability of getting tails, but some quick testing shows that you're getting more heads than expected: in your first ten tosses, you get seven heads and only three tails[1]. However, you know that it is perfectly possible for a fair coin to generate these results. How likely is it that the coin is unfair?

First, it's important to define exactly what you mean by a "fair" coin. Let A be the probability that the coin will come up heads. An ideal fair coin would have $A = 50\% = 0.5$, but if the coin is unfair then A could be anywhere between 0 and 1. However, while an ideal coin has $A = 0.5$, for a realistic coin there will necessarily be some slight imbalance which may cause it to be biased, so you should define an acceptable tolerance for fairness, depending on what you use the coin for. If you are just using the coin for low-importance day-to-day flipping (e.g. to decide where to eat lunch), you might be willing to accept a slightly-flawed coin with $0.45 \leq A \leq 0.55$, and require a 95% confidence that the coin lies within that range.

Next, define a prior distribution for different values of A describing how likely they are, based on what you know about the coin. Here, you're fairly suspicious about the coin, and think there's a good chance that it's unfair. For this example, a simple normal distribution with mean μ and standard deviation σ $P(A) = \mathcal{N}(\mu = 0.5, \sigma = 0.1)$ will be used, shown in Figure 3.1. This says that the coin has a 68% probability of $0.4 \leq A \leq 0.6$ and a 95% probability of having $0.3 \leq A \leq 0.7$.

You can also compare this to your desired tolerance for the coin's fairness by evaluating the cumulative density function (CDF). The CDF evaluates the probability that a given distribution is less than a certain value. Let $CDF(P, x)$ be the CDF for a probability distribution P evaluated at the point x (formally defined by Equation (3.2)). This prior therefore assigns a probability of $CDF(P(A), 0.45) = 30.9\%$ probability that the coin is

[1]For simplicity, all data in this example is simulated and does not reflect real coin tosses. Code for this demonstration can be found in Appendix A.

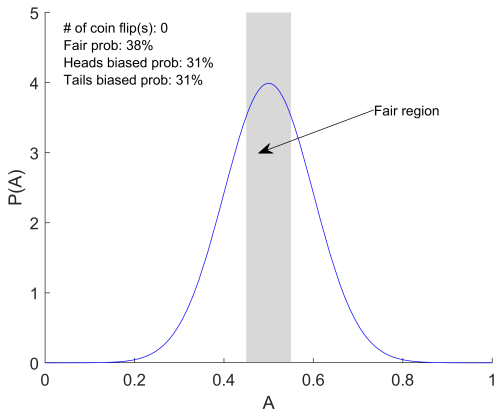


Figure 3.1: Prior distribution for the coin's fairness

biased towards tails, a $1 - CDF(P(A), 0.55) = 30.9\%$ probability that the coin is biased towards heads, and a $CDF(P(A), 0.55) - CDF(P(A), 0.45) \approx 38.3\%$ that the coin is fair: that is, there is only a 38% chance that the coin lies within your desired range of $0.45 \leq A \leq 0.55$.

$$CDF(P, x) = \int_{-\infty}^{x} P(t)\, dt \tag{3.2}$$

Next, you need to update this prior based on your coin tosses. Let B for this example be the result of a single toss, either heads or tails. The probability $P(B|A)$ is the likelihood of observing a result B for a given value of A (formally, this is called the probability of B *conditioned* on A). For this simple example, it is easy to calculate that conditional probability: for heads, it is simply $P(B = \text{heads}|A) = A$, while for tails it is $P(B = \text{tails}|A) = 1 - A$. For example, $P(B = \text{tails}|A = 0.8) = 0.2$. Based on this, the (unnormalized) posterior probability distribution $p(A|B)$ upon observing a single heads or a single tails flip are given by Equations (3.3) and (3.4) respectively. Next, you need to normalize these distributions. For this simple example, this can be done numerically using Equation (3.5). This process is visualized in Figure 3.2. The dotted red lines in those images show the unnormalized posterior distributions, and the solid red lines show the posteriors after numerical normalization.

$$p(A|B) \propto A \cdot \mathcal{N}(\mu = 0.5, \sigma = 0.1) \tag{3.3}$$

$$p(A|B) \propto (1 - A) \cdot \mathcal{N}(\mu = 0.5, \sigma = 0.1) \tag{3.4}$$

$$P(A|B) = \frac{p(A|B)}{\int_{-\infty}^{\infty} p(A|B)\, dA} \tag{3.5}$$

To extend this to consider multiple coin flips, the posterior $p(A|B)$ is simply used as the prior for the next iteration. Equation (3.6) shows the posterior distribution for n tests, where # heads and # tails are the number of heads and tails encountered in the flips respectively (note that this is essentially a non-normalized binomial distribution). Figure 3.3 shows the

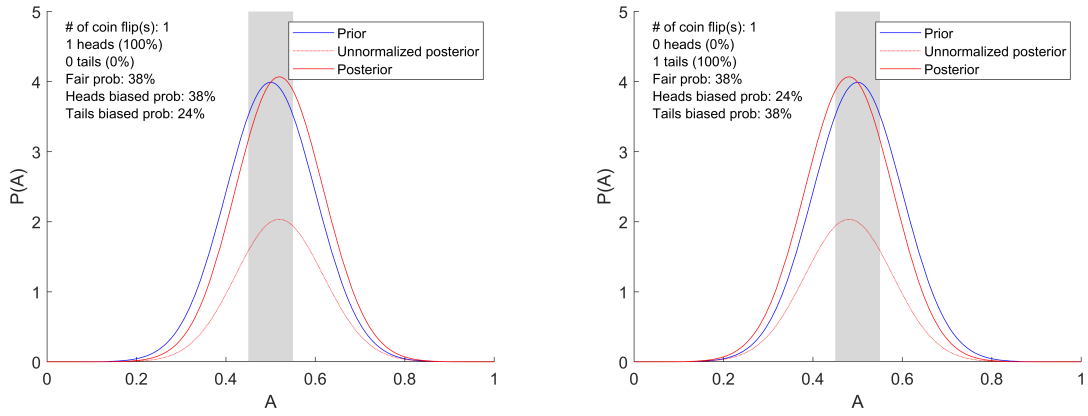


Figure 3.2: Posterior distribution after experiencing a single heads (a) or tails (b).

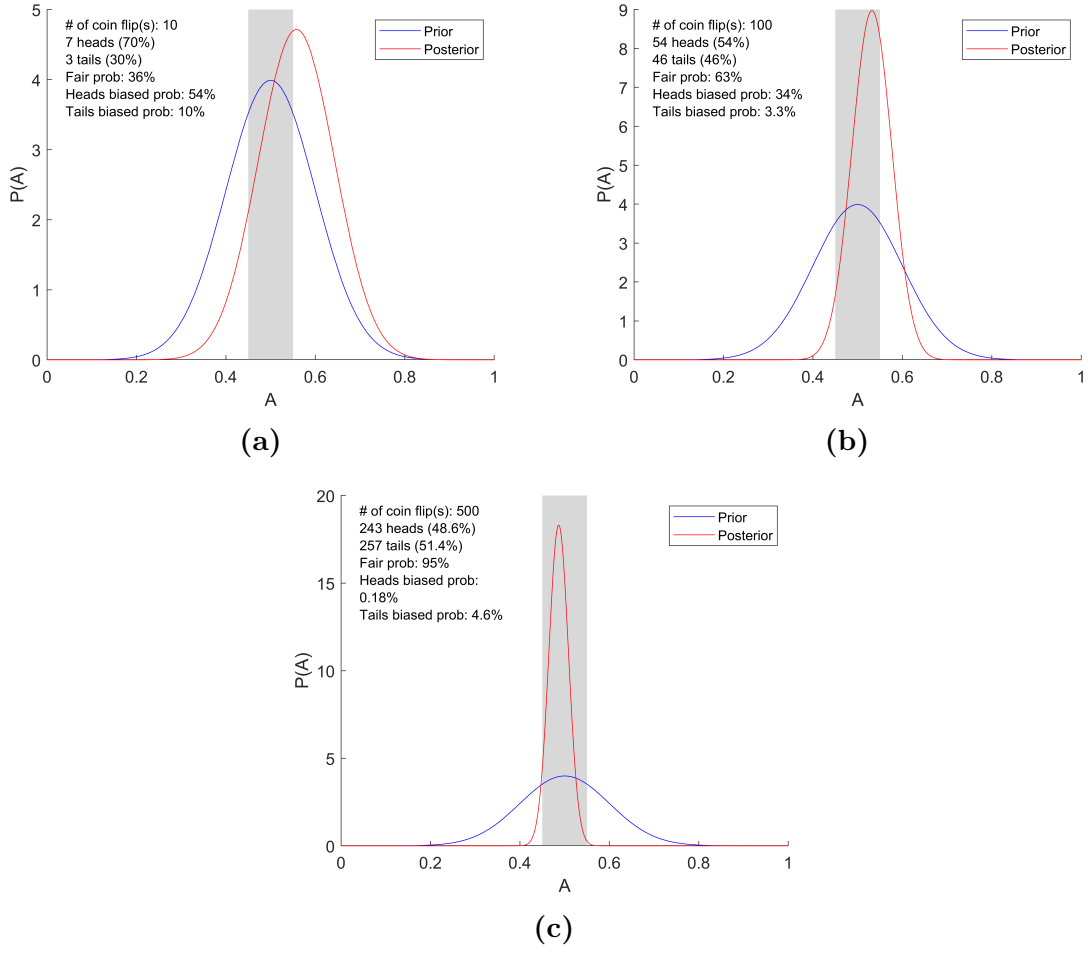


Figure 3.3: Posterior distribution for the coin's fairness after (a) 10, (b) 100, and (c) 500 flips using the Gaussian prior distribution

posterior distribution for A after different numbers of coin flips, starting with the initial ten flip run with seven heads and three tails and then showing how the probability shifts as more flips are added. Ultimately this coin is in fact fair and the initial streak of heads was simply luck, but it takes much more than 10 coin flips to conclusively determine this: even after 100 coin flips there is still only a 63% probability that the coin is fair, and it takes about 500 flips to reach the desired 95% confidence in the coin's fairness.

$$p(A|B_{1:n}) = A^{\#\text{heads}} \cdot (1-A)^{\#\text{tails}} \cdot P(A) \tag{3.6}$$

What if you had more stringent requirements for what is considered a "fair" coin? If you intended to use this coin for gambling, you might only be willing to accept a deviation of 0.5% from a 50/50 coin rather than 5%, with a 99% confidence that the coin fell within that limit. Many more tests will be required to converge to that tolerance. In fact, so many tests are required that Equation (3.6) no longer works. At 1073 tests the algorithm fails due to precision limits on floating-point numbers: the posterior probability before normalization just rounds down to 0 for all the points. Figure 3.4 shows the final result after 1072 tests, just before failure: the steps in the posterior occur because the ϵ value for a double-precision float is too large to accurately calculate these low probabilities. This sort of numerical precision limit is common with Bayesian updating, and can be overcome by performing the calculations with the log probabilities using Equation (3.7), where $\tilde{p}(x)$ is the log probability for a distribution $p(x)$. This log distribution is used to center the distribution so the maximum log probability is equal to 0 before converting back to non-log probabilities with Equation (3.8). This ensures that the proportional distribution is based around the most accurate region for floating point math before proceeding with the standard numerical normalization. Figure 3.5 shows how this algorithm can give accurate results at high numbers of samples, ultimately concluding that roughly 22000 flips are needed to achieve the desired 99% confidence.

$$\log(p(A|B_{1:n})) = \tilde{p}(A|B_{1:n})\#\text{heads} \cdot \log(A) + \#\text{tails} \cdot \log(1-A) + \tilde{P}(A) \tag{3.7}$$

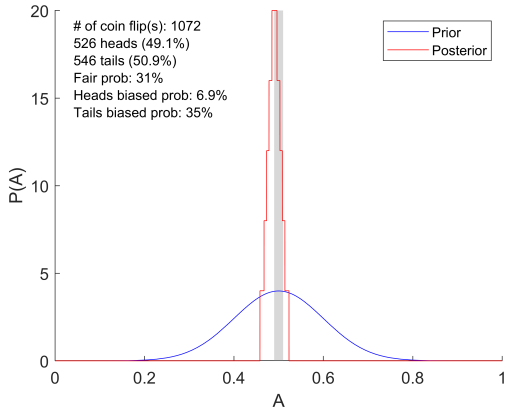


Figure 3.4: Posterior distribution after 1072 flips. The stepping is due to accuracy limitations with double-precision floating point numbers.

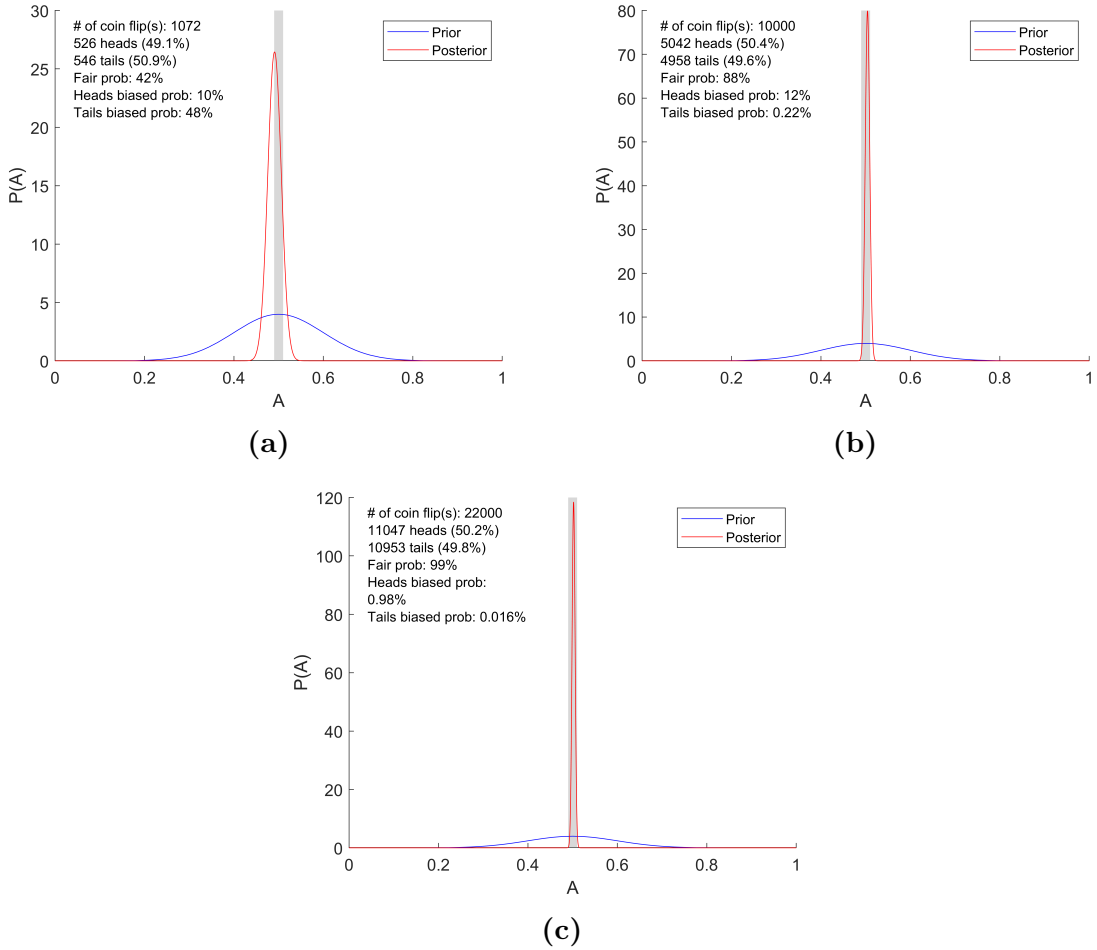


Figure 3.5: Posterior distribution for the coin's fairness, calculated using the logarithmic equation after (a) 1072, (b) 10000, and (c) 22000 flips using the Gaussian prior distribution

$$p(A|B_{1:n}) = \exp\left(\tilde{p}(A|B_{1:n}) - \max_A(\tilde{p}(A|B_{1:n}))\right) \tag{3.8}$$

Why were so many flips required? This is partly because of the inherent variability of coin flips: each flip reveals only a small amount of information about the underlying probability, so a large number of flips are needed to converge. It is also partly because of the choice of prior. The original normal distribution assumed the coin was almost certainly outside of the tight requirements, with only an 8% probability of lying within the tight $0.49 \leq A \leq 0.51$ range. This prior is suitable if, for example, you received the coin from an untrustworthy source that you suspect might have biased coins: effectively, this prior starts from the perspective that the coin is biased, and evidence must be presented to demonstrate that it's fair. Coupled with the high variance of coin flips, it takes a very large number of flips to achieve the required confidence that it's not biased.

Changing the prior will build different assumptions into the algorithm. If you believe that the coin is probably a standard quarter and just want to confirm that it isn't slightly biased by damage or bad manufacturing, then you could establish a more informed prior by looking at prior research. For example, Bartoš *et al.* published a large collection of coin-toss data in 2023 [29]. They conducted 4300 tosses with quarters and found that the coins had a 50.7% chance of landing on heads, with a 95% confidence interval between 49.2% and 52.2%. This would be a reasonable informed prior, corresponding to $P(A) = \mathcal{N}(\mu = 0.507, \sigma = 0.0075)$. Using that as a prior shows that there is already a 64% probability that $0.49 \leq A \leq 0.51$, before flipping any coins.

Alternatively, you could use an uninformed prior like the uniform distribution $P(A) = \mathcal{U}(0, 1)$, which assumes that any probability between 0 and 1 is equally likely. Uninformed priors are generally used when you don't have enough information about the system to make a guess about its performance. Here, that might be because the coin is very poorly manufactured and visibly deformed such that you can't predict how it will flip. Even starting with this naive assumption, it is still possible to learn about the coin given enough flips, as shown in Figure 3.6.

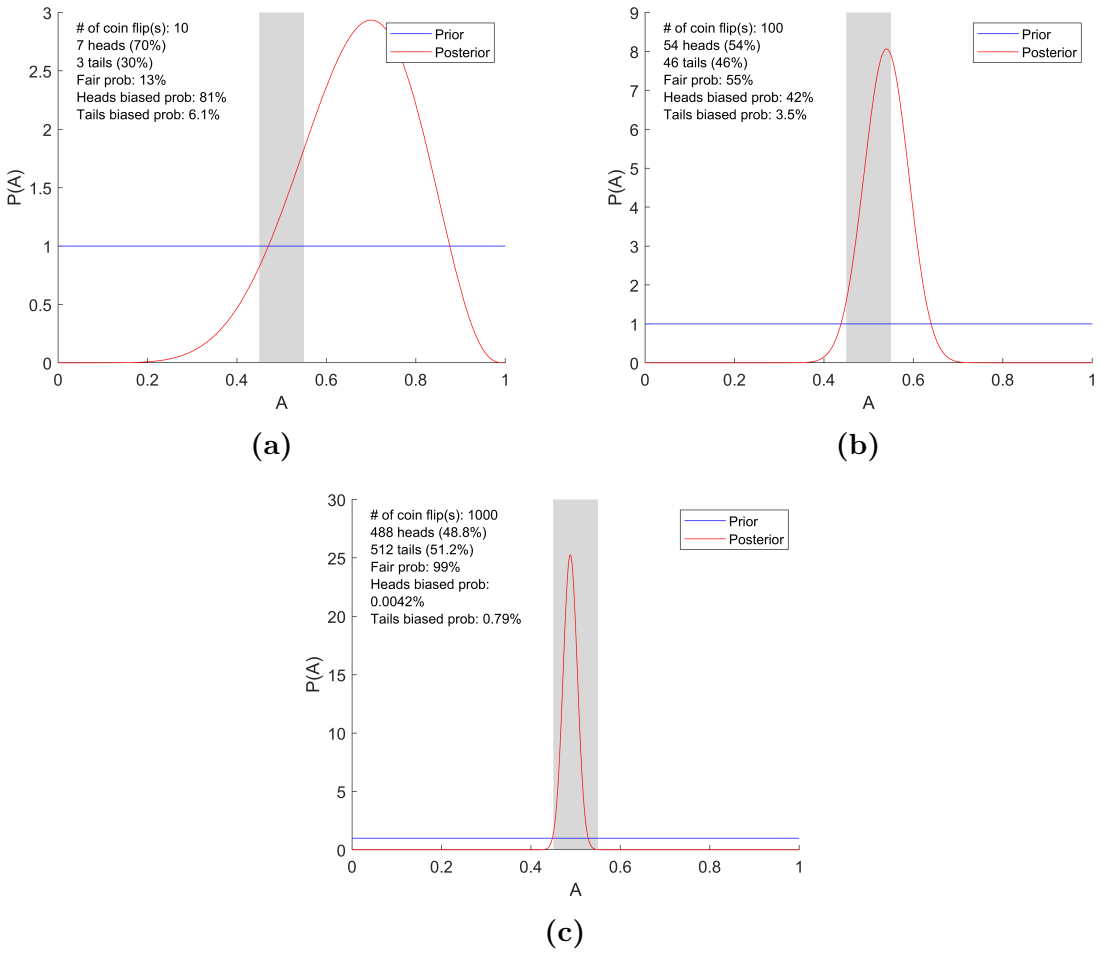


Figure 3.6: Posterior distribution for the coin's fairness after (a) 10, (b) 100, and (c) 1000 flips using the uniform prior distribution

This analysis could, of course, get significantly more complicated. For example, there is evidence that coins have a roughly 51% probability to land on the same side they started on when flipped [29]. This could be incorporated into the model by turning A into a conditional variable with four possible values: heads starting on heads, heads starting on tails, tails starting on heads, and tails starting on tails. Alternatively, you could instead use variables like the flipping force and weight distribution of the coin as inputs and propagate those variables through computer simulations to determine the likelihoods.

This demo has walked through a Bayesian inference problem. While this is a simple example, it demonstrates all the major components necessary for more complex Bayesian analysis. It has demonstrated how a simple problem (is a coin fair?) can be expressed probabilistically, and how priors can be defined based on what is known about the system. A likelihood function was established using basic domain knowledge, and the priors were updated based on a series of coin tosses, showing how the confidence in predict increases as more data is collected. The pros and cons of several different priors were discussed to show how you can set the prior based on your knowledge of the system. Several implementation details like using log probabilities were shown, and then it was shown that the model can be extended to include additional variables, depending on the requirements.

3.2 Bayesian machining modelling

Bayesian inference has been widely applied in many different fields but has seen only limited study in machining [30–32]. To date only five authors have applied Bayesian updating to the problem of machining stability (not counting papers directly related to this work).

The first Bayesian stability model was proposed by Karandikar *et al.* who described a non-physics-based Bayesian updating model for identifying optimal machining parameters [33–35]. This method established a probabilistic stability map which showed the likelihood that a given test would be stable, based on the information that was known about the system. The initial prior was a linear interpolation between 100% probability of stability at 0 mm axial depth and 0% probability of stability at 20 mm. This probabilistic stability map was used to select a cutting test, and the result of the cutting test was used to update the

probabilistic stability map. This process was iterated until the algorithm converged to the optimal machining parameters. This updating process is shown in Figure 3.7, where 20 test cuts were required to converge.

Physics-informed Bayesian models are a recent development which has seen limited research thus far. This method was first published by Li *et al.* [36]. Unlike the non-physics-informed model, these methods do not directly consider the probability of stability. Instead, the prior is an initial guess about the system parameters, specifically the tool tip frequency response function (FRF) and cutting force model. These uncertainties are then propagated through a stability map physics algorithm to obtain the probabilistic stability map and are updated by cutting test results. They applied this method to identify changes in spindle dynamics at different spindle speeds compared to a static tap test. The resulting stability map is shown in Figure 3.8. Chen iterated on this method by using a different physics model and replacing the computationally difficult Markov chain Monte Carlo (MCMC) sampling with Laplace estimation using a surrogate model [37]. They also demonstrated that the posterior parameter distribution can be used to calculate the stability map for different cutting conditions. Ahmadi *et al.* applied the Bayesian method to turning and used in-process vibration signals to learn the structural dynamics from stable cuts [38]. Akbari demonstrated how the Bayesian model can be expanded to to learn across several different machines when toolholders are shared between them [39]. These approaches are discussed in more detail in Chapter 4, walking through how they implement the Bayesian framework.

Physics-based Bayesian learning has several advantages compared to the non-physics-based method. First, because the process physics are incorporated into the updating, more information can be extracted from each test cut. This reduces the number of tests required to converge to the optimal machining parameters. Second, the updating provides information about the cutting force model and system FRF, which can then be used to calculate the stability map for different cutting conditions. However, the physics-based methods presented thus far have required many cutting tests to converge due to non-optimal test selection, have not considered other limitations on machining parameters beyond stability, and cannot uniquely identify the underlying stability map inputs. As a result, these methods are not suitable for practical use to select cutting parameters.

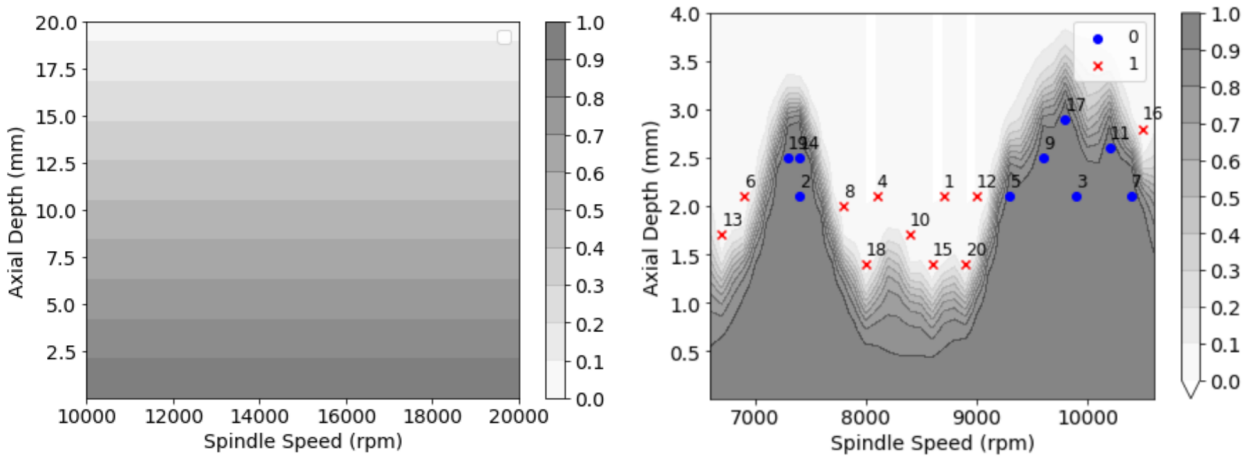


Figure 3.7: Prior (a) and posterior (b) probabilistic stability map, Figures 1 and 4 from [33]. The color indicates the probability of stability: darker colors indicate that a cut is more likely to be stable, while lighter colors indicate a cut is more likely to chatter.

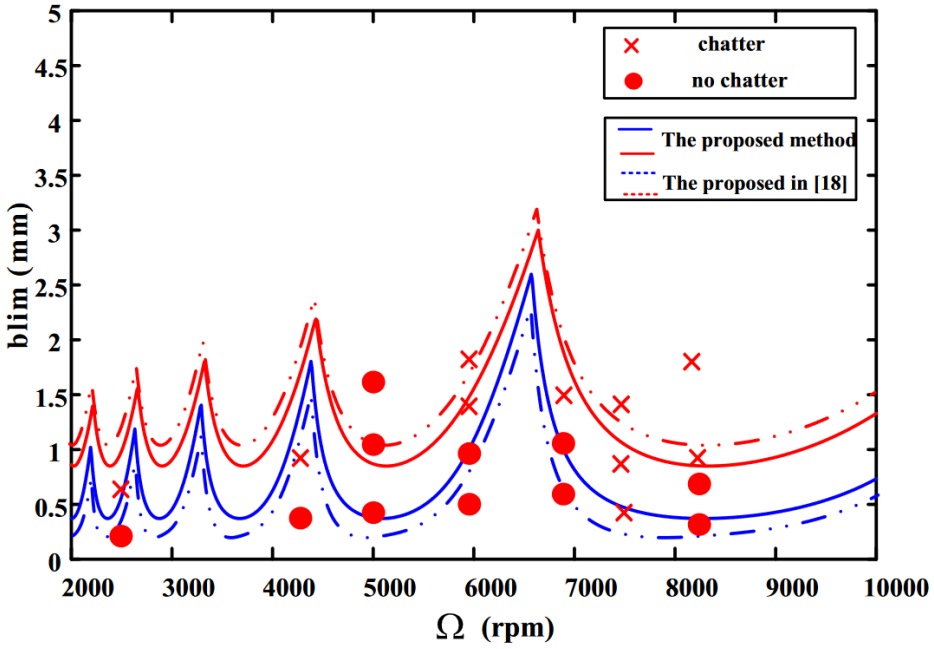


Figure 3.8: Tests and posterior probabilistic stability map, Figure 18 from [36]. The blue solid line represents the 50% probability of stability boundary from the prior, while the solid red line is the 50% probability of stability in the posterior.

Chapter 4

Bayesian machining modelling

As discussed in the last section, existing approaches have not presented a general method for modelling and optimizing uncertain machining processes. This section describes the overall Bayesian machining framework, extending it to consider an arbitrary number of factors in the machining process. There are five main steps, shown in Figure 4.1:

Defining the prior distribution A prior probability distribution $P(\boldsymbol{\theta})$ is defined which describes the user's initial beliefs about the machining system, and a set of samples are drawn from that distribution.

Physics models The samples are propagated through a set of physics models to generate probabilistic predictions about the machining process, e.g., the probability of stability or cutting torque.

Test selection The probabilistic predictions are used to select a cutting test based on the user's goals, for example to maximize MRR or improve knowledge of the system parameters. If no test is recommend, testing is terminated.

Defining the posterior distribution The results of the test cut is used to define a new posterior distribution which incorporates what was learned from the test.

Sampling the posterior distribution New samples are drawn from the posterior distribution and passed to the physics models to select a new cutting test.

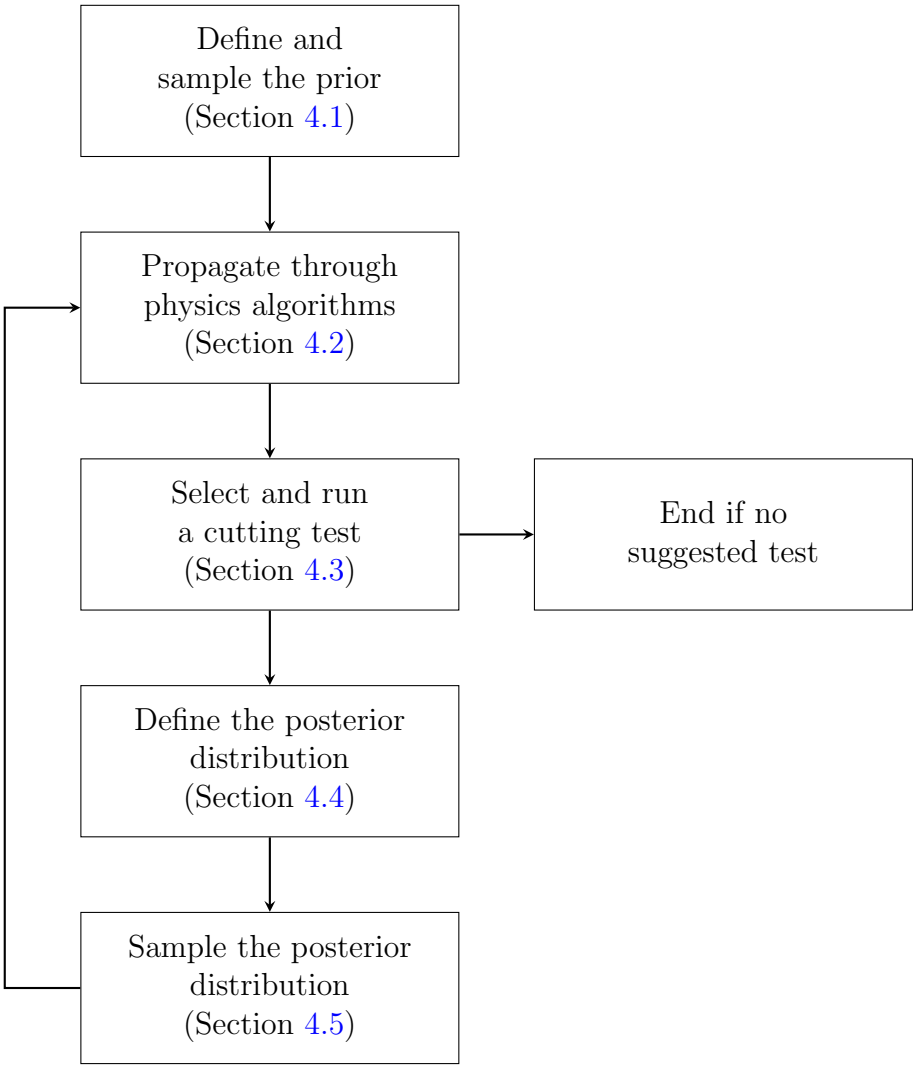


Figure 4.1: Flowchart outlining the steps in the Bayesian inference framework

Each of these steps is modular: there are, for example, many different ways to define and establish the system prior depending on exactly what is known about the system and what physics models are to be employed. The rest of the section will provide more information on each of these steps and give specific examples of different implementations (e.g., presenting several different test selection algorithms). In general, these implementations will focus on low-information machining optimization, where the Bayesian stability model is used to select productive cutting parameters with minimal measurement requirements, as well as summarizing the approaches that have been used in literature to date. However, the general model is highly extensible and can be used for different goals. To assist with understanding how the different components interact, Appendix C walks through the provided Matlab code for the overall framework.

4.1 Prior

Let $\boldsymbol{\theta}$ be a vector variable containing all relevant system parameters necessary to define the workpiece/machine/toolholder/cutting tool system, including the tool tip FRF and cutting force model. $\boldsymbol{\theta}$ can take various forms depending on what variables are known and what physics models are used. For a system with 2 degrees of freedom, $\boldsymbol{\theta}$ could take the form shown in Equation (4.1), with the cutting force consisting of the specific cutting force coefficient K_s and the cutting force angle β, and the tool tip FRF calculated from a set of modal parameters with natural frequency f_n, stiffness k, and viscous damping ratio ζ for an arbitrary number of degrees of freedom. This form can be extended to include additional variables, for example coefficients for process damping (see Section 5.2) or the cutting tool's ultimate tensile strength (Section 4.2.4).

$$\boldsymbol{\theta} = \begin{bmatrix} K_s & \beta & f_{n\,1} & k_1 & \zeta_1 & f_{n\,2} & k_2 & \zeta_2 \end{bmatrix} \tag{4.1}$$

The prior distribution $P(\boldsymbol{\theta})$ defines how likely any given set of system parameters are, encoding the epistemic uncertainties. For example, one possible form would be a multivariate normal (MVN) distribution with mean vector μ and covariance matrix Σ $P(\boldsymbol{\theta}) = \mathcal{N}(\mu, \Sigma)$, where the PDF of an MVN is given by Equation (4.2). Alternatively, the prior can be

assembled from individual distributions for each element in the $\boldsymbol{\theta}$ vector: Equation 4.3 shows this with d representing the length of $\boldsymbol{\theta}$ and $P_j(\boldsymbol{\theta}^j)$ being the prior distribution for the jth element. Figure 4.2 shows an example 3-dimensional prior combining uniform, exponential, and normal distributions, showing both the marginal distributions for each variable and conditional distributions which show correlation between pairs of variables. The higher the probability for some value of $\boldsymbol{\theta}$ is, the more likely it is to represent the true system.

$$PDF(\mathcal{N}(,\mu,\Sigma),x) = \frac{1}{\sqrt{(2\pi)^k \det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{4.2}$$

$$P(\boldsymbol{\theta}) = \prod_{j=1}^{d} P_j(\boldsymbol{\theta}^j) \tag{4.3}$$

Note that it may be better under some circumstances to truncate distributions to impose constraints on the values they can take. For example, the viscous damping ratio ζ often both has high uncertainty and low values, so a naive normal distribution prior may assign some notable probability to $\zeta \leq 0$, which is not physically meaningful for machining systems (e.g., the prior $P_\zeta(\mu = 0.01, \sigma = 0.005)$ assigns approximately 2% probability of negative damping). Therefore, priors should be truncated to assign a probably of 0 outside of reasonable values. Equation (4.4) defines a truncated MVN $\mathcal{N}^T$ with lower and upper bounds of x_{min} and x_{max} respectively. This can be easily sampled using rejection sampling by drawing samples, comparing them against the desired boundaries, and discarding them if not. Unless otherwise noted, all examples through this dissertation assume truncated distributions such that all elements must be be positive.

$$\mathcal{N}^T(x,\mu,\Sigma,x_{min},x_{max}) \propto \begin{cases} \mathcal{N}(x,\mu,\Sigma) & \text{all}(x \geq x_{min}) \,\&\, \text{all}(x \leq x_{max}) \\ 0 & \text{else} \end{cases} \tag{4.4}$$

Once the prior has been established, samples are drawn from it to use with the physics models described later. Since the prior is generally a standard distribution like MVN or uniform, it can be sampled using standard methods. Alternatively, some form of space-filling sampling like orthogonal sampling, Latin hypercube sampling, or Sobol sampling can

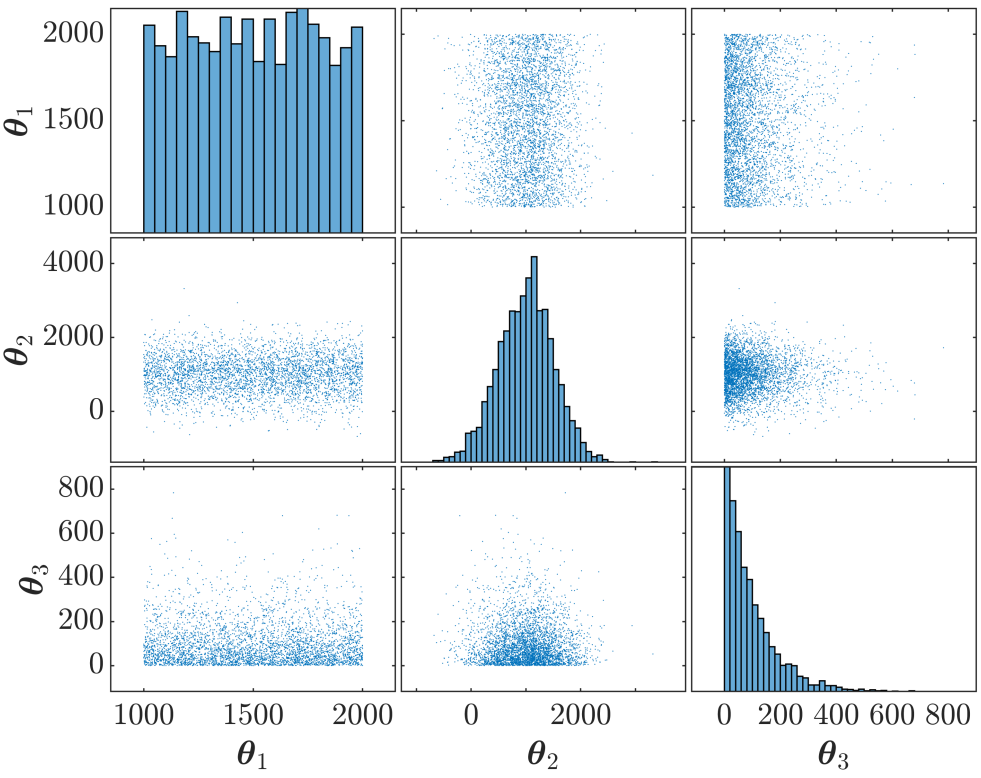


Figure 4.2: Example of a 3-dimensional prior distribution. The on-diagonal plots show the marginal distributions for each element, while the off-diagonal plots show the conditional distributions to help identify correlation between variables.

be used to draw samples evenly across the parameter space and then weight them based on their probability density, as shown in Section 5.3.

In order to provide the most accurate possible starting point, the prior distribution should incorporate any information known about the system before any cutting tests are made. The following subsections describe methods for establishing the prior for the cutting force model and frequency response function.

4.1.1 Cutting force model

These cutting force coefficients can be measured directly by recording the force during a cut with a cutting force dynamometer. Dynamometers are rare in machine shops, however, so a method for establishing the prior without directly measuring the cutting force is necessary. The cutting force prior can be set based on user experience [40], literature reviews of published values [41], or cutting force modelling using software like AdvantEdge from Third Wave Systems [42]. This software simulates orthogonal cutting for a range of chip thickness values using finite element modelling, which can then be used to fit the mechanistic model values. By performing Monte Carlo sampling, uncertainties in the tool's cutting edge geometry and the work material properties can be propagated to the mechanistic model [43]. The distribution of the resulting cutting force coefficients can then be used as the prior for the milling stability model.

4.1.2 Frequency response model

The frequency response prior defines the flexibility of the system. The exact form of this prior will vary depending on what system is being modeled and what physics models are employed.

One simple method is to model the endmill as a fixed-free beam using Euler-Bernoulli or Timoshenko beam theory. This approach is suitable for long tools where the holder/spindle/-machine can be approximated as rigid. Let the tool be modelled as a cylindrical beam with length L, cross-sectional area A, second moment of area I, elastic modulus E, density ρ, and solid damping factor η. Using Euler-Bernoulli beam theory, the displacement-to-force

receptance at the tip of the tool can be estimated for a frequency f using Equations (4.5) and (4.6) [44]. The biggest uncertainties here are generally the length L and equivalent diameter d: uncertainties can be established in these variables and propagated to the FRF. Timoshenko modelling can also be used: consult [45] for more information. Figure 4.3 shows an example with a 6.35 mm relieved shank endmill which has a stickout length of 66 mm. This example will be discussed more in Section 5.3.

$$\lambda(f) = \left((2\pi f)^2 \frac{\rho A}{EI(1+i\eta)} \right)^{1/4} \tag{4.5}$$

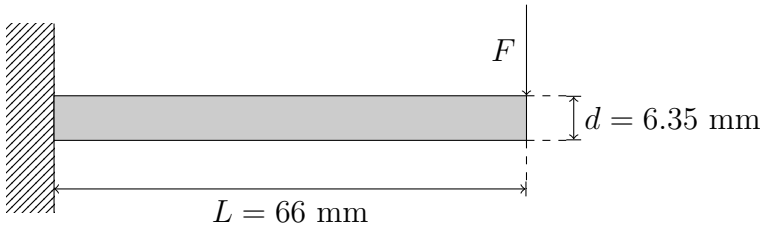
$$H_{22} = \frac{\sin(\lambda L)\cosh(\lambda L) - \cos(\lambda L)\sinh(\lambda L)}{\lambda^3 EI(1+i\eta)(\cos(\lambda L)\cosh(\lambda L)+1)} \tag{4.6}$$

A more sophisticated approach is receptance coupling substructure analysis (RCSA). This approach can model the effect of the holder and spindle on the FRF, making it more general than the fixed-free beam example above. RCSA takes frequence responses for a set of subcomponents and estimates the FRF when those components are assembled together. Each subcomponent has four separate receptances measured at different points on the component: displacement-to-force h_{ij}, displacement-to-couple l_{ij}, rotation-to-force n_{ij}, and rotation-to-couple p_{ij}, where i is the location that displacement/rotation is being measured and j is the location where the force or moment is applied. For example, the free-free cylindrical beam shown in Figure 4.4(a) has receptances measured at both ends of the beam in the X direction. There are 16 different receptances associated with that beam in the X direction: four direct receptances at each end of the beam (eight total) and four cross receptances at each end of the beam (eight total). While not shown here, there are also another 16 receptances for this beam in the Y direction for a total of 32 (since this particular beam is symmetric in X and Y those receptances will be identical), ignoring cross-coupling between the X and Y directions. In contrast, the fixed-free beam has only 8 total receptances since the receptance is only measured at one point. These receptances can be written in matrix form for each frequency f as shown in Equation (4.7).

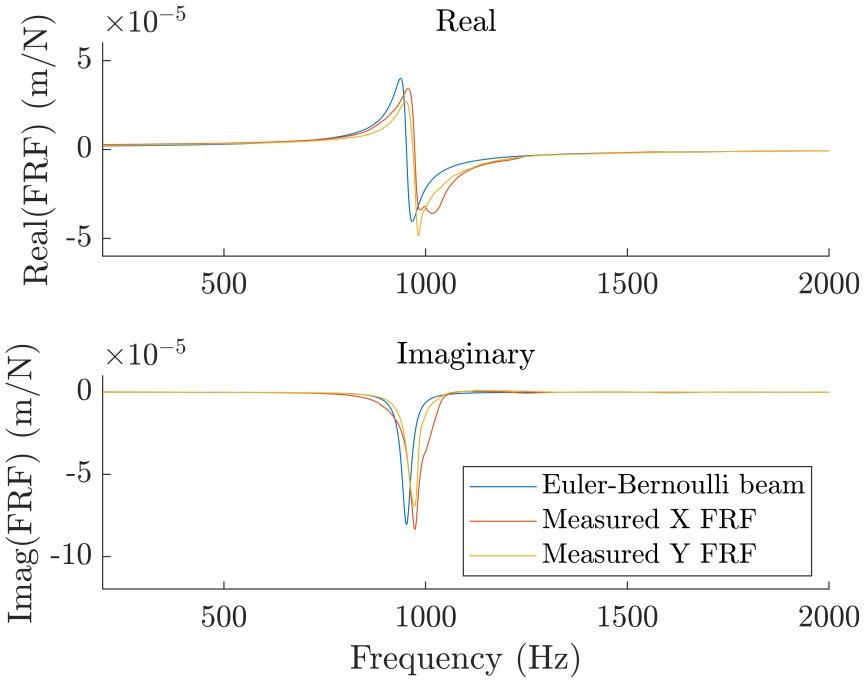
$$R_{ij}(f) = \begin{bmatrix} h_{ij}(f) & l_{ij}(f) \\ n_{ij}(f) & p_{ij}(f) \end{bmatrix} \tag{4.7}$$



(a)

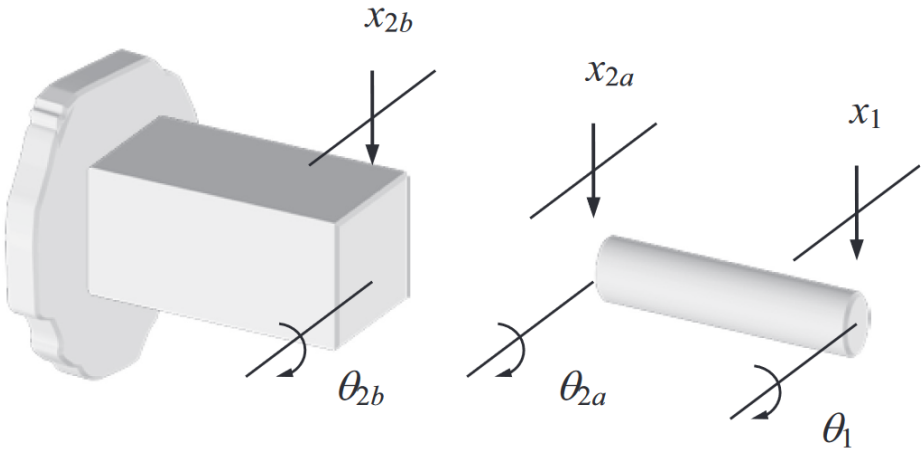


(b)

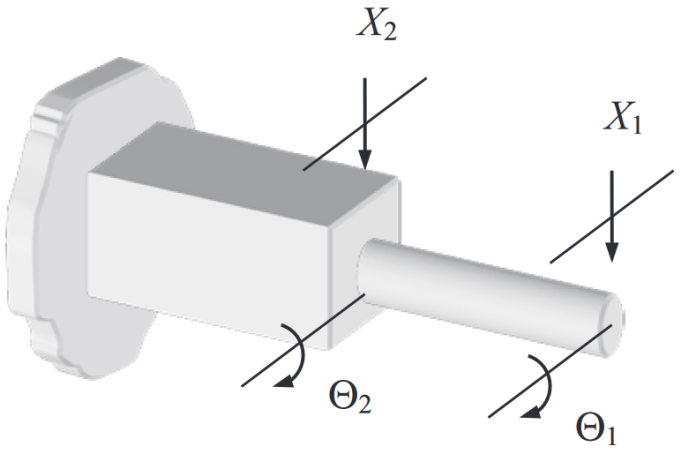


(c)

Figure 4.3: Modelling the tool FRF using Euler-Bernoulli beam theory. (a) A 6.35mm endmill with a 10:1 L/D ratio. (b) Modelling the tool as a fixed-free beam with an end load. (c) Comparison between the measured and predicted FRFs.



(a)



(b)

Figure 4.4: (a) Forces and moments applied to fixed-free and free-free beams in the X direction, Figure 7.15 from [3]. (b) Those components coupled together into a single body, Figure 7.14 from [3].

Once each component receptance is known they can be coupled together as shown in Figure 4.4. The flexibility of the coupling is defined by the dynamic stiffness matrix K (Equation (4.8)), where k_{xf} indicates the linear stiffness in X of the coupling to a force, $c_{\theta m}$ indicates the rotational damping of the coupling to a moment, and so on. This is then used to calculate G_{11}, which is the direct receptance of the assembly at point U_1 (Equation (4.9)).

$$K(f) = \begin{bmatrix} k_{xf} + i\, c_{xf}\, 2\pi f & k_{\theta f} + i\, c_{\theta f}\, 2\pi f \\ k_{xm} + i\, c_{xm}\, 2\pi f & k_{\theta m} + i\, c_{\theta m}\, 2\pi f \end{bmatrix} \tag{4.8}$$

$$G_{11}(f) = R_{11}(f) - R_{12a}(f)\left(R_{2a2a}(f) + R_{2b2b}(f) + K(f)^{-1}\right)^{-1} R_{2a1}(f) \tag{4.9}$$

This RCSA process can be applied to estimate the tool tip FRF for a given tool assembly by assembling the components, as shown in Figure 4.5. First, an artifact in the spindle is tap tested, and the artifact geometry is removed by inverse RCSA to give the machine/spindle/flange FRF (see [46] for details). The holder FRF is then coupled to this FRF: this connection is generally modeled as rigid because the holder and flange are a single component. Finally, the FRF of the cutting tool geometry is coupled to the holder using a flexible connection to give the tool tip FRF. While the spindle FRF is measured, the tool and holder FRFs can be modeled using finite element modeling based on CAD models provided by vendors or measurements of the component geometries.

This method does still technically require tap testing or other modal analysis to measure the FRF of the spindle, but only a single spindle measurement is necessary to estimate the FRF of any tool assembly used on the machine. Therefore, this service could feasibly be provided by the machine tool vendor during machine setup or by a third party with tap testing equipment. While spindle dynamics can change over time due to wear this process is usually gradual and could be remeasured regularly, e.g. during scheduled maintenance.

While the RCSA process is accurate, in practice there is considerable uncertainty about the specific inputs. First, the coupling parameters are difficult to measure and can vary significantly. (Note that $k_{\theta f}$ and $c_{\theta f}$ tend to dominate since there is generally no moment applied at the tool tip and tools commonly have a high length/diameter ratio.) Second, while the cutting flutes of the tool are commonly modeled as a solid cylinder with slightly

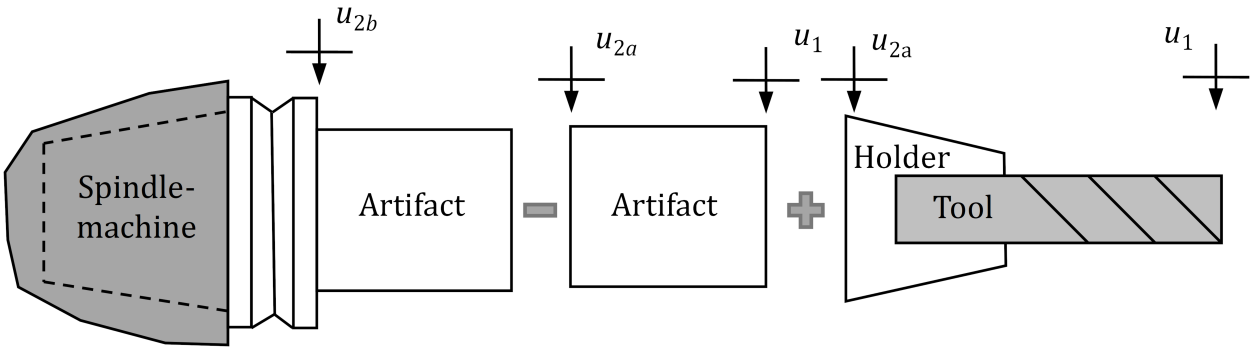


Figure 4.5: Receptance coupling substructure analysis for predicting the tool tip FRF, Figure 1 from [49].

smaller diameter than the tool itself, the best diameter of this equivalent stiffness cylinder will depend on the exact geometry of the tool and can't be measured directly [47, 48]. Third, while the spindle FRF is usually tap tested with the spindle not rotating, the bearing dynamics will change as the spindle turns at different speeds [12]. Therefore, uncertainties are defined about all these variables (currently set based on author experience).

Those uncertainties can then be used in two ways. Schmitz *et al.* propagated these uncertainties through the RCSA function to the tool tip FRF using Monte Carlo sampling and then fit sets of natural frequency, stiffness, and damping ratio values f_n, k, ζ (or mass/stiffness/damping values k, m, c) [49]. The uncertainties of those modal parameters was used in the the $\boldsymbol{\theta}$ vector. Equation (4.10) shows how k, m, c can be used to calculate the FRF for a given angular frequency ω. This approach is computationally fast and places the fewest constraints on the FRF since each mode is free to move around independently.

$$FRF(\omega) = \left(-(2\pi f)^2 m + 2ic\pi f + k\right)^{-1} \tag{4.10}$$

Alternatively, Akbari *et al.* directly used the RCSA input uncertainties in the $\boldsymbol{\theta}$ vector [39]. These uncertainties may be more generalizable than simple modal parameters: Akbari *et al.* were able to apply transfer learning to characterize the RCSA parameters for a given holder across multiple machines. However, the RCSA calculations are also slower than calculating FRFs using modal parameters and approximation errors in the RCSA function can potentially cause issues. For example, if the wrong equivalent diameter is chosen then the RCSA model may be unable to converge to the true FRF, regardless of the coupling parameters.

4.2 Physics models

Let there be some list of functions which simulate or estimate the results of a machining operation with cutting parameters ω and a set of system parameters $\boldsymbol{\theta}$. For example, there may be a function which predicts whether a cut would be stable or chatter, or which predicts the cutting torque. These functions can be used to propagate the uncertainty from the prior using Monte Carlo sampling, resulting in probabilistic predictions about the performance of

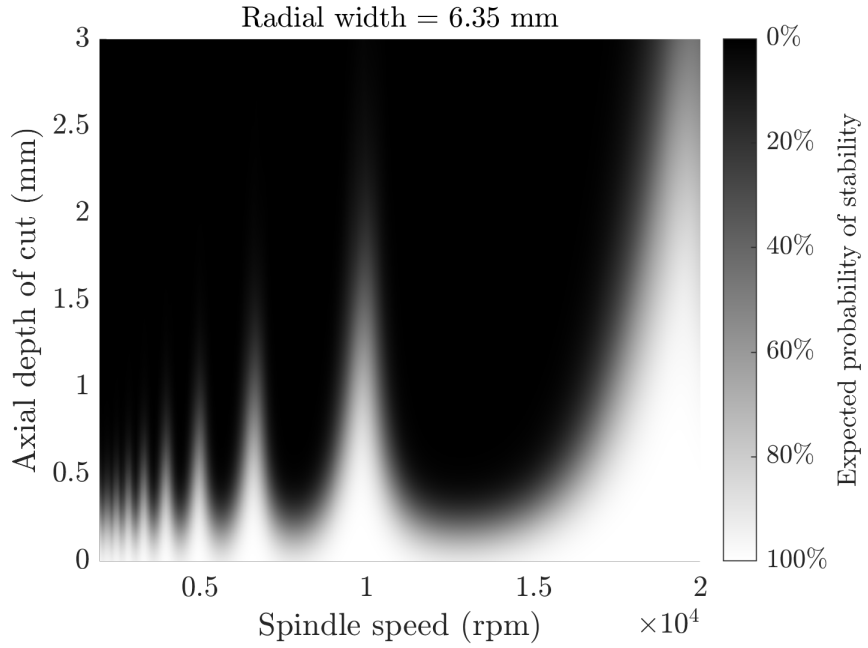
the machining system. Each of the samples from the prior is run through the selected physics models to establish uncertainty in the stability map, cutting torque, spindle bending moment, tool bending stress, and any other important factors. The following subsections describe some possible physics algorithms and demonstrate how they can establish uncertainties about the machining system.

4.2.1 Stability map algorithms

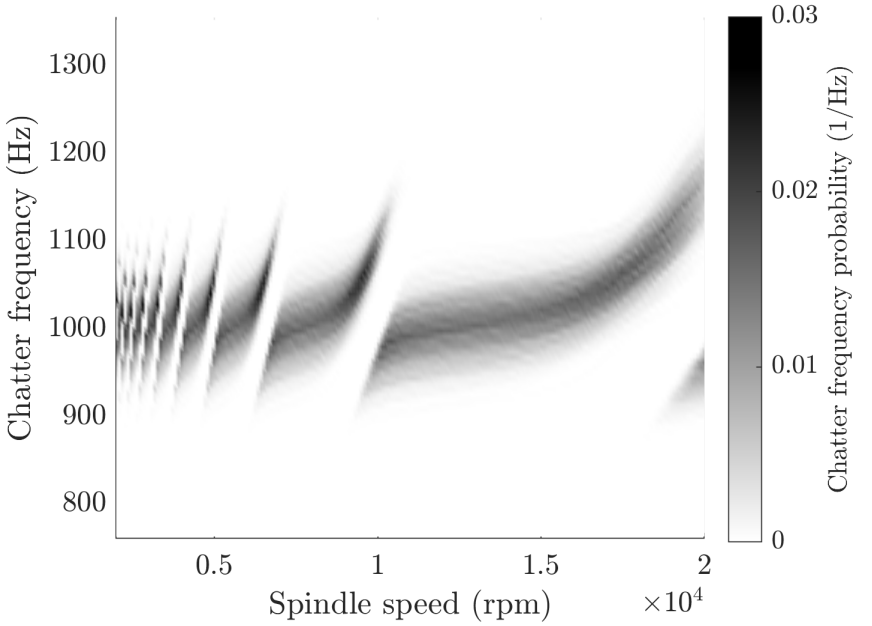
Most prior work on Bayesian machining modelling has focused on stability and chatter. A stability algorithm $S(\omega, \boldsymbol{\theta})$ predicts whether some given set of cutting parameters ω will be stable for some fixed set of system parameters $\boldsymbol{\theta}$. Generally, these algorithms can also predict what the chatter frequency $f_c(\omega, \boldsymbol{\theta})$ will be if chatter is encountered. These stability algorithms can be used to propagate the samples taken from the prior to the stability map and calculate the probability that a test cut at any given set of cutting parameters will be stable. For each point, the probability of stability $P_S(\omega)$ is calculated using Equation 4.11, where $L_S(S, \omega | \boldsymbol{\theta})$ is the probability that a cut will have a result of S (either stable or unstable) for a fixed set of system parameters $\boldsymbol{\theta}$. Likelihood functions will be discussed in more detail in Section 4.4: for now, assume that each sample is deterministic and assigns a likelihood of stability of either 100% or 0% based on whether $S(\omega, \boldsymbol{\theta})$ is true or false. A similar calculation can be done for the chatter frequencies, calculating the probability density of the predicted dominant chatter frequencies for each spindle speed. An example is shown in Figure 4.6.

$$P_S(\omega, \boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} L_S(S, \omega | \boldsymbol{\theta}_i) \tag{4.11}$$

As discussed in Section 2.1 there are several approximate stability models which can be used to predict stability for milling operations and estimate chatter frequencies. As a brief summary, these algorithms broadly fall into three main categories: frequency domain algorithms, discrete methods, and numerical simulation methods [5, 6]. First, these categories are briefly summarized, explaining the available algorithms and how they have been employed in Bayesian stability modelling to date. Afterwards, the algorithms which



(a)



(b)

Figure 4.6: Demonstration of how the prior is propagated through the stability algorithm, using the zero-order approximation frequency-domain algorithm. (a) Probabilistic stability map. The color of each point indicates the probability that a cut would be stable, with darker colors indicating a lower probability of stability. (b) Chatter frequency distribution. Darker colors indicate a higher likelihood of seeing specific chatter frequencies.

are employed in Chapters 5 and 6 are presented in greater detail, specifically the zero-order approximation frequency domain algorithm and time-domain simulations.

Frequency domain algorithms use control theory to analyze the frequency response function and determine which cutting parameters will be stable and what the chatter frequencies will be by linearizing the system dynamics and cutting forces. The most common algorithm is the well-established frequency domain zero-order approximation (ZOA) algorithm by Altintas and Budak [4]. A more complete presentation of the ZOA is presented below in Subsection 4.2.1. To-date, this has been used in most papers on physics-based Bayesian stability modelling [36, 39, 41, 49]. The ZOA is computationally efficient and is suitable for high radial immersion cutting conditions (i.e., when a is greater than 50% of the diameter of the tool). At lower radial immersions, the average cutting force approximation is less appropriate since flutes are engaged for only a small portion of the tool rotation, and bifurcations will appear in the stability map, as shown in Figure 4.7 [50]. This algorithm is also not suitable for modelling more advanced machining systems, such as cutting tools which have variable flute spacing or process damping. The ZOA has been extended to model more complex systems at the cost of increased calculation time. For example, Tyler *et al.* extended the ZOA to model velocity dependent process damping [51], Merdol *et al.* proposed the multifrequency algorithm which takes into account the higher-order cutting force harmonics for low radial immersion cutting [52], and Otto *et al.* created a multifrequency approach which can model complex tool geometries with variable helix angle, non-uniform flute spacing, and runout [53].

Discrete stability algorithms like semidiscretization use Floquet theory to convert the delayed periodic coefficient stability algorithm into a larger constant coefficient problem [54]. First, the tool rotation is discretized into a series of m steps. Equation (4.12) shows how the system's vibrations over the last revolution can be encoded as the vector $\mathbf{z}_i$. Note that the velocities $\dot{\mathbf{x}}$ and $\dot{\mathbf{y}}$ are only retained for the most recent step since the velocity history does not affect the motion or forces going forward. Floquet theory says that there is some transition matrix $\mathbf{\Phi}$ which models how a linear system will vibrate over the next revolution such that $\mathbf{z}_{i+m} = \mathbf{\Phi}\mathbf{z}_i$. Once $\mathbf{\Phi}$ is known, then its eigenvalues can be analyzed to determine whether it is stable based on its spectral radius (i.e., the absolute magnitude of the largest

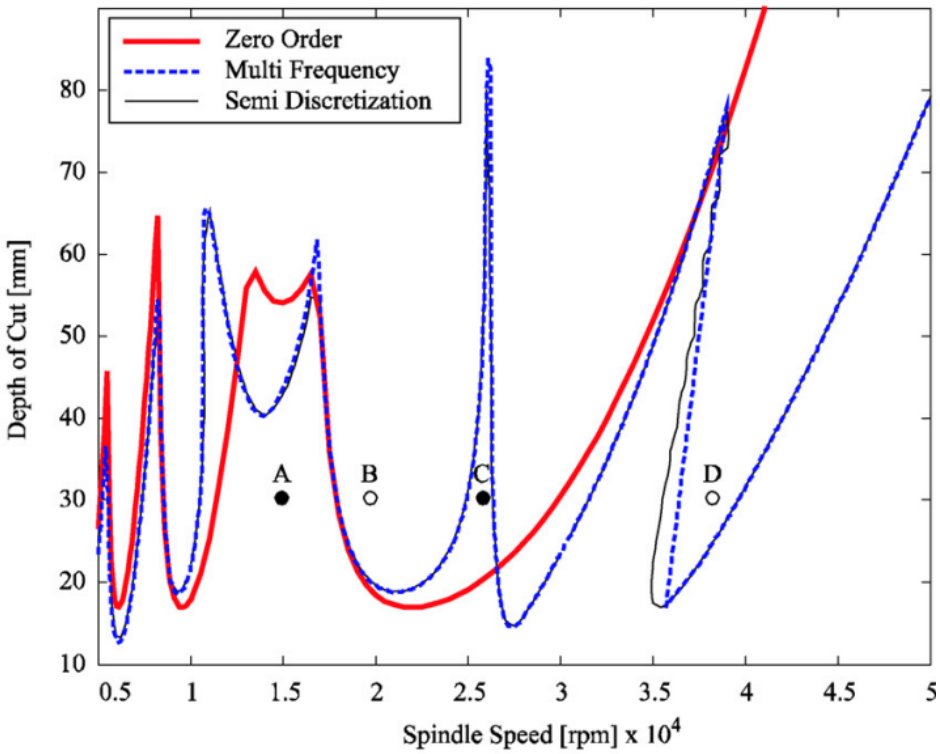


Figure 4.7: Bifurcations in the stability lobe, not accurately modelled by the ZOA algorithm. Figure 4 from [55].

eigenvalue). If the spectral radius is greater than one, then the system vibrations will grow larger every revolution and the cutting parameters will chatter. If the spectral radius is less than one then the vibrations will decrease and the cut will be stable. The chatter frequency can also be determined based on the phase angle of the largest eigenvalue.

$$\mathbf{z}_i = [x_i, y_i, \dot{x}_i, \dot{y}_i, x_{i-1}, y_{i-1}, \ldots, x_{i-m+1}, y_{i-m+1}]^T \tag{4.12}$$

Different discrete stability algorithms provide different methods for approximating $\mathbf{\Phi}$: consult [5, 6] for a discussion of the available algorithms. These methods can be extended to model difficult machining problems such as variable flute tools or low radial immersion cutting. However, they are also more computationally expensive than the ZOA, and still rely on linearizing the cutting force model and dynamics. Chen *et al.* employed one of the discretization-based algorithms for Bayesian stability modelling [37]. Due to the higher computational cost, they trained an artificial neural network surrogate model to approximate the spectral radius rather than directly running the discretization method for each sample (see Figure 4.8).

Numerical methods like time-domain simulation use Euler integration to solve the second-order differential equation for machining dynamics by dividing the problem into a series of small timesteps [3]. At each timestep, the instantaneous cutting force is estimated based on the current engagement of the cutter and part. Those forces are then used to estimate the position and velocity of the tool at the next timestep. This simulation run until the tool vibration has reached steady state, and then the cut's stability and chatter frequency are determined by analyzing the vibration history. This algorithm is presented in more detail in Section 4.2.1. Time-domain simulations give a great deal of information on the machining process, for example estimating the cutting forces and vibration amplitude, and can be easily extended to simulate complex machining phenomena like nonlinear dynamics or toolpaths with varying engagement which cannot be modelled by frequency domain or discrete approaches, at the cost of high computation time. Cornelius *et al.* used time-domain

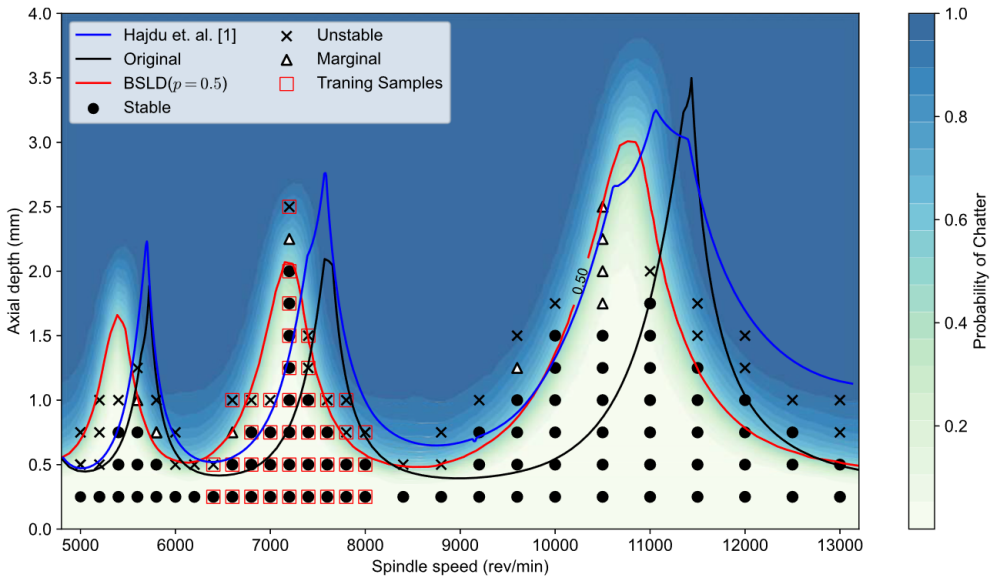


Figure 4.8: Bayesian stability updating using a surrogate model trained on a discrete stability algorithm, Figure 4.7 from [37]. The black curve is the prior stability map, while the red curve represents the posterior isocurve showing 50% of stability.

simulations to model process damping [40]. In order to make this approach feasible, a high-performance computing cluster was used to precalculate stability maps: see case study 2 in Section 5.2 for more details.

Zero-order approximation

The ZOA algorithm predicts the limiting axial depth $b_{lim}(n)$ and dominant chatter frequency f_c for given values of $\boldsymbol{\theta}$ [4]. This method removes the time-dependent cutting force variation that occurs as the tool rotates by approximating the cutting force values as the average force through the entire tool rotation (specifically, as the 0th term in a Fourier series). As a result, the ZOA is able to estimate the axial stability limits for a given range of spindle speeds and a set radial depth of cut in one pass, rather than having to iterate for each spindle speed/radial depth/axial depth combination like discrete methods or time domain simulations.

The following is a brief description of how to use the ZOA to calculate a stability map. First, the directional dynamic force coefficient matrix $[A_0]$ is constructed as shown in Equation (4.13). This matrix, which incorporates the cutting force model values K_s and β, describes how the average cutting force will change if the tool deflects a unit amount relative to the workpiece. For example, α_{xx} is the amount the average cutting force in the X direction will change for a unit deflection in the X direction. For a full derivation of how to calculate this, see [3, 4].

$$[A_0] = \frac{N_t}{2\pi}\begin{bmatrix} \alpha_{xx} & \alpha_{xy} \\ \alpha_{yx} & \alpha_{yy} \end{bmatrix} \tag{4.13}$$

This can be used together with the frequency response function matrix (Equation (4.14)) to give the characteristic equation of the milling system for each frequency f in the FRF, shown in Equation (4.15). Note that the FRF matrix assumes that input forces in the X direction will not cause a response in Y and vice versa (i.e., no crosstalk).

$$[FRF](f) = \begin{bmatrix} FRF_{xx}(f) & 0 \\ 0 & FRF_{yy}(f) \end{bmatrix} \tag{4.14}$$

$$\det\left(\frac{2\pi}{N_t}\left[A_0\right]\left[FRF\right](f) - \lambda(f)\left[I\right]\right) = 0 \tag{4.15}$$

There are two complex eigenvalue solutions $\lambda_{1,2}(f)$ for each frequency in the FRF, which are used to calculate the limiting axial depth of cut $\tilde{b}_{lim}$ and spindle speed $\tilde{n}$ as shown in Equations (4.16) and (4.17). These calculations are performed for each frequency using both eigenvalues and different values of j which represents the integer number of complete oscillations the tool will make every flute pass. Note that $\tilde{\epsilon}$ in Equation (4.17) is a function of the eigenvalue phase angle: see [4] for a full derivation. The lowest calculated b_{lim} value for any given spindle speed is the stability boundary above which chatter will occur, and the frequency from the FRF associated with that value will be the dominant chatter frequency, as shown in Figure 4.9.

$$\tilde{b}_{lim}(f) = \frac{2\pi}{N_t k_s \sin(\beta)\left|\lambda(f)\right|}\left(1 + \left(\frac{\lambda_{Im}(f)}{\lambda_{Re}(f)}\right)^2\right) \tag{4.16}$$

$$\tilde{n}(f) = \frac{f}{N_t\left(j + \frac{\tilde{\epsilon}(f)}{2\pi}\right)},\ j = 0, 1, 2, \ldots \tag{4.17}$$

Time domain simulation

Time domain simulations (TDS) simulate a test cut by dividing it into small incremental timesteps. At each timestep, the instantaneous engagement of the tool based on the cutter's current rotation and deflection is calculated and used to estimate the cutting force. That force is then applied to the system's modes and the vibration of the tool is calculated to estimate where the cutter will be at the next timestep. The following steps walk through the time domain simulation in more detail. First, the timestep and simulation length are decided based on the lowest and highest natural frequencies using Equations (4.18) and (4.19) respectively.

$$dt = \frac{1}{50 \cdot \max(f_n)} \tag{4.18}$$

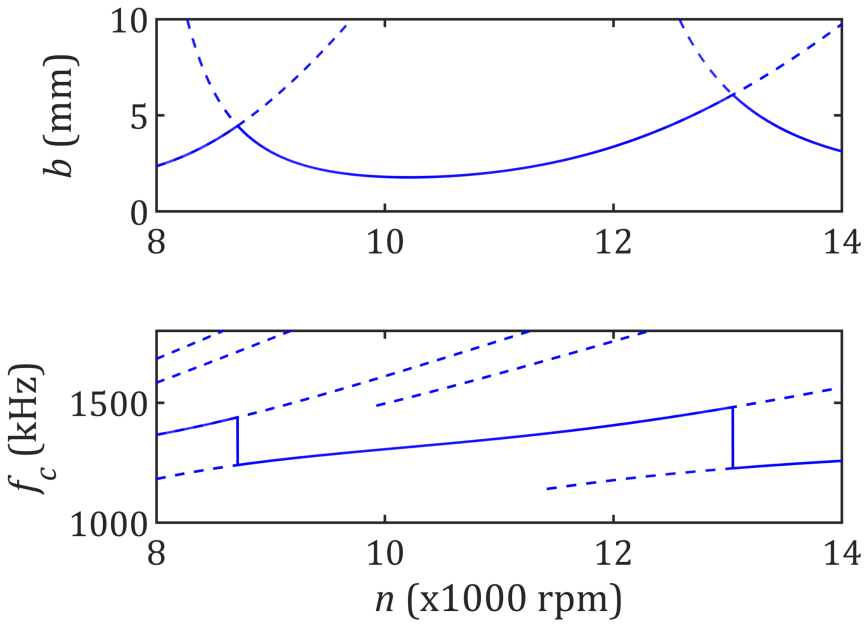


Figure 4.9: (Top) Stability map calculated using the ZOA algorithm. Dashed lines are possible stability limits given by the $\tilde{b}_{lim}$ values. The solid line is the b_{lim} value. (Bottom) Chatter frequencies for the stability map. Dashed lines are possible chatter frequencies, while the solid line is the dominant chatter frequency associated with b_{lim}.

$$t_{max} = 10000 \cdot \frac{1}{\min(f_n)} \tag{4.19}$$

Next, the tool is discretized into a set of small flute elements, as shown in Figure 4.10 When the tool has straight flutes (i.e., a helix angle of 0°) the height of each flute element can be set equal to the axial depth of cut. If the tool has helical flutes, then the height of each element is set based on the simulation timestep. After this, a surface displacement map is initialized. This records the deviation of the cut surface compared to the theoretical surface that would be obtained if the system was rigid, allowing for the regenerative effect to be modelled.

Next, the simulation begins. For simplicity, the below equations will have a tool with symmetric single degree of freedom dynamics and a rigid workpiece, but it is trivial to extend this approach for an arbitrary number of modes. At each timestep i:

1. Initialize the X and Y axis cutting forces for this timestep $F_{x\,i}$ and $F_{y\,i}$ to zero.

2. For each flute element j:

   (a) Calculate the instantaneous chip thickness h for that element, based on the current cutter angle, the current displacement of the tool, and the deviation of the workpiece surface at that location. If the tool is not currently engaged ($h < 0$) then continue to the next flute element.

   (b) Calculate the normal and tangential cutting forces $F_{n\,j}$, $F_{t\,j}$ on that element based on the cutting force model describe in Section 4.1.1.

   (c) Project $F_{n\,j}$ and $F_{t\,j}$ into the X and Y directions as $F_{n\,j}$ and $F_{n\,j}$.

   (d) Increment $F_{x\,i}$ and $F_{y\,i}$ by $F_{n\,j}$ and $F_{n\,j}$ respectively.

   (e) Update the surface displacement map in the area that was cut based on the current vibration of the tool.

3. Calculate the instantaneous accelerations $\ddot{x} = (F_x - c\dot{x} - kx)/m$ and $\ddot{y} = (F_y - c\dot{y} - ky)/m$.

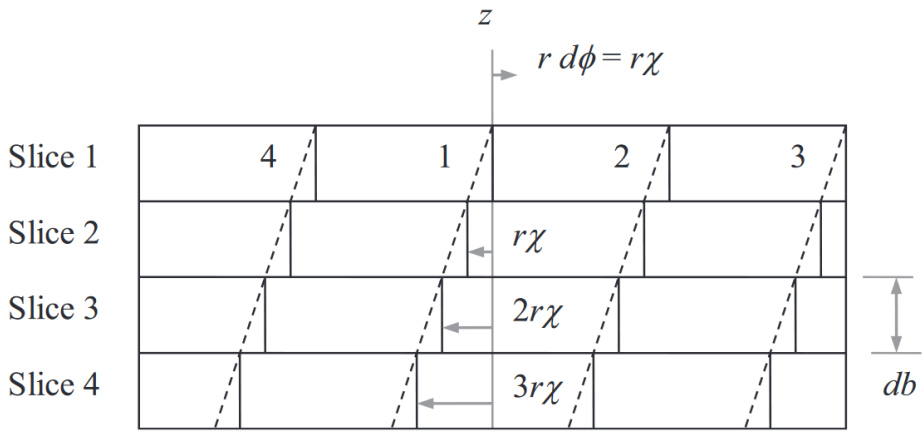


Figure 4.10: Cutting tool divided into discrete elements for time domain simulation, Figure 4.55 from [3].

4. Calculate the new velocities $\dot{x} = \dot{x} + \ddot{x} \cdot dt$ and $\dot{y} = \dot{y} + \ddot{y} \cdot dt$.

5. Calculate the new positions $x = x + \dot{x} \cdot dt$ and $y = y + \dot{y} \cdot dt$.

6. Advance to the next timestep and rotate the tool to its new angle.

Once the simulation has been run, the results are classified as stable or unstable. One way of doing this is using the M-metric cutoff [56]. This equation measures the average variation in displacement every toothpass. In an ideal stable cut, the displacement will perfectly repeat every tooth pass, while unstable cuts will vary. First, one of the signals in the simulation is selected, most commonly the feed-direction displacement. The chosen signal is truncated, getting rid of the first 25% to remove the initial transients, and then sampled at the tooth-passing frequency, obtaining a series of samples where $x_s(i)$ is the ith sample. Equation (4.20) is used to calculate the variation in displacement M, which is then compared to a preset stability cutoff typically somewhere between 1 and 10 µm. If M is greater than the cutoff, then the cut is classified as unstable, while M values less than the cutoff are classified as stable. The signal can also be analyzed to determine the chatter frequency.

$$M = \frac{1}{N-1} \sum_{i=2}^{N} |x_s(i) - x_s(i-1)| \tag{4.20}$$

This simulation and classification process is then repeated for every other set of cutting parameters that must be examined. Figure 4.11 shows an example stability map calculated with time domain simulation. This example closely matches the solution obtained from the ZOA algorithm.

4.2.2 Cutting power

Figure 4.12 shows the spindle power curve for a Haas VF-4SS machining center [57]. The different lines represent how much power the spindle is capable of outputting for various time intervals. The lowest line is the continuous duty cycle which can be maintained indefinitely, while the highest line can be maintained for three minutes before the spindle overheats and is damaged. It is therefore important to predict the spindle power required for a given

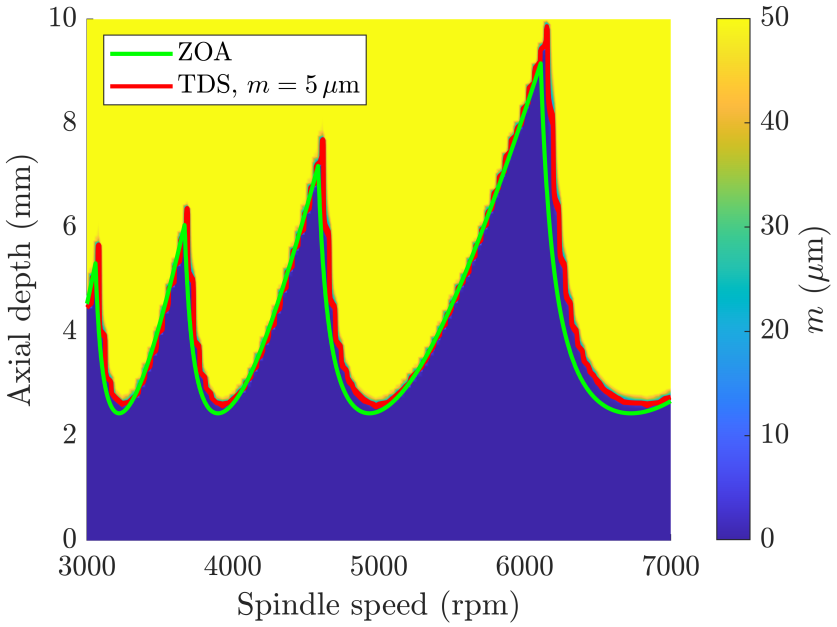


Figure 4.11: TDS-calculated stability map, compared to the ZOA algorithm.

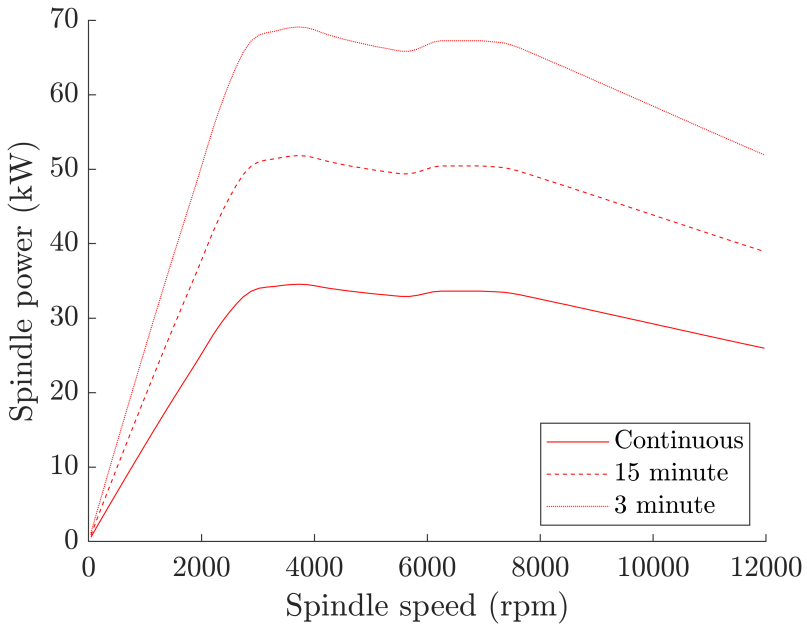


Figure 4.12: Spindle power curve for a Haas VF4SS machining center [57].

cut to confirm that the machine's capabilities are not exceeded. The instantaneous cutting torque and power changes continuously as the tool rotates and can be calculated using the cutting force model described in Section 4.1.1. However, due to the inertia of the rotating spindle assembly, the cut is generally limited by the average cutting power rather than the peak instantaneous power. The average cutting power $\bar{P}$ can be estimated for a given set of model inputs $\boldsymbol{\theta}$ and cutting parameters ω using Equation (4.21), where N_t is the number of flutes on the tool and d is the outside cutting diameter of the tool [3].

$$\bar{P}(\boldsymbol{\theta}, \omega) = k_{tc} \cdot MRR(\omega) + k_{te} N_t \omega_b \frac{\omega_n d \arccos(1 - 2\omega_a/d)}{120} \tag{4.21}$$

4.2.3 Spindle bending moment

Every spindle design has a bending moment limit based on how securely the tool holder is retained in the spindle: exceeding the limit will cause the tool holder to separate from the spindle and cause damage to both the toolholder and spindle tapers. Figure 4.13(a) shows how the cutting force produces a large bending moment on long tools, and Figure 4.13(b) shows the fretting damage that can occur to toolholders if the bending moment is exceeded. For reference, the common CAT40 spindle interface has a bending moment limit of 230 Nm, while an HSK63A spindle has limit of 420 Nm [58]. Unlike the spindle torque limits, it is unacceptable to exceed the bending moment limit even momentarily since it can cause permanent damage to the spindle interface.

The bending moment can be estimated using the mechanistic cutting force model described in Section 4.1.1. The tool is discretized into small elements with height db. For each element, the instantaneous tangential and normal cutting force components are calculated using Equations (4.22) and (4.23) respectively (where h is the chip thickness) and projected into the machine's X and Y directions (Figure 2.5). The forces on each element are then summed to obtain the overall cutting forces F_X and F_Y. Assuming that force is evenly distributed over the axial depth of cut, the instantaneous bending moment is calculated using Equation (4.24), where L_g is the cutting tool gage length measured from the spindle nose.



(a) **(b)**

Figure 4.13: (a) Long tools will experience high bending moments due to cutting forces [58]. (b) CAT40 toolholder exhibiting fretting damage at the top and bottom of the taper [59].

$$F_t = \begin{cases} K_s \sin(\beta) h \, db & h \geq 0 \\ 0 & h < 0 \end{cases} \tag{4.22}$$

$$F_n = \begin{cases} K_s \cos(\beta) h \, db & h \geq 0 \\ 0 & h < 0 \end{cases} \tag{4.23}$$

$$M_b(\boldsymbol{\theta}, \omega) = \left(L_g - \frac{\omega_b}{2} \right) \sqrt{F_X^2 + F_Y^2} \tag{4.24}$$

Note that this model ignores dynamic forces due to the vibration of the tool and holder. However, it serves as a reasonable first-order approximation for checking whether an operation is safe to run. To make this computationally-efficient, the bending moment can be calculated for unit values of k_{tc} and k_{nc} for different axial depths, then multiplied by the true values and added when calculating each sample. For unstable cuts, a constant factor can be multiplied by the predicted bending moment to estimate the higher cutting forces experienced in chatter.

4.2.4 Tool bending stress

With some cutting tools, the bending stress experienced by the tool shank due to the cutting forces and tool deflection can be enough to break the endmill, even during stable cuts. Several authors have created models for predicting bending stress in endmills during cutting [60, 61]. However, these models have generally focused on failure due to static forces and cutter edge fracture. At some combinations of stable cutting parameters, the dynamic cutting forces can excite the cutting tool's natural frequency and result in higher amplitude vibrations, and thus higher bending stress, than a static force model would predict. A novel model is proposed for estimating bending stress in stable cuts by approximating the tool as a fixed-free Euler-Bernoulli beam. This model is appropriate for tools with long stickout lengths such that the holder and machine tool can be approximated as rigid, shown in Figure 4.14.

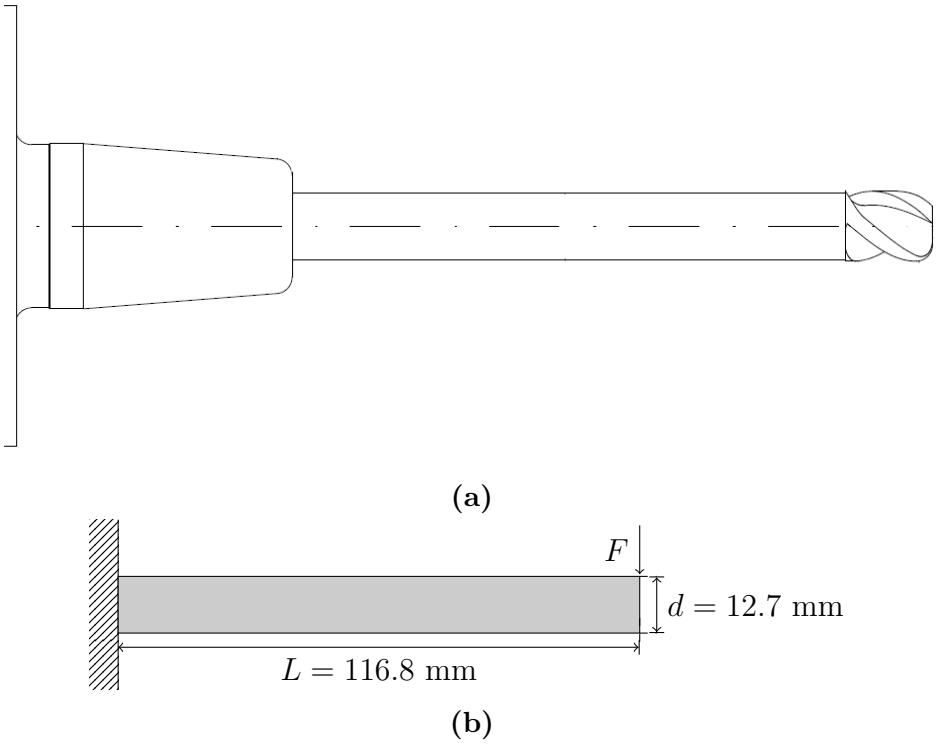


Figure 4.14: (a) A long endmill which is susceptible to breaking due to excessive deflection. (b) That same tool modeled as a clamped-free beam.

The bending stress is predicted using two steps. First, the tool's maximum displacement Δ_{max} is estimated using a frequency domain algorithm, similar to the frequency domain surface location error algorithm defined by Schmitz and Mann [62]. Let $F_x(f), F_y(f)$ be the cutting forces expressed in the frequency domain for a given frequency f. These frequency domain forces can be obtained either using a Fourier series approximation [3] or using the mechanistic force model described in Section 4.1.1. The frequency domain displacements are then obtained by multiplying the forces by the X and Y axis frequency responses as shown in Equation (4.25). The inverse Fourier transform is applied to $X(f), Y(f)$ giving the time domain displacements $x(t), y(t)$. Then the maximum tool displacement Δ_{max} is calculated using Equation (4.26).

$$X(f) = F_x(f) \cdot FRF_{xx}(f),\ Y(f) = F_y(f) \cdot FRF_{yy}(f) \tag{4.25}$$

$$\Delta_{max}(\boldsymbol{\theta}, \omega) = \max_t \sqrt{x(t)^2 + y(t)^2} \tag{4.26}$$

Next, this displacement is used to estimate the bending stress at the base of the beam. The first mode shape for a clamped-free beam is given by Equation (4.27), where $\Delta(x)$ is the relative magnitude of vibration for a given position x on the beam, L is the length of the beam, and $\lambda \approx 1.875$ and $\sigma \approx 0.734$ are mode shape constants for a clamped-free beam [63]. The moment at a given location on the beam is given by Equation (4.28), where E is the elastic modulus of the material and I is the second moment of area for the beam shape cross section. Based on this, Equation (4.29) gives the bending moment at the base of the beam for a displacement Δ at the beam tip. Finally, the maximum bending stress $\sigma_{max}(\omega)$ for a set of cutting parameters ω is calculated with Equation (4.30), where K_t is a stress concentration factor that may be applied for reduced-shank endmills and d is the diameter of the tool.

$$\Delta(x) = \cosh(\lambda x/L) - \cos(\lambda x/L) - \sigma(\sinh(\lambda x/L) - \sin(\lambda x/L)) \tag{4.27}$$

$$M_B(x) = EI\frac{d\Delta^2}{d^2x} \tag{4.28}$$

$$M_{b,\,x=0}(\Delta) = \frac{2\Delta E I \lambda^2}{L^2\left(\cosh(\lambda) - \cos(\lambda) - \sigma(\sinh(\lambda) - \sin(\lambda))\right)} \tag{4.29}$$

$$\sigma_{max}(\boldsymbol{\theta}, \omega) = K_t M_{b,\,x=0}(\Delta_{max}(\boldsymbol{\theta}, \omega))\frac{d}{2I} \tag{4.30}$$

σ_{max} can be compared to the tool material's ultimate tensile strength to predict whether the cutting tool will fail during the cut. In the example shown in Figure 4.15, a cut at 22500 rpm and 1 mm axial depth should be stable, but the bending stress would exceed the tool material's tensile strength and break it. By decreasing the spindle speed to 21000 rpm, it is possible to have a stable cut which will not break the tool. Note that the highest stresses occur when the cutting tool's natural frequency is close to a harmonic of the flute-passing frequency $f_t = N_t \frac{n}{60}$: while this situation is the best case for milling stability, it also causes the highest-amplitude vibrations [62]. Both factors must be considered when selecting cutting parameters.

Note that this method is not valid for unstable cuts since self-excited vibrations are not linear with respect to the cutting force. Calculating displacement using time-domain methods is computationally demanding. Instead, a simple safety factor can be defined which multiplies the bending stress predictions for unstable cuts by a constant multiplier. Since the Bayesian method will generally be suggesting tests which are close to the stability boundary rather than far above it, this can provide a reasonable safety margin.

In practice, this calculation can be performed quickly by precalculating the frequency-domain forces for unit values of the cutting force coefficients, feed per flute, and spindle speed. For example, let $F_x(f, [k_{tc}, k_{nc}, k_{te}, k_{ne}], f_z, n)$ be the X-axis frequency domain forces as a function of the four-element cutting force model and the spindle speed n. Since the cutting force is modelled as linear with respect to chip thickness, Equation (4.31) can then be used to calculate the cutting force for a given set of cutting force coefficients, frequency, spindle speed, and feedrate. This same precalculation optimization can also be applied to other

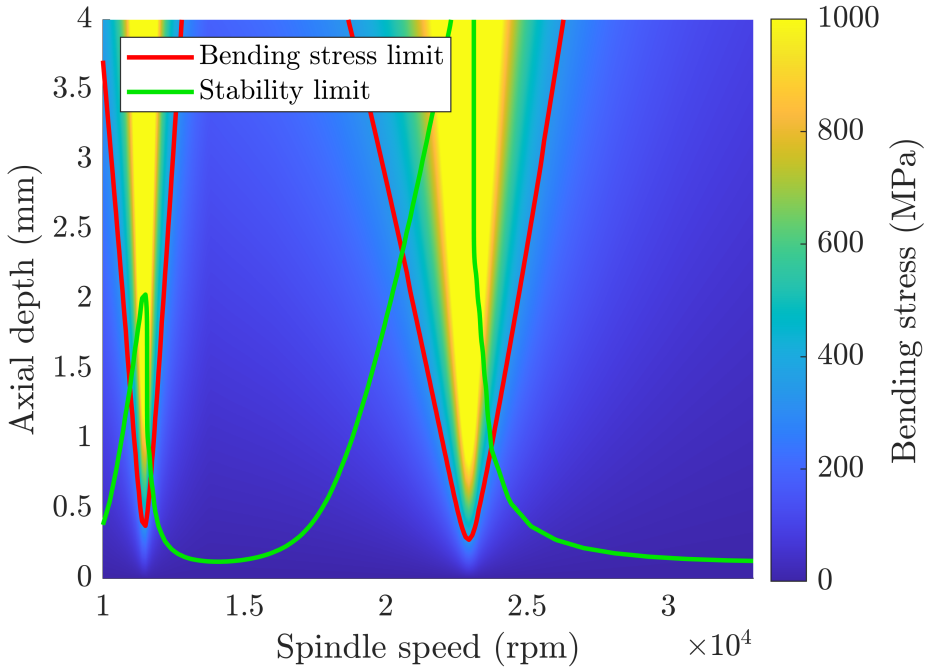


Figure 4.15: Predicted bending stress and acceptable cutting parameter limits for the two flute endmill shown in Figure 4.14

physics algorithms, such as surface location error, torque, or bending moment estimates.

$$F_x(f,[k_{tc},k_{nc},k_{te},k_{ne}],f_z,n) = f_z \cdot k_{tc} \cdot F_x(f/n,[1,0,0,0],1,1)+ \\ f_z \cdot k_{nc} \cdot F_x(f/n,[0,1,0,0],1,1) + k_{te} \cdot F_x(f/n,[0,0,1,0],1,1)+ \\ k_{ne} \cdot F_x(f/n,[0,0,0,1],1,1) \quad (4.31)$$

4.3 Test selection

To date, most work on Bayesian machining modelling has applied simple grid tests where tests are run at even intervals of axial depth and spindle speed, as shown in Figure 4.8. Generally, cutting tests are run at each spindle speed at increasing axial depths until chatter is reached, at which point testing moves to the next spindle speed. This approach is simple to run, but can be time-consuming due to the number of tests required to comprehensively span the space of possible cutting parameters. Furthermore, many of the cutting tests provide little or no new information about the system.

Authors have applied several different methods for selecting more informative tests, depending on their specific goals. Karandikar *et al.* has proposed two different testing algorithms for the non-physics-informed Bayesian stability model. The first is designed to maximize material remove rate, selecting cutting tests based on the expected improvement in MRR [30, 35]. Equation (4.32) shows how the expected value is calculated for each set of cutting parameters, where MRR_{best} is the highest MRR that has been already been identified. Figure 4.16 shows how the test is selected at the point with the highest value.

$$E(MRR(\omega)) = MRR(\omega) \cdot P_S(\omega) - MRR_{best} \quad (4.32)$$

Karandikar *et al.* also proposed an uncertainty-based test selection algorithm which selects cutting tests which will improve the confidence of the stability map the most [34]. A perfectly confident stability map will have a probability of stability of either 0% or 100% at all points, while a maximally unconfident map will have a probability of 50% everywhere. Let

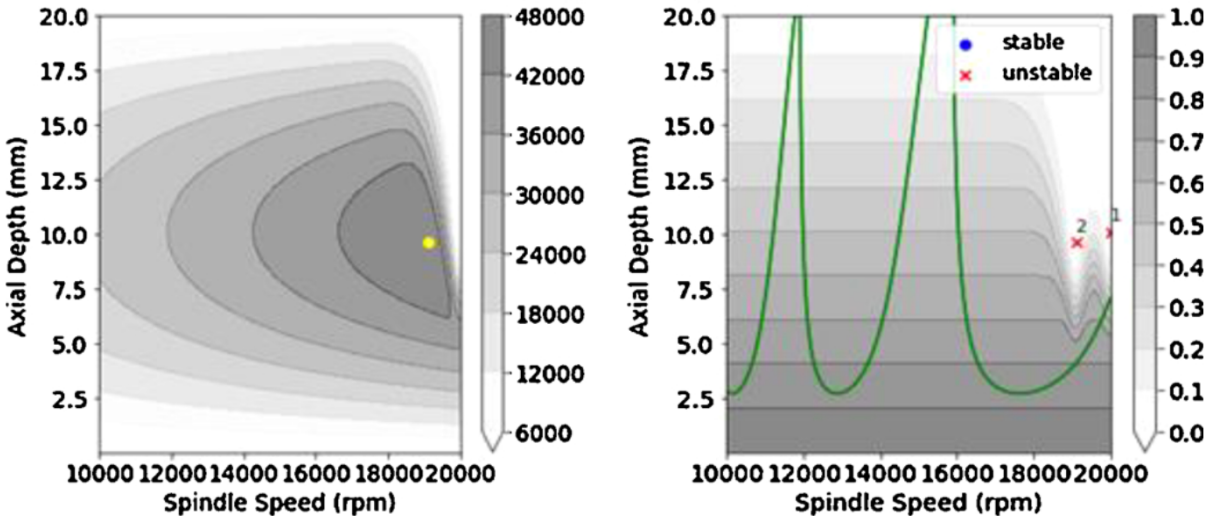


Figure 4.16: Expected value test selection, Figure 20 from [35]. The left image shows how the second test was selected at the point with the highest value, and the right image shows the posterior stability map after running that test.

the average uncertainty for the stability map U_S be given by Equation (4.33). For any given set of test parameters ω, the expected posterior stability uncertainty is given by Equation (4.34), where $U_S(\omega, S)$ is the uncertainty the stability map would have if a test at ω was stable and $U_S(\omega, \neg S)$ is the uncertainty if the test is unstable. This expected uncertainty is evaluated for all possible cutting parameters and the next test is selected which has the lowest expected posterior stability. This approach therefore focuses on exploring the stability map as a whole, rather than maximizing productivity. Figure 4.17 shows a series of test cuts selected via this method.

$$U_S = \int_\Omega \min(P_s(\omega), 1 - P_s(\omega))\, d\omega \tag{4.33}$$

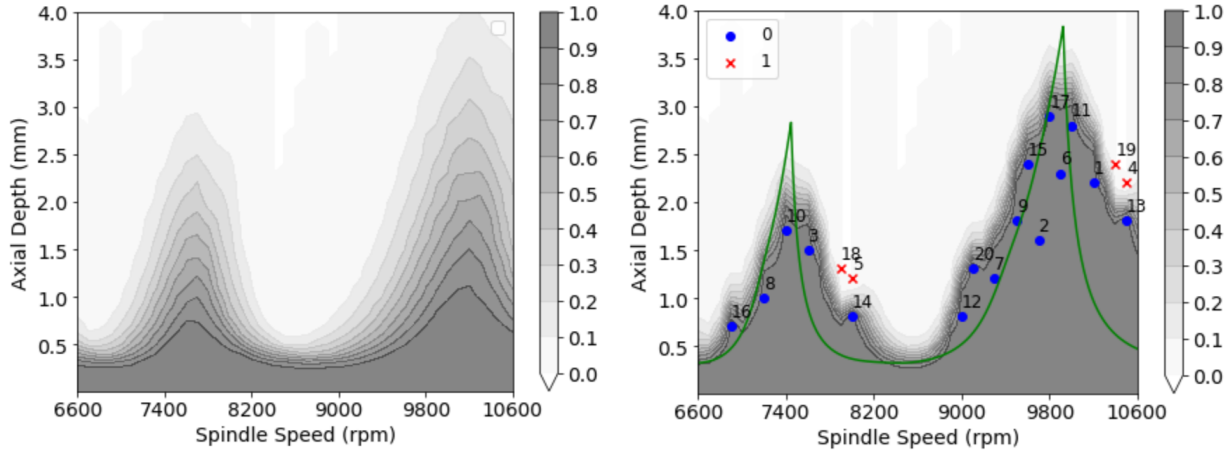
$$E[U_S] = P_S(\omega)U_S(\omega, S) + (1 - P_S(\omega))U_S(\omega, \neg S) \tag{4.34}$$

Akbari *et al.* applied a similar method to select tests for their physics-informed Bayesian stability model, selecting tests based on how much new information they give about the stability map. A Monte Carlo method was used to evaluate how any given test point was likely to affect the stability map: see Section 4.3.2 for an example of how this can be rapidly evaluated. They also demonstrated how this testing approach can be constrained to select tests with low axial depth to reduce the risk of damage. Rather than selecting the point with the highest information gain $\mathcal{I}_{max}$, the user selects a hyperparameter λ and selects a test point according to Equation 4.35. For example, if $\lambda = 0.9$ then the user selects the cutting test with the lowest axial depth which gives at least 90% of the maximum attainable information. Figure 4.18 shows a test selected via this method.

$$\omega_{test} = \arg\min_{\omega \in \Omega}(\omega_b) \text{ s. t. } \mathcal{I}(\omega) \geq \lambda \mathcal{I}_{max} \tag{4.35}$$

4.3.1 MRR maximization with risk tolerances

Cutting tests should be selected to both find higher-productivity machining parameters and to explore the stability map, while simultaneously controlling the risk of damaging the cutting tool via severe chatter. Let R_{stable} be a user-specified risk tolerance describing how willing



(a) **(b)**

Figure 4.17: Stability uncertainty test selection from [34]. (a) Prior, Figure 7. (b) Posterior, Figure 8. The green line is the nominal stability boundary calculated using the ZOA.

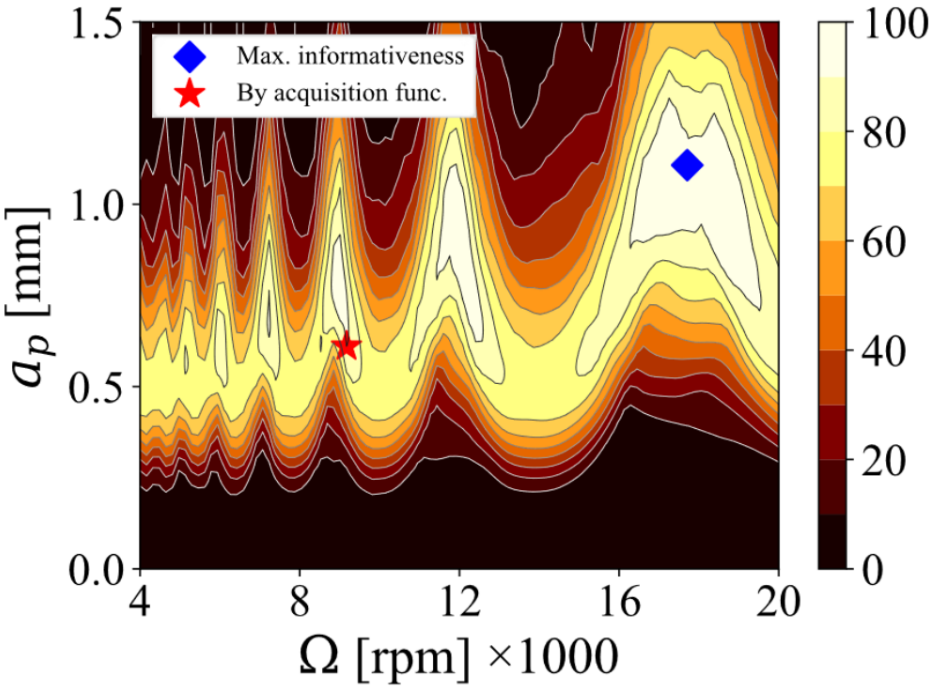


Figure 4.18: Entropy-based test selection, Figure 12 from [39]. The blue diamond is the point with highest possible information about the stability map, while the red star is the lowest axial depth point with at least $\lambda \mathcal{I}_{max}$ informativeness.

they are to accept that a cutting test may be unstable. For example, $R_{stable} = 0.75$ indicates that the user is only willing to run cutting tests which have at least a 75% chance of being stable. Equation (4.36) selects the most productive machining parameters that obey this risk tolerance. Higher risk tolerances will generally select more productive cutting parameters but also be more likely to encounter chatter, while lower risk tolerances will avoid chatter but may take longer to converge or find lower productivity parameters. This is demonstrated in Section 5.1.

$$\omega_{test} = \arg\max_{\omega \in \Omega}(MRR(\omega)) \text{ s. t. } P_S(\omega) \geq R_{stable} \tag{4.36}$$

This formula can be extended to include other constraints on the cutting parameter selection, such as the spindle torque curve and bending moment limit. Exceeding these parameters can cause the machine to alarm out or potentially even permanent damage to the cutting tool, holder, and machine. It is especially important to incorporate these machine limitations into the Bayesian method since it seeks to find the maximum possible cutting performance and may thus be considerably more aggressive than traditional machining parameters.

Cutting torque and bending moment can be estimated for each sample at each possible set of cutting parameters using the equations outlined in Sections 4.2.2 and 4.2.3. These are used to calculate the probability that the cut will not exceed spindle limits $P_{spindle}(\omega)$ using Equation (4.37), which compares the predicted torque and moments against the machine's maximum values $\bar{T}_{max}$ and M_{max} respectively. Any proposed cutting tests where $P_{spindle}(\omega) > R_{spindle}$ will be rejected, where $R_{spindle} \ll R_{stable}$ is a user-specified risk tolerance for damage to the machine. This gives the modified test selection algorithm shown in Equation (4.38):

$$P_{spindle}(\omega) = \frac{1}{N}\sum_{i=1}^{N} M(\boldsymbol{\theta}_i, \omega) < M_{max} || \bar{T}(\boldsymbol{\theta}_i, \omega) < \bar{T}_{max} \tag{4.37}$$

$$\omega_{test} = \arg\max_{\omega \in \Omega}(MRR(\omega)) \text{ s. t. } P_S(\omega) \geq R_{stable},\ P_{spindle}(\omega) \geq R_{spindle} \tag{4.38}$$

4.3.2 Expected reduction in uncertainty

Existing testing strategies have focused either on maximizing material removal rate or on the uncertainty of the stability map. However, one other possible objective is to select cutting tests in order to learn as much as possible about the underlying system. In order to achieve this, a cutting test can be selected which has the highest expected reduction in uncertainty. The expected reduction in uncertainty due to observing a stable or unstable cut is given by Equation (4.39), where $u(x|S(\omega))$ is the uncertainty that variable x would have if a test with the cutting parameters ω were to be stable and $u(x|\neg S(\omega))$ is the uncertainty in x if the test was unstable. Karandikar *et al.* tested a similar approach to learn process damping, but it was computationally expensive since it required fully evaluating their posterior [64].

$$E(\Delta u(x|\omega)) = u(x) - u(x|S(\omega)) \cdot P(S(\omega)) - u(x|\neg S(\omega)) \cdot (1 - P(S(\omega))) \tag{4.39}$$

Instead, the posterior can be quickly approximated for each set of cutting parameters by weighting the prior samples according to their conditional probability and evaluating the uncertainty of x while considering the weights. Equation (4.40) gives the updated weight w_i^* for each sample for stable and unstable cases, where $P_S(\omega|\,\boldsymbol{\theta}_i)$ is the predicted probability that at a test cut at the parameters ω will be stable according to the sample $\boldsymbol{\theta}_i$ and w_i is the initial weight for the sample (generally $w_i = 1$, at least for the initial Monte Carlo sampling). For example, $L_{stable}(\omega|\,\boldsymbol{\theta}_i) = 0.8$ indicates that the sample $\boldsymbol{\theta}_i$ predicts that the cutting parameters ω have an 80% chance of being stable. See Section 4.4.1 for more information: generally, the Gaussian CDF approach is most logical for this algorithm since it is automatically normalized and has an intuitive explanation for the assigned uncertainty.

$$w_i^* = w_i \begin{cases} L_{stable}(\omega|\,\boldsymbol{\theta}_i) & \text{Stable case} \\ 1 - L_{stable}(\omega|\,\boldsymbol{\theta}_i) & \text{Unstable case} \end{cases} \tag{4.40}$$

Once these weights are known, they are used to calculate the weighted uncertainties for both the stable and unstable cases. For the specific case where x is a univariate variable contained inside the $\boldsymbol{\theta}$ vector, the biased weighted standard deviation can be estimated using Equation (4.41), where μ_w is the weighted arithmetic mean for x. For example, this could

be used to select a cutting test which is most expected to reduce the uncertainty about the cutting force angle β. Note that, since this distribution will be calculated across thousands of samples, the use of the biased estimator is not particularly important.

$$\hat{\sigma}_w^2 = \frac{\sum_{i=1}^{N} w_i (x_i - \mu_w)^2}{\sum_{i=1}^{N} w_i} \tag{4.41}$$

Figure 4.19 shows an example probabilistic stability map, calculated using uncertainty in the specific cutting force and force angle K_s and β and the modal parameters f_n, k, and ζ. The stability uncertainty is set to $\sigma_b = 0.125$ mm. Figure 4.20 shows the expected percentage reduction in uncertainty for each of these variables. Calculating the expected uncertainty reduction map for each of these took approximately 18 seconds for each case. These tests can also be constrained based on other factors, for example to minimize the risk of damage to the machine or tool.

This equation can be extended to consider other other information gained from test cuts. Equation (4.42) shows how $u(x|\neg S(\omega)$ can be modified to incorporate the chatter frequency observed in unstable test cuts, where $u(x|\neg S(\omega, f_c))$ is the uncertainty that variable x would have if chatter frequency f_c and $L_{f_c}(f_c|\,\boldsymbol{\theta}_i)$ is the prior probability of encountering a specific chatter frequency (see Section 4.4.2 for details on the likelihood formula). In practice, this can be done using the samples from the prior distribution by getting a list of all of the predicted chatter frequencies for the cutting parameters ω, evaluating the posterior likelihood for each of them, and and averaging their values. Implementing this approach gives Figure 4.21. Each of these figures was evaluated using only 500 samples for computational speed, taking roughly three minutes per figure. This can also be trivially applied to include other factors, for example the spindle torque. Additionally, the tests can be selected to maximize safety. Equation 4.43, for example, selects the cutting parameters with the lowest axial depth which provide at least a 75% expected reduction in the natural frequency uncertainty.

$$u(x|\neg S(\omega)) = \int_0^\infty u(x|\neg S(\omega, f_c)) P(f_c)\, df_c \tag{4.42}$$

$$\omega_{test} = \underset{\omega \in \Omega}{\arg\min}(\omega_b) \text{ s. t. } E(\Delta u(f_n|\omega)) \geq 0.75 \cdot u(f_n) \tag{4.43}$$

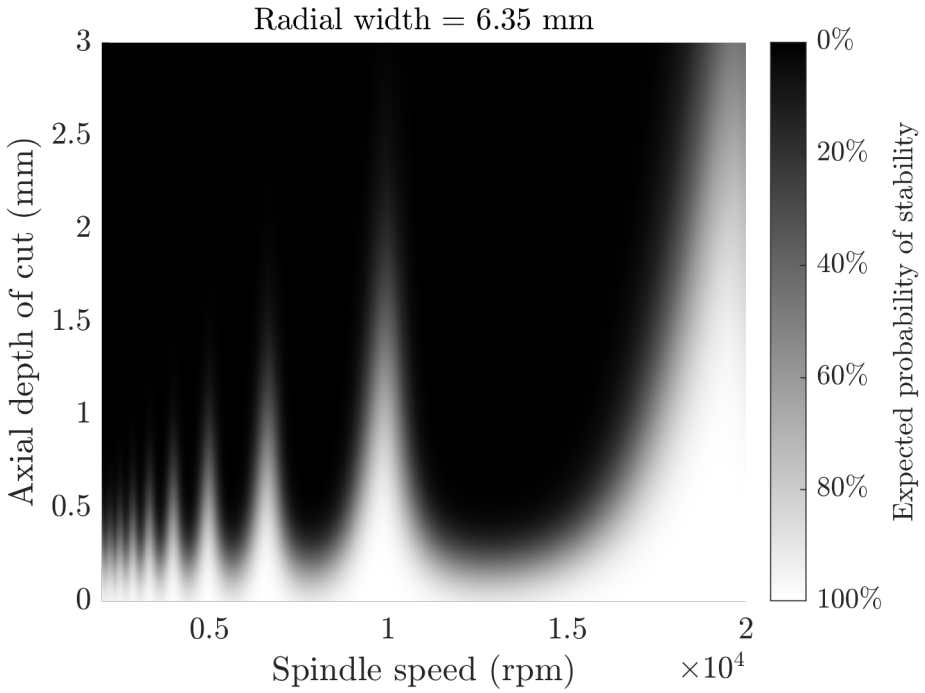


Figure 4.19: Probabilistic stability map for demonstrating uncertainty reduction test selection

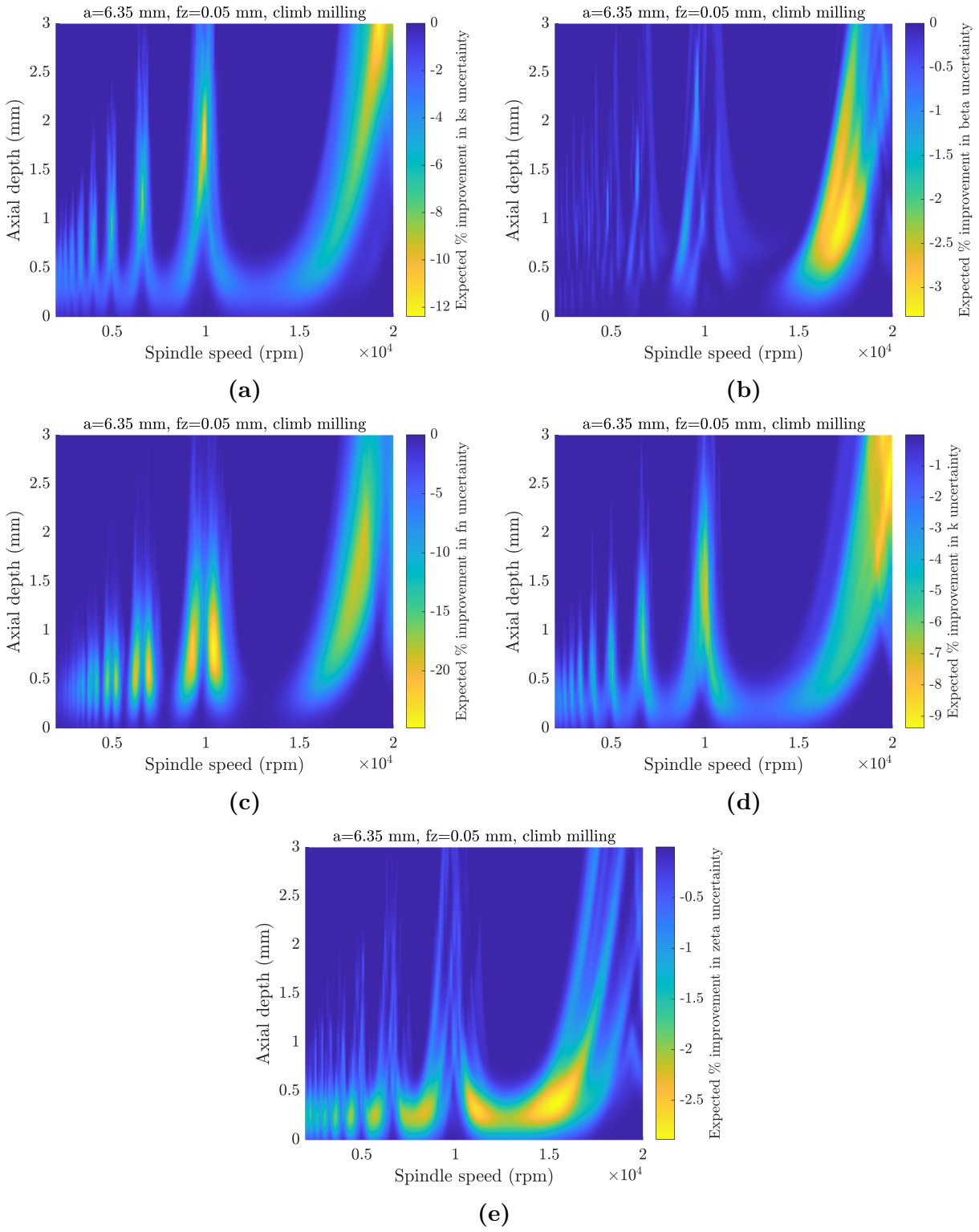


Figure 4.20: Expected reduction in system uncertainty, considering only binary stable/unstable test results, for (a) K_s, (b) β, (c) f_n, (d) k, and (e) ζ.

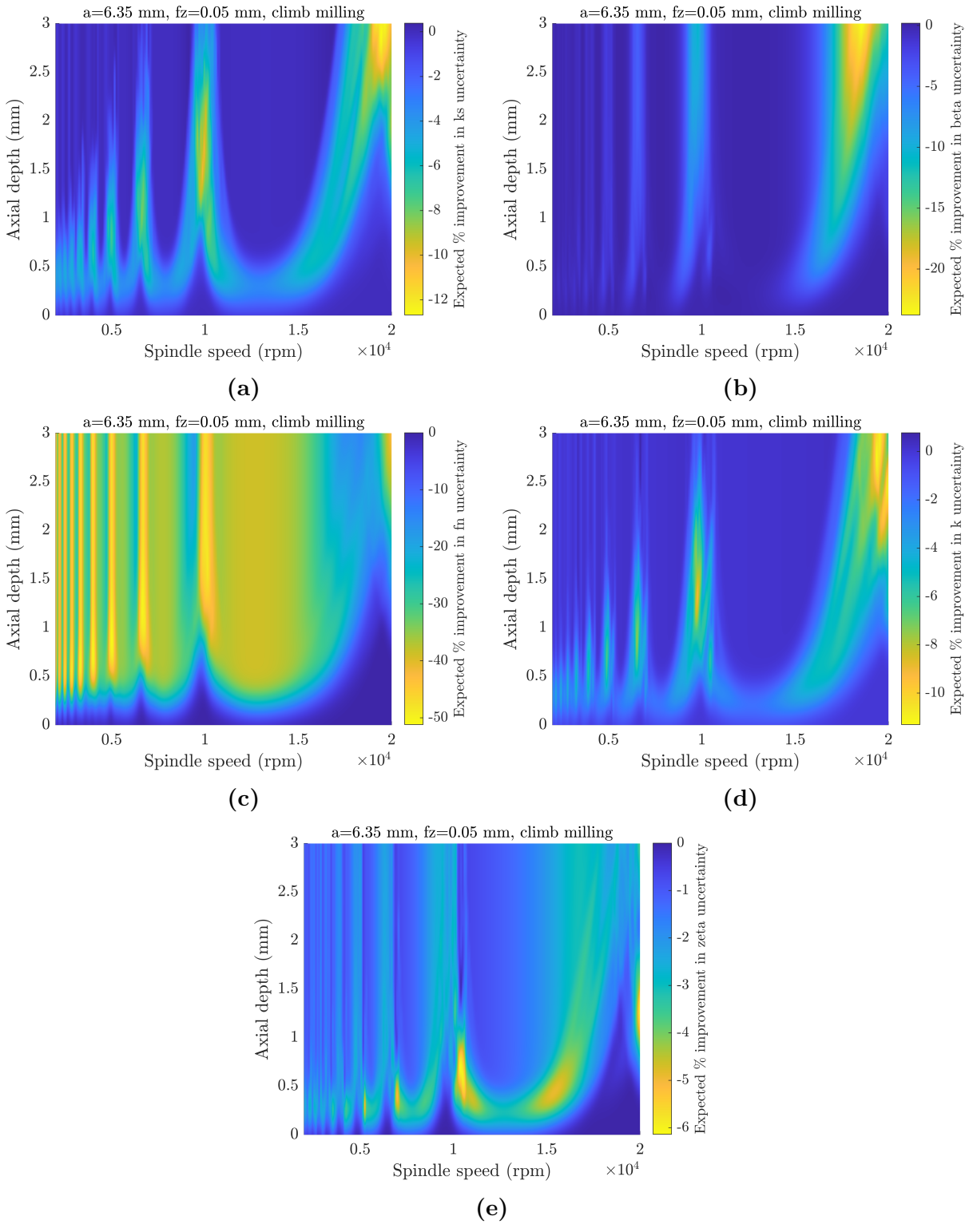


Figure 4.21: Expected reduction in system uncertainty considering both stability and chatter frequency likelihood for (a) K_s, (b) β, (c) f_n, (d) k, and (e) ζ.

4.4 Defining the posterior distribution

Let α be a variable containing all information gathered about the test cut, including the tested cutting parameters α_ω, binary stable/unstable classification α_S, chatter frequency α_{f_c}, and the average cutting torque $\alpha_{\bar{T}}$. This information can be used to define a new (non-normalized) probability distribution $p(\boldsymbol{\theta}\,|\alpha)$ using Bayes' theorem (Equation (4.44)) which incorporates the information learned from the cutting test. In that equation, $0 < L(\alpha|\,\boldsymbol{\theta}) \leq 1$ is known as the likelihood function, which predicts how probable the test result α is for a given set of system parameters $\boldsymbol{\theta}$. The more accurately $\boldsymbol{\theta}$ predicts the cutting test results, the higher $L(\boldsymbol{\theta}\,|\alpha)$ is and the more likely it is that $\boldsymbol{\theta}$ is close to the true system parameters. If multiple tests have been observed, the posterior distribution that incorporates all of these observed tests is given by Equation (4.45), providing both the standard and log forms for the probability.

$$p(\boldsymbol{\theta}\,|\alpha) = P(\boldsymbol{\theta})\, L(\alpha|\,\boldsymbol{\theta}) \tag{4.44}$$

$$p(\boldsymbol{\theta}\,|\alpha_{1:n}) = P(\boldsymbol{\theta}) \prod_{i=1}^{n} L(\alpha_i|\,\boldsymbol{\theta}),\ \tilde{p}(\boldsymbol{\theta}\,|\alpha_{1:n}) = \tilde{P}(\boldsymbol{\theta}) + \sum_{i=1}^{n} \tilde{L}(\alpha_i|\,\boldsymbol{\theta}) \tag{4.45}$$

The exact form of $L(\alpha|\,\boldsymbol{\theta})$ can be customized depending on what information is gathered about the test cut. Equation (4.46) shows how the overall likelihood is calculated using individual likelihood functions for stability $L_S(\alpha|\,\boldsymbol{\theta})$, chatter frequency $L_{f_c}(\alpha_{f_c}, \alpha_\omega|\,\boldsymbol{\theta})$, spindle power $L_{\bar{P}}(\alpha_{\bar{T}}, \alpha_\omega|\,\boldsymbol{\theta})$, and whether the tool broke $L_B(\alpha_B, \alpha_\omega|\,\boldsymbol{\theta})$. Since it is often preferable to use the log likelihood $\tilde{L}(\alpha|\,\boldsymbol{\theta})$, that version is shown in Equation (4.47). The following subsections will walk through how to calculate each of these individual likelihoods. Other elements can be added to the likelihood function, depending on what specific information is obtained about the cut: for example, the surface location error could be measured and added as another variable for the updating [62].

$$L(\alpha|\,\boldsymbol{\theta}) = L_S(\alpha_S, \alpha_\omega|\,\boldsymbol{\theta}_i)\, L_{f_c}(\alpha_{f_c}, \alpha_\omega|\,\boldsymbol{\theta})\, L_{\bar{P}}(\alpha_{\bar{P}}, \alpha_\omega|\,\boldsymbol{\theta})\, L_B(\alpha_B, \alpha_\omega|\,\boldsymbol{\theta}) \tag{4.46}$$

$$\tilde{L}(\alpha|\,\boldsymbol{\theta}) = \tilde{L}_S(\alpha_S, \alpha_\omega|\,\boldsymbol{\theta}_i) + \tilde{L}_{f_c}(\alpha_{f_c}, \alpha_\omega|\,\boldsymbol{\theta}) + \tilde{L}_{\bar{P}}(\alpha_{\bar{P}}, \alpha_\omega|\,\boldsymbol{\theta}) + \tilde{L}_B(\alpha_B, \alpha_\omega|\,\boldsymbol{\theta}) \tag{4.47}$$

4.4.1 Stability classification likelihood

The stability likelihood function predicts how likely it is that a given set of cutting parameters are stable for a given set of system parameters $\boldsymbol{\theta}$. If the stability prediction was perfect, then this value would be either zero or one based on whether the point is above or below the predicted stability boundary. However, in practice there is some aleatoric uncertainty in the stability prediction itself, due to both approximation errors in the stability algorithm and uncertainties about the process which are not modeled in $\boldsymbol{\theta}$ (for example, the diameter of the cutting tool is not perfectly known, but this will generally not be a variable in $\boldsymbol{\theta}$). Therefore, even if a value of $\boldsymbol{\theta}$ perfectly matches the measured system values it is possible that it may not perfectly predict stability results. As a result, it is necessary to have some stability likelihood function which is tolerant of small errors. This also has a computational advantage: using a binary likelihood is more difficult to sample since it results in sharp dropoffs in the probability rather than smooth gradients.

Several different forms have been proposed for the stability likelihood function. The works by Karandikar, Schmitz, Cornelius, and Li have primarily used the Gaussian dropoff shown in Equation (4.48) along with its log form[1]. This approach assigns a likelihood of one unless the test result disagrees with the stability prediction, and otherwise assigns a likelihood based on how close to correct the limiting axial depth is. Δ_b is the error in the stability limit and σ_b is a hyperparameter which governs how quickly errors in the stability boundary decrease the likelihood [36, 40, 41, 49, 65]. If the prediction is correct, then Δ_b=0. Otherwise, Δ_b is the difference between the tested axial depth and the predicted limiting value, b_{lim}, at that spindle speed. Figure 4.22 demonstrates how Δ_b is obtained for an unstable test cut. This approach assumes that stability is a binary value and information is only obtained about a sample when it is contradicted. Cornelius *et al.* extended this approach to consider stability models which have multiple limiting axial depths at a given spindle speed [40].

$$L_S(\alpha_S, \alpha_\omega | \, \boldsymbol{\theta}_i) = \exp\left(-\frac{1}{2}\left(\frac{\Delta_b}{\sigma_b}\right)^2\right), \; \tilde{L}_S(\alpha_S, \alpha_\omega | \, \boldsymbol{\theta}_i) = -\frac{1}{2}\left(\frac{\Delta_b}{\sigma_b}\right)^2 \tag{4.48}$$

[1]While Li *et al.* expressed it slightly differently, in practice their formulation functions similarly.

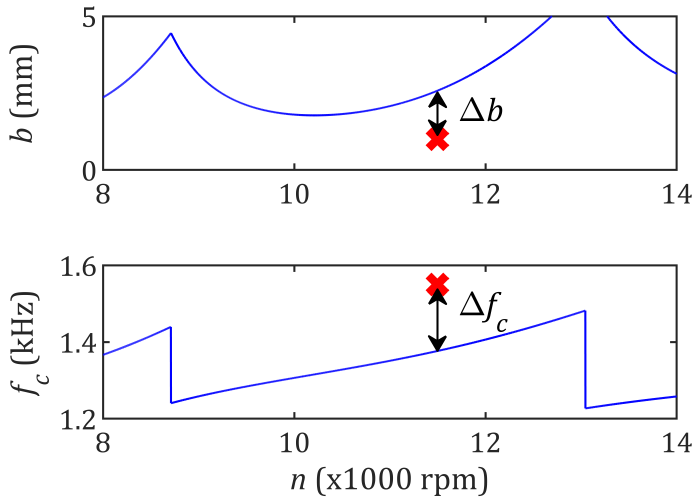


Figure 4.22: Calculating Δ_b, Δ_{f_c} for a given sample, used for calculating the stability likelihood using the Gaussian dropoff method. (Top) Δ_b is the difference between the predicted limiting depth of cut, b_{lim} (line), and the unstable axial depth ($\times$). (Bottom) Δ_{f_c} is the difference between the actual ($\times$) and predicted (line) chatter frequencies.

Akbari *et al.* instead employed a sigmoid likelihood function based on the difference between the calculated stability limit and the tested axial depth. Equation (4.49) gives the formula for a stable cut, where k is a stability hyperparameter governing how quickly the likelihood drops off. For unstable cuts the likelihood is flipped (i.e., $L(\neg\alpha_S|\,\boldsymbol{\theta}) = 1 - L(\alpha_S|\,\boldsymbol{\theta})$). In this approach, the likelihood at the limiting axial depth b_{lim} is equal to 0.5, indicating that cuts exactly at the stability boundary have a 50/50 probability of being stable or unstable.

$$L_S(\alpha_S, \alpha_\omega|\,\boldsymbol{\theta}_i) = \frac{1}{1 + \exp\left(\frac{b_{lim}(\boldsymbol{\theta}) - \alpha_b}{k}\right)} \tag{4.49}$$

An alternative stability likelihood formula is based on the uncertainty in the stability limit attributable to factors not considered in the prior $\boldsymbol{\theta}$ distribution, e.g. due to approximations in the stability algorithm itself. For a given set of system parameters $\boldsymbol{\theta}_i$, let the true stability limit $b^*_{lim}(\boldsymbol{\theta}_i)$ be defined as a normal distribution centered around the predicted stability limit and with uncertainty σ_b, as shown in Equation (4.50). Stability testing does not directly reveal the true stability limit b^*_{lim}, instead revealing whether the test was performed under or above the stability limit. This directly corresponds to the cumulative density function (CDF) of the stability limit, so the likelihood of seeing a stable or unstable cut is given by equations (4.51) and (4.52) respectively. In practice, this likelihood behaves similarly to the logistic sigmoid function used by Akbari, but provides a more intuitive meaning for the hyperparameter. Figure 4.23 compares these three different stability approaches. This approach could also be extended, for example by expressing the uncertainty as a percentage of the stability limit rather than a fixed value, or by including uncertainties in both axial depth and spindle speed [66].

$$b^*_{lim}(\boldsymbol{\theta}_i) = \mathcal{N}\left(\mu = b_{lim}(\boldsymbol{\theta}_i), \sigma = \sigma_b\right) \tag{4.50}$$

$$L_S(S, \alpha_\omega|\,\boldsymbol{\theta}_i) = 1 - CDF(\omega_b, \mu = b_{lim}(\boldsymbol{\theta}_i), \sigma = \sigma_b) \tag{4.51}$$

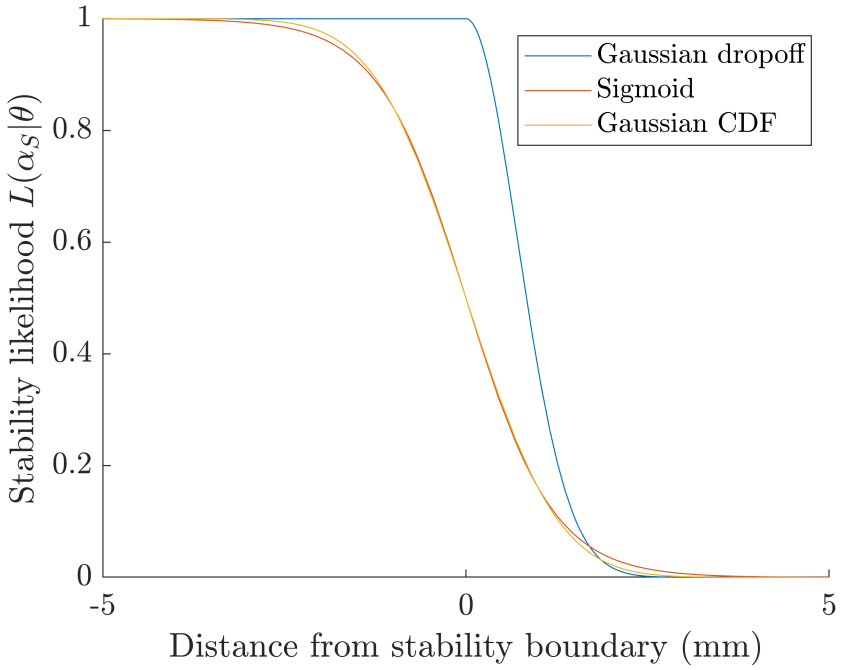


Figure 4.23: Comparison between different stability likelihood functions for a stable cut (Gaussian dropoff, sigmoid, and CDF). The Gaussian dropoff and CDFs both use $\sigma_S = 1$ mm, while the sigmoid uses $k = 0.6$ mm so it closely matches the Gaussian CDF.

$$L_S(\neg S, \alpha_\omega | \, \boldsymbol{\theta}_i) = CDF(\omega_b, \mu = b_{lim}(\boldsymbol{\theta}_i), \sigma = \sigma_b) = 1 - L_{stable}(\omega | \, \boldsymbol{\theta}_i) \tag{4.52}$$

While those methods are based solely on the calculated stability limit, other likelihood functions can be established if there is additional information available. Chen *et al.* used a sigmoid likelihood function based on the spectral radius calculated by the discrete stability algorithm they used (essentially the gain for the system) [37]. The probability of chatter is given by Equation (4.53), where λ is the spectral radius and a is a hyperparameter affecting the dropoff rate.

$$L_S(\alpha_S, \alpha_\omega | \, \boldsymbol{\theta}_i) = \frac{1}{1 + \exp\left(-a(\lambda - 1)\right)} \tag{4.53}$$

Ahmadi *et al.* used a similar approach for their Bayesian turning stability model, establishing the likelihood with a Gaussian stability model [38]. However, rather than just treating the cut as stable or unstable they recorded the vibration of the cut and extracted the magnitude and phase of the spectral radius for the cut using operational modal analysis. These were then compared to the predictions from their discrete stability algorithm, allowing for the system dynamics to be learned efficiently even from stable cuts. However, operational modal analysis on milling setups requires extensive instrumentation to measure the vibration of the tool, so this approach is difficult to apply to milling [13].

4.4.2 Chatter frequency likelihood

The chatter frequency is related to the natural frequency [3, 15]. Directly observing the chatter frequency allows the Bayesian algorithm to quickly locate the peaks in the stability map. This can be easily measured by recording the cut using a microphone at a high sampling rate (e.g. 10 kHz). However, there is some uncertainty associated with the chatter frequency prediction $f_c(\boldsymbol{\theta}, \alpha_\omega)$. This uncertainty can be expressed using the normal distribution shown in Equation (4.54), where $f_c^*(\boldsymbol{\theta}_i, n)$ is the true chatter frequency at a spindle speed n. It is generally convenient to have the likelihood lie in the range from zero to one, so the non-normalized form of the Gaussian PDF is used to construct the likelihood function shown in

Equation (4.55), with the log form in Equation (4.56). Note that if chatter is not observed, then there is not chatter frequency, so $L(\alpha_{f_c}|\,\boldsymbol{\theta}) = 1$. This approach has been used by Schmitz and Akbari [39, 49].

$$f_c^*(\boldsymbol{\theta}_i, n) = \mathcal{N}(\mu = f_c(\boldsymbol{\theta}_i, n), \sigma = \sigma_{f_c}) \tag{4.54}$$

$$L_{f_c}(\alpha_{f_c}, \alpha_\omega|\,\boldsymbol{\theta}) = \begin{cases} \exp\left(-1/2\left(\frac{f_c(\boldsymbol{\theta},\alpha_\omega)-\alpha_{f_c}}{\sigma_{f_c}}\right)^2\right) & \neg\alpha_S \\ 1 & \alpha_S \end{cases} \tag{4.55}$$

$$\tilde{L}_{f_c}(\alpha_{f_c}, \alpha_\omega|\,\boldsymbol{\theta}) = \begin{cases} -1/2\left(\frac{f_c(\boldsymbol{\theta},\alpha_\omega)-\alpha_{f_c}}{\sigma_{f_c}}\right)^2 & \neg\alpha_S \\ 0 & \alpha_S \end{cases} \tag{4.56}$$

4.4.3 Spindle power likelihood

One challenge with existing Bayesian models is that the stability map is often not one-to-one: i.e., there are often many sets of inputs $\boldsymbol{\theta}$ that provide similar stability maps. This results in the posterior failing to uniquely identify the stability inputs $\boldsymbol{\theta}$, even if the posterior stability map is accurate. For example, the cutting force coefficient K_s is generally linearly coupled with the modal stiffness k: doubling both the stiffness and cutting force will result in the same stability behavior (assuming the natural frequency f_n and damping ratio ζ are held constant), so the exact values can't be distinguished solely via binary stable/unstable test cuts. This can be resolved by observing additional information about each test cut, specifically the spindle cutting power. Since the torque is directly related to the cutting force, adding this information would can allow the Bayesian model to uniquely identify K_s and k.

The spindle drive current signal, shown in Figure 4.24, can be used to calculate the spindle cutting power during machining. This signal can be provided by the machine tool controller itself or an external inductive current sensor like a Rogowski coil. However, this information can be difficult to analyze since it is filtered by both the mechanical system (spindle/tool dynamics) and spindle drive motor parameters [67, 68]. This can

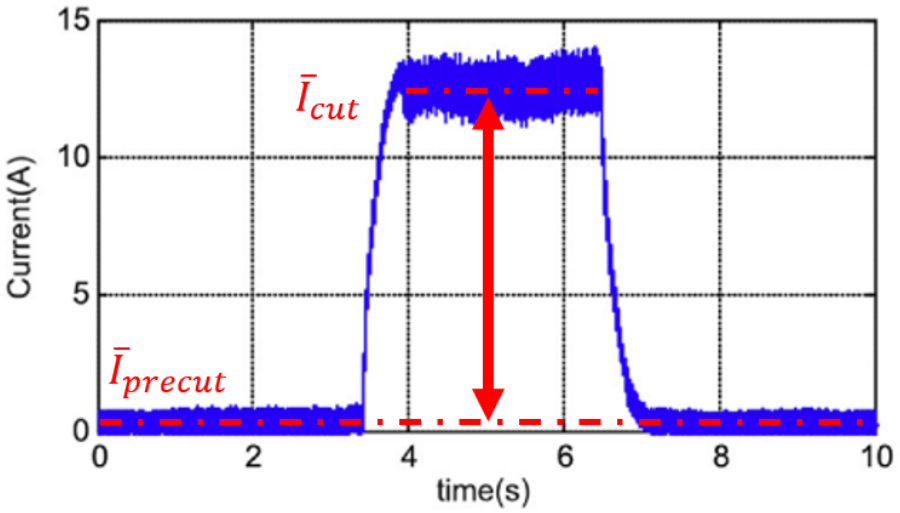


Figure 4.24: Spindle current drive example, Figure 10(a) from[67]

be compensated for using the cutting torque/drive current transfer function, allowing both stable/unstable classification and chatter frequency identification. In practice this is complex to implement, and the best methods are limited to a bandwidth of 2500 Hz [67]. Since high-speed milling cuts can have flute passing frequencies over 1500 Hz, this bandwidth may not be sufficient to replace microphone measurements.

Instead, the spindle current is averaged over the cut duration to estimate the mean spindle power $\alpha_{\bar{P}}$. Equation (4.57) is used to calculate the cutting power where V is the spindle motor voltage and $\bar{I}_{cut}$ and $\bar{I}_{precut}$ are the average current during and before the machining cut. The average cutting torque can be calculated with Equation (4.58).

$$\alpha_{\bar{P}} = \sqrt{3}V(\bar{I}_{cut} - \bar{I}_{precut}) \tag{4.57}$$

$$\alpha_{\bar{T}} = \frac{60\bar{P}}{2\pi n} \tag{4.58}$$

The average cutting torque for a given test can be estimated from the cutting force coefficients. There is, however, some uncertainty $\sigma_{\bar{P}}$ associated with both the measured torque value and the torque prediction. Based on this, the likelihood function can be established using a normal distribution, similar to the chatter frequency likelihood. This is shown in Equation (4.59).

$$L_{\bar{P}}(\alpha_{\bar{P}}, \alpha_{\omega} | \, \boldsymbol{\theta}) = \exp\left(-\frac{1}{2}\left(\frac{\alpha_{\bar{P}} - \bar{P}(\boldsymbol{\theta}, \alpha_{\omega})}{\sigma_{\bar{P}}}\right)^2\right) \tag{4.59}$$

4.4.4 Bending stress likelihood

The precise bending stress value is not directly measured during cutting. However, some information can still be obtained from the binary broken/not broken information observed from a given cutting test. Since material properties have uncertainty associated with them, let the ultimate tensile strength of the cutting tool be defined by a normal distribution $UTS = \mathcal{N}(\mu, \sigma)$. For this work, this uncertainty will not be incorporated as a variable inside of the Bayesian updating. The probability that the tool will break is thus given by Equation

(4.60), where $CDF(\mathcal{N}, x)$ is the cumulative density function of the distribution $\mathcal{N}$ evaluated at x. Using this, the bending stress likelihood is given by Equation (4.61).

$$P_{break}(\boldsymbol{\theta}, \omega) = UTS < \sigma_{max}(\boldsymbol{\theta}, \omega) = CDF(\mathcal{N}(\mu, \sigma), \sigma_{max}(\boldsymbol{\theta}, \omega)) \tag{4.60}$$

$$L_B(\alpha_B, \alpha_\omega | \, \boldsymbol{\theta}) = \begin{cases} P_{break}(\boldsymbol{\theta}, \omega) & \alpha_{break} \\ 1 - P_{break}(\boldsymbol{\theta}, \omega) & \neg\alpha_{break} \end{cases} \tag{4.61}$$

4.5 Sampling the posterior distribution

Once the likelihood function and posterior distribution are defined, new samples must be drawn from the posterior in order to calculate the updated probabilistic stability map and other statistics necessary to select a new test point. However, since the posterior distribution is non-parametric there is no closed-form solution to directly draw samples from the distribution and simple Monte Carlo sampling is not sufficient. Furthermore, drawing new samples is computationally complex since evaluating the sample's probability requires calculating its stability map. Therefore, a computationally efficient method for quickly approximating the posterior distribution is required.

Several different methods have been proposed to overcome this, with their own relative merits. The following subsections will describe four of the main approaches that have been used for updating and sampling the posterior distributions.

4.5.1 Rejection sampling

Karandikar *et al.* applied rejection in their non-physics-guided random-walk approach [65]. For each sample in the prior, a random sample is drawn from the uniform distribution $u = \mathcal{U}(0, 1)$ and compared to the likelihood l for the most recent test. If $a > l$, then this sample is discarded. After this discarding is performed for all prior samples, the remaining samples will approximate the posterior distribution.

This rejection sampling updating method is very fast to perform since it simply involves comparing each sample's prediction to the actual test results. However, the samples rapidly

deplete as additional samples are added and will eventually do a poor job of approximating the posterior distribution. This also cannot account for situations where the posterior distribution changes significantly from the prior.

The same approach can be applied to draw new samples from the posterior by performing Monte Carlo sampling on the prior distribution and then accepting or rejecting the samples based on their likelihoods. However, this is inefficient and can have very low acceptance rates.

4.5.2 Updating sample weights

Rather than rejecting samples outright, they can instead be weighted based on their likelihoods [40]. After the original Monte Carlo sampling, each sample $\boldsymbol{\theta}_i$ will have a weight of $w_i = 1$ since probability density is indicated by the density of the samples. After observing a test result α, the weight of each sample will be updated to $w_i = L(\alpha|\,\boldsymbol{\theta}_i)$, and updated statistics can be calculated using the weighted mean $\bar{\boldsymbol{\theta}}_w$ and standard deviation $\hat{\sigma}_w$ for the samples using Equations (4.62) and (4.63) respectively, where n is the total number of weighted samples.

$$\bar{\boldsymbol{\theta}}_w = \frac{\sum_{i=1}^{n} w_i\,\boldsymbol{\theta}_i}{\sum_{i=1}^{n} w_i} \tag{4.62}$$

$$\hat{\sigma}_w = \left(\frac{\sum_{i=1}^{n} w_i \left(\boldsymbol{\theta}_i - \bar{\boldsymbol{\theta}}_w\right)^2}{\frac{n-1}{n} \sum_{i=1}^{n} w_i} \right)^{1/2} \tag{4.63}$$

In practice, this has similar pros and cons to rejection sampling. The updating can be done very quickly since it simply compares precalculated stability maps to the test results and does not need to calculate any new maps. Compared with rejection sampling, this approach can provide somewhat higher accuracy since the weighting will provide a better approximation of the likelihood than the binary accept/reject. These can make it a good choice for computationally-expensive physics algorithms since it allows all the stability maps to be precalculated ahead of time [40]. However, it runs into the same approximation issues when the posterior distribution is significantly different than the prior: eventually

only a small number of samples will have significant weights and will thus provide a poor approximation of the posterior distribution.

4.5.3 Laplace's approximation

Chen *et al.* applied Laplace's approximation to sample the prior distribution [37]. This method fits a multivariate normal distribution $q(\boldsymbol{\theta})$ to the data such that $q(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}\,|\alpha)$, based on analysis at a single data point. This approach is computationally efficient since it can be performed with a small number of samples. There are two primary steps in this approach. First, identify the maximum likelihood estimator (MLE) $\hat{\boldsymbol{\theta}}$, i.e. the point on the posterior with the highest probability density. This can be done via numerical methods like Newton's method, or by evaluating the posterior probability for the samples drawn from the prior distribution. This point will act as the mean vector for the posterior.

Second, calculate A, the negative second derivative matrix of the log probability at $\hat{\boldsymbol{\theta}}$, shown in Equation (4.64). If d is the length of the $\boldsymbol{\theta}$ vector, then A should be a square matrix with d-by-d size. This can be done numerically by evaluating several points immediately around $\boldsymbol{\theta}$. A acts as the covariance matrix for q, giving the probability density function shown in Equation (4.65).

$$A = -\nabla\nabla \log\left(P(\boldsymbol{\theta}\,|\alpha)|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}\right) \tag{4.64}$$

$$Q = \mathcal{N}(\mu = \hat{\boldsymbol{\theta}}, \Sigma = A) \approx p(\boldsymbol{\theta}\,|\alpha) \tag{4.65}$$

While this approach is computationally efficient, since it assumes that the posterior has the form of a multivariate normal distribution it is not capable of accurately sampling more complex posteriors. For example, under some circumstances the posterior may be multimodal with more than one discrete local maximum (see Section 5.4). The Gaussian approximation method will select one mode to fit the MVN distribution and entirely ignore any other modes. Depending on the approach used to identify the MLE, selection of fit mode may not even be consistent. This can potentially be overcome by fitting multiple modes and mixing them, at the cost of increased computational time to identify and fit all the modes. Additionally,

it requires that the posterior likelihood be smoothly differentiable: likelihood functions with sharp changes in the second derivative are not suitable for Laplace approximation.

4.5.4 Markov Chain Monte Carlo (MCMC) sampling

MCMC samplers have been the most common approach for sampling the posterior distribution for physics-guided Bayesian stability models. As such, they are described in more detail, aiming to provide a complete (albeit informal) introduction to MCMC and links to resources for more information. First, there is a brief introduction to the basic Metropolis MCMC algorithm, along with a simple toy example to demonstrate the concept. Limitations of this simple algorithm are discussed, as well as a brief overview of the many available MCMC samplers. A parallel-adaptive MCMC sampler designed specifically for rapid iterative Bayesian updating is presented. Finally, several suggestions for implementing and optimizing these sampling algorithms are presented.

Introduction to MCMC

Markov Chain Monte Carlo (MCMC) sampling is a common set of methods for efficiently sampling complex high-dimensional non-parametric probability distributions and create a chain of samples that approximate the posterior [69]. Rather than generating independent samples, MCMC methods create a chain of correlated samples by randomly walking around the target probability distribution. As more and more steps are taken, the distribution of the samples converges to the overall probability distribution. The size of each proposed step in the random walk procedure is determined by a preset proposal distribution and steps are accepted or rejected based on the ratio of the probability of the old and new positions on the chain. The samples are naturally clustered in high-probability regions of the target probability distribution. MCMC methods are thus less susceptible to the so-called "curse of dimensionality" and sample higher-dimension distributions with higher efficiency and acceptance rates than rejection sampling. Additionally, MCMC methods do not require the target distribution to be normalized (i.e., it is acceptable to have $\int_x P(x) \neq 1$). Since numerical normalization becomes computationally expensive as more samples are taken,

this is a significant advantage. Furthermore, unlike Laplace approximation MCMC does not assume any given form for the target distribution and can therefore sample complex non-parameteric distributions, for example multimodal distributions.

The most common MCMC sampler is the Metropolis algorithm. This method proposes steps based on a present proposal distribution and accepts or rejects them based on the ratio of the probability at the current and candidate points. The following steps describe how to run the sampler for an arbitrary non-normalized target distribution $f(x)$.

1. Initialize the algorithm:
   - Initialize an output chain where x_n is the nth element.
   - Select an arbitrary starting point and set it as the first element of the chain x_1.
   - Select an arbitrary symmetric proposal distribution $g(x_{cand}|x)$ which describes the probability of transitioning from a given point x_n to a new position x_{cand}. This is commonly a Gaussian distribution centered at x.
   - Set the step counter $i = 1$ and max step count s.
2. Run the sampler. For each iteration i:
   (a) Generate a candidate point x_{cand} from the proposal distribution $g(x_{cand}|x_i)$.
   (b) Calculate the target probability density at the candidate point $f(x_{cand})$ and the last point on the chain $f(x_i)$.
   (c) Calculate the acceptance probability $a = \min\left(f(x_{cand})/f(x_i), 1\right)$ Note that it is generally better to perform this calculation in logarithmic space to avoid floating-point errors, i.e. $a = \min\left(\exp\left(\log(f(x_{cand}))/\log(f(x_i))\right)\right)$.
   (d) Draw a sample u from the uniform distribution $\mathcal{U}(0, 1)$.
   (e) Compare u and a:
      - If $u \leq a$, accept x_{cand} and append it to the output chain as $x_{i+1} = x_{cand}$.
      - If $u > a$, reject x_{cand} and duplicate the last element of the chain, $x_{i+1} = x_i$.
   (f) Increment the step counter $i = i + 1$.

As a concrete demonstration, let the target distribution $f(x) = \mathcal{N}(\mu = 3, \sigma = 2)$, the arbitrary starting position be $x_1 = 0$, and the proposal distribution be a unit Gaussian centered around the last element in the chain $g(x_{cand}|x) = \mathcal{N}(x_{cand}, \mu = x, \sigma = 1)$. The code for this demo is available in Appendix B. The starting point in the chain is shown in Figure 4.25a, with $x_1 = 0$ and a proposal distribution centered around that point. Next, a candidate point $x_{cand} = 0.538$ is drawn from the proposal distribution and its probability is compared to the probability at x_1, as shown in Figure 4.25b. Since $f(x_{cand} = 0.538)/f(x_1 = 0) = 0.0935/0.0648 \geq 1 \Rightarrow a = 1$, the point is accepted with 100% probability and appended to the chain as $x_2 = x_{cand} = 0.538$.

The next proposed point is $x_{cand} = -1.72$, shown in Figure 4.25c. The probability of this candidate is much lower than the current probability, so $a = f(x_{cand} = -1.72)/f(x_2 = 0.538) = 0.0123/0.0935 = 0.132$. A value $u = 0.913$ is drawn from $\mathcal{U}(0, 1)$. Since $u > a$, this move is rejected. Instead, another copy of x_2 is appended to the chain as $x_3 = x_2 = 0.538$, as shown in Figure 4.25d.

After only three samples, the marginal distribution of the chain is not a good approximation of $f(x)$, but as the sampler continues to run, the marginal distribution of the chain approaches $f(x)$. This is illustrated in Figure 4.26, which shows how the chain develops at 25, 100, 1000, and 10000 samples. These figures also show the acceptance rate, which is the percentage of candidate points which were accepted. This specific example gives an acceptance rate of about 85%.

Metropolis drawbacks

While Metropolis sampling can be more computationally efficient than algorithms like rejection sampling, there are several practical problems that make it infeasible for sampling complex distributions. These challenges must be overcome when designing an efficient MCMC sampler. First, if the starting point is set in a region with low probability it can bias the output chain. For example, see Figure 4.27. This shows a unit normal distribution where the chain is initialized at $x_1 = -10$ and run for 250 samples. Since the starting chain is set in a low probability region, the resulting sampled distribution is biased and extremely low values below -5 are over-represented. In this specific case it would be simple

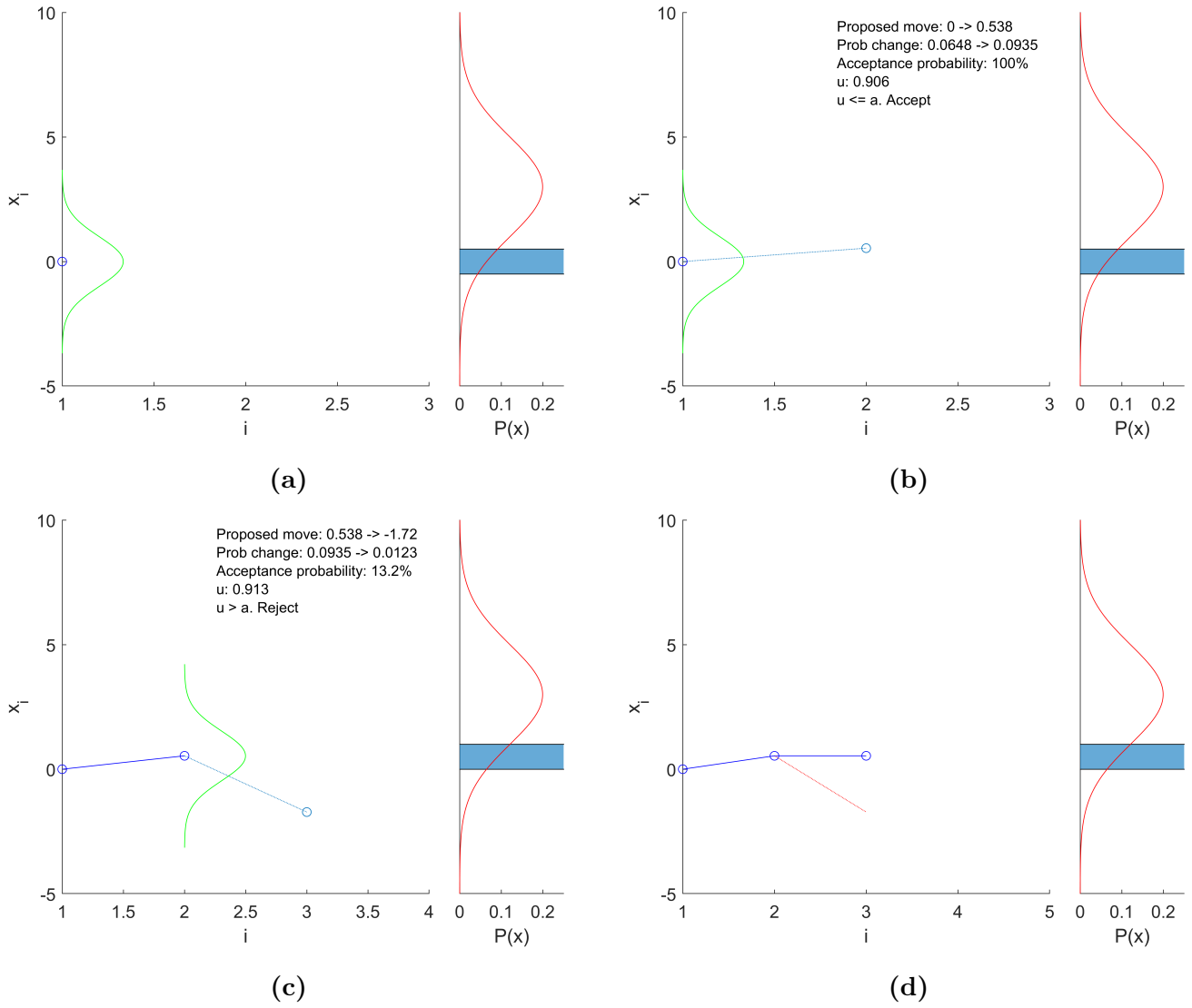


Figure 4.25: Demonstration of the Metropolis MCMC algorithm. (a) Initial starting point and proposal distribution. (b) Evaluating the first candidate point. (c) Evaluating the second candidate point. (d) The chain after rejecting the last move.

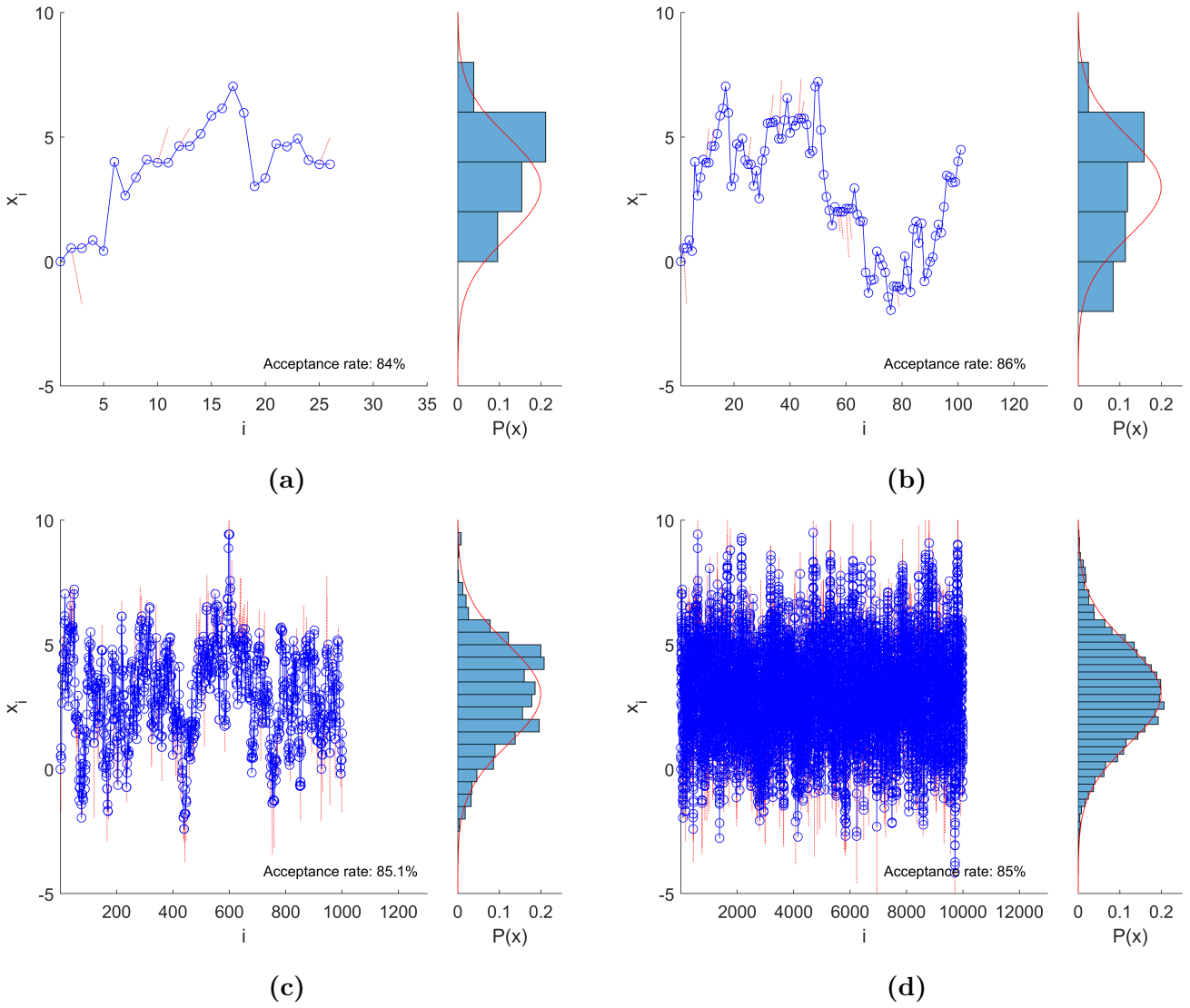


Figure 4.26: Demonstration of the Metropolis MCMC algorithm showing how it approaches the target marginal distribution over time after 25 (a), 100 (b), 1000 (c), and 10000 (d) samples.

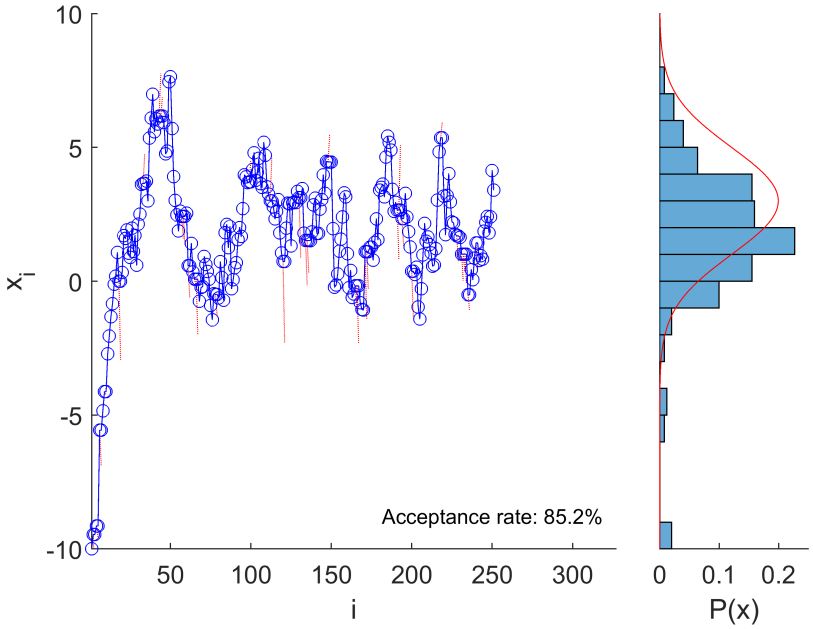


Figure 4.27: Demonstration of how a poor starting point can bias the output chain

to start the chain at the mean of the target distribution, but since posterior distributions after Bayesian updating can be complex and have sharp dropoffs in probability it can be difficult to select an appropriate starting point. Many authors address this using "burn-in", where the beginning of the chain is discarded in the hope that it will have shifted to higher probability regions. While this can remove the low-probability points that are biasing the chain, it can be difficult to determine when to stop the burn-in and there is no theoretical guarantee that it will succeed at all: see Chapter 1 in [70] for a more complete discussion.

Second, the choice of proposal distribution affects how long it takes for the MCMC sampler to converge to the target distribution due to a tradeoff between acceptance rate and autocorrelation. Proposal distributions which are too wide will have low autocorrelation, indicating that each new point is significantly different from the last and provides significant new information, but will also have low acceptance rates since the steps will likely move into low-probability regions. Narrow distributions will have high acceptance rates but high autocorrelation since each candidate is close to the previous point. Figure 4.28 demonstrates this, comparing the exploration of a unit normal distribution in 500 samples. Using a proposal distribution with $\sigma = 0.25$ gives a high acceptance rate of 92.6%, but does not accurately approximate the target distribution. Using $\sigma = 1$ gives an acceptance rate of 71.4% and does a decent job of converging. $\sigma = 4$ has an acceptance of only 30.4% and has worse convergence. For a high-dimensional target distribution which is similar to a multivariate Gaussian it is ideal to target an acceptance rate of roughly 25-30% [71], but for the sort of complex distributions common in Bayesian posteriors this can be difficult to select.

Finally, the Metropolis algorithm is a serial algorithm since each new proposal depends on the previous step in the chain. This means that Metropolis sampling can't take advantage of modern parallel-computing resources. While there are various strategies to allow for parallelization, any MCMC strategy will require some degree of synchronization and thus cannot be perfectly parallelizable (i.e., doubling the number of threads will not halve the computation time).

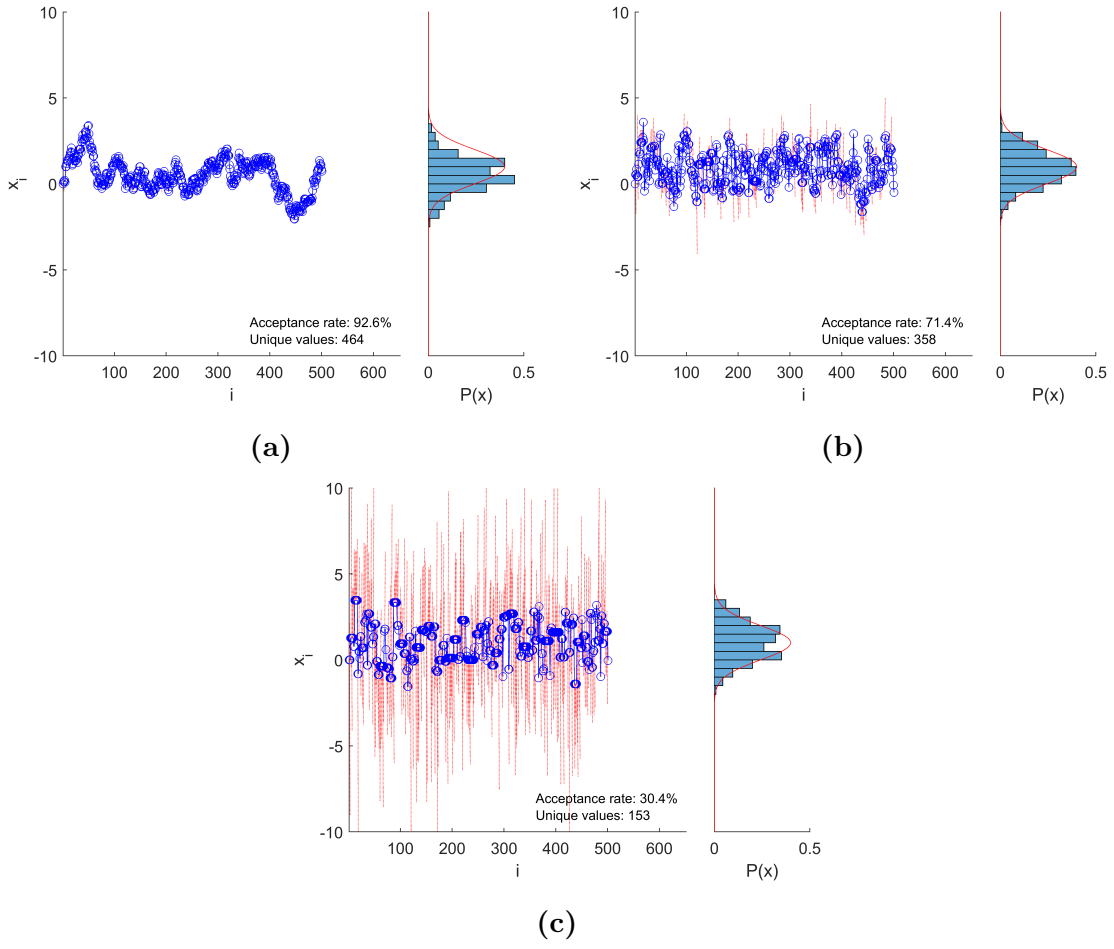


Figure 4.28: Comparison of MCMC proposal distributions for exploring a unit normal distribution. (a) $\sigma = 0.25$, (b) $\sigma = 1$, (c) $\sigma = 4$

Many different MCMC samplers have been proposed which help overcome these challenges. Following is a brief, non-exhaustive reference list of several common MCMC strategies and how they improve on the Metropolis algorithm:

- The Metropolis-Hastings algorithm extends the Metropolis algorithm to allow non-symmetric proposal distributions [69]
- Multiple-Try Metropolis allows for limited parallelization by proposing multiple candidate points for each iteration and accepting one of them [72]
- Hamiltonian Monte Carlo uses the gradient of the target probability distribution to help select points with higher acceptance rates [73]
- Adaptive MCMC algorithms alter the proposal distribution during sampling in order to converge more quickly [74]
- Delayed-rejection MCMC reduces the rejection rate by sequentially testing multiple candidates at each step of the chain [75]
- Mode jumping MCMC uses two types of proposal distributions to efficiently sample non-unimodal distributions [76]
- Ensemble MCMC use multiple walkers which move around the probability space relative to each other to efficiently sample skewed distributions [77]
- Transdimensional MCMC can automatically vary the dimension of the random variable, extending it as necessary to better explain the observations [78]

The different Bayesian machining papers have utilized several different MCMC samplers. Li *et al.* used a parallel ensemble algorithm [36]. Ahmadi *et al.* applied a transitional MCMC method which applies weights to the samples in the chain to improve convergence [38]. Akbari *et al.* used an evolutionary ensemble sampler which tests out and evolves different strategies for combining the multiple walkers to converge quickly and fit the posterior samples with a Gaussian distribution [39]. Schmitz *et al.* applied the parallel adaptive algorithm described in the next section [49]. To date, there is no work which compares these different

approaches. There are also a variety of high-performance MCMC samplers available for different software suites, such as ParaDRAM (parallel delayed-rejection adaptive MCMC) [79] and Stan (Hamiltonian Monte Carlo with automatic differentiation) [80].

Parallel adaptive MCMC

This section describes an MCMC algorithm which is designed to rapidly approximate the posterior distribution to allow for sequential updating and testing. Rejection sampling on the prior samples is used to select an appropriate starting point and proposal distribution. At each step of the MCMC sampler, multiple points are proposed and accepted in parallel, and the proposal distribution is continually fine-tuned based on the current state of the chain to improve acceptance. This method provides good performance and convergence, requires no tuning for specific target distributions, and is simple to implement. Code for the sampler is included along with this dissertation on TRACE.

First, the samples from the prior distribution are filtered using rejection sampling to approximate the updated posterior distribution, shown in the below steps. Note that this process normalizes the distribution to help ensure that sufficient samples are retained. This is generally acceptable when the posterior distribution is not changing dramatically compared with the prior. However, if the posterior distribution is significantly different from the prior then the remaining samples will not be a good approximation of the posterior. This is generally a reasonable assumption when using the Bayesian framework to automatically select cutting tests, since tests are being selected which have a reasonable probability of stability.

1. For each prior sample $\boldsymbol{\theta}_i$, calculate the log likelihood for the last test result $\tilde{L}(\omega|\,\boldsymbol{\theta}_i)$.
2. Find the sample with the highest likelihood $\boldsymbol{\theta}_{L\,max}$ and calculate the normalized likelihoods for each sample $\tilde{L}^*(\boldsymbol{\theta}_i) = \tilde{L}(\boldsymbol{\theta}_i) - \tilde{L}(\boldsymbol{\theta}_{L\,max})$.
3. Create a new empty list of samples $\boldsymbol{\theta}^*$.
4. For each sample $\boldsymbol{\theta}_i$, draw a sample a from the uniform distribution $\mathcal{U}(0,1)$. If $a < L^*(\omega|\,\boldsymbol{\theta}_i)$, then append $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}^*$.

The starting location for the output chain is set using one of the retained samples in $\boldsymbol{\theta}^*$, denoted by $\boldsymbol{\theta}_{last}$. Since the retained samples are known to be in high probability locations on the posterior, any may serve as a good starting position.

Next, the proposal distribution is automatically set and recalculated after every step using an adaptive MCMC method [74]. This method uses a normal distribution centered around the last element of the chain as defined in Equation (4.66), where $\boldsymbol{\theta}_{last}$ is the last point on the current chain, Σ is the covariance matrix for all accepted samples, and d is the number of elements in the $\boldsymbol{\theta}$ vector. Σ is calculated using both the current chain of samples and $\boldsymbol{\theta}^*$ to improve accuracy with the final posterior covariance. The factor $5.76/d$ has been shown to give optimal mixing for targets which are close to multivariate normal distributions [71]. Since this proposal distribution considers the correlation between different variables, it also naturally accounts for skewed probability distributions.

$$P_{prop} = \mathcal{N}(\boldsymbol{\theta}_{last}, 5.76/d \cdot \Sigma) \tag{4.66}$$

Third, the MCMC is parallelized to propose and accept multiple points at each step using the framework proposed by Calderhead [81]. For each step, j candidate samples, denoted by $\boldsymbol{\theta}^{cand}$, are drawn from P_{prop}. To reduce floating point error their probabilities are normalized using Equation (4.67). The candidates' relative probabilities are then calculated using Equation (4.68). Note that $\boldsymbol{\theta}_0^{cand} = \boldsymbol{\theta}_{last}$: this represents the probability of staying at the last position on the chain rather than transitioning to any of the candidates. This approach works well since it allows the candidate probabilities to be evaluated in parallel: running the physics algorithm is generally the most time-consuming step for Bayesian machining modelling, so parallelizing this step can significantly improve performance.

$$p(\boldsymbol{\theta}_i^{cand}) = \exp\left(\log\left(P(\boldsymbol{\theta}_i^{cand}\,|\alpha_{1:n})\right) - \max_i \log\left(P(\boldsymbol{\theta}_i^{cand}\,|\alpha_{1:n})\right)\right) \tag{4.67}$$

$$I(\boldsymbol{\theta}_i^{cand}) = \frac{p(\boldsymbol{\theta}_i^{cand})}{\sum_{l=0}^{j} p(\boldsymbol{\theta}_l^{cand})} \tag{4.68}$$

Finally, k samples are drawn from the stationary distribution I formed by the candidates: note that any given candidate can drawn multiple times. For example, if $j = 3$ then

three candidate samples will be drawn from the proposal distribution. If they have non-normalized probabilities $P(\boldsymbol{\theta}_{0:3}^{cand}) = [1, 2, 1, 1]$, then $p(\boldsymbol{\theta}_{0:3}^{cand}) = [0.5, 1, 0.5, 0.5]$ and $I(\boldsymbol{\theta}_{0:3}^{cand}) = [0.2, 0.4, 0.2, 0.2]$ (with the first element in all lists representing $\boldsymbol{\theta}_0^{cand} = \boldsymbol{\theta}_{last}$, i.e. the probability of staying at the current point in the chain). If $k = 4$, then a likely outcome of this could be $[1, 0, 1, 3]$, with the higher-probability $\boldsymbol{\theta}_1^{cand}$ getting represented twice, the previous point on the chain getting added once, and $\boldsymbol{\theta}_2^{cand}$ not getting included at all. The accepted elements are then appended to the output chain, the last element serves as the center point for the new proposal distribution, and sampling proceeds until the desired number of samples are obtained.

The following steps summarize all these steps to draw s samples from the posterior distribution.

1. Perform rejection sampling on the samples from the prior distribution, putting the remaining samples into a new set of samples $\boldsymbol{\theta}^*$.

2. Select one of the points in $\boldsymbol{\theta}^*$ as a starting point for the MCMC sampler.

3. Calculate the covariance matrix $\Sigma = \text{Cov}(\boldsymbol{\theta}_{out} \cup \boldsymbol{\theta}^*)$.

4. Define a proposal distribution $P_{prop} = \mathcal{N}(\mu = \boldsymbol{\theta}_{last}, 5.76/d \cdot \Sigma)$, where $\boldsymbol{\theta}_{last}$ is the last element of the $\boldsymbol{\theta}_{out}$ chain.

5. Draw j candidate samples from P_{prop}, $\boldsymbol{\theta}_{1:j}^{cand}$ and calculate their probabilities (i.e., calculate the physics algorithms and compare their predictions to the test results). For accuracy, normalize these probabilities in logarithmic space so that the maximum value has a probability of 1 using Equation (4.67).

6. For each candidate sample, calculate its relative probability using Equation (4.68). Note that $\boldsymbol{\theta}_0^{cand}$ in this case represents $\boldsymbol{\theta}_{last}$, i.e. the current position of the MCMC chain.

7. Accept s samples from the candidates and append them to $\boldsymbol{\theta}_{out}$, based on their relative probabilities $I(\boldsymbol{\theta}_i^{cand})$. Note that this may mean that more than one copy of a specific candidate gets appended to the output.

8. If $\boldsymbol{\theta}_{out}$ contains at least s unique samples, terminate the sampler. Otherwise, go back to step 3.

This approach is computationally efficient, generally achieving acceptance rates between 30-40%. The calculation time for a single iteration will depend on the required number of samples and how computationally expensive the physics algorithms are. For example, 4000 samples using the Altintas ZOA stability algorithm generally takes less than a minute to calculate on a standard laptop, adding in additional calculations like bending stress estimations could take several minutes, and using time domain simulations could require many hours even with a high-performance computing cluster.

Possible optimizations

While the previous subsections have outlined different available sampling strategies, the performance of the final sampler is heavily dependent on the implementation and there are many opportunities for optimization. While sampling the posterior and calculating the physics algorithms are listed as separate steps in Figure 4.1, in practice these steps may overlap or be performed simultaneously. During MCMC, the sampler must only run the parts of the physics algorithms needed to calculate the posterior likelihood. In some cases, this essentially means running the entire physics algorithm: for example, the ZOA calculates an entire stability map in one pass, and this can then be saved with the sample so it doesn't need to get calculated again later. However, discrete stability algorithms and time domain simulations are run separately for each set of cutting parameters, so calculating the likelihood for a sample might only require simulating a small part of the stability map around the cutting test parameters. If the sample is accepted, then the rest of the stability map and any other physics functions can be calculated. This optimization favors likelihood functions like the spectral radius approach used by Chen *et al.* since they rely only on a single simulation rather than needing multiple calculations to calculate the limiting axial depth [37]. A similar approach could be applied for time-domain simulations using the m metric value. Similarly, it may be possible to reduce calculation time by omitting irrelevant areas of the stability map: if the goal is to maximize MRR then only parameters where $MRR(\omega) > MRR_{best}$ or which are necessary to evaluate the likelihood must be calculated.

Chapter 5

Results and Discussion: Laboratory Testing

This chapter presents case studies that were performed in laboratory settings at the University of Tennessee, Knoxville (UTK) and Oak Ridge National Laboratory (ORNL). Four major case studies are presented:

- Case study 1 uses the chatter-risk test selection algorithm to select highly productive machining parameters for an aluminum test cut, demonstrating how including chatter frequency in the Bayesian updating can help improve convergence and improve productivity. This testing was done with several different hyperparameter settings to explore their impact on the updating. This case study was published in 2022 [49].

- Case study 2 uses Bayesian updating to learn the cutting forces and process damping behavior from a set of cutting tests drawn from previously published work by Tyler *et al.* [51]. This set of tests demonstrates that the Bayesian framework can accurately uncover hidden information about the machining system which is physically meaningful, not just fitting parameters, as well as providing some suggestions on how to scale the Bayesian algorithm to use high-performance computing resources. This case study was accepted for publication in 2024 [40].

- Case study 3 demonstrates priors can be established using simple methods, using Euler-Bernoulli beam theory to establish the prior for an extremely long endmill

and quickly select cutting parameters. Additionally, this test set demonstrates how using hybrid exploration/exploitation test selection algorithms can more efficiently identify the system parameters and improve productivity compared to purely greedy approaches.

- Case study 4 demonstrates how the spindle torque can be incorporated into the Bayesian updating to help learn the cutting force model. Additionally, it shows that the Bayesian method can quickly identify productive cutting parameters even with a very naive prior: in this case, a uniform distribution assuming there are two modes somewhere between 1000 and 2000 Hz.

The code for case studies 3 and 4 is included along with this dissertation on TRACE. See Appendix C for a brief introduction to the codebase.

5.1 Case study 1: Risk tolerance and chatter frequency

This section demonstrates how the Bayesian stability model can be used to help identify highly productive machining parameters in a small number of test cuts and explores the effect of risk tolerance and using the chatter frequency for updating. Testing was performed on a DMG Mori Ultrasonic 65 5-axis machining center using a three flute 12.7 mm endmill clamped in a thermal heatshrink holder. The workpiece was aluminum 7050-T7451, a high-strength alloy commonly used for aerospace applications. Table 5.1 provides a complete description of the testing parameters and Figure 5.1 shows a picture of the machining setup.

The force model prior was established based on author experience with similar work materials, and the FRF uncertainty was set using the RCSA method described in Section 4.1.2. These uncertainties were propagated to the stability map and chatter frequency using the ZOA frequency domain algorithm to obtain the probabilistic stability map. Cutting tests were then selected using the chatter-risk MRR maximization testing strategy (Section 4.3). Testing was continued until the next proposed test cut would improve MRR by less than 10%. All tests used the Gaussian dropoff stability likelihood function shown in Section 4.4.1 with $\sigma_b = 0.125$ mm.

Table 5.1: Testing parameters for the MRR-maximization case study

Machine	DMG Mori Ultrasonic 65
Cutting tool	Robbjack FMHV-304-16
Holder	Haimer A63.140.1/2z.4 thermal heatshrink
Tool diameter	12.7 mm
Flute count	3
Flute length	19.05 mm
Tool stickout length	65 mm
Coolant	External flood
Workpiece material	Aluminum 7050-T7451
Feed per flute	0.100 mm
Radial depth of cut	9.525 mm (75% radial immersion)
$\boldsymbol{\theta}$ form	$[K_s, \beta, f_{n1}, k_1, \zeta_1, f_{n2}, k_2, \zeta_2]$
Prior	FRF: Receptance coupling
	Cutting force: author experience
Physics functions	Stability map and chatter frequency (ZOA algorithm)
Test selection	MRR maximization with chatter risk tolerance
Likelihood function	Stability: Gaussian dropoff, $\sigma_b = 0.125$ mm
	Chatter frequency: Gaussian PDF, $\sigma_{f_c} = 50$ Hz
Sampling strategy	Prior: Monte Carlo
	Posterior: Parallel adaptive MCMC

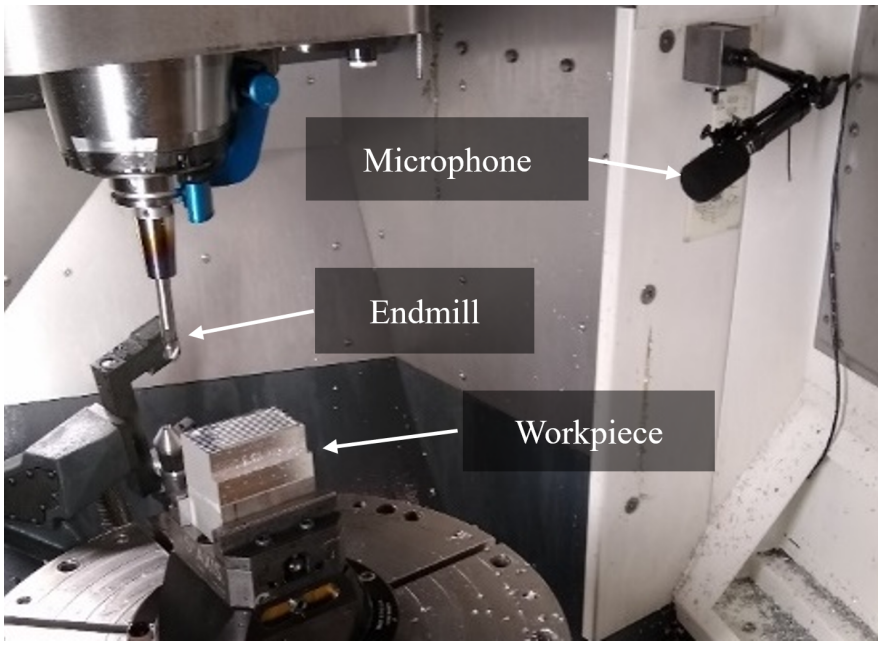


Figure 5.1: Machining setup for the MRR-maximization case study

Three sets of tests were run. The first used a chatter risk preference of 50% (i.e., the user was willing to accept test cuts which had up to a 50% probability of chattering). The Bayesian updating considered only the binary stable/unstable classification. This test series is shown in Figure 5.2. Six tests were run before halting, with three unstable cuts. The best cutting parameters were 11256 rpm at 2.5 mm axial depth, with an MRR of 80.6 cm^3/min.

Next, a set of tests was ran which incorporated the chatter frequency into the Bayesian updating with $\sigma_{f_c} = 50$ Hz, shown in Figure 5.3. All other simulation settings were kept the same. The first three test cuts in this series are identical to the first case, diverging only after the first unstable cut is run. This test series halted after five tests, only one of which was unstable. The best operating parameters were 10985 rpm at 3.4 mm axial depth, with a material removal rate of 107 cm^3/min. Figure 5.4 shows why this second set of tests was more successful. After observing the unstable cutting test, the posterior for the first set was unsure where the true stability lobe was and where to direct the next cutting test: the best spindle speed might be lower, higher, or at the same speed that was already tested. In contrast, observing the chatter frequency gives information about the system natural frequency and thus the spindle speed that stability peaks will occur at. The posterior thus is confident that the true stability peak occurs at a lower spindle speed and accurately directs the next cutting tests into that region.

Finally, a set of cutting tests was run with a more aggressive risk tolerance of 25% (i.e., the user is willing to run cutting tests that have only a 25% probability of being stable), shown in Figure 5.5. Other parameters were kept the same as the second test series. This series also halted after five tests, but the recommended tests were generally more aggressive: chatter occurred on the second cut rather than the third, and a higher MRR of 131 cm^3/min was obtained. Despite the higher risk tolerance only one unstable cut was observed, due to the fact that the true stability limit was significantly higher than the prior.

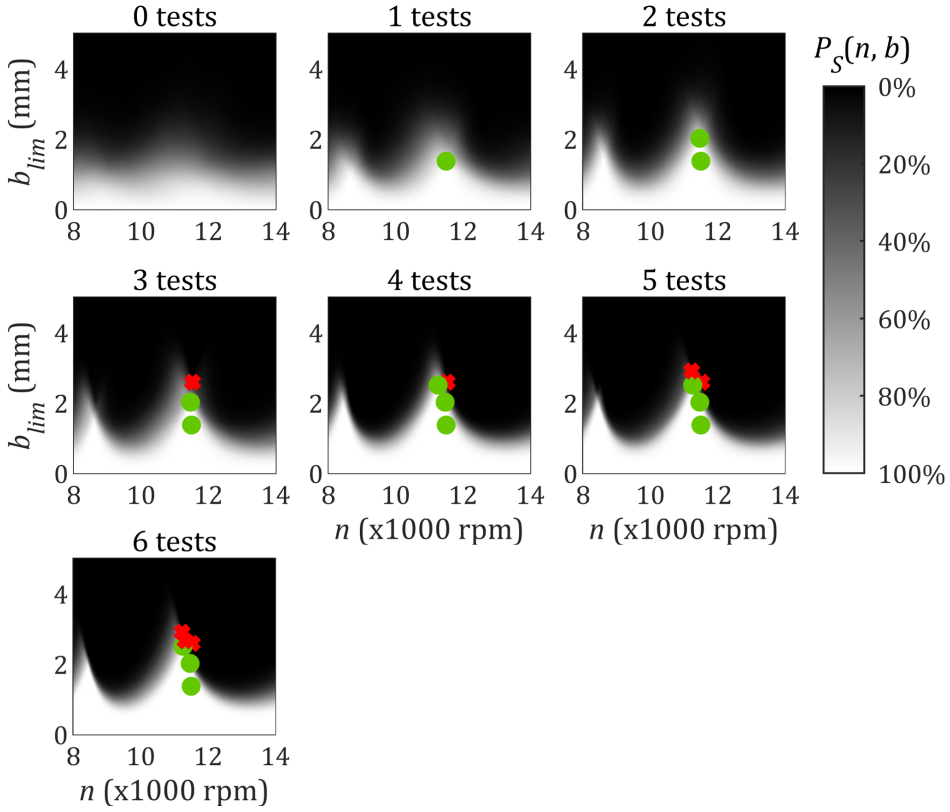


Figure 5.2: Cutting test series with risk tolerance of 50% and no chatter frequency updating

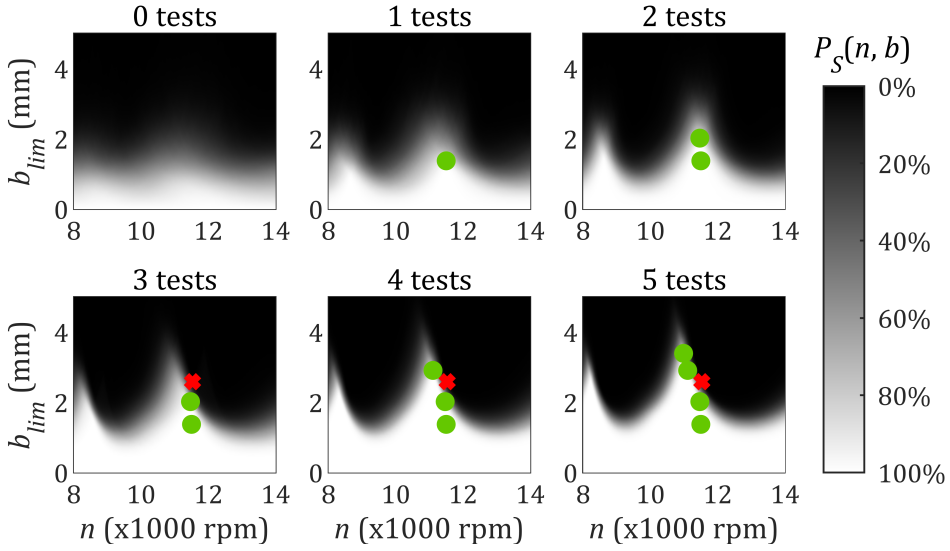


Figure 5.3: Cutting test series with risk tolerance of 50% and chatter frequency updating using $\sigma_{f_c} = 50$ Hz

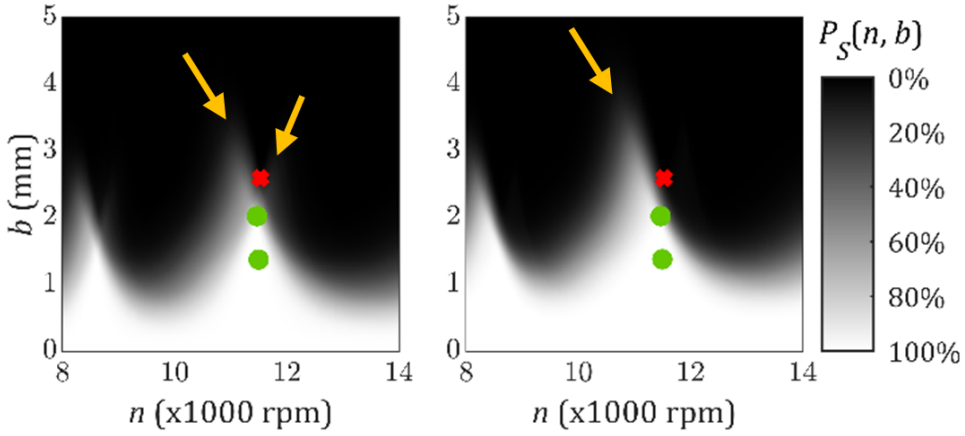


Figure 5.4: Comparison of the posterior stability maps after observing an unstable cut, based on whether the chatter frequency is not (left) or is (right) included in the Bayesian updating

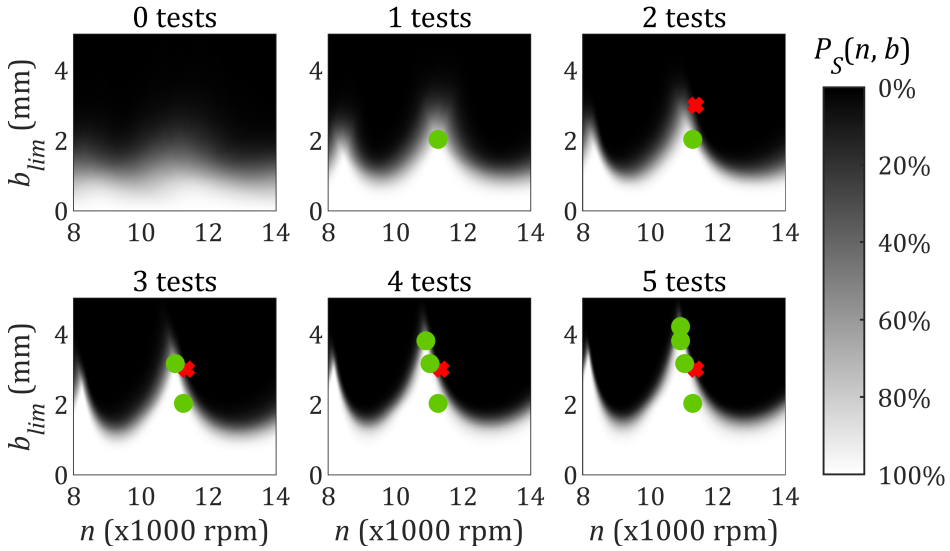


Figure 5.5: Cutting test series with risk tolerance of 75% and chatter frequency updating using $\sigma_{f_c} = 50$ Hz

5.2 Case study 2: Process damping identification

Process damping is a phenomenon in machining where vibrational energy is dissipated due to contact between the tool and workpiece [3]. This effect occurs when the tool flank face indents into the workpiece as it vibrates, resulting in an additional force opposing the vibration (illustrated in Figure 5.6). Process damping can have significant effect on process stability, converting some cuts which would otherwise be unstable into stable cuts. As a result, cutting tool manufacturers are increasingly designing tools with modified edge geometries that exploit this phenomenon to allow for increased depths of cut and MRR [82]. However, process damping is difficult to model or predict since it depends on the exact geometry of the cutting edge like the cutting edge radius, relief angle, and surface finish, as well as on high-speed interactions between the tool and workpiece which are difficult to model. There is therefore significant uncertainty about the true stability boundary in systems where process damping is significant. This section describes how Bayesian inference can be used to learn the process damping and force model from a series of cutting tests, converging towards the same values identified using other methods. Additionally, it also demonstrates how the Bayesian method can be scaled to utilize high-performance computing resources in order to allow for more accurate and computationally expensive machining simulations like time domain simulations, and how the learning process can be improved by using only the most relevant and accurate tests to learn specific variables. These results were accepted for publication in February 2024 [40].

This example uses a set of cutting tests from Tyler *et al.* [51], specifically the set of tests shown in their Figures 12 and 15. Their case study created a flexible workpiece by setting up the part on a nominally single degree-of-freedom flexure. They performed cutting tests at both low and high spindle speeds using a single flute indexable endmill with a 15° clearance angle. High speed cutting tests (above 5000 rpm) were selected using the stability map and agree with classical stability solutions, while the low speed tests were run in a grid pattern and show the exponentially increasing stability limit at low speeds which is characteristic of process damping. Table 5.2 details the test setup and algorithms. The setup and cutting test results are shown in Figure 5.7.

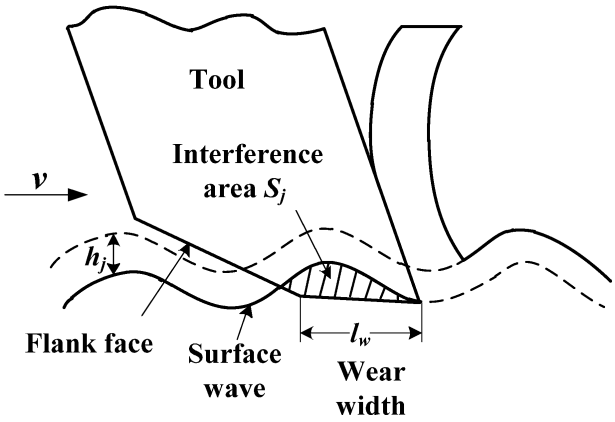
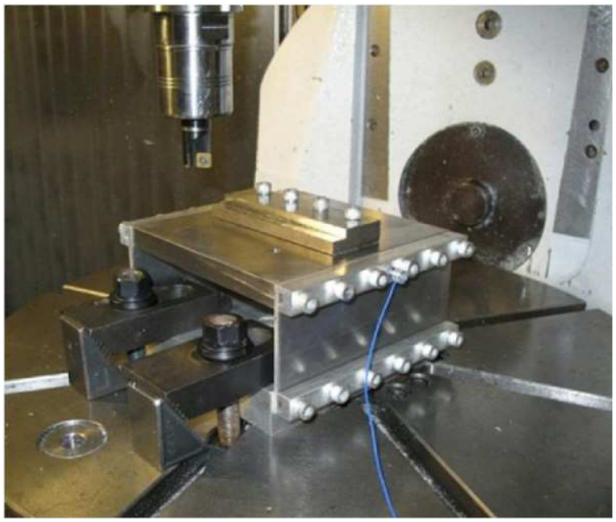


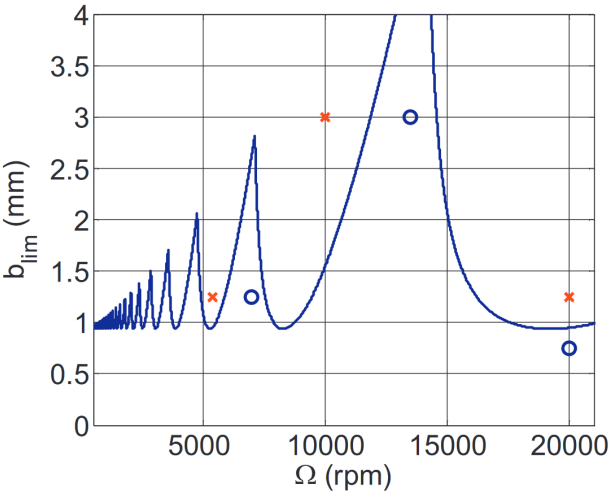
Figure 5.6: Process damping mechanic at the cutting edge, from [82]

Table 5.2: Testing parameters for laboratory case study 2, from [51]

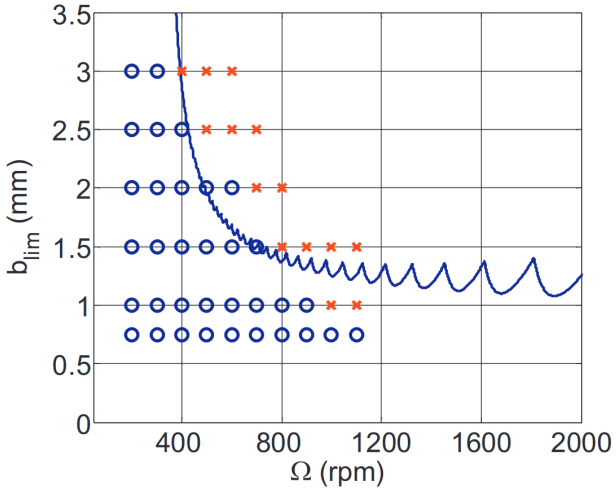
Cutting tool	Cutting Tool Technologies DRM-030
Tool diameter	19.05 mm
Flute count	1
Flute length	8.6 mm
Workpiece material	AISI 1018 steel
Feed per flute	0.05 mm
Radial depth of cut	9.525 mm (50% radial immersion)
$\boldsymbol{\theta}$ form	$[K_s, \beta, c_r]$
Prior	FRF: Fixed
	Cutting force: author experience
Physics functions	Stability map (time domain simulation with $m = 1$ μm cutoff)
Likelihood function	Stability: Gaussian dropoff, $\sigma_b = 0.125$ mm
Sampling strategy	Prior: Grid sampling
	Posterior: Sample weighting



(a)



(b)



(c)

Figure 5.7: Process damping tests from [51]. (a) Test setup with workpiece flexure (Figure 11 in source). (b) High spindle speed test results (Figure 12 in source). (c) Low spindle speed cutting test results showing the effect of process damping (Figure 17 in source).

First, the stability physics algorithm was selected. In order to ensure maximum flexibility, the time domain simulation shown in Section 4.2.1 was used. Process damping was incorporated into this model using the nonlinear wavelength-dependent model introduced by Altintas *et al.* [83]. Let V be the surface speed of the cutting tool's outer diameter, given by Equation (5.1) (where d is the tool diameter in meters) and c_n be a process damping coefficient expressed in N/m. Equation (5.2) gives the additional radial process damping force on a single flute element with height db and tool rotation angle ϕ. $\dot{x}$ and $\dot{y}$ be the relative vibrational velocities between the tool and workpiece in the X and Y directions respectively. When the chip thickness is not positive (i.e., $h < 0$), the flute element is not in contact with the workpiece and the force goes to zero. This is calculated for each flute element and added to the radial force during the time domain simulation.

$$V = \frac{2\pi\Omega d}{60} \tag{5.1}$$

$$F_D = \begin{cases} -c_n\, db \frac{\dot{x}\sin(\phi) - \dot{y}\cos(\phi)}{V} & h \geq 0 \\ 0 & h < 0 \end{cases} \tag{5.2}$$

Next, a prior was established. The workpiece dynamics had been tap tested and are not subject to change due to spindle speed like spindle dynamics are. While they do change somewhat as material is removed from the part, the change was small (less than three percent of the modal mass), and as such the dynamics were regarded as fixed. Uncertainties were instead assigned to the specific cutting force coefficient and angle as $P_{prior}(K_s) = \mathcal{N}(\mu = 2500, \sigma = 200)$ N/mm^{2}m and $P_{prior}(\beta) = \mathcal{N}(\mu = 65, \sigma = 5)$ degrees. The prior for the process damping term c_n used an exponential distribution $P_{prior}(c_n) = \mathcal{E}(\lambda = 75000)$ N/m, using the PDF defined in Equation (5.3).

$$\mathcal{E}(\lambda) = \begin{cases} \lambda \exp(-\lambda x) & x \leq 0 \\ 0 & x < 0 \end{cases} \tag{5.3}$$

Next, the uncertainties were updated using a separated learning approach. The high speed cutting tests are primarily affected by K_s and β, while the low speed tests are are

affected by all three variables. All three variables can be learned simultaneously, but this causes errors from the process damping model to bias K_s and β: see Table 5.3. Additionally, sampling using all three variables with 40 samples per variable requires 64000 samples, which is computationally demanding. Instead, the uncertainties in K_s and β are first based on the high speed cutting tests. Rather than using randomly-distributed equally-weighted samples obtained via Monte Carlo sampling, samples were taken in a grid pattern with 40 samples per variable for a total of 1600 stability maps. All stability maps were calculated using the University of Tennessee, Knoxville's ISAAC-NG high-performance computing cluster, using six nodes with Intel Xeon Gold 6248R processors. Once the stability maps were calculated, they were downloaded from the cluster and the Bayesian updating was performed on a laptop with an Intel i7-10750H processor, taking less than one minute. Next, the process damping coefficient was learned from the low speed cutting tests. To reduce the computational complexity, K_s and β were fixed at their maximum likelihood estimators from the posterior. Since c_n was the only uncertainty at this point, only 40 samples were required, and these were calculated on a laptop rather than using the cluster. The posterior from the process damping updating is shown in Figure 5.8, and Table 5.3 summarizes the results. The separated learning approach was able to accurately identify the process damping value, with a posterior mean of 309000 N/m compared to the reference value of 330000 N/m. Using this separated learning approach was able to more accurately identify all three values compared to the combined learning approach (i.e., where all three variables where learned simultaneously from all cutting tests), while significantly reducing the number of required samples. Ultimately, this Bayesian approach was able to learn the process damping value.

Table 5.3: Comparison of Bayesian inference approaches for learning process damping [40].

Variable	Prior	Combined learning posterior	Separate learning: Cutting force posterior	Separated learning: Process damping posterior	Reference value
K_s (N/mm^2)	$\mathcal{N}(2500, 200)$	$\mathcal{N}(2970, 115)$	$\mathcal{N}(2671, 156)$	2671	2531.0
β (deg)	$\mathcal{N}(65, 5)$	$\mathcal{N}(60.2, 3.1)$	$\mathcal{N}(62.8, 3.9)$	62.8	62.0
c_r (N/m)	$\mathcal{E}(75000)$	$\mathcal{N}(411000, 30500)$	N/A	$\mathcal{N}(309000, 10400)$	330000

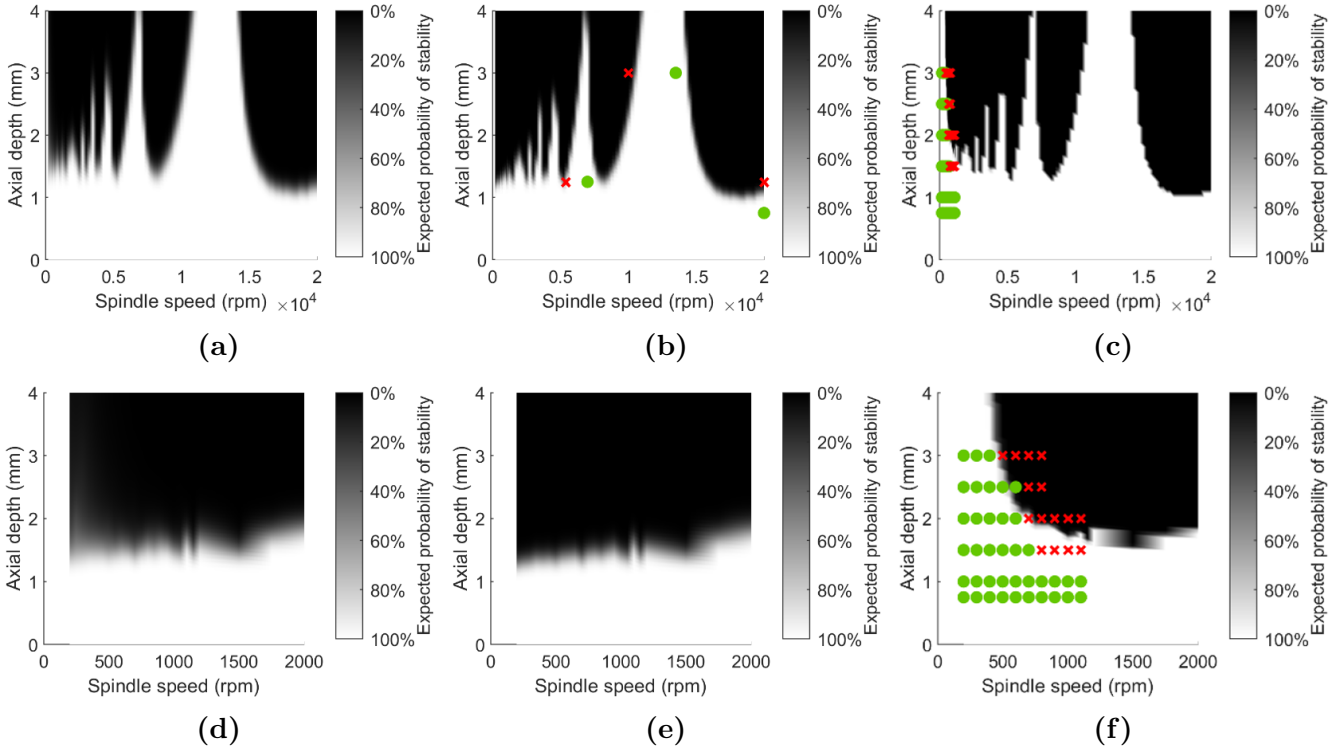


Figure 5.8: Stability maps showing the separated learning process for learning cutting tests. (a) and (d) show the prior distribution at high and low speeds respectively. (b) and (e) show the posterior after the cutting force learning. (c) and (f) show the posterior after the process damping learning.

5.3 Case study 3: Beam priors and uncertainty reduction test selection

There were two goals for this study. First, it demonstrates that simple approximate models can be used to establish the prior for simple cases. Second, it explores the effect that different test selection algorithms can have, showing how an exploration-focused algorithm can outperform greedy MRR-maximizing algorithms by reducing system uncertainty before switching to exploitation to select optimal cutting parameters. Cutting tests were performed using a 6.35 mm diameter three flute reduced shank endmill, with a stickout length of 66 mm. All testing parameters are shown in Table 5.4, and Figure 5.9 shows the setup.

Table 5.5 shows the prior for this case study. The cutting force prior was based on author experience with aluminum alloys. The FRF prior defined using Euler Bernoulli beam theory, as described in Section 4.1.2, fitting modal parameters f_n, k, ζ to the simulated FRF. These modal parameters were used as the mean for the dynamics prior, and the standard deviation for each variable was established based on a percentage of the mean value. This establishes a relatively wide prior with significant uncertainty on the stability map, shown in Figure 5.10.

Two sets of cutting tests were conducted. Table 5.6 lists all the tested cutting parameters for both sets. The first set selected cutting tests based on the MRR-maximizing strategy defined in Section 4.3.1 with a risk tolerance of 50%. Tests were halted when the next proposed test would improve MRR by less than 10%. Testing stopped after five tests with a peak MRR of 0.81 cm^3/min. Figure 5.11 shows the test series.

The second set of cutting tests was split into separate exploration and exploitation phases. The first three cutting tests were selected to maximize the reduction in uncertainty for the natural frequency, as described in Section 4.3.2, providing maximum information about what spindle speeds the stability lobes would occur at. All three tests chattered and the chatter frequencies were used to help identify the natural frequency. Testing then switched to exploitation, using the MRR-maximizing strategy to identify productive machining parameters. The next cut was stable, and had identical MRR to the fourth test cut from the first set. However, after that the hybrid testing strategy pulled ahead. After five test cuts, it achieved an MRR of 0.90 cm^3/min, 11% ahead of the first strategy. When

Table 5.4: Testing parameters for the laboratory case study 3

Machine	Haas VF4SS
Cutting tool	Accupro 62787387
Holder	Schunk 1407720 hydraulic
Tool diameter	6.35 mm
Flute count	3
Flute length	9.525 mm
Tool stickout length	66 mm
Coolant	None
Workpiece material	Aluminum 6061-T6511
Feed per flute	0.0508 mm
Radial depth of cut	3.175 mm (50% radial immersion)
$\boldsymbol{\theta}$ form	$[K_s, \beta, f_n, k, \zeta]$
Prior	FRF: Euler-Bernoulli beam theory
	Cutting force: author experience
Physics functions	Stability map and chatter frequency (ZOA algorithm)
Test selection	MRR maximization with 50% chatter risk tolerance
	Uncertainty reduction in f_n
Likelihood function	Stability: Gaussian dropoff, $\sigma_b = 0.05$ mm
	Chatter frequency: Gaussian CDF, $\sigma_{f_c} = 20$ Hz
Sampling strategy	Prior: Monte Carlo
	Posterior: Parallel adaptive MCMC

Table 5.5: Prior definition for the 6.35 mm endmill demonstration

Variable	Mean	Uncertainty %	Standard deviation
K_s (N/mm^2)	600	20	120
β (deg)	68	10	6.8
f_n (Hz)	948.9	10	94.9
k (N/m)	1.78×10^5	20	3.56×10^4
ζ	2.49%	20	0.498 %

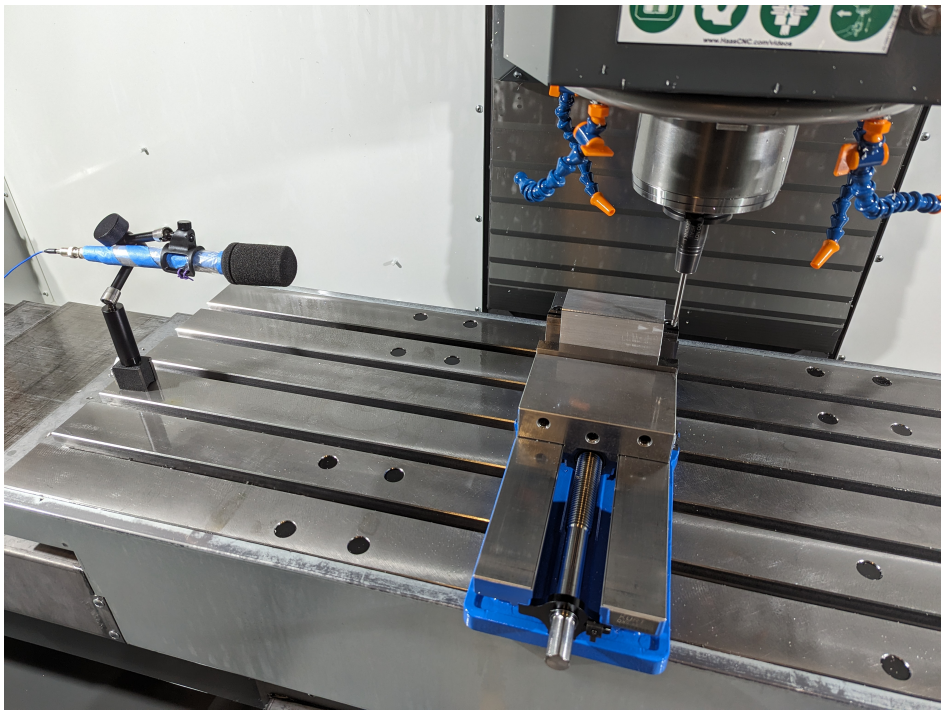


Figure 5.9: Setup for lab test 3

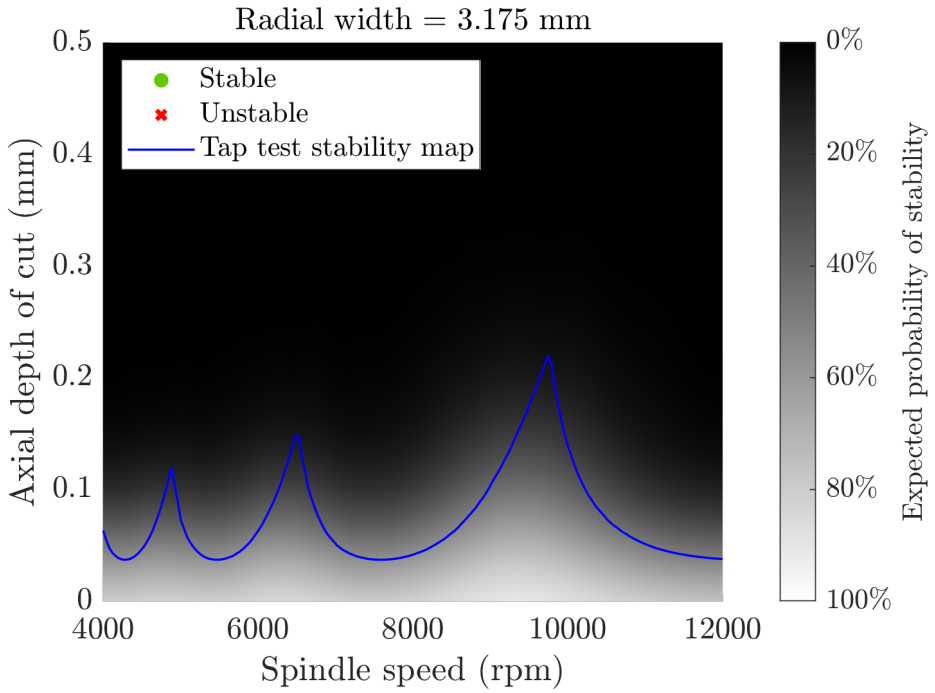


Figure 5.10: Prior probabilistic stability map for laboratory test case 3

Table 5.6: List of cutting tests for test set 3. The f_n column gives the uncertainty for the natural frequency after each test cut, expressed as the mean μ plus or minus one standard deviation σ.

Test #	MRR maximizing						Hybrid					
	n (rpm)	b (mm)	Stable	f_c (Hz)	Peak MRR (cm^3/min)	f_n ($\mu \pm \sigma$)	n (rpm)	b (mm)	Stable	f_c (Hz)	Peak MRR (cm^3/min)	f_n ($\mu \pm \sigma$)
0						949 ± 94.9						949 ± 94.9
1	9668	0.086	Yes		0.40	949 ± 94.4	6131	0.071	No	1034		1001 ± 25.3
2	9709	0.136	No	1067	0.40	1002 ± 17.6	4040	0.106	No	981		987 ± 11.3
3	10030	0.146	Yes		0.81	1009 ± 14.2	4965	0.182	No	1070		997 ± 4.9
4	10070	0.167	Yes		0.81	1010 ± 11.7	9990	0.167	Yes		0.81	998 ± 4.7
5	10111	0.187	No	973	0.81	1000 ± 5.9	9990	0.187	Yes		0.90	998 ± 4.5
6							9990	0.207	Yes		1.0	997 ± 4.1
7							9990	0.232	Yes		1.12	998 ± 4.1

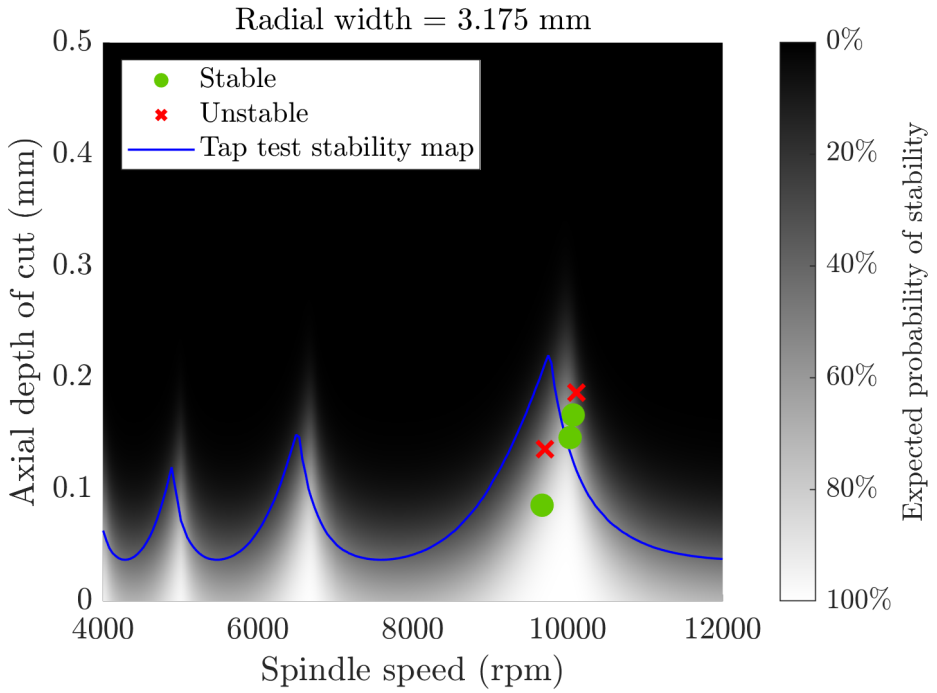


Figure 5.11: Cutting test series for laboratory case study 3, selected using the MRR-maximizing strategy.

testing was finally halted after seven tests, this strategy had found parameters with an MRR of 1.12 cm^3/min, 38% higher than the first strategy. The posterior probabilistic stability map is shown in Figure 5.12.

Note that the posterior stability lobe is shifted to a slightly higher spindle speed than the nominal stability map calculated by tap testing the tool. This occurs because the accelerometer mass loading artificially lowered the natural frequency of the tool during measurement, resulting in a less accurate FRF and stability map. This testing used a model 352C23 accelerometer from PCB Piezotronics which has a mass of 0.2 grams, 3.8% of the modal mass of the tool (5.27 g). This should shift the natural frequency by roughly 18 Hz, lowering the 2nd stability lobe by 180 rpm and closely matching the actual cutting test results.

5.4 Case study 4: Spindle power and uniform priors

One other signal that can be easily obtained from most machine tools is the spindle power. This case study shows how the spindle power can be measured using a Rogowski coil and incorporated into the Bayesian updating to help converge to the true cutting force model. It also showcases how the Bayesian model can quickly converge, even with a very low information prior, and demonstrates that this approach can work with ferrous work materials. Figure 5.13 shows the tool, workpiece, and microphone set up on the machine tool. Table 5.7 describes all the parameters for the Bayesian model. Figure 5.14 shows the Rogowski coil setup in the machine cabinet. This specific model has inputs to measure the spindle voltage as well as the current, allowing the true spindle motor power to be calculated.

Table 5.8 shows the prior distribution for this study. The cutting force prior was based on values from a commercial machining modeling software program, and included both the cutting force coefficients (formulated as K_s and β) and the tangential edge force coefficient k_{te} since that affects the spindle power. The FRF prior was established using a uniform distribution for the natural frequency, assuming that there were two modes located somewhere between 1000 and 2000 Hz. The prior stiffness and damping ratio were determined using an Euler-Bernoulli beam model for the tool and assigning a 20% uncertainty

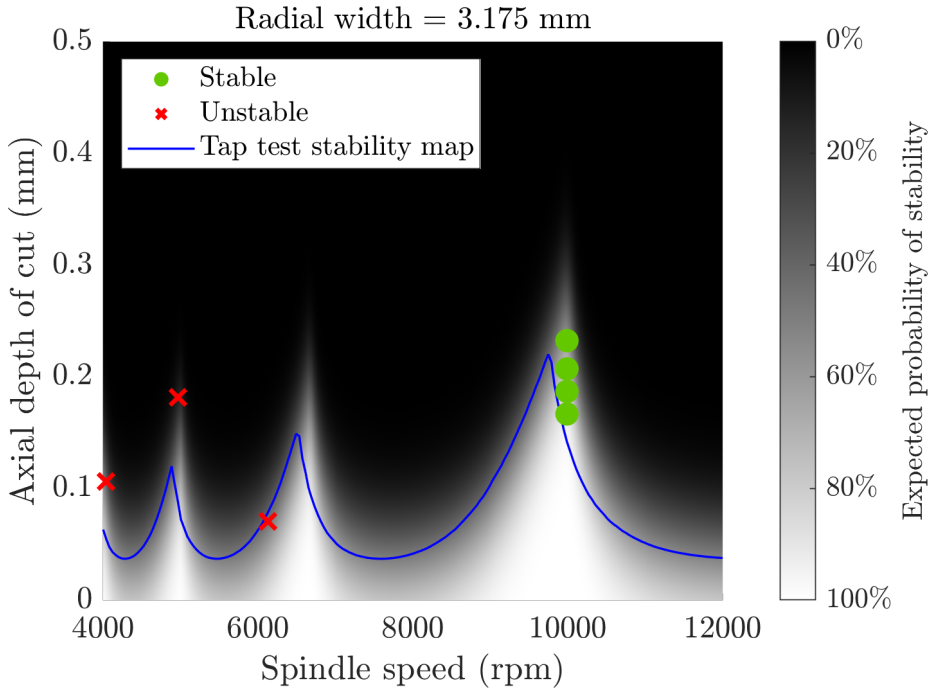
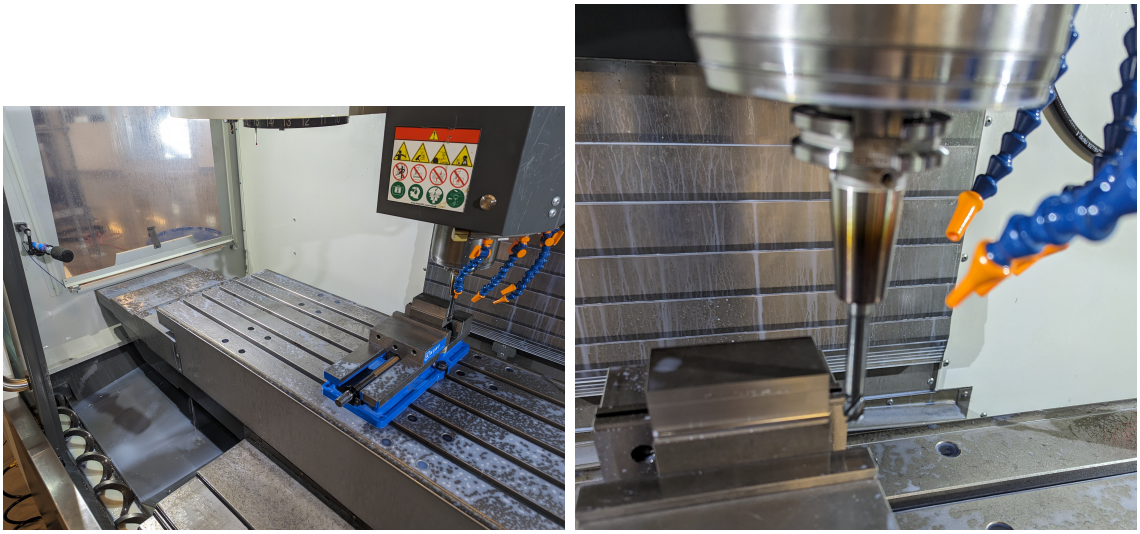


Figure 5.12: Cutting test series for laboratory case study 3, selected using the hybrid exploration/exploitation testing series.



(a) **(b)**

Figure 5.13: Setup for lab test 4. (a) Full setup. (b) Closeup of the tool and workpiece.

Table 5.7: Testing parameters for laboratory case study 4

Machine	Haas VF4
Cutting tool	YG1 GMF62935
Holder	Haimer 40.840.12.4 thermal heatshrink
Tool diameter	12 mm
Flute count	4
Flute length	16 mm (64 mm reach)
Tool stickout length	73.5 mm
Coolant	Flood
Feed per flute	0.03 mm, 0.06 mm
Radial depth of cut	6 mm
$\boldsymbol{\theta}$ form	$[K_s, \beta, k_{te}, f_{n1}, k_1, \zeta_1, f_{n2}, k_2, \zeta_2]$
Prior	FRF: Uniform distribution, two modes between 1000 and 2000 Hz Cutting force: author experience
Physics functions	Stability map and chatter frequency (ZOA algorithm) Spindle torque prediction
Test selection	MRR maximization with 50% chatter risk tolerance Uncertainty reduction for f_n
Likelihood function	Stability: Gaussian CDF, $\sigma_b = 0.05$ mm Chatter frequency: Gaussian PDF, $\sigma_{f_c} = 20$ Hz Spindle power: Gaussian PDF, $\sigma_P = 10$ W
Sampling strategy	Prior: Monte Carlo Posterior: Parallel adaptive MCMC

Table 5.8: Prior for laboratory test set 4

Variable	Prior
K_s (N/mm^2)	$\mathcal{N}(\mu = 1800, \sigma = 360)$
β (deg)	$\mathcal{N}(\mu = 68, \sigma = 6.8)$
k_{te} (N/mm)	$\mathcal{N}(\mu = 50, \sigma = 12.5)$
f_{n1} (Hz)	$\mathcal{U}(1000, 2000)$
k_1 (N/m)	$\mathcal{N}(\mu = 5 \times 10^6, \sigma = 1 \times 10^6)$
ζ_1	$\mathcal{N}(\mu = 0.03, \sigma = 0.01)$
f_{n2} (Hz)	$\mathcal{U}(1000, 2000)$
k_2 (N/m)	$\mathcal{N}(\mu = 5 \times 10^6, \sigma = 1 \times 10^6)$
ζ_2	$\mathcal{N}(\mu = 0.03, \sigma = 0.01)$
Penalty function	$f_{n1} > f_{n2}$, see Equation (5.4)

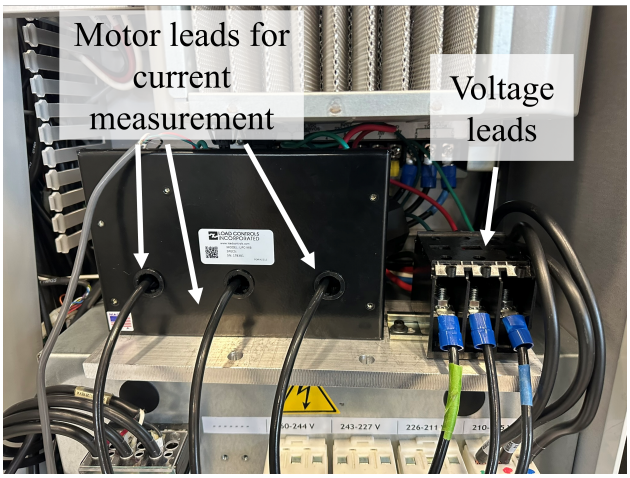


Figure 5.14: Rogowski coil setup in the machine tool control cabinet for measuring spindle power.

for stiffness and a 33% uncertainty for damping. Since the ratio between the tool stickout length and diameter was only 5:1 this approximation is not accurate, but serves to provide a rough initial guess for the Bayesian model. In order to keep the modes sorted such that $f_{n1} \geq f_{n2}$, a penalty function was applied to the prior distribution, shown in Equation (5.4). This penalty function decreases the probability the closer the two values get to keep the two modes separated. During the initial Monte Carlo sampling, this penalty was applied using rejection sampling (i.e., by drawing a sample, generating a random value a from $\mathcal{U}(0,1)$ and discarding that sample if $a \geq P_{penalty}(\boldsymbol{\theta})$), and during the MCMC sampling it was multiplied into the prior probability. Figure 5.15 shows the stability uncertainty for that prior.

$$P_{penalty}(\boldsymbol{\theta}) = 1 - \exp\left(-\frac{1}{2}\left(\frac{\max(\boldsymbol{\theta}_{f_{n2}} - \boldsymbol{\theta}_{f_{n1}}, 0)}{100}\right)^2\right) \tag{5.4}$$

In order to learn both the cutting force and edge force coefficients, each suggested cutting test was run at two different feedrates, 0.03 mm and 0.06 mm. The first two tests were selected using the MRR maximization approach, but after that it was not able to suggest any more productive cutting parameters since the Bayesian model is not confident enough about the natural frequency of the other mode to suggest any more productive cutting parameters. Figure 5.16 shows the probabilistic stability map and natural frequency distribution after chatter is first observed. f_{n2} has been accurately identified after observing the chatter frequency at 7000 rpm (1255 Hz), but f_{n1} is now a complex multimodal distribution. There is therefore no clear stability peak to explore and improve MRR, at least not with the 50% stability risk tolerance. Instead, the next test was selected to reduce the uncertainty for f_{n1} before switching back to MRR maximization. Table 5.9 shows the sequence of cutting tests, terminating after five recommendations (ten total cutting tests).

Next, those eight cutting tests were used to perform Bayesian updating both with and without including spindle power as an updating variable. Figure 5.17 shows the posterior stability maps for both updating sequences, while Table 5.10 compares the posterior uncertainties for both cases to values measured with a cutting force dynamometer and modal tap-testing. Note that the stability map calculated using tap-testing is not entirely accurate:

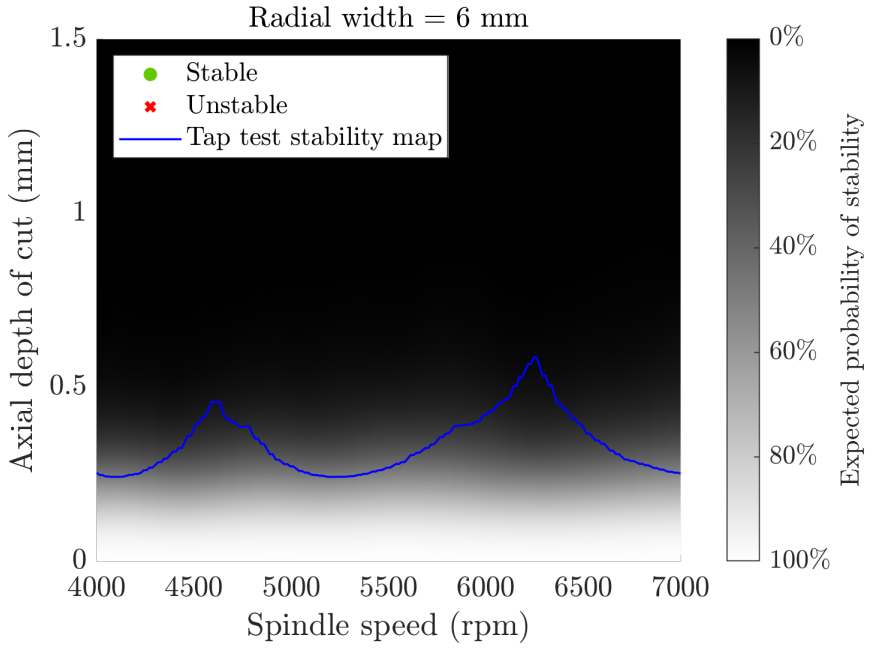
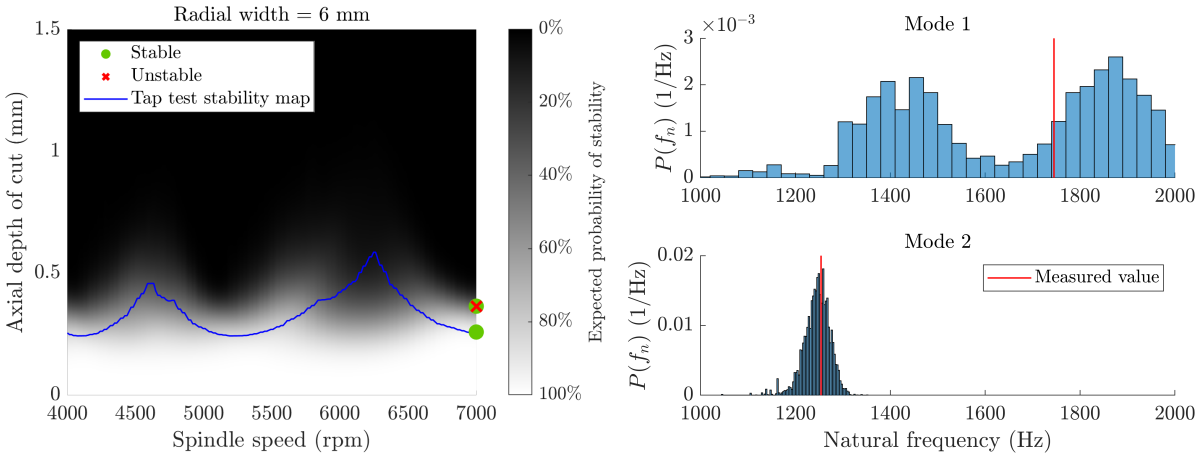


Figure 5.15: Prior probabilistic stability map for case study 4.



(a) **(b)**

Figure 5.16: Posterior after the first unstable cut. (a) Posterior probabilistic stability map. The point with both an unstable and stable cut was stable at 0.06 mm feed and unstable at 0.03 mm feed. (b) Posterior distributions for the natural frequencies.

Table 5.9: Test cuts for laboratory case study 4

Test #	n (rpm)	b (mm)	0.06 mm feed			0.03 mm feed		
			Stable	f_c(Hz)	Spindle power (W)	Stable	f_c(Hz)	Spindle power (W)
1	7000	0.258	Yes		114	Yes		51
2	7000	0.364	Yes		153	No	1255	80
3*	6140	0.409	Yes		244	Yes		159
4	6005	0.559	Yes		331	Yes		192
5	6261	0.939	Yes		368	No	1215	232

* Selected to reduce uncertainty for f_{n1}

Table 5.10: Comparison of the prior and posterior distributions for each system variable.

Variable	Prior ($\mu \pm \sigma$)	Without spindle power ($\mu \pm \sigma$)	With spindle power ($\mu \pm \sigma$)	Measured Value
k_{tc} (N/mm^2)	1656 ± 338	1239 ± 155	1954 ± 12.36	2278
k_{nc} (N/mm^2)	675 ± 238	532 ± 104	880 ± 22.0	464
k_{te} (N/mm)	50.0 ± 12.6	47.2 ± 8.45	26.4 ± 0.349	30.0
f_{n1} (Hz)	1631 ± 252*	1651.1 ± 12.7	1639 ± 5.57	1745
k_1 (MN/m)	4.97 ± 0.974	5.73 ± 0.508	7.70 ± 0.0629	12.6
ζ_1 (%)	3.00 ± 0.990	3.91 ± 0.613	4.61 ± 0.0209	2.53
f_{n2} (Hz)	1374 ± 258*	1228 ± 5.88	1242 ± 2.25	1254
k_2 (MN/m)	5.00 ± 0.996	4.89 ± 0.669	5.52 ± 0.0405	3.33
ζ_2 (%)	3.01 ± 0.982	3.28 ± 0.465	3.93 ± 0.0209	4.34

* Numerical approximation since the prior was not a normal distribution

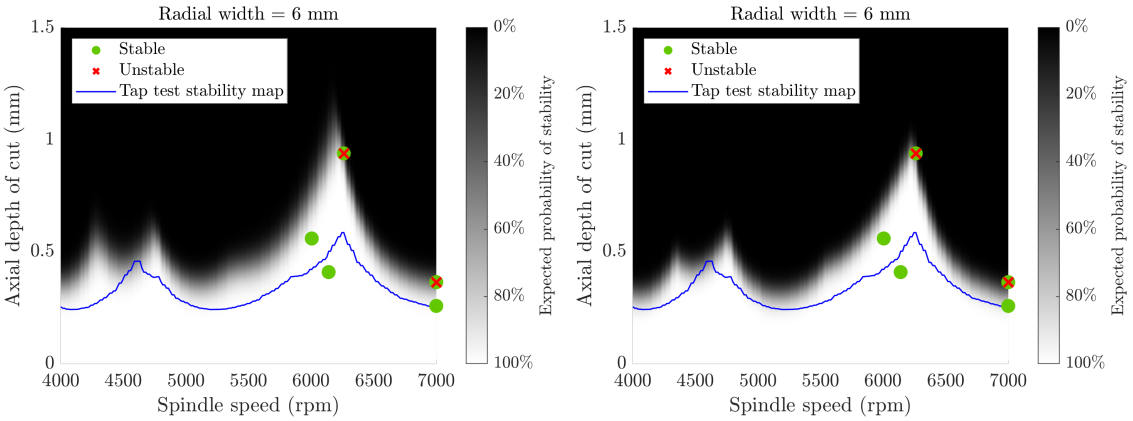


Figure 5.17: Posterior stability map and cutting tests, (a) with and (b) without using the measured spindle power in the Basyesian updating. Cases where both a stable and unstable test are overlapping indicate that the 0.03 mm feedrate test chattered while the 0.06 mm feedrate test was stable.

while the stability peaks occur at roughly the correct spindle speeds, actual cutting tests are stable at much higher axial depths than predicted.

Figure 5.18 shows the distribution of samples taken from the prior distribution for reference, while Figure 5.19 shows the posterior distributions without spindle power updating[1]. This posterior significantly underestimates the cutting force due to how much higher the stability boundary is. Furthermore, the cutting force is correlated with the stiffness and damping ratios: if the cutting force is higher then the dynamic stiffness of the system must also be higher to have the correct limiting axial depth. Also note that f_{n1}, the natural frequency for the first mode, is inaccurate. Chatter with this mode has never been observed so there is no chatter frequency to identify the natural frequency and the natural frequency is inferred solely from the binary stability results, meaning that either the stability valleys for mode 1 don't line up with any of the cutting tests that were run or that mode 1 is much stiffer than the other modes so that its stability valleys lie above all the tests that were run. In actuality this mode is roughly four times stiffer than mode 2, but the prior assigned a low probability of the stiffness being that high so the Bayesian model converged to a natural frequency where the stability valleys don't line up with any of the test cuts.

Adding in the spindle power as another updating variable gives the posterior shown in Figure 5.20. This distribution converges much more tightly, finding unique values for essentially every variable. While it still somewhat underestimates the tangential cutting and edge coefficients, the estimate is much closer than the first data set, within 15% of the value measured with a cutting force dynamometer. This is at least partly caused by the stability limit being higher than expected: using the spindle power reduces the errors caused by the inaccurate boundary, but it will still cause some bias for both the dynamic stiffness and cutting force. Decreasing the power likelihood uncertainty σ_P further could reduce this bias, at the cost of computational issues (discussed in depth later). This is also likely affected by the tool's 2 mm corner radius, which causes the exact cutter engagement with the workpiece to vary depending on axial and radial depth. Otherwise, the same general inaccuracies are present here as in the first case: the stiffnesses and damping ratios are not accurately

[1]For convenience, the cutting force has been recalculated in terms of k_{tc} and k_{nc} rather than K_s and β.

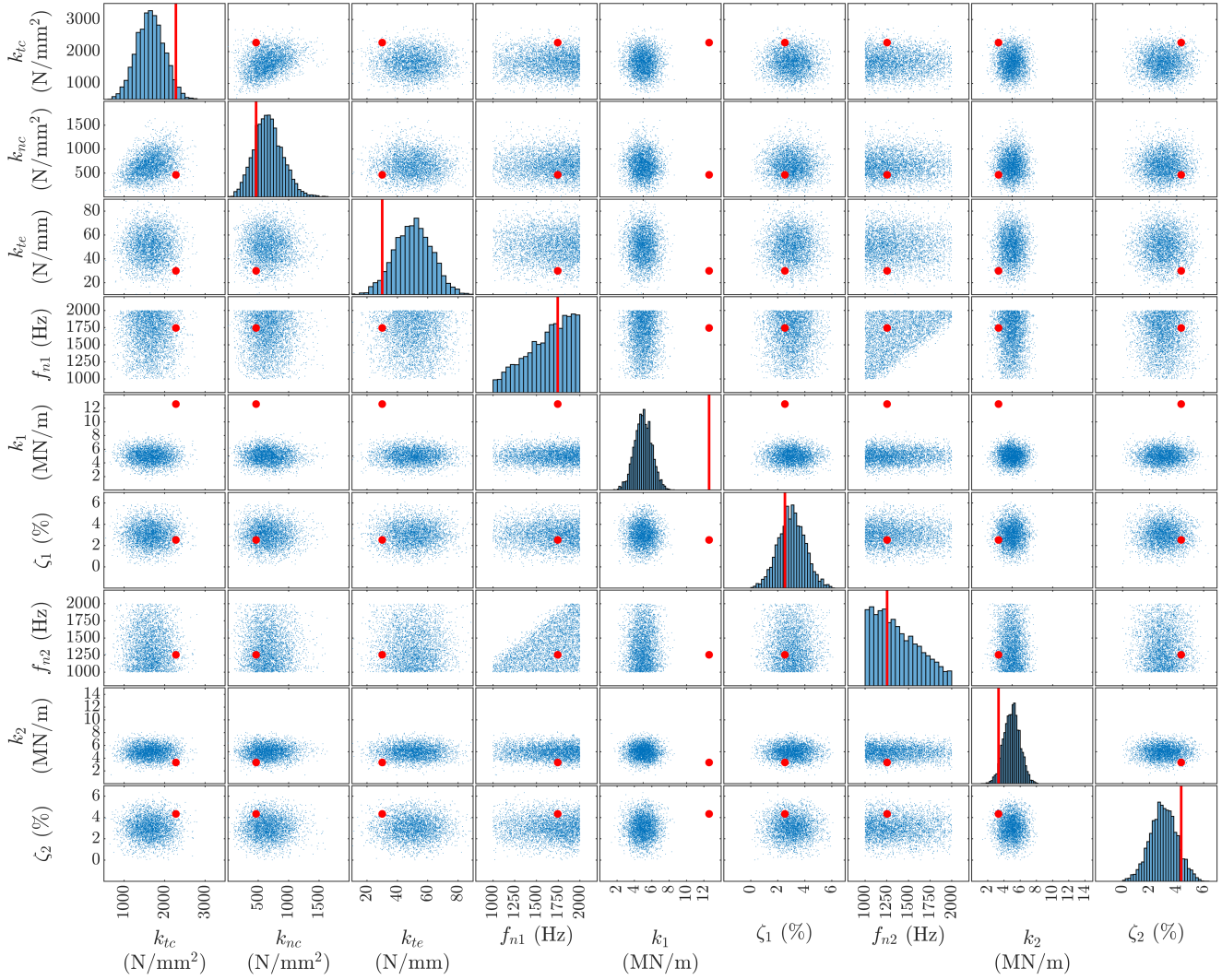


Figure 5.18: Distribution of samples drawn from the prior distribution for case study 4

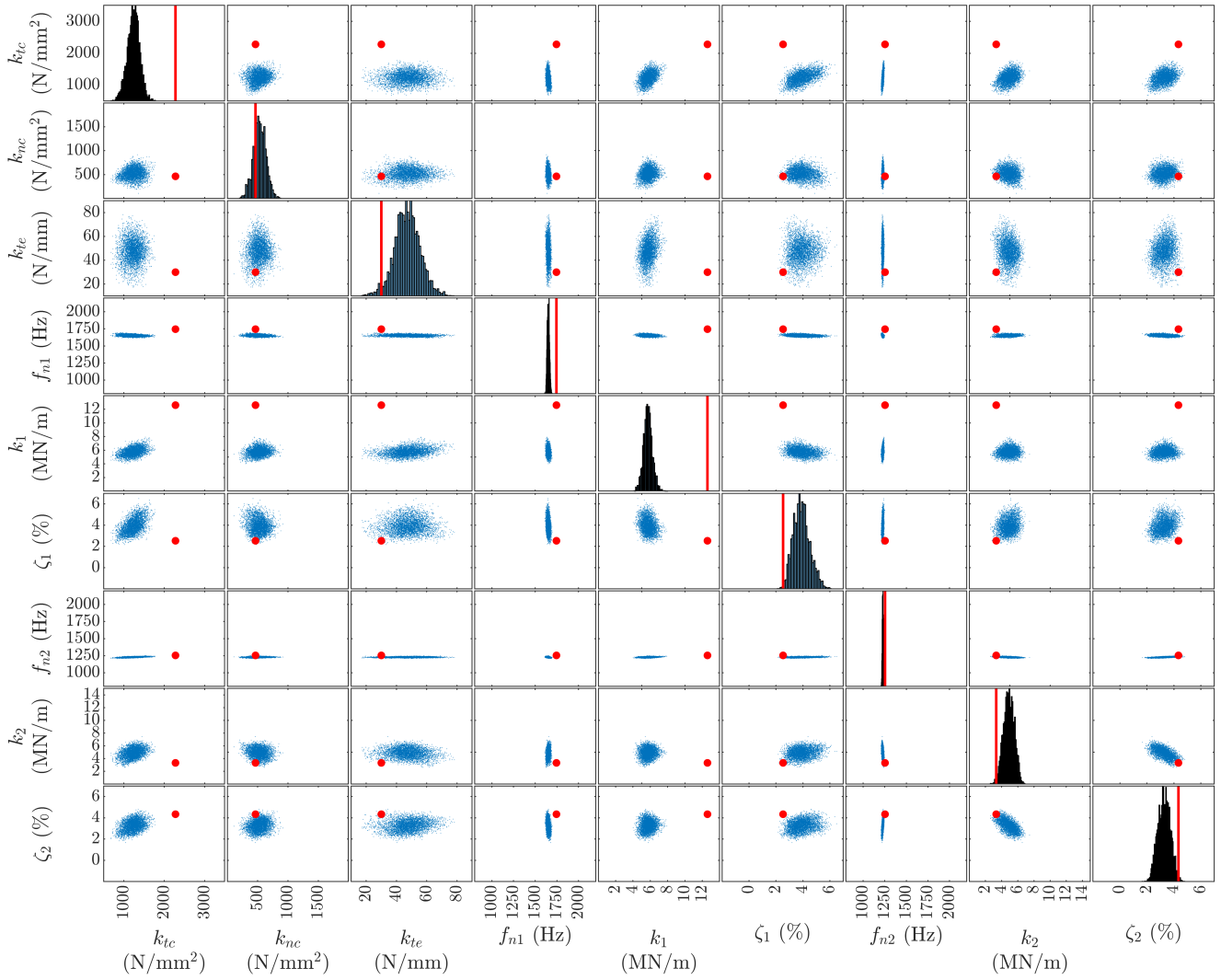


Figure 5.19: Posterior distribution for case study 4, without spindle power in the Bayesian updating

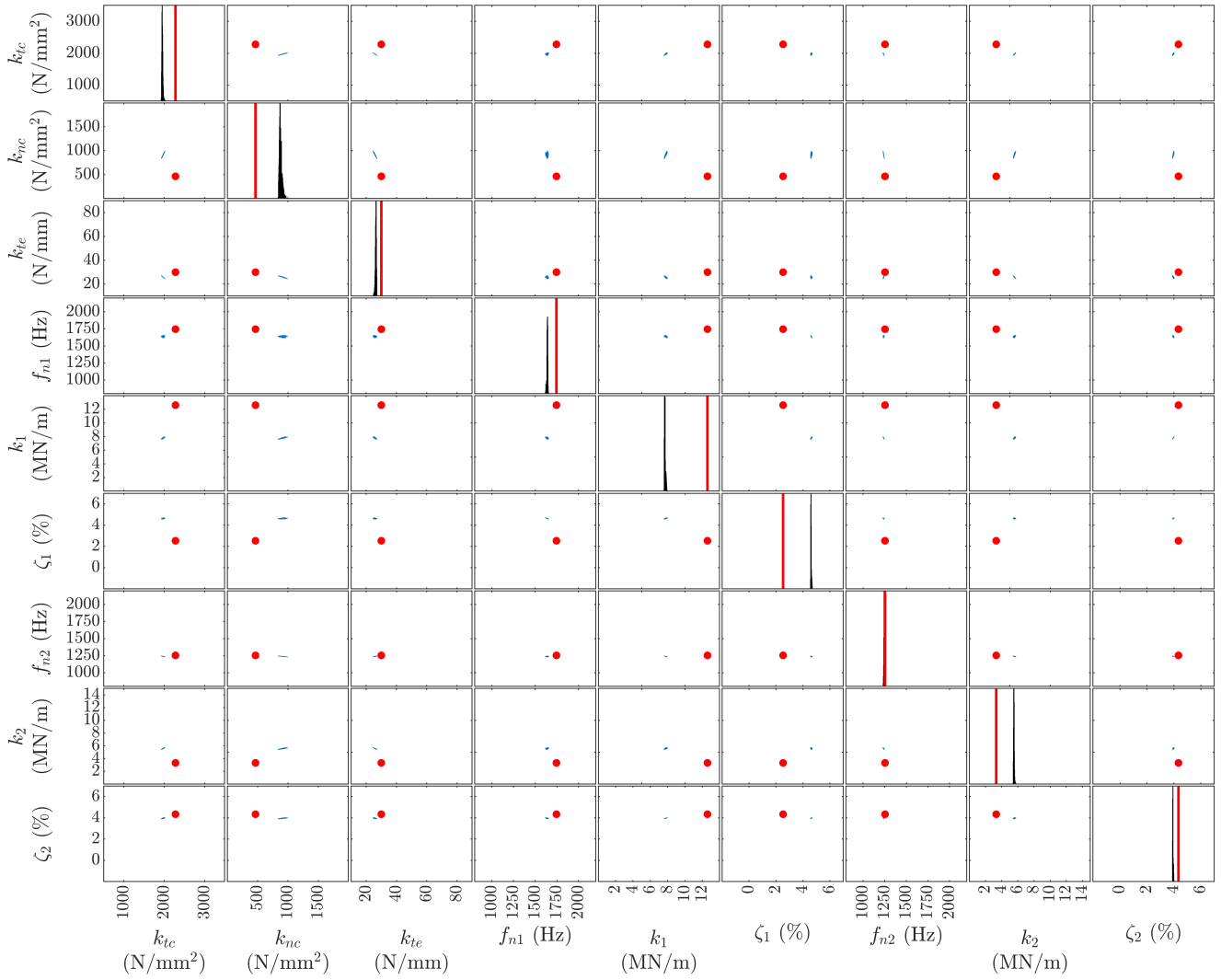


Figure 5.20: Posterior distribution for case study 4, with spindle power in the Bayesian updating

measured due to the higher stability limit, and the mode 1 chatter frequency has not been observed so the natural frequency is not well known.

Three challenges were experienced here. In some cases, the stability of the cutting test depended on the feedrate, with tests at 0.06 mm being stable and tests at 0.03 mm chattering. Here, this effect proved beneficial since the unstable tests helped learn the system natural frequency. However, the ZOA stability algorithm assumes that stability is independent of feedrate and thus generates the same stability predictions for both the 0.03 and 0.06 mm feed cases. The Bayesian model is confronted with conflicting information about whether a test point is stable and will fit a compromise model which assumes that axial depth is the stability limit, even though it may still be possible to cut deeper at the higher feedrates.

Second, it can sometimes be difficult to identify the dominant chatter frequency. Figure 5.21 shows the frequency-domain microphone data collected from test cut 5 at 0.03 mm feed per flute. Frequency content at content at flute passing frequencies reflects normal cutting behavior, while content at other frequencies indicates chatter. The highest amplitude chatter frequency is 1632 Hz, but the true dominant chatter frequency (i.e., the frequency associated with the lowest b_{lim} value at that spindle speed) is actually 1215 Hz. The 1632 Hz data is a sideband frequency, which occur at constant offsets from the flute passing frequencies in unstable cuts [84]. The relative magnitudes of these sidebands depends on the system dynamics, the selected cutting parameters and resulting frequency-domain forces, and the frequency response of the microphone. As a result, the highest magnitude frequency is not necessarily the dominant chatter frequency. For this study, the dominant chatter frequency was always selected as the lowest non-flute-passing frequency with significant magnitude.

Finally, the adaptive MCMC sampler described in Section 4.5.4 struggled to quickly and accurately sample this distribution. This sampler assumes that the target distribution is fairly similar to an MVN and has lower acceptance rate the more complex the posterior is. Due to uniform prior some variables had complex multimodal distributions, causing the acceptance rate sometimes fell below 10% on some of the updates here rather than the usual 30-40%. Also, the rejection sampling approach used to select the starting point and help calculate the proposal distribution assumes that the distribution changes gradually with each update. Introducing the spindle torque as another variable causes the posterior to change

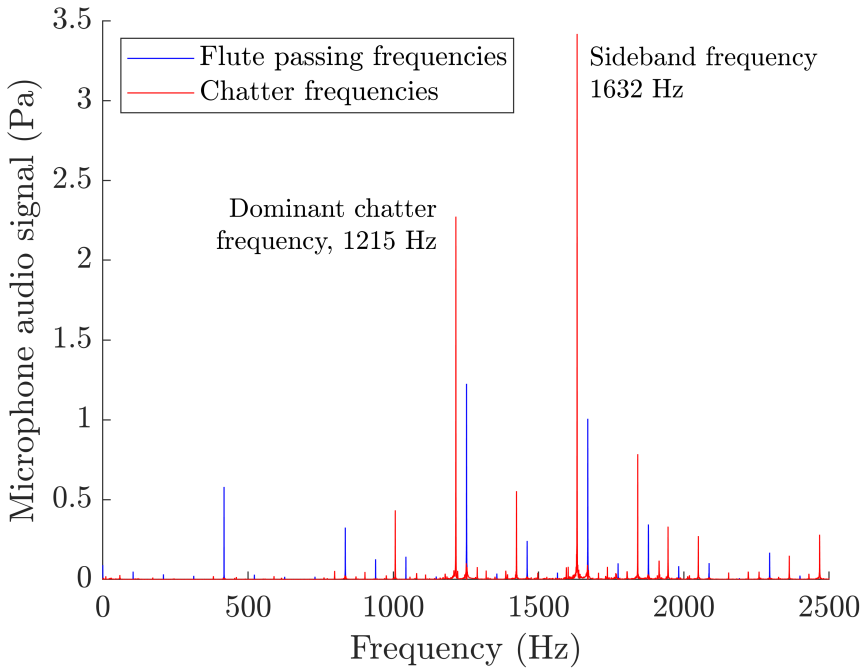


Figure 5.21: Frequency-domain microphone data collected from test cut 5 at 0.03 mm feed per flute. While the peak at 1632 Hz has the highest magnitude, the 1215 Hz peak is actually the dominant chatter frequency.

more rapidly. If σ_P is too small, the likelihoods become small and too many samples get rejected, which makes the proposal distribution too small and preventing the sampler for fully exploring the target distribution. The value used here, $\sigma_P = 10$ W, is the smallest value possible able to accurately sample the posterior, and may still be underestimating the actual width of the posterior.

Chapter 6

Results and Discussion: Industrial Testing

In industrial settings, it is essential to optimize machining performance. As part of a Department of Energy project, the Bayesian process was used to optimize the machining process for a relevant meter-scale aluminum 7050-T7451 component in collaboration with an industrial partner. The CNC milling machine used in the study had a high-speed spindle providing over 100 kW of power at 33000 rpm. This component had several challenging features requiring long tools which posed significant chatter risks, making it a good candidate for the Bayesian optimization method. Additional details are not included due to confidentiality requirements. Table 6.1 summarizes the Bayesian model parameters used for this optimization.

The basic process was designed by the industrial partner based on their experience with similar components. They selected cutting tools and programmed toolpaths to machine the component. From the initial set of thirteen tools, eight cutters were selected for optimization, prioritizing tools with the longest estimated cycle times and highest difficulty for selecting stable parameters. Table 6.2 shows the dimensions of each of the selected cutting tools. A set of baseline cutting parameters was established for each of the eight target tools, based on recommendations from the tool manufacturer. The 25.4 mm cutting tools had baselines for roughing cuts at 100% radial immersion, while the 19.05 mm and 12.7 mm tools had cutting parameters for both roughing and finishing cuts at 20% radial immersion, shown in

Table 6.1: Bayesian model parameters for laboratory case study

$\boldsymbol{\theta}$ form	$[K_s, \beta, k_{\theta f}, c_{\theta f}]$
Prior	FRF: uncertainty in RCSA coupling parameters
	Cutting force: author experience
Physics functions	Stability map and chatter frequency (ZOA algorithm)
	Spindle torque prediction
	Bending moment prediction
	Tool bending stress prediction
Test selection	MRR maximization with 50% chatter risk tolerance
	Reduction in $k_{\theta f}$ uncertainty
Likelihood function	Stability: Gaussian CDF, σ_b depending on tool
	Chatter frequency: $\sigma_{f_c} = 20$ Hz
	Spindle torque: 5 Nm
	Tool breakage: $\sigma_B = 10$ MPa
Sampling strategy	Prior: Monte Carlo
	Posterior: Parallel adaptive MCMC

Table 6.2: Tool geometry for the industrial case study

Tool #	Diameter (mm)	Flute length (mm)	Stickout length (mm)	Flute count
1	25.4	31.75	57.15	3
4	25.4	31.75	107.95	3
5	25.4	28.575	133.35	2
7	19.05	25.4	57.15	3
8	19.05	25.4	69.85	3
9	19.05	25.4	92.71	2
10	19.05	25.4	117.475	2
14	12.7	15.875	116.84	2

Tables 6.3 and 6.4[1]. These baselines reflect naive starting parameters based solely on the tool diameter and flute count, and do not consider tool length. In some cases, the baseline parameters lie above the stability boundary and would chatter. Therefore, the tables also report the limiting axial depth b_{lim} at 33000 rpm calculated using the tap tested stability map. This provides a rough estimate for the maximum productivity that would be achievable by a machine shop without using tap testing equipment or performing time-consuming trial and error searching to find stable lobes.

The FRFs were calculated using RCSA, with uncertainties in the force-to-rotation coupling stiffness $k_{\theta f}$ and damping $c_{\theta f}$. Each of the tools was tap tested and coupling parameters were selected to fit the measured FRF as accurately as possible. A linear model was then fit to the coupling parameters to predict them based on the tool diameter, its square, the tool stickout length, the tool gage length, and the radio between the tool stickout length and diameter. Table 6.5 shows the best fit and predicted coupling parameters for the selected tools. The damping was clearly the most difficult to predict, with several large outliers where the model either over or under predicted the damping by more than 100%. The model predictions for each tool were used as the mean values for their prior distributions, and the standard deviation was set as a percentage of the mean value using the standard deviation of the percent errors for the model predictions. This resulted in an uncertainty for each tool of 17.2% of the mean for $k_{\theta f}$ and 170% for $c_{\theta f}$. The only exception is tool 14, since the model predicted negative damping. For that tool, the mean was set to 0.7 Ns/m to provide a low initial estimate for the damping. The cutting force prior was based on author experience, using normal distributions for the specific cutting force $P(K_s) = \mathcal{N}(850, 170)$ N/mm^2 and force angle $P(\beta) = \mathcal{N}(68, 6.8)$ deg.

Next, each of these tools was optimized to find the most productive machining parameters, selecting cutting tests using the MRR-maximizing algorithm. Straight-line cutting tests in the X direction were used for simplicity, and were recorded with a microphone to measure chatter frequency and help classify the cuts as stable or unstable. In addition to stability, the test selection also considered constraints on spindle torque, spindle bending

[1]Note that tool 6 was removed from the process plan, thus why there are 14 tool numbers for only 13 tools.

Table 6.3: Roughing baseline for all tools based on the cutting tool manufacturer's recommendations

Tool #	Spindle speed (rpm)	Axial depth (mm)	Radial depth (mm)	Feed per flute (mm)	MRR (cm^3/min)	b_{lim} at 33000 rpm (mm)	Stable?
1	33000	10.16	10.16	0.244	3115	15.8	Yes
4	33000	10.16	10.16	0.244	3115	2.1	No
5	33000	10.16	10.16	0.244	2077	1.6	No
7	33000	7.62	9.525	0.163	1168	10.9	Yes
8	33000	7.62	9.525	0.163	1168	2.5	No
9	33000	7.62	9.525	0.163	779	3.1	No
10	33000	7.62	9.525	0.163	779	0.90	No
14	33000	5.08	6.35	0.122	260	0.022	No

Table 6.4: Finishing baseline for all tools based on the cutting tool manufacturer's recommendations

Tool #	Spindle speed (rpm)	Axial depth (mm)	Radial depth (mm)	Feed per flute (mm)	MRR (cm^3/min)	b_{lim} at 33000 rpm (mm)	Stable?
7	33000	7.62	3.81	0.091	263	25.4	Yes
8	33000	7.62	3.81	0.091	263	23.3	Yes
9	33000	7.62	3.81	0.091	175	18.7	Yes
10	33000	7.62	3.81	0.091	175	6.4	No
14	33000	5.08	2.54	0.061	34.6	0.14	No

Table 6.5: Comparison of predicted and actual RCSA parameters for the cutting tools

Tool number	Best fit $k_{\theta f}$ (N/rad)	Predicted $k_{\theta f}$ (N/rad)	Best fit $c_{\theta f}$ (Ns/rad)	Predicted $c_{\theta f}$ (Ns/rad)
1	10000000	10667053	50	78.2
4	7000000	6767115	50	30.7
5	3700000	4830364	5	7.16
7	4000000	4490085	140	91.0
8	3500000	3803288	60	81.1
9	3000000	2573884	70	63.5
10	1600000	1232743	10	44.3
13	1250000	1232849	20	14.5
14	600000	803861	2	-7.26

moment, bending stress on the cutting tool, and vibration on the machine tool's spindle-mounted accelerometer. Ideally, cuts should be maintained at under 6G acceleration, and if this accelerometer measures sustained vibrations over 10G (with 1G being the acceleration due to gravity), then the machine will alarm out. In some cases, the suggested cutting parameters were modified slightly based on user experience with the machine to reduce vibration amplitude using the probabilistic stability map as a guide, generally by dropping spindle speed slightly or choosing to do start at a lower depth and work up to a recommended test point.

Those selected cutting parameters were then used to machine the first test part and prove out the program. During the initial part machining, several issues occurred. First, the original machine that the Bayesian optimization was performed on was undergoing routine maintenance, so the part was machined on another machine of the same make and model. However, despite being nominally identical there were small shifts in the machine tool's spindle dynamics that were sufficient to cause one cutting tool to chatter. Figure 6.1 shows the selected cutting parameters for tool 8 overlaid on stability maps calculated from tap tests for both machines, showing how the difference in machine dynamics was sufficient to shift the stability lobe enough to induce chatter (note that vibration amplitude prevented cutting deeper than 10 mm, even though the nominal stability map predicts that deeper cutting is possible). The Bayesian approach used here is intended to maximize MRR for a given setup, and the recommended cutting parameters cannot necessarily be used on other machines or setups. It is usually more practical to sacrifice a small amount of productivity to ensure that a single program and set of cutting parameters can be safely run on all machine tools. In order to fix this while requiring minimum reprogramming, the spindle speed was dropped so that it would be stable on either machine.

Second, tool 1 experienced higher amplitude vibrations than predicted. The slotting cuts performed for optimization had an amplitude of roughly 5G. This was verified on the new machine, with straight slotting cuts having a slightly higher amplitude of roughly 5.2G. However, when cutting the actual part there were spikes in amplitude up over 10G when the tool engaged or disengaged with the part. This was particularly prominent on the first Z depth cutting pass.

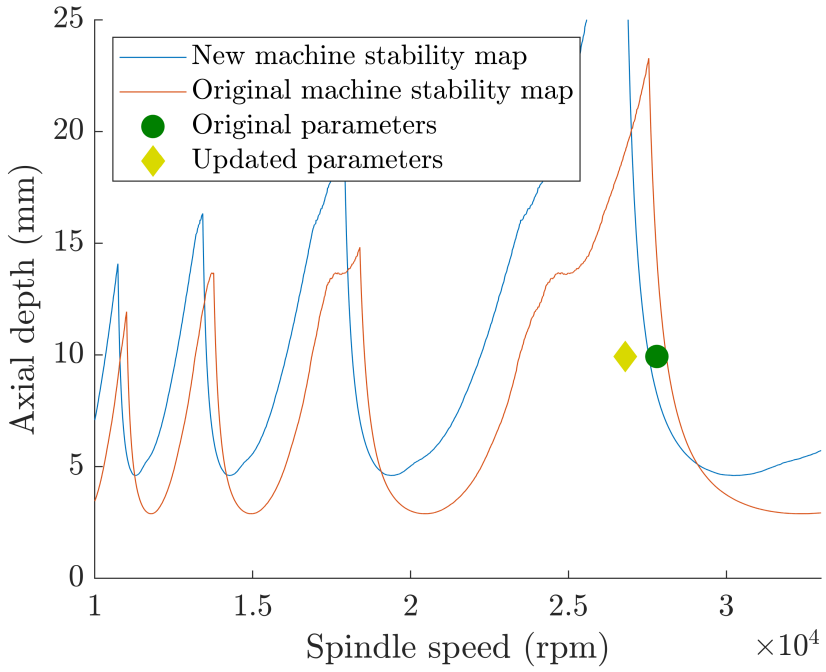


Figure 6.1: Shift in the stability map due to switching between two nominally identical machine tools.

Third, tool 14 chattered while machining the corners of some pockets. This tool had a 3 mm corner radius on the cutting edges. As the tool stepped down, this resulted in the engagement between the tool and workpiece being larger than expected: despite the nominal 0.305 mm stepdown the tool could be engaged along the entire corner radius. This is even worse in tight corners. The part featured several internal corners with radii only slightly larger than the tool (6.86 mm vs the tool's 6.35 mm radius). Figure 6.2 shows the difference in engagement as the tool moves into the pocket corner. The red areas show the area of the tool which is engaged with the part, taking into account the tool's corner radius. As the tool enters the corner more of the tool's corner radius becomes engaged, increasing the cutting forces on the tool and causing chatter. For comparison, the engagement area for the straight wall cut is 0.23 cm^2 while the engagement in the corner is 0.38 cm^2. This results in higher cutting forces which induce chatter in the corners. This effect cannot be mitigated by reducing the tool's axial depth because of the 3 mm corner radius, and due to the geometry of the pocket it was not possible to reduce the radial engagement. This same effect had been encountered by the industrial partner as well, who were unable to machine the target geometry without the tool breaking.

Instead, an alternate set of cutting parameters for slotting was selected using the Bayesian process, running the tool at 11271 rpm at 0.191 mm axial depth. This shifted the operating parameters into the second stability lobe rather than the first (which was located at roughly 22500 rpm) and aligned the second harmonic of the flute passing frequency with the natural frequency of the tool. In slotting cuts the second force harmonic is very low, and while this cut is not purely slotting this approach still results in lower amplitude vibrations. While the tool still chattered in the corners the amplitude was lower than it had been at the higher spindle speeds and the tool was able to machine the pocket without breaking. Several modifications to the tool geometry were recommended to the industrial partner which would improve performance going forward.

Tables 6.6 and 6.7 compare the MRR for the final optimized roughing and finishing parameters to the baseline. In most cases, the Bayesian method was able to identify parameters more productive than the baseline, but some tools were less productive than the manufacturer's naive recommendations. For those tools, there are no stable cutting

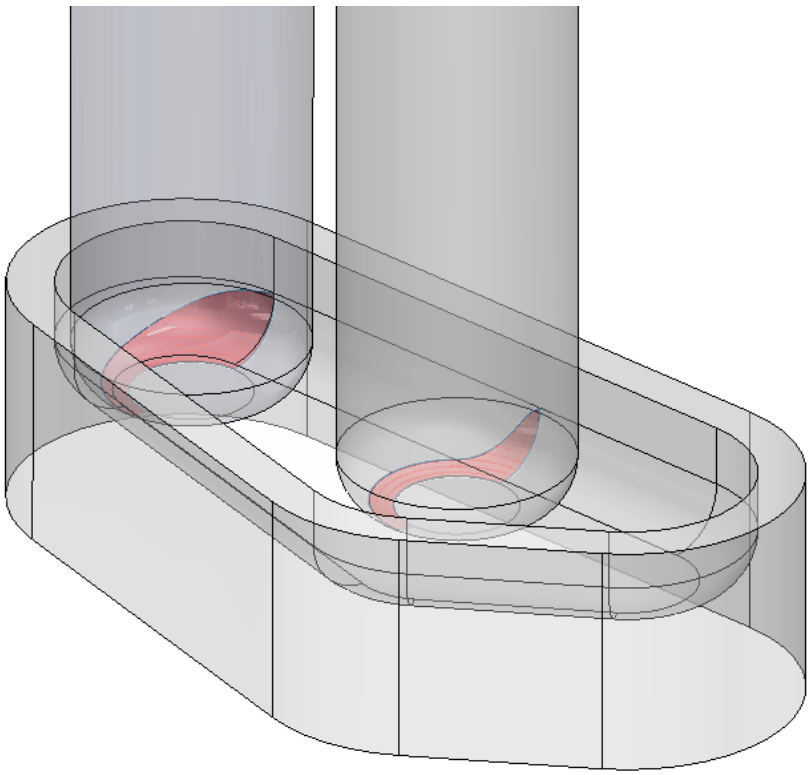


Figure 6.2: Engagement between the tool and workpiece as the tool moved into the corner of a pocket.

Table 6.6: Final optimized roughing parameters for the industrial case study

Tool #	Spindle speed (rpm)	Axial depth (mm)	Radial depth (mm)	Feed per flute (mm)	MRR (cm^3/min)	% of baseline MRR	% of 33000 rpm MRR	# of tests
1	28955	18.3	25.4	0.244	9835	316	102	6
4	18600	3.05	25.4	0.127	549	18	80	10
5	26181	2.54	25.4	0.244	824	40	123	5
7	28492	19.8	19.05	0.191	6130	525	157	9
8	26799	9.93	19.05	0.191	2898	248	329	5
9	30000	5.64	19.05	0.191	1228	158	168	3
10	24800	3.56	19.05	0.140	469	60	297	10
14	11271	0.191	12.7	0.0508	2.8	1	297	3

Table 6.7: Final optimized finishing parameters for the industrial case study

Tool #	Spindle speed (rpm)	Axial depth (mm)	Radial depth (mm)	Feed per flute (mm)	MRR (cm^3/min)	% of baseline MRR	% of 33000 rpm MRR	# of tests
7	33000	25.4	3.81	0.091	876	333	100	1
8	31035	25.4	3.81	0.091	824	313	102	5
9	33000	25.4	3.81	0.091	584	333	136	1
10	31497	15.6	3.81	0.091	343	196	234	7
14	20171	2.87	2.54	0.0508	14.9	29	1281	5

parameters which meet the baseline productivity. To provide a more realistic comparison, the tables also compare the optimized MRR to the best productivity that would be achievable at maximum spindle speed (i.e., using the 33000 rpm b_{lim} axial depths reported in Tables 6.3 and 6.4). Generally, the shorter cutting tools had MRRs close to the 33000 rpm MRR since they are less likely to experience chatter, while longer cutting tools show a higher improvement since it is more important to find a stability lobe. There is only one case where the suggested parameters did not at least match the 33000 rpm MRR, tool 4 for roughing. This particular case experienced significant tool vibration which prevented the test cuts from exploring the very peak of the stability lobe.

Next, the industrial partner will reprogram the toolpaths using the final machining parameters and machine four more test parts to demonstrate that the process is repeatable. This is expected to be completed by May 2024.

In summary, the Bayesian optimization process was able to recommend highly productive cutting parameters for a realistic component. However, this process still required some user guidance to minimize vibrations and select parameters which can be used across multiple machines. Additionally, cutting parameters were selected without considering surface finish or tolerance requirements. As a result, there were some minor surface mismatches and gouges where different toolpaths intersect, which may or may not be acceptable on a real component depending on tolerances. Workpiece dynamics were also not considered here: on many parts thin walls and floors can chatter or deflect, but this test component had relatively thick walls. These challenges reinforce that simple straight-line test cuts are not sufficient to fully model the complexities of a real machining process, and represent opportunities for future research.

Chapter 7

Conclusion and Recommendations

7.1 Summary

In order to select optimal machining parameters, it is necessary to predict how those parameters will affect the outcome of the machining process. However, accurate machining simulations require knowledge of difficult-to-measure variables like the tool's natural frequency and the cutting force model. This dissertation has described a physics-based Bayesian framework for statistically modelling and optimizing machining processes under a state of uncertainty. This approach helps users select machining parameters based on their imperfect knowledge of the machining system, iteratively running selecting and running cutting tests then using the results of those tests to update the user's beliefs about the system. The framework is highly flexible and can be customized based on the user's goals and prior knowledge to safely select highly productive and efficient machining parameters. The basic Bayesian framework consists of five main steps:

1. Establish prior uncertainties for the frequency response and cutting force model based on any information known about the system

2. Propagate those uncertainties through chosen physics models like stability algorithms to obtain probabilistic predictions about machining cuts.

3. Select and run a cutting test based on the user's goals, for example to improve material removal rate or to reduce uncertainty about the machining system.

4. Use the cutting test results to define a new probability distribution.

5. Draw new samples from the posterior distribution and iterate the test selection/updating process until the user's goals have been met.

Each of these different steps was discussed in detail, showing how the steps interact and how different physics algorithms, test selections, likelihood functions, and distribution samplers can be selected to customize the Bayesian model based on the user's goals. Next, the proposed process was demonstrated, both in a laboratory setting and at an industrial partner. Specific novel demonstrations in these case studies include:

- demonstrating how the FRF prior can be set using physics-based models like RCSA or beam theory, or using a highly uninformed prior,
- showing how additional information from each test cut, like the chatter frequency and spindle power, can be used to help learn more about the system,
- using an uncertainty-reduction test selector to quickly reduce uncertainty on the system natural frequency and help identify more productive cutting parameters, and
- optimizing the machining operations for a realistic aluminum aerostructure, significantly improving productivity and safety versus the cutting tool manufacturer's recommended cutting parameters.

7.2 Future work

This dissertation has focused on establishing the general Bayesian framework and showcasing its possibilities. Future work will focus on extending the framework to address the limitations identified during these case studies and adding new capabilities. One major focus will be on creating fast and accurate machining simulations to replace approximate solutions like the ZOA. One possibility could be implementing scalable versions of existing algorithms, for example by leveraging GPUs to parallelize time domain simulations. Alternatively, surrogate models can be used to approximate the output of expensive physics algorithms using machine learning.

More efficient sampling algorithms will also open up new possibilities. MCMC samplers can be adapted to efficiently handle multimodal distributions and rapid changes in the probability distributions. Transdimensional MCMC can also be employed to allow problems to naturally learn how complex the machining system is, e.g. by automatically extending the number of natural frequencies to better explain the system.

Finally, the Bayesian method will be extended to focus on holistically modelling the entire machining process for a real part, taking into account the complex motion and varying engagement encountered during real part programs. Rather than simply maximizing MRR, cutting tests can be selected to minimize part cost by factoring in the actual part machining time as well as factors like tool wear and cost. Part tolerances can also be used to set constraints on test selection, for example by setting feedrate based on surface finish requirements. Ultimately, this approach will provide a complete solution for statistically modelling machining operations and selecting productive machining parameters for complex processes.

Bibliography

[1] M. D. Packard and B. B. Clark, "Mitigating versus Managing Epistemic and Aleatory Uncertainty," *Academy of Management Review*, vol. 45, pp. 872–876, Oct. 2020. http://journals.aom.org/doi/10.5465/amr.2020.0266. 1

[2] Industrial Base and Sustainability Program, "Annual Industrial Capabilities Report to Congress for Fiscal Year 2020," tech. rep., United States, Jan. 2021. https://www.businessdefense.gov/docs/resources/USA002573-20_ICR_2020_Web.pdf. 2

[3] T. L. Schmitz and K. S. Smith, *Machining Dynamics: Frequency Response to Improved Productivity*. Cham: Springer International Publishing, 2019. https://doi.org/10.1007/978-3-319-93707-6. ix, x, xi, 4, 5, 9, 11, 38, 46, 48, 52, 55, 59, 77, 103

[4] Y. Altintaş and E. Budak, "Analytical Prediction of Stability Lobes in Milling," *CIRP Annals*, vol. 44, no. 1, pp. 357–362, 1995. doi.org/10.1016/S0007-8506(07)62342-7. ix, 7, 44, 48, 49

[5] C. Yue, H. Gao, X. Liu, S. Y. Liang, and L. Wang, "A review of chatter vibration research in milling," *Chinese Journal of Aeronautics*, vol. 32, pp. 215–242, Feb. 2019. https://doi.org/10.1016/j.cja.2018.11.007. 8, 42, 46

[6] L. Zhu and C. Liu, "Recent progress of chatter prediction, detection and suppression in milling," *Mechanical Systems and Signal Processing*, vol. 143, p. 106840, Sept. 2020. https://doi.org/10.1016/j.ymssp.2020.106840. 8, 42, 46

[7] "Manufacturing Laboratories Incorporated." https://mfg-labs.weebly.com/. 8

[8] M. A. Rubeo and T. L. Schmitz, "Mechanistic force model coefficients: A comparison of linear regression and nonlinear optimization," *Precision Engineering*, vol. 45, pp. 311–321, July 2016. https://doi.org/10.1016/j.precisioneng.2016.03.008. 11

[9] S. Aggarwal, N. Nešić, and P. Xirouchakis, "Cutting torque and tangential cutting force coefficient identification from spindle motor current," *The International Journal of Advanced Manufacturing Technology*, vol. 65, pp. 81–95, Mar. 2013. https://doi.org/10.1007/s00170-012-4152-x. 11

[10] D. Aslan and Y. Altintas, "Prediction of Cutting Forces in Five-Axis Milling Using Feed Drive Current Measurements," *IEEE/ASME Transactions on Mechatronics*, vol. 23, pp. 833–844, Apr. 2018. https://doi-org.utk.idm.oclc.org/10.1109/TMECH.2018.2804859. 11

[11] O. Özşahin, H. Özgüven, and E. Budak, "Analysis and compensation of mass loading effect of accelerometers on tool point FRF measurements for chatter stability predictions," *International Journal of Machine Tools and Manufacture*, vol. 50, pp. 585–589, June 2010. doi.org/10.1016/j.ijmachtools.2010.02.002. 14

[12] H. Cao, B. Li, and Z. He, "Chatter stability of milling with speed-varying dynamics of spindles," *International Journal of Machine Tools and Manufacture*, vol. 52, pp. 50–58, Jan. 2012. https://doi.org/10.1016/j.ijmachtools.2011.09.004. 14, 41

[13] J. Berthold, M. Kolouch, V. Wittstock, and M. Putz, "Identification of modal parameters of machine tools during cutting by operational modal analysis," *Procedia CIRP*, vol. 77, pp. 473–476, 2018. https://doi.org/10.1016/j.procir.2018.08.268. 14, 77

[14] Y. Peng, B. Li, X. Mao, H. Liu, C. Qin, and H. He, "A method to obtain the in-process FRF of a machine tool based on operational modal analysis and experiment modal analysis," *The International Journal of Advanced Manufacturing Technology*, vol. 95, pp. 3599–3607, Apr. 2018. https://doi.org/10.1007/s00170-017-1405-8. 14

[15] S. Smith and T. Delio, "Sensor-Based Chatter Detection and Avoidance by Spindle Speed Selection," *Journal of Dynamic Systems, Measurement, and Control*, vol. 114, pp. 486–492, Sept. 1992. https://doi.org/10.1115/1.2897373. ix, 14, 15, 77

[16] Y. Tarng and T. Li, "The change of spindle speed for the avoidance of chatter in end milling," *Journal of Materials Processing Technology*, vol. 41, pp. 227–236, Feb. 1994. https://doi.org/10.1016/0924-0136(94)90063-9. 14

[17] S. Yamato, K. Nakanishi, N. Suzuki, and Y. Kakinuma, "Development of Automatic Chatter Suppression System in Parallel Milling by Real-Time Spindle Speed Control with Observer-Based Chatter Monitoring," *International Journal of Precision Engineering and Manufacturing*, vol. 22, Jan. 2021. https://doi.org/10.1007/s12541-021-00469-2. 14

[18] B. Bergmann and S. Reimer, "Online adaption of milling parameters for a stable and productive process," *CIRP Annals*, vol. 70, no. 1, pp. 341–344, 2021. https://doi.org/10.1016/j.cirp.2021.04.086. ix, 14, 15

[19] N. P. Greis, M. L. Nogueira, S. Bhattacharya, and T. Schmitz, "Physics-Guided Machine Learning for Self-Aware Machining," in *2020 AAAI Spring Symposium on AI and Manufacturing*, 2020. 16

[20] K. Kawai, N. Irino, K. Iiyama, A. Matsubara, H. Mizuyama, K. Mori, and M. Fujishima, "A prediction model for high efficiency machining conditions based on machine

learning," *Procedia CIRP*, vol. 101, pp. 54–57, 2021. https://doi.org/10.1016/j.procir.2020.09.188. 16

[21] J. Friedrich, J. Torzewski, and A. Verl, "Online Learning of Stability Lobe Diagrams in Milling," *Procedia CIRP*, vol. 67, pp. 278–283, 2018. https://doi.org/10.1016/j.procir.2017.12.213. 16

[22] B. Denkena, B. Bergmann, and S. Reimer, "Analysis of different machine learning algorithms to learn stability lobe diagrams," *Procedia CIRP*, vol. 88, pp. 282–287, 2020. doi.org/10.1016/j.procir.2020.05.049. 16

[23] N. P. Greis, M. L. Nogueira, S. Bhattacharya, C. Spooner, and T. Schmitz, "Stability modeling for chatter avoidance in self-aware machining: An application of physics-guided machine learning," *Journal of Intelligent Manufacturing*, vol. 34, pp. 387–413, Jan. 2023. https://doi.org/10.1007/s10845-022-01999-w. 16

[24] I. Oleaga, C. Pardo, J. J. Zulaika, and A. Bustillo, "A machine-learning based solution for chatter prediction in heavy-duty milling machines," *Measurement*, vol. 128, pp. 34–44, Nov. 2018. https://doi.org/10.1016/j.measurement.2018.06.028. 16

[25] M. Postel, B. Bugdayci, and K. Wegener, "Ensemble transfer learning for refining stability predictions in milling using experimental stability states," *The International Journal of Advanced Manufacturing Technology*, vol. 107, pp. 4123–4139, Apr. 2020. https://doi.org/10.1007/s00170-020-05322-w. 16

[26] M. Postel, B. Bugdayci, F. Kuster, and K. Wegener, "Neural network supported inverse parameter identification for stability predictions in milling," *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 71–87, May 2020. https://doi.org/10.1016/j.cirpj.2020.02.004. 16

[27] S. Rezaei, A. Cornelius, J. Karandikar, T. Schmitz, and A. Khojandi, "Using GANs to predict milling stability from limited data," *Journal of Intelligent Manufacturing*, Jan. 2024. doi.org/10.1007/s10845-023-02291-1. 16

[28] C. Deng, J. Tang, S. Lu, Y. Ma, L. Lin, and J. Miao, "Improved milling stability analysis for chatter-free machining parameters planning using a multi-fidelity surrogate model and transfer learning with limited experimental data," *International Journal of Production Research*, vol. 62, pp. 1126–1143, Feb. 2024. https://doi.org/10.1080/00207543.2023.2176698. 17

[29] F. Bartoš, A. Sarafoglou, H. R. Godmann, A. Sahrani, D. K. Leunk, P. Y. Gui, D. Voss, K. Ullah, M. J. Zoubek, F. Nippold, F. Aust, F. F. Vieira, C.-G. Islam, A. J. Zoubek, S. Shabani, J. Petter, I. B. Roos, A. Finnemann, A. B. Lob, M. F. Hoffstadt, J. Nak, J. de Ron, K. Derks, K. Huth, S. Terpstra, T. Bastelica, M. Matetovici, V. L. Ott, A. S. Zetea, K. Karnbach, M. C. Donzallaz, A. John, R. M. Moore, F. Assion, R. van Bork, T. E. Leidinger, X. Zhao, A. K. Motaghi, T. Pan, H. Armstrong, T. Peng, M. Bialas, J. Y. C. Pang, B. Fu, S. Yang, X. Lin, D. Sleiffer, M. Bognar, B. Aczel, and E.-J. Wagenmakers, "Fair coins tend to land on the same side they started: Evidence from

350,757 flips," *arXiv preprint*, 2023. https://doi.org/10.48550/arXiv.2310.04153. 25, 27

[30] J. M. Karandikar, T. L. Schmitz, and A. E. Abbas, "Application of Bayesian Inference to Milling Force Modeling," *Journal of Manufacturing Science and Engineering*, vol. 136, p. 021017, Apr. 2014. doi.org/10.1115/1.4026365. 27, 62

[31] J. Karandikar, A. Chaudhuri, T. No, S. Smith, and T. Schmitz, "Bayesian optimization for inverse calibration of expensive computer models: A case study for Johnson-Cook model in machining," *Manufacturing Letters*, vol. 32, pp. 32–38, Apr. 2022. doi.org/10.1016/j.mfglet.2022.02.001.

[32] Y. Li, W. Mou, J. Li, C. Liu, and J. Gao, "An automatic and accurate method for tool wear inspection using grayscale image probability algorithm based on bayesian inference," *Robotics and Computer-Integrated Manufacturing*, vol. 68, p. 102079, Apr. 2021. https://doi.org/10.1016/j.rcim.2020.102079. 27

[33] J. Karandikar, M. Traverso, A. Abbas, and T. Schmitz, "Bayesian Inference for Milling Stability Using a Random Walk Approach," *Journal of Manufacturing Science and Engineering*, vol. 136, p. 031015, June 2014. doi.org/10.1115/1.4027226. x, 27, 29

[34] J. Karandikar, A. Honeycutt, S. Smith, and T. Schmitz, "Milling stability identification using Bayesian machine learning," *Procedia CIRP*, vol. 93, pp. 1423–1428, 2020. https://doi.org/10.1016/j.procir.2020.04.022. xii, 62, 65

[35] J. Karandikar, A. Honeycutt, T. Schmitz, and S. Smith, "Stability boundary and optimal operating parameter identification in milling using Bayesian learning," *Journal of Manufacturing Processes*, vol. 56, pp. 1252–1262, Aug. 2020. https://doi.org/10.1016/j.jmapro.2020.04.019. xii, 27, 62, 63

[36] K. Li, S. He, H. Liu, X. Mao, B. Li, and B. Luo, "Bayesian uncertainty quantification and propagation for prediction of milling stability lobe," *Mechanical Systems and Signal Processing*, vol. 138, p. 106532, Apr. 2020. doi.org/10.1016/j.ymssp.2019.106532. x, 28, 29, 44, 73, 92

[37] G. Chen, Y. Li, X. Liu, and B. Yang, "Physics-informed Bayesian inference for milling stability analysis," *International Journal of Machine Tools and Manufacture*, vol. 167, p. 103767, Aug. 2021. https://doi.org/10.1016/j.ijmachtools.2021.103767. xi, 28, 46, 47, 77, 83, 96

[38] K. Ahmadi, "Bayesian updating of modal parameters for modeling chatter in turning," *CIRP Journal of Manufacturing Science and Technology*, vol. 38, pp. 724–736, Aug. 2022. doi.org/10.1016/j.cirpj.2022.06.006. 28, 77, 92

[39] V. O. A. Akbari, M. Kuffa, and K. Wegener, "Physics-informed Bayesian machine learning for probabilistic inference and refinement of milling stability predictions," *CIRP Journal of Manufacturing Science and Technology*, vol. 45, pp. 225–239, Oct. 2023. https://doi.org/10.1016/j.cirpj.2023.07.004. xii, 28, 41, 44, 65, 78, 92

[40] A. Cornelius, J. Karandikar, C. Tyler, and T. Schmitz, "Process damping identification using Bayesian learning and time domain simulation," *Journal of Manufacturing Science and Engineering, accepted*, 2024. viii, 35, 48, 73, 82, 97, 103, 108

[41] A. Cornelius, J. Karandikar, M. Gomez, and T. Schmitz, "A Bayesian Framework for Milling Stability Prediction and Reverse Parameter Identification," *Procedia Manufacturing*, vol. 53, pp. 760–772, 2021. https://doi.org/10.1016/j.promfg.2021.06.073. 35, 44, 73

[42] T. T. No, *Digital Cutting Force Modeling for Milling Operations*. PhD thesis, University of Tennessee Knoxville, Aug. 2021. https://trace.tennessee.edu/utk_graddiss/6518/. 35

[43] T. No, M. Gomez, J. Karandikar, J. Heigel, R. Copenhaver, and T. Schmitz, "Propagation of Johnson-Cook flow stress model uncertainty to milling force uncertainty using finite element analysis and time domain simulation," *Procedia Manufacturing*, vol. 53, pp. 223–235, 2021. https://doi.org/10.1016/j.promfg.2021.06.025. 35

[44] R. E. D. Bishop and D. C. Johnson, *The Mechanics of Vibration*. Cambridge: Cambridge Univ. Press, paperback re-issue, 1. paperback ed ed., 2011. 36

[45] T. L. Schmitz and G. S. Duncan, "Three-Component Receptance Coupling Substructure Analysis for Tool Point Dynamics Prediction," *Journal of Manufacturing Science and Engineering*, vol. 127, pp. 781–790, Nov. 2005. https://doi.org/10.1115/1.2039102. 36

[46] U. V. Kumar and T. L. Schmitz, "Spindle dynamics identification for Receptance Coupling Substructure Analysis," *Precision Engineering*, vol. 36, pp. 435–443, July 2012. https://doi.org/10.1016/j.precisioneng.2012.01.007. 39

[47] L. Zhongqun, L. Shuo, and C. Yizhuang, "Receptance Coupling for End Mill Using 2-Section Step Beam Vibration Model," in *2009 Second International Conference on Intelligent Computation Technology and Automation*, (Changsha, Hunan, China), pp. 160–163, IEEE, 2009. https://doi-org.utk.idm.oclc.org/10.1109/ICICTA.2009.276. 41

[48] E. Betters, *A Two-Diameter Helical Endmill Beam Model for Tool Tip Dynamics Prediction with Application to Milling*. PhD thesis, University of Tennessee Knoxville, Dec. 2023. https://trace.tennessee.edu/utk_graddiss/9177/. 41

[49] T. Schmitz, A. Cornelius, J. Karandikar, C. Tyler, and S. Smith, "Receptance coupling substructure analysis and chatter frequency-informed machine learning for milling stability," *CIRP Annals - Manufacturing Technology*, p. S0007850622000257, Apr. 2022. https://doi.org/10.1016/j.cirp.2022.03.020. x, 40, 41, 44, 73, 78, 92, 97

[50] A. Honeycutt, *PERIOD-N BIFURCATIONS IN MILLING*. PhD thesis, University of North Carolina at Charlotte, 2018. http://hdl.handle.net/20.500.13093/etd:1520. 44

[51] C. T. Tyler and T. L. Schmitz, "Analytical process damping stability prediction," *Journal of Manufacturing Processes*, vol. 15, pp. 69–76, Jan. 2013. doi.org/10.1016/j.jmapro.2012.11.006. viii, xiii, 44, 97, 103, 104, 105

[52] S. D. Merdol and Y. Altintas, "Multi Frequency Solution of Chatter Stability for Low Immersion Milling," *Journal of Manufacturing Science and Engineering*, vol. 126, pp. 459–466, Aug. 2004. https://doi.org/10.1115/1.1765139. 44

[53] A. Otto, S. Rauh, S. Ihlenfeldt, and G. Radons, "Stability of milling with non-uniform pitch and variable helix Tools," *The International Journal of Advanced Manufacturing Technology*, vol. 89, pp. 2613–2625, Apr. 2017. https://doi.org/10.1007/s00170-016-9762-2. 44

[54] T. Insperger and G. Stépán, "Updated semi-discretization method for periodic delay-differential equations with discrete delay," *International Journal for Numerical Methods in Engineering*, vol. 61, pp. 117–141, Sept. 2004. doi.org/10.1002/nme.1061. 44

[55] Y. Altintas, G. Stepan, D. Merdol, and Z. Dombovari, "Chatter stability of milling in frequency and discrete time domain," *CIRP Journal of Manufacturing Science and Technology*, vol. 1, no. 1, pp. 35–44, 2008. https://doi.org/10.1016/j.cirpj.2008.06.003. xi, 45

[56] A. Honeycutt and T. L. Schmitz, "A New Metric for Automated Stability Identification in Time Domain Milling Simulation," *Journal of Manufacturing Science and Engineering*, vol. 138, p. 074501, July 2016. https://doi.org/10.1115/1.4032586. 53

[57] Haas Automation, "VF-40t series binder." https://www.haascnc.com/content/dam/haascnc/productivity/spindles/torque-chart-jpg/torque-chart-machine-pdp-binder-/VF-40T. xi, 53, 54

[58] Makino, "Machining Titanium - Part 2 Tool-Bending Moment." https://www.makino.com/getmedia/6c5d8a20-990b-4437-9953-0dda1930d397/Machining-Titanium-Part-2-Bending-Moment-(2021).pdf?ext=.pdf, 2021. xi, 55, 56

[59] JM Performance Products, "High torque retention knobs." https://www.jmperformanceproducts.com/product-information/high-torque-retention-knobs. xi, 56

[60] G. Jemal, *Stress Analysis of Metal Cutting Tools*. PhD thesis, University of British Columbia, 1992. doi.org/10.14288/1.0080910. 57

[61] J. Nemes, S. Asamoah-Attiah, E. Budak, and L. Kops, "Cutting Load Capacity of End Mills with Complex Geometry," *CIRP Annals*, vol. 50, no. 1, pp. 65–68, 2001. https://doi.org/10.1016/S0007-8506(07)62072-1. 57

[62] T. L. Schmitz and B. P. Mann, "Closed-form solutions for surface location error in milling," *International Journal of Machine Tools and Manufacture*, vol. 46, pp. 1369–1377, Oct. 2006. https://doi.org/10.1016/j.ijmachtools.2005.10.007. 59, 60, 72

[63] R. D. Blevins, *Formulas for Natural Frequency and Mode Shape*. Malabar, Florida: Krieger Publishing, reprint ed ed., 2001. 59

[64] J. Karandikar, C. Tyler, and T. Schmitz, "Process damping coefficient identification using Bayesian inference," *Transactions of the North American Manufacturing Research Institution of SME*, vol. 41, pp. 55–65, Jan. 2013. 67

[65] J. M. Karandikar, *THE FUNDAMENTAL APPLICATION OF DECISION ANALYSIS TO MANUFACTURING*. PhD thesis, University of North Carolina at Charlotte, 2013. http://hdl.handle.net/20.500.13093/etd:453. 73, 81

[66] T. Schmitz, "A geometric approach to milling stability uncertainty," *Journal of Manufacturing Processes*, vol. 105, pp. 307–312, Nov. 2023. https://doi.org/10.1016/j.jmapro.2023.09.032. 75

[67] D. Aslan and Y. Altintas, "On-line chatter detection in milling using drive motor current commands extracted from CNC," *International Journal of Machine Tools and Manufacture*, vol. 132, pp. 64–80, Sept. 2018. https://doi.org/10.1016/j.ijmachtools.2018.04.007. xii, 78, 79, 80

[68] Y.-H. Jeong and D.-W. Cho, "Estimating cutting force from rotating and stationary feed motor currents on a milling machine," *International Journal of Machine Tools and Manufacture*, vol. 42, pp. 1559–1566, Nov. 2002. https://doi.org/10.1016/S0890-6955(02)00082-2. 78

[69] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An Introduction to MCMC for Machine Learning," *Machine Learning*, vol. 50, no. 1/2, pp. 5–43, 2003. doi.org/10.1023/A:1020281327116. 84, 92

[70] S. Brooks, ed., *Handbook for Markov Chain Monte Carlo*. Boca Raton: Taylor & Francis, 2011. https://doi.org/10.1201/b10905. 90

[71] A. Gelman, W. R. Gilks, and G. O. Roberts, "Weak convergence and optimal scaling of random walk Metropolis algorithms," *The Annals of Applied Probability*, vol. 7, Feb. 1997. doi.org/10.1214/aoap/1034625254. 90, 94

[72] J. S. Liu, F. Liang, and W. H. Wong, "The Multiple-Try Method and Local Optimization in Metropolis Sampling," *Journal of the American Statistical Association*, vol. 95, pp. 121–134, Mar. 2000. https://doi.org/10.2307/2669532. 92

[73] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo," *Physics Letters B*, vol. 195, pp. 216–222, Sept. 1987. https://doi.org/10.1016/0370-2693(87)91197-X. 92

[74] H. Haario, E. Saksman, and J. Tamminen, "An Adaptive Metropolis Algorithm," *Bernoulli*, vol. 7, p. 223, Apr. 2001. https://doi.org/10.2307/3318737. 92, 94

[75] H. Haario, M. Laine, A. Mira, and E. Saksman, "DRAM: Efficient adaptive MCMC," *Statistics and Computing*, vol. 16, pp. 339–354, Dec. 2006. https://doi.org/10.1007/s11222-006-9438-0. 92

[76] A. Hubin and G. Storvik, "Mode jumping MCMC for Bayesian variable selection in GLMM," *Computational Statistics & Data Analysis*, vol. 127, pp. 281–297, Nov. 2018. https://doi.org/10.1016/j.csda.2018.05.020. 92

[77] J. Goodman and J. Weare, "Ensemble samplers with affine invariance," *Communications in Applied Mathematics and Computational Science*, vol. 5, pp. 65–80, Jan. 2010. http://dx.doi.org/10.2140/camcos.2010.5.65. 92

[78] P. J. Green, N. L. Hjort, and S. Richardson, eds., *Highly Structured Stochastic Systems*. No. 27 in Oxford Statistical Science Series, Oxford ; New York: Oxford University Press, 2003. 92

[79] A. Shahmoradi and F. Bagheri, "ParaDRAM: A Cross-Language Toolbox for Parallel High-Performance Delayed-Rejection Adaptive Metropolis Markov Chain Monte Carlo Simulations." https://doi.org/10.48550/arXiv.2008.09589, Aug. 2020. 93

[80] A. Gelman, D. Lee, and J. Guo, "Stan: A Probabilistic Programming Language for Bayesian Inference and Optimization," *Journal of Educational and Behavioral Statistics*, vol. 40, pp. 530–543, Oct. 2015. https://doi.org/10.3102/1076998615606113. 93

[81] B. Calderhead, "A general construction for parallelizing Metropolis-Hastings algorithms," *Proceedings of the National Academy of Sciences*, vol. 111, pp. 17408–17413, Dec. 2014. https://doi.org/10.1073/pnas.1408184111. 94

[82] M. Li, W. Zhao, L. Li, N. He, and M. Jamil, "Chatter Suppression during Milling of Ti-6Al-4V Based on Variable Pitch Tool and Process Damping Effect," *Machines*, vol. 10, p. 222, Mar. 2022. https://doi.org/10.3390/machines10040222. xiii, 103, 104

[83] Y. Altintas, M. Eynian, and H. Onozuka, "Identification of dynamic cutting force coefficients and chatter stability with process damping," *CIRP Annals*, vol. 57, no. 1, pp. 371–374, 2008. https://doi.org/10.1016/j.cirp.2008.03.048. 106

[84] H. Caliskan, Z. M. Kilic, and Y. Altintas, "On-Line Energy-Based Milling Chatter Detection," *Journal of Manufacturing Science and Engineering*, vol. 140, p. 111012, Nov. 2018. https://doi.org/10.1115/1.4040617. 126

Appendices

A Bayesian coin flip modelling code

This section presents the code for the coin flip example presented in Section 3.1.

```
%% Define the inputs
possibleVals = [0 1]; % tails is 0, heads is 1
weights = [0.5 0.5]; % true value for A
fairRange = [0.49 0.51]; % what range is considered fair
flipCount = 10000;

% Define the prior distribution
prior = @(x) normpdf(x, 0.5, 0.1); % Original distribution
% prior = @(x) normpdf(x, 0.507, 0.0075); % Tight normal
   distribution
% prior = @(x) ones(size(x)); % Uniform distribution

%% Draw samples
flips = randsample(possibleVals,flipCount,true,weights);

%% Update the probabilities
% Non-logarithmic solution
```

```
% posteriorNonNormalized = @(x) x.^(sum(flips==1)) .* (1-x).^(
    sum(flips==0)) .* prior(x);
% normalizationFactor = integral(posteriorNonNormalized,0,1);
% posteriorNormalized = @(x) posteriorNonNormalized(x) ./
    normalizationFactor;

% Logarithmic solution
logPosteriorNonNormalized = @(x) log(x)*sum(flips==1) + log((1-
    x))*sum(flips==0) + log(prior(x));
x = [0:0.001:1];
maxLogProb = max(logPosteriorNonNormalized(x));
posteriorCenteredLog = @(x) exp(logPosteriorNonNormalized(x) -
    maxLogProb);
logNormalizationFactor = integral(posteriorCenteredLog, 0, 1);
posteriorNormalized = @(x) posteriorCenteredLog(x) /
    logNormalizationFactor;

%% Calculate the bias probabilities
pBiasedHeads = integral(posteriorNormalized,fairRange(2),1);
pBiasedTails = integral(posteriorNormalized,0,fairRange(1));
pFair = integral(posteriorNormalized,fairRange(1),fairRange(2))
    ;

%% Create the output figure
figure(1)
clf
hold on
```

```
fplot(prior, [0, 1], 'b')
% fplot(posteriorNonNormalized, [0, 1], 'r:')
fplot(posteriorNormalized, [0, 1], 'r')
xregion(fairRange(1), fairRange(2))

ax = gca;
ax.FontSize = 12;
xlabel('A')
ylabel('P(A)')

strings{1} = strcat("# of coin flip(s): ", num2str(flipCount));
strings{2} = strcat(num2str(sum(flips==1)), " heads (", num2str
   (sum(flips==1)/flipCount*100,3), "%)");
strings{3} = strcat(num2str(sum(flips==0)), " tails (", num2str
   (sum(flips==0)/flipCount*100,3), "%)");
strings{4} = strcat("Fair prob: ", num2str(pFair*100,2), "%");
strings{5} = strcat("Heads biased prob: ", num2str(pBiasedHeads
   *100,2), "%");
strings{6} = strcat("Tails biased prob: ", num2str(pBiasedTails
   *100,2), "%");
annotation("textbox",[0.15, 0.65, 0.3, 0.27], String=strings,
   FontSize=10, EdgeColor='none')
% annotation("textarrow", [0.7 0.5], [0.7 0.6], String="Fair
   region", FontSize=10)
legend(["Prior" "Posterior"], FontSize=10)
print("-dpng", "coinflipDemo", "-r600") % Save the figure
```

B MCMC demonstration code

This section presents the code for the coin flip example presented in Section 4.5.4.

```
targetDist = @(x) normpdf(x, 1, 1);
proposalStdDev = 1;
startPosition = 0;

outputChain = startPosition;
chainLength = 500;

% Plotting options
plotRejected = true;
plotNextCandidate = false;
plotProposalCurve = false;
displayNextMoveStats = false;
displayAcceptanceRate = true;
displayUniqueSampleCount = true;

figure(1)
clf
hold on

acceptedCount = 0;
for i = 1:chainLength
    cand = randn(1) * proposalStdDev + outputChain(i);
    a = targetDist(cand) / targetDist(outputChain(i));

```

```
25      u = rand();
26      if a >= u
27          acceptLast = true;
28          outputChain(i+1) = cand;
29          acceptedCount = acceptedCount + 1;
30      else
31          acceptLast = false;
32          outputChain(i+1) = outputChain(i);
33          if plotRejected % Optionally plot any rejected
               transitions
34              plot([i i+1], [outputChain(i) cand], 'r:')
35          end
36      end
37  end
38
39  displayStrings = {};
40
41  % Generate a new candidate
42  if plotNextCandidate
43      cand = randn(1) * proposalStdDev + outputChain(end);
44      plot([chainLength+1 chainLength + 2], [outputChain(end)
           cand], ':', Color=[0 0.447 0.741])
45      plot(chainLength+2, cand, 'o', MarkerEdgeColor=[0 0.447
           0.741])
46      if displayNextMoveStats
```

```
47          displayStrings{end+1} = strcat("Proposed move: ",
               num2str(outputChain(end),3), " -> ", num2str(cand,
               3));
48          displayStrings{end+1} = strcat("Prob change: ", num2str
               (targetDist(outputChain(end)),3), " -> ", num2str(
               targetDist(cand), 3));
49          u = rand(1);
50          acceptProb = min(1, targetDist(cand)/targetDist(
               outputChain(end)));
51          displayStrings{end+1} = strcat("Acceptance probability:
                ", num2str(acceptProb*100, 3), "%");
52          displayStrings{end+1} = strcat("u: ", num2str(u, 3));
53          if u <= acceptProb
54              displayStrings{end+1} = strcat("u <= a. Accept");
55          else
56              displayStrings{end+1} = strcat("u > a. Reject");
57          end
58      end
59  end
60
61  % plot completed chain
62  plot(outputChain, 'bo-')
63  xlabel('i')
64  ylabel('x_i')
65  ylim([-10 10])
66  if chainLength ~= 0
67      xlim([1, ceil(chainLength * 1.3)+2])
```

```
68 else
69     xlim([1, 3])
70 end
71
72 % Adjust axes
73 set(gca, 'Units', 'normalized')
74 axpos = get(gca, 'Position');
75 axpos(3) = 0.6;
76 axpos(2) = axpos(2) + 0.03;
77 set(gca, 'Position', axpos)
78 ax = gca;
79 ax.FontSize = 12;
80
81
82 % Plot acceptance rate
83     textSize = [0.36 0.05];
84     textPos = [(axpos(3:4) + axpos(1:2) - textSize) textSize];
85     ta = annotation('textbox', textPos, 'String',
          displayStrings, EdgeColor='none', FontSize=10);
86
87 if displayAcceptanceRate
88     acceptRate{1} = strcat("Acceptance rate: ", num2str(100*
          acceptedCount / (chainLength),3), "%");
89     textSize = [0.28 0.08];
90     textPos = [(axpos(3:4) + axpos(1:2) - textSize) textSize];
91     textPos(2) = axpos(2) + 0.03;
92     if displayUniqueSampleCount
```

```
93          acceptRate{2} = strcat("Unique values: ", num2str(
                length(unique(outputChain))));
94      end
95      annotation('textbox', textPos, 'String', acceptRate,
            EdgeColor='none', FontSize=10);
96
97  end
98
99  % Plot the proposal histogram
100 xyPoint = [length(outputChain), outputChain(end)];
101 xl = xlim();
102 yl = ylim();
103
104 xyNorm = (xyPoint - [xl(1),yl(1)]) ./[range(xl),range(yl)];
105 xyFigNorm = axpos(1:2) + axpos(3:4).*xyNorm;
106
107 axSize = [0.1, 0.4];
108 xyFigNorm(2) = xyFigNorm(2) - axSize(2)/2;
109
110 ax2 = axes(Units="normalized", Position=[xyFigNorm, axSize],
        Color='none');
111 hold on
112 if plotProposalCurve
113     fplot(@(t) normpdf(t, 0, proposalStdDev), @(t) t, [-5 5], '
            g')
114 end
115 totalRange = range(yl) * axSize(2) / axpos(4);
```

```
116 ylim([-totalRange totalRange]/2)
117 hA = gca;
118 set(hA, 'XTick', [], 'XTickLabel', []);
119 set(hA, 'YTick', [], 'YTickLabel', []);
120 set(get(hA, 'XAxis'), 'Visible', 'off');
121 set(get(hA, 'YAxis'), 'Visible', 'off');
122
123
124 ax3pos = axpos;
125 ax3pos(1) = ax3pos(1) + ax3pos(3) + 0.05;
126 ax3pos(3) = 1 - ax3pos(1) - 0.05;
127
128 ax3 = axes(Units="normalized", Position=ax3pos);
129 hold on
130 histogram(outputChain, Normalization="pdf", Orientation="
      horizontal")
131 fplot(targetDist, @(t) t, yl, 'r')
132 xlim([0 0.5])
133 ylim(yl)
134 xlabel('P(x)')
135 ax3.FontSize = 12;
136 set(ax3, 'YTick', [], 'YTickLabel', []);
137
138 print("-dpng", "mcmcDemo", "-r600")
```

C Bayesian framework code guide

Code for laboratory case studies 3 and 4 is included along with this dissertation on UTK's Tennessee Research and Creative Exchange (TRACE) open-access repository, helping demonstrate the framework shown in this dissertation and providing a basis for future work. The code is provided under the MIT license, and includes no warranty or liability.

This appendix walks through the provided Matlab code for the Bayesian framework in order to demonstrate how it functions. This framework is generally aimed at being flexible and easy to extend, rather than aiming for maximum performance. For example, samples are stored in `structs` which contain all relevant information about them, such as the calculated stability maps and likelihoods. This is convenient for programming, but also adds memory and computational overhead compared to storing, e.g., a single matrix with all predicted stability limits.

Note that this code uses anonymous functions to make the code generalizable. Anonymous functions define a new function inside of the code which can be called later. For example, `increment = @(x)x+1` defines a new function which can be called later in the code as `increment(5)` to return the value `6`. Consult Matlab's documentation for more information on anonymous functions.

C.1 SimulationSettings.m

This subsection walks through the SimulationSettings.m file. This file contains all information necessary to customize the simulation for a specific application. Ideally, this should be the only file that gets edited to configure the Bayesian framework.

The `param` struct contains common information on the simulation which is shared between different stages of the calculation. First, the cutting tool is defined.

```
11 % Tool
12 param.ToolDiameter = 0.25*25.4e-3; % Tool diameter, meters
```

```
13 param.FluteCount = 3; % Flute count
14 param.HelixAngle = 0; % Flute helix angle, radians
```

Next, the range of cutting parameters to simulate is defined as a large 5-dimensional grid. `param.CutParamGrid` contains separate grids for spindle speed, radial and axial depth, feed per flute, and cut direction (i.e., climb vs conventional cutting). These can be indexed as, for example, `param.CutParamGrid.rpm(rpmInd, aInd, bInd, fzInd, cutDirInd)`. Note that there may be bugs when running the code with multiple radial widths, feedrates, and cut directions.

```
16 % Cutting parameters
17 param.RadialWidth = param.ToolDiameter * [0.5]; % Radial widths
       of cut, meters
18 assert(all(param.ToolDiameter >= param.RadialWidth), 'Radial
   width of cut must be smaller than tool diameter')
19 param.CutDirection = [2]; % Feed directions. 1 = conventional
   milling, 2 = climb milling
20 param.FeedPerTooth = 0.001*25.4e-3; % Feed rate per flute,
   meters
21 param.Rpms = linspace(4000, 8100, 200); % List of spindle speed
   , rpm
22 param.Aps = linspace(0e-3, 0.5e-3, 100);
23 param.CutParamGrid = cuttingParameterGrid(... % Defines the
   grid of parameters using full-factorial combinations
24     param.Rpms, ... % Spindle speeds
25     param.RadialWidth, ... % Radial width
26     param.Aps, ... % Axial depth
27     param.FeedPerTooth, ... % Feed per flute
28     param.CutDirection); % Cut direction
```

The prior distribution is defined using the `Distribution_MVN_Truncated class`, which defines a variety of different methods for drawing samples and evaluating the PDF and log PDF of the distribution. There are also classes available for single-dimensional normal, exponential, uniform, and mixed distributions. The `zeros(...)` and `realmax(...)` inputs define the lower and upper bounds for the truncated distribution respectively.

```
40 %% Define the prior and true values
41 % [ks (N/m^2), beta (rad), fn (Hz), k (N/m), zeta]
42 priorMeans = [600e6 deg2rad(68) 950 5e5 0.015];
43 priorStdDevs = priorMeans .* [0.2 0.1 0.1 0.2 0.2]; % Define
       prior stdDevs as a percentage of the mean
44 priorCov = diag(priorStdDevs.^2);
45 priorDist = Distribution_MVN_Truncated(priorMeans, priorCov,
       zeros(size(priorMeans)), realmax*ones(size(priorMeans)));
```

The `sampleParser` class helps interpret and convert between different variables regardless of how they're presented in the $\boldsymbol{\theta}$ vector: for example, a physics function can request k_{tc} for a given sample and the sample parser will automatically convert that value from K_s and β, regardless of where those values are positioned.

```
40 parser = sampleParser('KsBeta', [1 2], 'fnKZeta', [3 5], ["ks"
       "beta" "fn" "k" "zeta"]);
```

Next, the physics functions are defined. The `GenericPhysicsDelegate` function evaluates a range of different physics functions for a given sample and saves the requested outputs: in this case, `Stability` and `ChatterFrequency`. Each of the physics functions can have different options injected here, like the `FrequencyRange='auto'` argument: see the specific function .m files for more information on these options. These functions can also draw on the output from previous physics functions. Here, the `frfFromSymmetricModesSub` function first calculates the frequency response function which

is then used by `ZOA_StabilityLikelihoodFunc` to calculate the stability map. However, the FRF is not retained to reduce memory usage: adding `"FrequenciesFRF"` and `"FRFxx"` to the last argument for `GenericPhysicsDelegate` would also save the frequency vector and X-axis FRF for each sample. Note that the frequencies may vary from sample to sample because the `FrequencyRange` is set to `'auto'` mode: if necessary, a predefined range of frequencies can be specified instead. See each different physics function for a list of possible outputs and options they offer.

```
% Physics algorithms
physicsFuncs = {...
    @(sample) physicsFuncFrfFromSymmetricModes(sample, parser,
        FrequencyRange='auto'), ...
    @(sample, previousCalcs) physicsFunZOA_StabilityLikelihood(
        sample, parser, previousCalcs, param, RpmIncrement1D =
        10, StabilityUncertainty='inParam', AssumeSymmetric=true
        , ReduceSymmetricFRF=true)};
physicsDelegate = @(sample) GenericPhysicsDelegate(sample,
    physicsFuncs, ["Stability" "ChatterFrequency"]);
clear physicsFuncs
```

Cutting test are selected by calling the `testSelector` function. This example uses the `switchingTestSelector` function, which takes in a list of different test selectors and allows the user to pick which one they'd like to use or manually select a test. It also applies constraints on the tests, for example that the risk of chattering must be between 0% and 50%. In this case, the uncertainty reduction test selector is told to ignore the safety constraints in order to observe chatter and the chatter frequency as quickly as possible.

```
% Algorithm to select the next test
testSelectors = {
```

```
65      @(iteration, sampledVals, testResults, acceptableTests)
            testSelectUncertaintyReduction( ...
66          iteration, ...
67          sampledVals, ...
68          testResults, ...
69          acceptableTests, ...
70          param, ...
71          parser, ...
72          ChatterFreqUncertainty = 20, SampleCountToUse=500,
                OverrideConstraints = true), ...
73      @(iteration, sampledVals, testResults, acceptableTests)
            testSelectMaximizeMRR( ...
74          iteration, ...
75          sampledVals, ...
76          testResults, ...
77          acceptableTests, ...
78          param, ...
79          ChatterRiskTolerance = 0.5)};
80  selectorNames = ["Uncertainty reduction" "Maximize MRR"];
81  constraints = @(samples) testConstraints(samples, param); % Can
        constrain test selection based on things like spindle power
        or bending moment, not used here.
82
83  testSelector = @(iteration, sampledVals, testResults) ...
84      testSelectorSwitching(iteration, sampledVals, testResults,
            testSelectors, selectorNames, param, constraints);
```

Next, the test classifier is defined. This function evaluates the result of the selected cutting test. This example uses `manualTestClassifier`, which prompts the user to run the test and input the results manually.

```
% Algorithm to label tests as stable or unstable
testClassifier = @(iteration, test) manualTestClassifier(
    iteration, test, param);
```

Next, the likelihood function is defined. This function takes in a single $\boldsymbol{\theta}$ vector of inputs (in this case, the vector `[k_s, beta, f_n, k, zeta]`) and calculates the likelihood. In order to save time, the output is not simply a single likelihood value but a `struct` which contains all relevant information about the sample, including the $\boldsymbol{\theta}$ values (`sample.Values`), the log probability (`sample.LogProbability`), any calculated information like the stability map (contained in a sub`struct`, e.g. `sample.CalculatedValues.Stability`), and other diagnostic info (`sample.Diagnostics`). If the likelihood function is called on a `struct` sample which already has all of this calculated information, it will update the probability but will not rerun the physics algorithms. In this example, the stability likelihood is precalculated in the ZOA physics function rather than getting specified here.

```
% PDF formula. The prior is not injected here because it's not
    used while calculating updated probabilities for the old
    points
pdfFormula = @(sample, prior, testedPoints)
    acceptanceFormulaLikelihood(sample, prior, testedPoints, ...
     physicsDelegate, ... % Algorithm to generate the full
        stability maps
     param, ... % General parameters
     chatterFreqStdDev = 20, ... % Standard deviation for
        chatter frequency agreement, Hz), ...
```

```
94      StabilityUncertainty='precalculated'); % Whether the PDF
          should learn from cuts that don't break the tool
```

This section defines the sampling algorithms. Two different algorithms are defined. The `initialSampleGenerator` is used to draw samples from the prior distribution, while `updateSampleGenerator` samples the posterior. In addition to the information described above in the likelihood section, each sample will also have a weight assigned to it (`sample.Weight`).

```
96  % Sample generators
97  sampleCount = 4000;
98  initialSampleGenerator = @() initialSampleGenerator(priorDist,
      pdfFormula, SampleCount = sampleCount, Parallelize = true);
99  updateSampleGenerator = @(priorSamples, newTestResults,
      oldTestResults) mcmcSampleGenerator(priorSamples,
      newTestResults, oldTestResults, priorDist, pdfFormula,
      SampleCount = sampleCount);
100 clear sampleCount
```

Finally, the user defines what figures they want generated. Here, it plots the probabilistic stability map, a scatter plot showing all the samples, and a plot of the chatter frequencies.

```
102 % Figure plotting
103 plotFuncs = {@PlotProbStabilityMap, @PlotMatrixCorrelations, @
      chatterFrequencyPlot};
104 plotFuncNames = ["Probability of stability" "Variable
      plotmatrix" "Chatter frequencies"];
105 figureGeneration = @(iteration, sampledVals, nominalVals,
      testResults) PlotFiguresSub(plotFuncs, plotFuncNames,
      iteration, sampledVals, nominalVals, testResults, param);
```

C.2 StartSimulation.m

This subsection walks through the `StartSimulation.m` file. This is the file that the user executes to begin running the Bayesian model once they've fully configured the simulation settings. It first clears the existing workspace, adding the `Functions` folder to the path, and creating the output `Figures` folder.

```
%% Clear all data to start with blank slate
clear
close all

%% Add the function source code folder to the path
addpath("Functions");

%% Create output directory
if not(isfolder('Figures'))
    mkdir('Figures')
end
```

The `taskList` object keeps track of what the simulation is currently doing and outputs the current state to the console. It also provides functions for displaying waitbars in the console with calculated ETA, including for parallelized calculations. This is created as a global variable so that other functions can access it and add what they're doing to the list.

```
%% Define the time tracker
global taskList;
taskList = TaskList();
```

It runs the `SimulationSettings` file to initialize the current simulation.

```
%% Load the simulation parameters
SimulationSettings();
```

The nominal value is run through the physics functions to calculate the nominal stability map and other values, and then the initial set of samples is drawn from the prior distribution.

```
%% Generate starting stability maps
% Calculate the nominal values
nominalVals = pdfFormula(trueStabilityMapVals, priorDist, []);
taskList.Add("Generating initial samples");
sampledVals = initialSampleGenerator();
```

Finally, the list of test results is initialized. Each test is stored as a `struct` with all relevant information about the test. The `CutParamInd` saves which index on the `CutParamGrid` this test is associated with. The starting state is saved to a `.mat` file in the Figures folder, and then the program calls the `ResumeSimulation.m` file to begin the Bayesian updating.

```
%% Initialize tested point array. This may need to get
   customized depending on what info is saved for each test.
taskList.Update('Save initial data');
testResults = struct('RPM', {}, 'b', {}, 'a', {}, 'fz', {}, '
   MRR', {}, 'CutDirection', {}, 'Stable', {}, '
   ChatterFrequency', {}, 'Broken', {}, 'CutParamInd', {});
iteration = 0;
chatterFrequency = 0;
save('Figures\\StartingData')

taskList.Remove;
taskList.Remove;

ResumeSimulation();
```

C.3 ResumeSimulation.m

This subsection walks through the `ResumeSimulation.m` file. This file is responsible for performing the main updating loop. If the user wishes to restart the simulation, they can load a saved state and directly execute this file to start from where they left off.

The main updating loop is run a user-selected number of times. During each loop, it selects, classifies, and updates based on a new test point. Figures are output at the end of each loop: for the initial iteration, it skips adding a new test point and just outputs the figures. Lines 37–49 save the test data to the Figures folder to allow it to be restarted later. The user can specify in the simulation settings whether they want to save the physics model calculations or not to save harddrive space. If the physics values are not saved, they will automatically be recalculated by the `CalculateSamples` function on line 16 when this file is executed.

```
4  global taskList;
5  taskList.Clear();
6  taskList.Add(testName);
7  taskList.Add('Performing Bayesian updating');
8  taskList.Add('Iteration 0');
9
10 %%
11 for iteration = iteration:param.MaxIterations
12     taskList.Update(['Iteration ' num2str(iteration) ' / '
           num2str(param.MaxIterations)]);
13     taskList.Add('Starting iteration');
14
15     %% Calculate the physics models for all the samples if
           loading from data
```

```
16      sampledVals = CalculateSamples(sampledVals, physicsDelegate
            );
17
18      % The first run-through with iteration=0 does not add a
            test point
19      if iteration ~= 0
20          % Select test
21          taskList.Update('Selecting test point');
22          recommendedTest = testSelector(iteration, sampledVals,
                testResults);
23
24          % Classify test
25          taskList.Update('Classifying the test point');
26          testResults(end+1) = testClassifier(iteration,
                recommendedTest);
27
28          % Draw new samples
29          taskList.Update('Sampling new distribution');
30          sampledVals = updateSampleGenerator(sampledVals,
                testResults(end), testResults);
31          end
32
33      %% Save figures and data
34      taskList.Update('Saving figures');
35      figureGeneration(iteration, sampledVals, nominalVals,
            testResults);
36
```

```
37      % Save data
38      iteration = iteration+1; % Increment the index so that
           restarting from the saved file will start properly
39      taskList.Update('Saving current data');
40      if param.SaveCalculatedData
41          save(['Figures\\' num2str(iteration-1)]);
42      else % Remove the CalculatedValues field to save space.
43          save(['Figures\\' num2str(iteration-1)], '-regexp', ['
               ^(?!', 'sampledVals','$).']); % The calculated
               values on the samples are omitted from saving to
               save file space
44          sampledValsTemp = sampledVals;
45          sampledVals = rmfield(sampledVals, 'CalculatedValues');
46          save(['Figures\\' num2str(iteration-1)], 'sampledVals',
                '-append');
47          sampledVals = sampledValsTemp;
48          clear sampledValsTemp
49      end
50
51      taskList.Remove;
52  end
```

Vita

Aaron Cornelius is originally from Menlo Park, CA, and earned a B.S. in Mechanical Engineering from Worcester Polytechnic Institute in 2016. After this, he worked as an applications engineer for a manufacturing company until 2019, helping set up and optimize production machining cells and assisting with metalcutting research. He started graduate school at the University of Tennessee, Knoxville in 2020 with Dr. Tony Schmitz. His research has focused on statistical Bayesian modelling machining operations to help select more productive machining parameters and how structured light scanning can be used to help facilitate machining of additively manufactured preforms.