Real-Time Energy Prediction for a Milling Machine Tool Using Sparse Gaussian Process Regression

Jinkyoo Park and Kincho H. Law
Civil and Environmental Engineering
Stanford University
Stanford, CA, USA
{Jkpark11, law}@stanford.edu

Raunak Bhinge, Mason Chen, and David Dornfeld
Mechanical Engineering
University of California, Berkeley
Berkeley, CA, USA
{raunakbh, mas.chen, dornfeld}@berkeley.edu

Sudarsan Rachuri
Systems Integration Division
National Institute of Standards and Technology
Gaithersburg, MD, USA
sudarsan.rachuri@nist.gov

***Abstract*— This paper describes a real-time data collection framework and an adaptive machining learning method for constructing a real-time energy prediction model for a machine tool. To effectively establish the energy consumption pattern of a machine tool over time, the energy prediction model is continuously updated with new measurement data to account for time-varying effects of the machine tool, such as tool wear and machine tool deterioration. In this work, a real-time data collection and processing framework is developed to retrieve raw data from a milling machine tool and its sensors and convert them into relevant input features. The extracted input features are then used to construct the energy prediction model using Gaussian Process (GP) regression. To update the GP regression model with real-time streaming data, we investigate the use of sparse representation of the covariance matrix to reduce the computational and storage demands of the GP regression. We compare computational efficiency of sparse GP to that of full GP regression model and show the effectiveness of the sparse GP regression model for tracking the variation in the energy consumption pattern of the target machine.**

Keywords-machine learning; Sparse Gaussian regression; energy prediction model; milling machine; MTConnect.

I. INTRODUCTION

Reliable energy prediction for machine tools can have many benefits in manufacturing, such as providing feedback to improve machine tool energy efficiency, energy-efficient process planning and real-time monitoring. With the advent of data acquisition and interoperability standards such as MTConnect [1], it is possible to collect real-time data from a machine tool controller and the instrumented sensors with a common timestamp. The synchronized data between the actual machine operations and the cutting simulation, allows the development of casual relationships between the machining parameters and the energy consumption pattern. To effectively characterize the energy consumption pattern of a machine tool over time, two aspects need to be investigated: (1) a data acquisition system that collects raw data and rapidly processes them to extract the input features relevant to the energy prediction model in real time, and (2) an energy prediction model that can be adaptively updated with newly measured data to account for the most recent energy consumption pattern of the tool.

The data collection capability through the MTConnect standard has helped researchers to investigate the energy consumption pattern of a machine tool using the data measured from the machine. For example, Vijayaraghavan and Dornfeld have tracked the variations in energy usage by different machining operations [2]. Diaz et al. have developed an empirical model to predict energy for a face milling operation [3]. Using a novel block-wise experimentation and data-condensation technique, we have employed Gaussian Process (GP) regression to construct a generalized energy prediction model that can estimate the energy consumption of the target machine for different machine operations and combinations of operational parameters [4-5]. Although data-driven energy prediction models have shown their potential to reliably represent the energy consumption pattern of a target machine tool, the energy prediction models are constructed offline in that the energy consumption and machining parameters are retrieved and processed prior to applying the GP regression analysis. In this study, we investigate an adaptive approach that can be potentially used to construct the energy prediction model in real time.

For offline data processing, the entire raw data retrieved from the target machine is saved before extracting the features that are useful to characterize the energy consumption pattern of the target machine. This approach not only requires large data storage, but also prevents adapting continuously the energy prediction model in real time. To handle the large volume of manufacturing process data, a real-time data collection, condensation and categorization system that can be linked with an adaptive

energy prediction model is desirable. In this study, we propose a real-time data-collection and processing framework that uses three data buffers for raw data, derived data and simulated data, respectively. The computational algorithm and simulator that are built in the framework sequentially process the data stored in the buffers to produce the information-condensed features that can readily be used to update the energy prediction model. The framework is demonstrated by showing how the energy consumption data is collected from the machine at the shop factory and processed to predict the energy consumption in real time at a remote location.

Continuously updating a predictive model with streaming data is challenging, especially for non-parametric models that require retaining the training data for prediction. Particularly, when a GP regression model is employed, the training phase requires a large data set and is computationally expensive because the GP regression model requires the inverse of the covariance matrix to optimize the hyper-parameters and to predict the target response. For N training data points, the storage requirement for the covariance matrix is $O(N^2)$ and the computational cost for inverting the covariance matrix is $O(N^3)$. To employ GP regression for constructing a prediction model in real time, it is necessary to reduce the computational cost and storage demands for training the model and to regulate the size of training data by retaining only the most informative data. To reduce the computational and storage demands, two methods have been proposed in the literature. The first approach is to construct a regression model by combining local GP regression models, each of which is constructed using only a certain portion of the training data [6-10]. The approach of a local GP model has been employed to construct an energy prediction model of a milling machine [4]. Although this approach reduces the computational time significantly, training each local GP regression model will eventually suffer from the complexity issues as the size of the data used for each local GP increases. The second approach is to approximate the covariance matrix, which contains the correlations between every pair of the training input, by a product of row-rank matrices [11-15] and is referred to as sparse GP regression. Due to the sparse representation for the covariance matrix, the computational time and storage requirement for this approach is much smaller than those of full GP regression without approximation. In this study, we explore the use of sparse GP regression to reduce the training time so that the global GP regression model can be efficiently updated as the newly measured training data set is included. We also study the effect of regulating the size of the training input on the accuracy of the prediction model.

This paper is organized as follows: First, the experimental design and the system framework for real time data acquisition and processing are presented. Then, the sparse GP regression method is briefly introduced. Using the collected data from a milling machine, the sparse GP regression model is constructed and its prediction accuracy and the computational cost are compared to those of the full GP model. The paper is concluded with a brief summary and discussion.

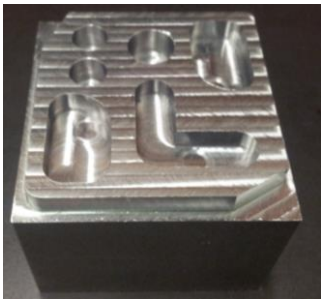


Figure 1. Test part design for experimentation [4].

TABLE I. EXPERIMENTAL DETAILS

Work piece Material	Cold Finish Mild Steel 1018
Work piece Dimensions	63.5mm x 63.5mm square cut to a length of 56mm
Machine Make	Mori Seiki NVD 1500
Machine Type	Micro NC Milling Machine
Tool Material	Solid Carbide
Tool Diameter	3/8" (9.525 mm)

II. EXPERIMENTAL DESIGN AND DATA PROCESSING

With recent advances in sensing and data management technologies, the energy consumption corresponding to each NC (Numerical Control) code block (a line of G-code is commonly called a "Block") and its corresponding operational machine parameters can be measured in real time and efficiently processed and retrieved remotely [4-5]. This motivates us to design novel experiments to collect the data that can realistically represent the operation by a machine tool. In this section, we briefly describe the experimental set up for collecting the training data set. We then discuss the real time data collection process and the real time conversion of raw data into input features.

A. Experimental Setup

Fig.1 shows the part designed to collect the training data for constructing the data-driven energy prediction model. The part is machined using different machine operations: cutting with feed, air cut and auxiliary operations such as rapid tool motion. The cutting with feed operation includes face milling, contouring, pocketing, slotting and plunge. To machine the part, a set of machine operations are required, each of which is described by NC code control input. The associated energy consumption for each operation is measured with the NC code input. A total of 18 parts are machined with 12,299 NC blocks, which serve as the training data to build the energy prediction model. Details of the experiment are listed in Table 1. The detailed information about test set up has been previously described in [4].

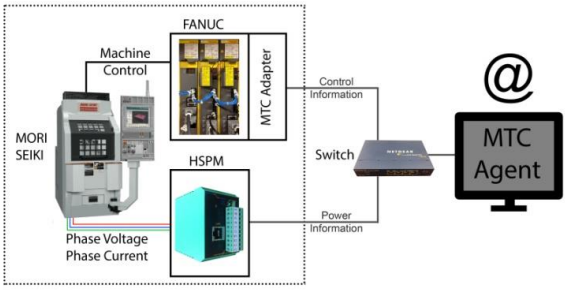


Figure 2. Data collection and processing procedures [4].

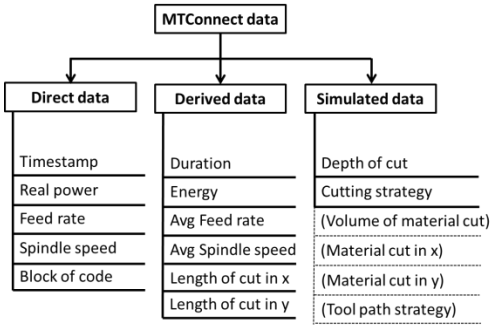


Figure 3. Classified data type.

B. Data Collection

Fig. 2 describes the overall hardware framework designed to collect the contextualized energy consumption data with machining parameters for constructing the data-driven energy prediction model. The machining process data including the process parameters, NC blocks and tool positions are collected from the FANUC controller and the power time series is collected from the System Insights High Speed Power Meter. The machining parameters and the power time series are synchronously collected through the MTConnect Agent. The hardware setup for data collection has been described in [4-5].

C. Real-Time Data Processing Architecture

Depending on the level of post processing, as shown in Fig. 3, the retrieved data from the target machine is categorized into three groups, namely direct data, derived data and simulated data. The direct data is measured from the machine in a form of time series with 100 Hz sampling rate. The derived data is obtained through statistical analyses and simple calculations with the direct data. Finally, simulated data is obtained by simulating the entire cutting process with an initial geometry. The derived and simulated data can effectively (or compactly) describe the cutting operation in the corresponding NC code block, serving as input features for the data-driven energy prediction model in this study. The detailed procedures of data collection and cutting simulation are described in [4-5].

A real-time data collection and processing system should effectively process the data stream collected from the MTConnect agent. Fig. 4 shows the real-time data-collection and processing architecture (currently, this architecture is implemented on local computer based on Python interface code) designed to support the adaptive data driven model. As shown in the figure, there are three buffers between the data steam from MTConnect and the prediction model. The first buffer stores the direct data from the MTConnect client using the IP Address of the MTConnect agent. The direct data stored in the first buffer is sequentially converted to derived data through simple statistical computations, which is then stored by blocks in the second buffer. Using the derived data that sequentially arrives at the second buffer, the cutting simulation proceeds to generate simulated data which cannot be calculated from the data stream directly. This condensed and categorized block-by-block information is finally stored into the third buffer, which is used by the adaptive prediction model. Using the stored data in the last buffer, the adaptive prediction model updates its model to account for time-varying characteristics in the target machine. For all three buffers, the buffer sizes vary depending on the time duration of the blocks. The three-level data stream processing architecture works well for the MTConnect data stream, especially since each module has highly varying run-times and data structures.

D. Demonstration of Data Collection and Processing

The real-time data collection and processing framework has been demonstrated in the O'Reilly Solid Conference on Hardware, Software and the Internet of Things, held from June 23, 2015 to June 25, 2015 in San Francisco. Fig. 5(a) shows the demonstration booth sponsored by AMT. At the conference, we showed in real time how the target milling machine tool was being operated in the machine shop located at UC Berkeley while at the same time, the data acquisition and processing agent and energy prediction procedure were performed at San Francisco (Solid Conference Site) (see Fig. 5(b)). Whenever the retrieved and processed data for a NC code block is received at the last data buffer, the energy prediction function estimates the energy consumption for that NC code block in real time. The predicted and the measured energy consumptions are then shown simultaneously to illustrate the capability of the energy prediction model for monitoring as shown in Fig. 5(c) and Fig. 5(d). For every single cut, the prediction window, enlarged as shown in Fig. 5(d), shows the monitored loads on the tool axes, predicted energy values and the constructed energy density function.

The energy prediction function used during the demonstration was pre-constructed using GP regression with the data collected from the 18 test parts [5]. During the demonstration, we focus on showing how the raw data from the target machine is converted into the input features and how the input features are used to predict the energy consumption for every single machine operation in real time. The demonstration shows that the proposed data

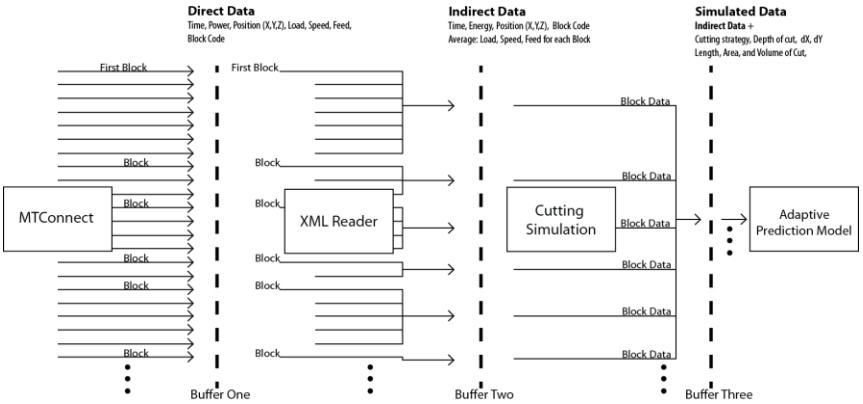


Figure 4. Real-time data processing architecture

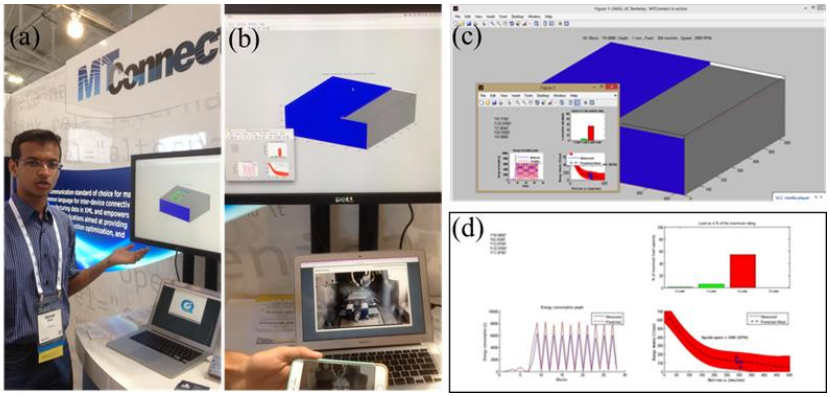


Figure 5. Demonstration of real-time data collection and monitoring with the real-time energy prediction.

acquisition and processing framework enables the energy prediction model to predict the energy consumption for every single operation of the target machine in real time.

The energy prediction model can be updated whenever new data is appended to the data buffer. The updating frequency can be determined based on the computational requirements and the level of responsiveness that the prediction model is required. For example, if the target system requires real time control, updating the model with every single data point would be beneficial. However, for monitoring and predicting applications, updating the prediction model with a bulk of new data set would be sufficient. One issue is that as the measurement data increases, the GP regression can become computationally expensive. In the following section, we focus on discussing a methodology to reduce the computational and storage requirements for updating the GP regression model to greatly facilitate the updating the energy prediction model in real time.

III. Sparse Gaussian Process Regression

To continuously track the energy consumption pattern of a target machine, it is imperative for the energy prediction model for the target machine to be updated with new measurement data. Although GP regression, as a non-parametric model, can model complex input and output relationships using a small number of hyper-parameters and the data itself as basis functions, constructing a GP regression model is computationally demanding and requires significant storage as the number of training input increases. To reduce the computational and storage requirements, approximated GP regression approaches, such as sparse GP [11-15], have been proposed. In this section, we briefly introduce the sparse GP method.

A. Gaussian Process (GP) Regression

GP is a collection of random variables (stochastic process), any finite number of which has a joint Gaussian distribution [9]. GP can effectively represent the distribution over functions by constructing a joint distribution on the function values. Assume that we can observe a noisy response $y^i = f(\boldsymbol{x}^i) + \epsilon^i$ corresponding to the input $\boldsymbol{x}^i$. Using the N -noisy responses $\boldsymbol{y} = (y^1, \dots, y^i, \dots, y^N)$ corresponding to the N-inputs $\boldsymbol{X} = (\boldsymbol{x}^1, \dots, \boldsymbol{x}^i, \dots, \boldsymbol{x}^N)$, GP regression constructs a probability distribution $p(f^*|\boldsymbol{y})$ of the target function value $f^* = f(\boldsymbol{x}^*)$ at the unobserved target input $\boldsymbol{x}^*$.

In GP regression, the prior on the latent function values $\boldsymbol{f} = (f^1, \dots, f^i, \dots, f^N)$ and the target function value f^* corresponding to the target input $\boldsymbol{x}^*$ is represented as a Gaussian Process (GP) [9]:

$$p(\boldsymbol{f}, f^*) = N\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}^{NN} & \mathbf{K}^{N*} \\ \mathbf{K}^{*N} & \mathbf{K}^{**} \end{bmatrix}\right) \quad (1)$$

where the covariance matrix $\mathbf{K}^{NN}$ is given as

$$\mathbf{K}^{NN} = \begin{bmatrix} k(\boldsymbol{x}^1, \boldsymbol{x}^1) & \dots & k(\boldsymbol{x}^1, \boldsymbol{x}^N) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}^N, \boldsymbol{x}^1) & \dots & k(\boldsymbol{x}^N, \boldsymbol{x}^N) \end{bmatrix} \quad (2)$$

In addition, the covariance coefficients corresponding to the target input $\boldsymbol{x}^*$ are given as $\mathbf{K}^{*N} = [k(\boldsymbol{x}^*, \boldsymbol{x}^1), \dots, k(\boldsymbol{x}^*, \boldsymbol{x}^N)]$ and $\mathbf{K}^{**} = k(\boldsymbol{x}^*, \boldsymbol{x}^*)$. The (i, j) th component of the covariance matrix $\mathbf{K}^{NN}$ quantifies the covariance $\text{cov}(\boldsymbol{x}^i, \boldsymbol{x}^j) = \text{E}[(f(\boldsymbol{x}^i) - m(\boldsymbol{x}^i))(f(\boldsymbol{x}^j) - m(\boldsymbol{x}^j))]$, which is approximated by the evaluation of the kernel function $k(\boldsymbol{x}^i, \boldsymbol{x}^j)$. The kernel function $k(\boldsymbol{x}^i, \boldsymbol{x}^j)$ quantifies the geometric similarity between two inputs, $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$, and uses the quantified similarity as the approximated value for the covariance, i.e., $k(\boldsymbol{x}^i, \boldsymbol{x}^j) \approx \text{cov}(\boldsymbol{x}^i, \boldsymbol{x}^j)$. The type of kernel function $k(\boldsymbol{x}^i, \boldsymbol{x}^j)$ used to build a GP regression model and its parameters can strongly affect the overall representation.

An effective kernel function is chosen by considering the characteristics of the target function. Noting that the energy consumption varies smoothly with the changes in the machining parameters [3], we use a squared exponential kernel function that can effectively describe a continuously varying function. The squared exponential kernel function evaluates the covariance between the two input feature vectors $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$ as [16]:

$$k(\boldsymbol{x}^i, \boldsymbol{x}^j) = \sigma_s^2 \exp\left(-\frac{1}{2}(\boldsymbol{x}^i - \boldsymbol{x}^j)^T \text{diag}(\boldsymbol{\lambda})^{-2}(\boldsymbol{x}^i - \boldsymbol{x}^j)\right) \quad (3)$$

The kernel function is described by the hyper-parameters, σ_s and $\boldsymbol{\lambda}$. The term σ_s^2 is referred to as the signal variance that quantifies the overall magnitude of the covariance value. The vector $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_r, \dots \lambda_N)$ is referred to as the characteristic length scales to quantify the relevancy of the input features in $\boldsymbol{x} = (x_1, \dots, x_r, \dots x_n)$ for predicting the response y.

To model the relationship between the latent function values $\boldsymbol{f} = (f^1, \dots, f^i, \dots, f^N)$ and its observed values $\boldsymbol{y} = (y^1, \dots, y^i, \dots, y^N)$, GP regression defines the likelihood function as

$$p(\boldsymbol{y}|\boldsymbol{f}) = N(\boldsymbol{f}, \sigma_\epsilon^2 \boldsymbol{I}) \quad (4)$$

which assumes independent and identically distributed Gaussian noise with variance σ_ϵ^2. The variance σ_ϵ^2 quantifies the level of noise assumed to exist in the observed output response and its value is estimated while optimizing the hyper parameters, σ_s and $\boldsymbol{\lambda}$, for the kernel function. The noise variance is often treated together with σ_s and $\boldsymbol{\lambda}$ as hyper-parameters, i.e., $\boldsymbol{\theta} = \{\sigma_s, \boldsymbol{\lambda}, \sigma_\epsilon\}$. The optimum hyper parameters $\boldsymbol{\theta}^* = \{\sigma_s^*, \boldsymbol{\lambda}^*, \sigma_\epsilon^*\}$ are determined by maximizing the (marginal) log-likelihood of the training data as

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmax}} \log p(\boldsymbol{y}) \quad (5)$$

where $p(\boldsymbol{y})$ is computed as $\int p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f})\, d\boldsymbol{f}$.

Using the prior distribution on the function values $p(\boldsymbol{f}, f^*)$ and the likelihood function $p(\boldsymbol{y}|\boldsymbol{f})$ (i.e., the noise model), the posterior distribution on the joint function values $(\boldsymbol{f}, f^*)$ given the observed function values $\boldsymbol{y}$ can be computed using Bayes rule as

$$p(\boldsymbol{f}, f^*|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}, f^*)}{p(\boldsymbol{y})} \quad (6)$$

Marginalizing the latent function values $\boldsymbol{f}$, the distribution on the target function value f^* (corresponding to the target input $\boldsymbol{x}^*$) can be computed as

$$\begin{aligned} p(f^*|\boldsymbol{y}) &= \int p(\boldsymbol{f}, f^*|\boldsymbol{y}) d\boldsymbol{f} \\ &= \frac{1}{p(\boldsymbol{y})} \int p(\boldsymbol{y}|\boldsymbol{f}) p(\boldsymbol{f}, f^*) d\boldsymbol{f} \end{aligned} \quad (7)$$

Since both the prior and the likelihood are Gaussian, $p(f^*|\boldsymbol{y})$ is also Gaussian represented as

$$p(f^*|\boldsymbol{y}) = N(\mu^*, (\sigma^*)^2) \quad (8)$$

where the mean μ^* and the variance $(\sigma^*)^2$ of the target value f^* are expressed as [6].

$$\mu^* = \mathbf{K}^{*N}(\mathbf{K}^{NN} + \sigma_\epsilon^2 \boldsymbol{I})^{-1} \boldsymbol{y} \quad (9a)$$

$$(\sigma^*)^2 = \mathbf{K}^{**} - \mathbf{K}^{*N}(\mathbf{K}^{NN} + \sigma_\epsilon^2 \boldsymbol{I})^{-1} \mathbf{K}^{N*} \quad (9b)$$

B. Sparse GP Approximation

Computing the mean μ^* and the variance $(\sigma^*)^2$ in Eqs. (9a) and (9b) requires the inversion of the $N \times N$ matrix $(\mathbf{K}^{NN} + \sigma_\epsilon^2 \boldsymbol{I})$. As the number of training data points becomes large, the Gaussian Process regression becomes inefficient. To overcome the computational demand of the GP regression, researchers have proposed various ways to approximate the exact GP regression model. One approach is to approximate the kernel matrix using a row-rank matrix constructed using a small number of input points. The inputs used to construct the row-rank matrix are named support points, pseudo inputs, inducing inputs, or others. The approach is often based on approximating the prior $p(\boldsymbol{f}, f^*)$ on the latent function values $\boldsymbol{f}$ and the target function value f^* [11].

Let $\bar{\boldsymbol{f}} = (\bar{f}^1, \dots, \bar{f}^i, \dots, \bar{f}^M)$ denotes the M inducing function values corresponding to the M inducing inputs

$\bar{\boldsymbol{X}} = (\bar{\boldsymbol{x}}^1, \ldots, \bar{\boldsymbol{x}}^i, \ldots, \bar{\boldsymbol{x}}^M)$. The prior $p(\boldsymbol{f}, f^*)$ can be expressed by including the inducing inputs $\bar{\boldsymbol{f}}$ as:

$$p(\boldsymbol{f}, f^*) = \int p(\boldsymbol{f}, f^*, \bar{\boldsymbol{f}})\, d\bar{\boldsymbol{f}} = \int p(\boldsymbol{f}, f^*|\bar{\boldsymbol{f}}) p(\bar{\boldsymbol{f}})\, d\bar{\boldsymbol{f}} \quad (10)$$

According to [11], almost all methods of sparse GP regression assume that $\boldsymbol{f}$ and f^* are conditionally independent given $\bar{\boldsymbol{f}}$ such that

$$p(\boldsymbol{f}, f^*|\bar{\boldsymbol{f}}) \approx p(\boldsymbol{f}|\bar{\boldsymbol{f}}) p(f^*|\bar{\boldsymbol{f}}) \quad (11)$$

This conditional independence assumption is the source that reduces the computational complexity of computing the joint probability distribution in Eq. (10). The concept is very similar to probabilistic graphical model that computes the joint probability as a product of conditional probability distributions to reduce the computational requirements. The impact of this approximation would differ depending on the data set. If the data actually possesses this behavior, the approximation would be very good.

Substituting Eq. (11) into Eq. (10), the prior $p(\boldsymbol{f}, f^*)$ is approximated as

$$\begin{aligned} p(\boldsymbol{f}, f^*) &= \int p(\boldsymbol{f}, f^*|\bar{\boldsymbol{f}}) p(\bar{\boldsymbol{f}})\, d\bar{\boldsymbol{f}} \\ &\approx \int p(\boldsymbol{f}|\bar{\boldsymbol{f}}) p(f^*|\bar{\boldsymbol{f}}) p(\bar{\boldsymbol{f}})\, d\bar{\boldsymbol{f}} \end{aligned} \quad (12)$$

Note that the latent function values $\boldsymbol{f}$ and the target value f^* are related only through the inducing function values $\bar{\boldsymbol{f}}$. In other words, the relationship between $\boldsymbol{f}$ and f^* is expressed by the two compact relationships, one between $\boldsymbol{f}$ and $\bar{\boldsymbol{f}}$ and the other between f^* and $\bar{\boldsymbol{f}}$.

Depending on how we model $p(\boldsymbol{f}|\bar{\boldsymbol{f}})$ and $p(f^*|\bar{\boldsymbol{f}})$ in Eq. (12), the different inference methods for the sparse GP regression result. In this study, we use the fully independent training conditional (FITC) approximation proposed in [12] to approximate $p(\boldsymbol{f}|\bar{\boldsymbol{f}})$ as

$$p(\boldsymbol{f}|\bar{\boldsymbol{f}}) \approx \prod_i^N p(f^i|\bar{\boldsymbol{f}}) \quad (13)$$

and uses the exact expression for $p(f^*|\bar{\boldsymbol{f}})$. Inserting the approximated $p(\boldsymbol{f}|\bar{\boldsymbol{f}})$ into Eq. (12), the prior $p(\boldsymbol{f}, f^*)$ on the function values becomes [12]:

$$p(\boldsymbol{f}, f^*) \approx N\left(\boldsymbol{0}, \begin{bmatrix} \mathbf{Q}^{NN} + \mathrm{diag}(\mathbf{K}^{NN} - \mathbf{Q}^{NN}) & \mathbf{Q}^{N*} \\ \mathbf{Q}^{*N} & \mathbf{K}^{**} \end{bmatrix}\right) \quad (14)$$

where $\mathbf{Q}^{NN} = \bar{\mathbf{K}}^{NM}(\bar{\mathbf{K}}^{MM})^{-1}\bar{\mathbf{K}}^{MN}$, $\mathbf{Q}^{N*} = \bar{\mathbf{K}}^{NM}(\bar{\mathbf{K}}^{MM})^{-1}\bar{\mathbf{K}}^{M*}$ and $\mathbf{Q}^{*N} = \bar{\mathbf{K}}^{*M}(\bar{\mathbf{K}}^{MM})^{-1}\bar{\mathbf{K}}^{MN}$. The $M \times M$ covariance matrix $\bar{\mathbf{K}}^{MM}$ is constructed by the kernel evaluations between every pair of the inducing inputs $\bar{\boldsymbol{X}} = (\bar{\boldsymbol{x}}^1, \ldots, \bar{\boldsymbol{x}}^i, \ldots, \bar{\boldsymbol{x}}^M)$, i.e., the (i, j) component of $\bar{\mathbf{K}}^{MM}$ is $k(\bar{\boldsymbol{x}}^i, \bar{\boldsymbol{x}}^j)$. In addition, the $N \times M$ matrix $\bar{\mathbf{K}}^{NM}$ is constructed by the kernel evaluations between the training inputs $\boldsymbol{X} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^i, \ldots, \boldsymbol{x}^N)$ and the inducing inputs $\bar{\boldsymbol{X}} = (\bar{\boldsymbol{x}}^1, \ldots, \bar{\boldsymbol{x}}^i, \ldots, \bar{\boldsymbol{x}}^M)$ and the $1 \times M$ matrix $\bar{\mathbf{K}}^{*M}$ is constructed as $\bar{\mathbf{K}}^{*M} = [k(\boldsymbol{x}^*, \bar{\boldsymbol{x}}^1), \ldots, k(\boldsymbol{x}^*, \bar{\boldsymbol{x}}^M)]$.

Substituting the approximated $p(\boldsymbol{f}, f^*)$ in Eq. (14) into Eq. (7), the distribution $p(f^*|\boldsymbol{y})$ on the target value f^* given $\boldsymbol{y}$ is expressed as $p(f^*|\boldsymbol{y}) = N(\mu^*, (\sigma^*)^2)$ with the mean μ^* and variance $(\sigma^*)^2$ of the target value f^* being represented, respectively, as [12]:

$$\mu^* = \mathbf{Q}^{*N}[(\mathbf{Q}^{NN} + \mathrm{diag}(\mathbf{K}^{NN} - \mathbf{Q}^{NN}) + \sigma_\epsilon^2 \boldsymbol{I})]^{-1}\boldsymbol{y} \quad (15)$$

$$\begin{aligned} (\sigma^*)^2 = {} & \mathbf{K}^{**} \\ & -\mathbf{Q}^{*N}[(\mathbf{Q}^{NN} + \mathrm{diag}(\mathbf{K}^{NN} - \mathbf{Q}^{NN}) + \sigma_\epsilon^2 \boldsymbol{I})]^{-1}\mathbf{Q}^{N*} \end{aligned} \quad (16)$$

Because $\mathbf{Q}^{NN} = \bar{\mathbf{K}}^{NM}(\bar{\mathbf{K}}^{MM})^{-1}\bar{\mathbf{K}}^{MN}$ is a rank M matrix, the computational demands for evaluating Eqs. (15) and (16) can be significantly reduced when $M \ll N$.

C. Optimization of the Inducing Variables and Hyper Parameters

So far, we have assumed that the inducing inputs $\bar{\boldsymbol{X}} = (\bar{\boldsymbol{x}}^1, \ldots, \bar{\boldsymbol{x}}^i, \ldots, \bar{\boldsymbol{x}}^M)$ are given. The inducing inputs can be selected from the training inputs $\boldsymbol{X} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^i, \ldots, \boldsymbol{x}^N)$, which require solving combinatorial optimization problem. Another approach is to directly optimize $\bar{\boldsymbol{X}}$ by treating $\bar{\boldsymbol{X}}$ as continuous optimization variables [12]. In this study, we use the later approach following the optimization procedure in [12]. Using the likelihood $p(\boldsymbol{y}|\boldsymbol{f}) = N(\boldsymbol{f}, \sigma_\epsilon^2 \boldsymbol{I})$ and the approximated prior $p(\boldsymbol{f}) \approx N(\boldsymbol{0}, \mathbf{Q}^{NN} + \mathrm{diag}(\mathbf{K}^{NN} - \mathbf{Q}^{NN}))$ that can be obtained from Eq. (14), the marginal likelihood $p(\boldsymbol{y})$ can be computed as [12]:

$$\begin{aligned} p(\boldsymbol{y}) &= \int p(\boldsymbol{y}, \boldsymbol{f})\, d\boldsymbol{f} \\ &= \int p(\boldsymbol{y}|\boldsymbol{f}) p(\boldsymbol{f})\, d\boldsymbol{f} \\ &= N(\boldsymbol{0}, \mathbf{Q}^{NN} + \mathrm{diag}(\mathbf{K}^{NN} - \mathbf{Q}^{NN}) + \sigma_\epsilon^2 \boldsymbol{I}) \end{aligned} \quad (17)$$

Note that the marginal likelihood $p(\boldsymbol{y})$ depends on the inducing inputs $\bar{\boldsymbol{X}}$ and the hyper parameters $\boldsymbol{\theta}$, which determine the components in $\mathbf{K}^{NN}$ and $\mathbf{Q}^{NN}$. The optimum inducing inputs $\bar{\boldsymbol{X}}^*$ and the hyper-parameters $\boldsymbol{\theta}^*$ can be obtained by maximizing the marginal loglikelihood as

$$(\bar{\boldsymbol{X}}^*, \boldsymbol{\theta}^*) = \underset{(\bar{\boldsymbol{X}}, \boldsymbol{\theta})}{\mathrm{argmax}} \log p(\boldsymbol{y}) \quad (18)$$

which can be computed using a gradient based optimization algorithm with the analytically available gradient of $\log p(\boldsymbol{y})$ with respect to $\bar{\boldsymbol{X}}$ and $\boldsymbol{\theta}$ [12]. The optimized inducing inputs $\bar{\boldsymbol{X}}^*$, which are not necessarily part of training inputs $\boldsymbol{X}$, will then be used to construct the sparse GP regression. In solving Eq. (18), the size of inducing

inputs is predetermined by the user and the M inputs randomly selected out of training inputs $\boldsymbol{X} = \{\boldsymbol{x}^1, \dots, \boldsymbol{x}^N\}$ are used as the initial values. 1,579 data points are the total data points that we can possibly use. We assume that all of 1,579 data points are not necessary to describe the energy consumption pattern of the target machine. Therefore, we are experimenting using only a portion of the training data points to construct the energy prediction function. The chosen number of training data points is denoted as N, on which the formulations of both Full GP and sparse GP based. The optimization is conducted using the MATLAB code for Sparse pseudo-input Gaussian processes (SPGP) available in [17]. When M ($M \ll N$) inducing inputs are obtained to construct a sparse GP regression model, the computation demand for training reduces from $O(N^3)$ to $O(NM^2)$, and the storage requirement reduces from $O(N^2)$ to $O(NM)$.

IV. Energy Density Prediction Model

The sparse GP regression is employed to efficiently construct and represent the energy prediction model for a machine tool. In this section, we investigate how the relative ratio M/N between the number of inducing inputs M and the number of training inputs N affect the accuracy and the training time of the energy prediction function constructed by sparse GP regression. We conduct this study using the energy consumption data of the target milling machine. Furthermore, we also discuss the effect of the size of training inputs N on the accuracy of the prediction model. If we can regulate the number of training input data that need to be retained in the model with acceptable accuracy, the energy prediction model can be updated with much reduced computational demand $O(\bar{N}M^2)$ and storage requirement $O(\bar{N}M)$ where $\bar{N}$ is the size of training input data to be retained in the model.

A. Data

The direct, derived, and simulated data, obtained from MTConnect and simulation, are used to construct the energy prediction model of the milling machine. There are five basic input (predictor) variables, namely feed rate, spindle speed, depth of cut, cutting direction and cutting strategy. Feed rate, spindle speed and depth of cut are quantifiable measurements, while the cutting direction and cutting strategy are qualitative (or categorical) features. Park *et al.* have constructed generalized energy prediction models using both continuous and categorical variables [5]. Here, we only use three quantitative input features because (1) it is easier to visualize a small dimensional target function, and (2) Eq. (18) can be reliably solved when the input features are continuous variables. The focus of this study is to illustrate the computational efficiency of the sparse GP regression by comparing to that of the full GP regression. If a large amount of data is available, energy prediction functions for every combination of the cutting directions and cutting strategies can be constructed. The input features employed in this study are summarized as follows:

- $x_1 \in \mathbb{R}$ Feed rate: the average velocity at which the tool is fed, which can be retrieved from the controller data.
- $x_2 \in \mathbb{R}$ Spindle speed: the average rotational speed of the tool, which can be retrieved from the controller data.
- $x_3 \in \mathbb{R}$ Depth of cut: the actual depth of material that the tool is cutting, which can be obtained from the cutting simulation.

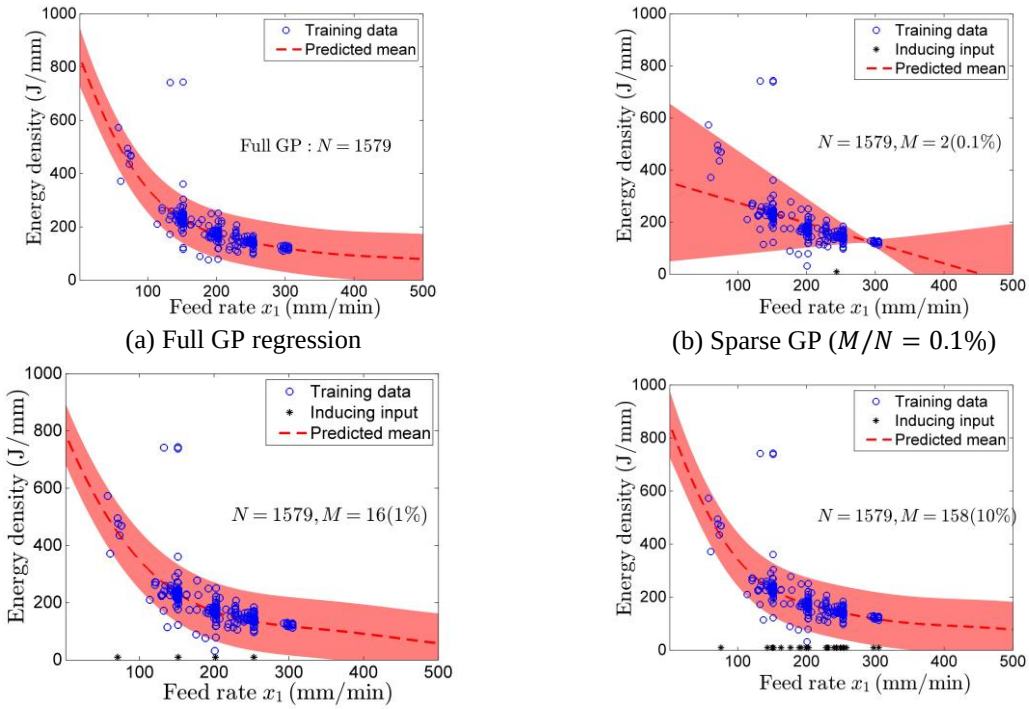
The output (response) variable is energy per unit length of cut $y^i = E^i/l^i$ where E^i and l^i are, respectively, the average energy consumption and the length of cut for the ith NC code block. For the energy prediction model, the estimated density is scaled by the length of cut. Note that we model the relationship between the averaged values of the process parameters and the average energy density across the duration of the block. For the 18 parts machined for the experiments, we select 1,579 pairs of the input feature vector $\boldsymbol{x} = (x_1, x_2, x_3)$ and the corresponding energy density $y = E/l$. The selected data set $\boldsymbol{D} = \{(\boldsymbol{x}^i, y^i) | i = 1, \dots, N\}$, where $N = 1{,}579$, serves as training data for this study.

B. Training

Using the selected data set $\boldsymbol{D} = \{(\boldsymbol{x}^i, y^i) | i = 1, \dots, N\}$, where $N = 1{,}579$, the energy density prediction function $\hat{y} = f(\boldsymbol{x})$ is constructed. First, we construct the full GP regression model that does not use the sparse approximation for the covariance matrix, and use its performance (i.e., representation and training time) as the reference to compare the performance of the sparse GP repression model. By changing the size M of the inducing inputs $\bar{\boldsymbol{X}} = (\bar{\boldsymbol{x}}^1, \dots, \bar{\boldsymbol{x}}^i, \dots, \bar{\boldsymbol{x}}^M)$ that will be used to construct the sparse version of the covariance matrix, we construct the energy density functions based on the sparse GP regression. We then investigate how the size M of the inducing inputs $\bar{\boldsymbol{X}} = (\bar{\boldsymbol{x}}^1, \dots, \bar{\boldsymbol{x}}^i, \dots, \bar{\boldsymbol{x}}^M)$ affects the representation and the training time of the energy prediction function.

Fig. 6 compares the energy density function constructed by full GP regression and sparse GP regression with different number of inducing inputs, i.e., the ratio M/N given the training data points, $N = 1{,}579$. For visualization purpose, the constructed energy density function $\hat{y} = f(\boldsymbol{x})$, which is a three dimensional function, is shown here as 1-D curve $\hat{y} = f(x_1, x_2 = 3{,}000, x_3 = 1.5)$ with the fixed spindle speed $x_2 = 3{,}000$ (RPM) and depth of cut $x_3 = 1.5$ (mm). The circles on the figures represent the training data points that have $x_2 = 3{,}000$ and $x_3 = 1.5$; the dotted line represents the predicted mean energy density; and the shaded band represents the 95% confidence bound computed as $[\mu - 1.96\sigma, \mu + 1.96\sigma]$; and the asterisk marks denote the locations (x_1 values) of the inducing inputs that have $x_2 = 3{,}000$ and $x_3 = 1.5$.

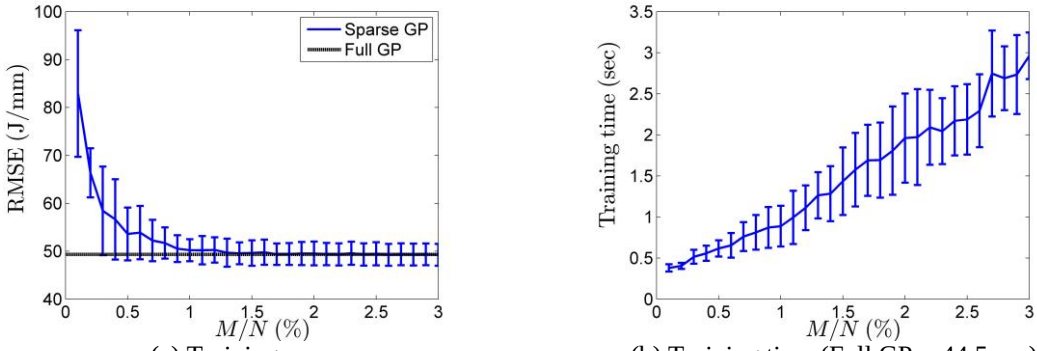
As shown in Fig. 6(b), when a small number of inducing inputs ($M = 2$) are used, the constructed mean is biased significantly. As the number of inducing inputs increase, as shown in Fig. 6(c), the constructed mean and confidence



(a) Full GP regression (b) Sparse GP ($M/N = 0.1\%$)

(c) Sparse GP ($M/N = 1\%$) (d) Sparse GP ($M/N = 10\%$)

Figure 6. The impact of the ratio M/N on the constructed energy density function.



(a) Training error (b) Training time (Full GP = 44.5 sec)

Figure 7. The impact of the ratio M/N on the training error and time.

bound resemble those of the full GP regression model in Fig. 6(a). When the ratio M/N is 0.1, as shown in Fig. 6(d), the constructed energy density function (i.e., both the mean and the confidence bound) is close to the energy density function constructed using the full GP regression model.

Using 80% of training inputs (i.e., $T = 0.8N$), and the training time for the different ratios of M/N. The training error is to quantify the fitness of the constructed regression model to the training input. Using 50 randomly selected training input data sets, 50 training errors are computed for different ratios of M/N, ranging from 1% to 3%. The mean and $\pm$ one standard deviation of the training errors for different ratios M/N are then represented in Fig. 7(a). As shown in the figure, when the ratio M/N exceeds 1%, the training error for the sparse GP model converges to that of the full GP regression model, which implies that the energy density function constructed by the sparse GP regression has similar representation as its full GP model. Fig. 7(b) shows the trend of the training time with the increase in the ratio M/N. The training time for sparse GP regression is significantly shorter than the training time of 44.5 sec for the full GP regression. In short, using sparse GP regression, we can construct the energy density function whose representation is as good as that for the energy density function constructed by the full GP model with significantly reduced training time.

C. Testing Using Blind Test Part

The constructed energy prediction model based on sparse GP regression is employed to predict the energy consumption for the target machine to produce a generic test part. The geometry of the test part is different from that of the 18 parts used to collect the training data. Therefore, the energy prediction capability of the prediction model can be

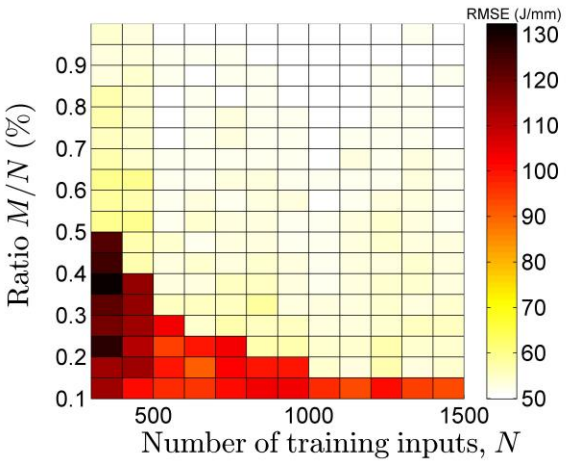


Figure 8. The test error depending on N and M/N.

tested for generalized geometry. First, we construct energy prediction functions using different combinations of N (the number training input) and M/N (the ratio of inducing inputs to the number of training inputs). Here, a random selection method is used to select the N training inputs among the available 1,579 input data points. Using the constructed energy (density) prediction models $\hat{y} = f(\boldsymbol{x})$, we estimate the energy density for each NC code block in test data set. The test error is then estimated as $\sqrt{\frac{1}{Q}\sum_{i=1}^{Q}(f(\boldsymbol{x}^i) - y^i)^2}$, where y^i is the measured energy density for the ith NC code block in the test data set, and Q is the total number of data points in the test data set. For each combination of N and M/N, we construct 30 energy prediction functions using 30 sets of randomly selected N inputs, and compute the 30 test errors and average them. The average test errors for different energy prediction functions are compared in Fig. 8. As shown in the figure, the test error decreases as the number of retained training input N and the ratio M/N of inducing inputs to the training inputs increase. Another interesting observation is that as the number of training input N decreases, a larger ratio M/N is required for the energy prediction function to predict the energy density values more accurately.

Fig. 9 compares the predicted and the measured energy density for each individual NC code block in the test data set. Each subfigure shows the prediction results of the energy density function constructed using different values of N (the number training input) and M/N (the ratio of inducing inputs to the training inputs). The figure clearly shows that given the same number of training inputs N, the prediction accuracy increases as the ratio M/N increases. In addition, given the ratio M/N, the predicted energy density becomes more accurate as the number of training inputs N increases.

V. DISCUSSIONS

To overcome the high computational and memory demands of the GP regression, we use the approximated GP regression based on a sparse representation of the covariance matrix. The simulation studies using the energy consumption data collected from the target milling machine shows that the energy prediction function constructed using sparse GP regression can predict the energy density as accurate as the full GP regression does, with significantly reduced computational and storage requirements.

The sparse GP regression approximates the covariance matrix using inducing inputs. The number of inducing inputs M determines the level of approximation and thus the computational efficiency. In this study, the value M is pre-determined and specified by the user. Further study is needed to find the optimal number of inducing inputs. We have also investigated how the number of training data points N selected randomly from the total training data set may affect the accuracy of the prediction model. The results show that the accuracy of the prediction model does not increase proportionally with the size of training data set. In order for the GP regression model to become an adaptive model with continuously streaming data, efficient methods to regulate the size of training input data N that will be retained in the model and to select the most informative training data need to be studied further.

ACKNOWLEDGMENT

The authors acknowledge the support by the Smart Manufacturing Systems Design and Analysis Program at the National Institute of Standards and Technology (NIST), Grant Numbers 70NANB12H225 and 70NANB12H273 awarded to University of California, Berkeley, and to Stanford University respectively. In addition, the authors appreciate the support of the Machine Tool Technologies Research Foundation (MTTRF) and System Insights for the equipment used in this research. Certain commercial systems are identified in this paper. Such identification does not imply recommendation or endorsement by NIST; nor does it imply that the products identified are necessarily the best available for the purpose. Further, any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NIST or any other supporting U.S. government or corporate organizations. The authors also thank Ronay Ak for providing valuable comments.

REFERENCES

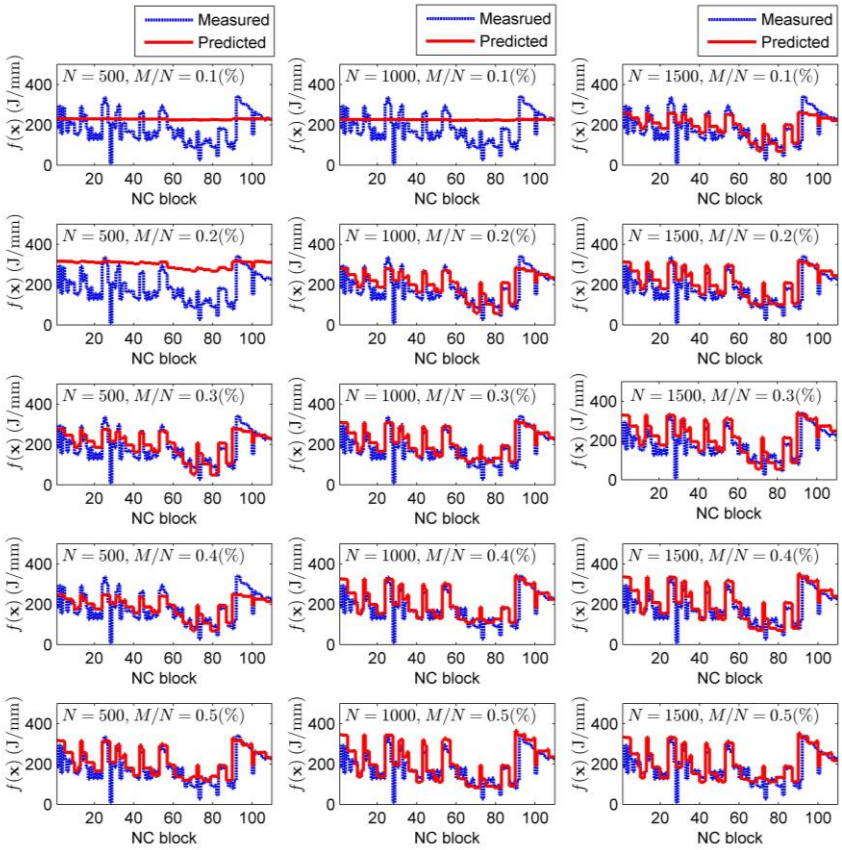


Figure 9. The comparison between the measured and predicted energy density depending on N and M/N.

[1] A. Vijayaraghavan, W. Sobel, A. Fox, D. Dornfeld, and P. Warndorf, "Improving machine tool interoperability using standard interface protocols: MTConnect," *Proceeding of 2008 International Symposium on Flexible Automation*, 2008, Atlanta, USA.

[2] A. Vijayaraghavan, and D. Dornfeld, "Automated energy monitoring of machine tools," *CIRP Annals – Manufacturing Technology*, 59, 2010, pp. 21-24.

[3] N. Diaz, E. Redelsheimer and D. Dornfeld, "Energy Consumption Characterization and Reduction Strategies for Milling Machine Too Use," *Proceeding of 18th CIRP International Conference on Life Cycle Engineering*, 2011, Braunschweig, Germany.

[4] R. Bhinge, J. Park, N. Biswas, M. Helu and D. Dornfeld, K. Law and S. Rachuri, "AnIntelligent Machine Monitoring System Using Gaussian Process Regression for Energy Prediction," *in Proceeding of IEEE International Conference on Big Data* (IEEE BigData 2014), 2014, Washington, DC.

[5] J. Park, R. Bhinge, N. Biswas, M. Srinivasan, M. Helu, S. Rachuri, D. Dornfeld and K. Law, "A generalized data-driven energy prediction model with uncertainty for a milling machine tool using Gaussian Process," *in Proceeding of ASME 2015 International Manufacturing Science and Engineering* Conference, 2015, Charlotte, NC.

[6] V. Tresp, "A Bayesian committee machine," *Neural Computation*, 12(11), 2000, pp. 2719–2741.

[7] D. Nguyen-tuong, and J. Peters, "Local Gaussian process regression for real time online model learning and control," *Advances in Neural Information Processing Systems*, MIT Press, 2008.

[8] J. Shi, R. Murray-Smith, and D. Titterington, "Hierarchical Gaussian process mixtures for regression," *Statistics and Computing*, 15, 2005, pp. 31-41.

[9] A. Ranganathan, M. Yang, and J. Ho, "Online sparse Gaussian Process regression and its applications," *IEEE Transactions on Image Processing*, 20(2), 2011, pp. 391-404

[10] H. Xiao, and C. Echert, "Lazy Gaussian Process Committee for Real-Time online regression," *in Proceeding of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, Bellevue, USA, 2013.

[11] J. Quiñonero-Candela and C. Rasmussen, "A unifying view of sparse approximate Gaussian Process regression," Journal of Machine Learning Reseach, 6, 2005, pp. 1939-1959.

[12] E. Snelson and Z. Ghahramani, "Sparse Gaussian Processes using pseudo-inputs," *in Proceeding of the Neural Information Processing Systems Conference*, 18, 2006, pp. 1257-1264

[13] E. Snelson and Z. Ghahramani, "Local and global sparse Gaussian process approximations," *in Proceeding of the Eleventh International Conference on Artificial Intelligence and Statistics*, 2, 2007, pp. 524-531.

[14] L. Csató and M. Opper, "Sparse Online Gaussian Processes," Technical Report NCRG, Neural Computing Reseach Groput, Dept of Computer Science and Applied Mathmatics, Aston University, 2009.

[15] M.K. Titsias, "Variational learning of inducing variables in sparse Gaussina Processes," *in Proceeding of the Twelveth International Conference on Artificial Intelligence and Statistics*, 2009, pp. 567-574.

[16] R. M., Neal. (1996). *Bayesian learning for neural networks*. Springer-Verlag, New York.

[17] E. Snelson. (2006). MATLAB code for Sparse pseudo-input Gaussian processes (SPGP) [Computer program]. Available at http://www.gatsby.ucl.ac.uk/~snel.