Voxel-based analysis and modeling of MRR computational accuracy in milling process

Article *in* CIRP Journal of Manufacturing Science and Technology · October 2019

DOI: 10.1016/j.cirpj.2019.07.003

4 authors:

Zhenguo Nie
Carnegie Mellon University
24 PUBLICATIONS **65** CITATIONS
SEE PROFILE

Roby Lynn
Georgia Institute of Technology
15 PUBLICATIONS **76** CITATIONS
SEE PROFILE

Tommy Tucker
Tucker Innovations
14 PUBLICATIONS **63** CITATIONS
SEE PROFILE

T. Kurfess
Georgia Institute of Technology
266 PUBLICATIONS **2,665** CITATIONS
SEE PROFILE

Voxel-based Analysis and Modeling of MRR Computational Accuracy in Milling Process

Zhenguo Nie [a], Roby Lynn [a], Tommy Tucker [b], Thomas Kurfess [a,*]

[a] *George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA, 30332-0405, USA*

[b] *Tucker Innovations, Inc., Waxhaw, NC, 28173, USA*

* Corresponding author: Thomas Kurfess

E-mail addresses:

zhenguo.nie@me.gatech.edu; zhenguonie@gmail.com (Z. Nie).

roby.lynn@gatech.edu (R. Lynn).

tommy@tuckerinnovations.com (T. Tucker).

kurfess@gatech.edu (T. Kurfess).

Abstract

Material removal rate (MRR) is a commonly used metric for determining the efficiency of a toolpath design, as it is usually used to determine the amount of machining time spent doing useful work. Voxel-based computer-aided manufacturing (CAM) software enables simple computation of MRR by counting the number of voxels removed (N) in one feed step. However, depending on the geometry of the cutting tool and the voxel size used in the CAM software, there can be disagreement between theoretical and simulated MRR values. The cause of oscillation in voxel-based simulated MRR is the misjudgment of voxel removal due to the discrete nature of the voxel model. MRR has a linear relationship with N, so the analysis of MRR can be equivalent to the analysis of N as other conditions keep invariable. A series of simulation experiments were conducted on GPGPU by using computer programs that were developed to simulate the milling process. Discrepancies between voxel-based simulated MRR and the theoretical MRR were investigated. The dependence with respect to the ratio (M) between the radius of the tool and voxels size, the use of single or double precision floating-point computations, and the end shape of the cutting tool was studied.

Results indicate that double precision improves the ability of judgment on voxel removal, thereby decreases the standard deviation of the N curve. Besides, ball-end milling has good computational accuracy with a lower standard deviation than flat-end milling. The relative range, mean absolute error, and standard deviation analysis were used to describe the oscillation regularity of N curves. The probability distribution of normal distance from the voxel center to tool boundary was studied. The result indicates that the distance obeys a uniform distribution when M is large enough. Modeling of MRR computation with computational error domain was proposed to represent the oscillation behavior of MRR, and the result shows that the model can well predict the magnitude and period of N curves during the milling process.

Keywords: MRR, CAM, Voxel, GPGPU, Modeling.

1. Introduction

Computer-aided manufacturing (CAM) software is a powerful tool used for the generation of toolpath for computer numerical control (CNC) machine tools. Toolpath generation for a complex surface is a large time-consumption for current general-purpose CPU (central processing unit)-based CAM software. Hossain et al. (2016b) point out that the current advent of parallel computation with graphics processing units (GPUs) enable the use of GPU-based algorithms for toolpath design that can improve computation speed for both additive and subtractive manufacturing processes. In recent years, some works have solved CAD and CAM acceleration problems using GPGPU. Krishnamurthy et al. (2009)describe a unified and optimized method for evaluating and displaying trimmed NURBS surfaces using GPGPU. It turns out GPU evaluation and rendering speeds have an obvious elevation to CPU. Kurfess et al. (2007) use NURBS (Non-Uniform Rational B-Spines) curves and surface-based calculation as the main methods for rendering, surface offsetting, tool path programming. Tarbutton et al. (2010) propose a graphics-based approach to the tool path and trajectory-planning problem found in machining and robotics applications.

A voxel model uses a three-dimensional array of small cubes to represent a part volume; these cubes, or voxels, are the three-dimensional analog of two-dimensional pixels in an image. The use of voxels for a CAM application enables higher surface complexity, simplified collision checking, and more robust analysis of material removal than would be possible with typical parametric CAM (Kurfess, 2018). Voxel-based CAM has a number of advantages over traditional parametric CAM: it is better able to represent complex, freeform surfaces that would be difficult to describe with analytical curves; it enables simpler collision checking between a cutting tool and a workpiece; and the calculation and simulation of material removal along a toolpath consists of a simple summation of removed voxels (Jang et al. , 2000, Lynn et al. , 2018a, Lynn et al. , 2018b). To solve the enormous data storage problem in voxel-based computation, Hossain et al. (2016a) develop hybrid dynamic trees for an extreme-resolution 3D sparse data modeling. Abi-Chahla (2008) indicates that GPU has a great potential in the engineering field as its powerful parallel computing ability. Lynn et al. (2017) point out the use of GPUs for general purpose computation is of particular interest for CAM using voxel models; instead of using analytical surfaces to define part geometry. Fig. 1 shows how a voxel model represents a part of a collection of small cubes. The sparse voxel octree can enable higher memory efficiency: fine voxels are near the boundary of the part and the cutting tool, and coarse voxels are in the interior of the workpiece (Konobrytskyi, 2013). This work relies on a voxel-based CAM software known as SculptPrint that uses GPUs to accelerate toolpath generation.

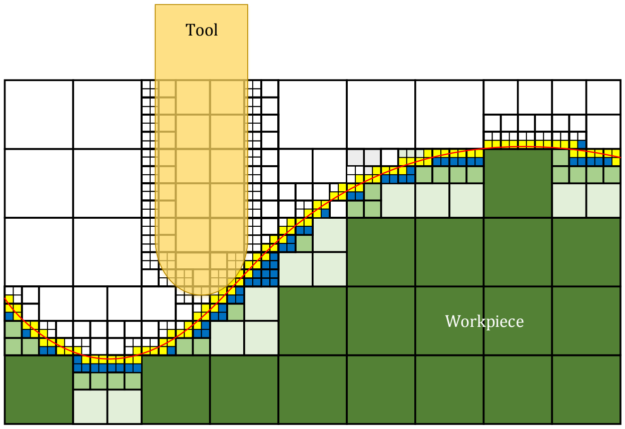


Fig. 1. Geometrical representation in voxel-based CAM.

Material removal rate (MRR), the volumetric rate of material flow from the workpiece, is a commonly used metric for the efficiency of a toolpath as it can be used to determine the amount of machining time spent doing useful work. Choudhury and Appa Rao (1999) find that MRR has a significant effect on the tool life, cutting force and energy consumption, etc. Kara and Li (2011) have recently proposed an empirical model to characterize the relationship between energy consumption and process variables for material removal processes. The results indicate that the specific energy consumption (SEC) has an inverse relationship with MRR. Parametric CAM system commonly applies a complicated NURBS interpolator to compute MRR, which has been tried one after another by Tsai et al. (2001), Tikhon et al. (2004), and Ko et al. (2005).

Voxel-based CAM enables simple computation of MRR for a toolpath of any complexity. The whole toolpath is composed of motion steps. In each step, the tool center moves just an identical distance of the voxel size. The total volume of all removed voxels over one-step can be summed up. The time length of the step is the ratio of the voxel size to the feedrate. Finally, MRR is obtained as the ratio of the volume to the time. For example, consider a simple raster toolpath shown on the prismatic workpiece: for this ball-end milling along a linear trajectory, which has constant cutting depth and feedrate, so the theoretical MRR of the milling process should be constant. A voxel-based CAM software known as SculptPrint (Lynn et al. , 2016) is used to simulate such a steady-state milling process. The cutting depth is 0.1 inch, and feedrate is 0.5 inches/s. The simulated MRR curve using the voxel model is computed and plotted in the bottom in Fig. 2. However, the simulated MRR is not constant and instead exhibits oscillation.

The cause of oscillation in voxel-based MRR computation would be the misjudgment of voxel removal. Due to the discrete nature of the voxel model, the theoretical MRR may not always match the simulated MRR. The theoretical analysis and the simulation experimental demonstration are both presented in this article. All the simulation experiments in this article were conducted on GPU using CUDA (Compute Unified Device Architecture) respectively with single precision and double precision. In addition, the basic judging criterion of voxel removal is the distance from the voxel center to tool boundary. The probability distribution of the normal distance from the voxel center to tool boundary is studied in this article for better understanding the nature of MRR oscillation.

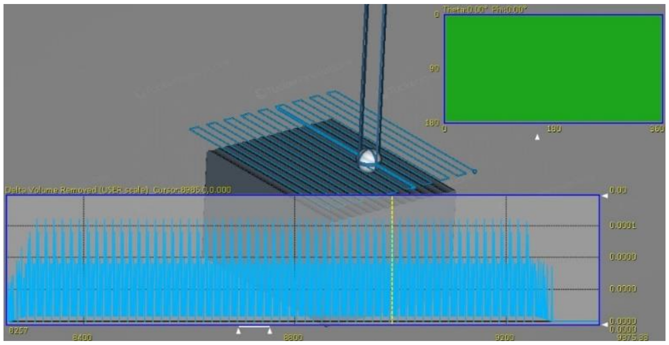


Fig. 2. Simulated MRR curve using voxel-based CAM software SculptPrint.

This work presents a voxel-based model of MRR computational accuracy using GPGPU computing. Mathematical expressions of MRR with voxel representation was derived using Cavalieri's principle. Analysis under both single and double precision computational errors of the amount of voxel removal is presented to explain the cause of oscillation in the computed MRR curves. A scale factor M is defined as a ratio of the tool radius to the voxel size. The simulation experimental results demonstrate that the amplitude and oscillation of N curves during stable cutting stage are related to M, computation accuracy, and end shape of the cutting tool. The probability distribution of normal distance from the voxel center to tool boundary was studied, and the result indicates that the normal distance obeys a uniform distribution. A computation model was proposed to describe the oscillating behavior of N curves; the error domain is embedded into the model to describe the voxel misjudgment during MRR computation.

2. Related works

Material Removal Rate

MRR is defined as the volume of material removed per unit time. The higher cutting parameters, the higher MRR. As shown in Fig. 3, MRR in flat-end milling process with a cylindrical milling cutter is the product of the depth of cut, the width of cut, and the feedrate. The mathematical expression of MRR for linear flat-end milling is represented in Equation (1).

$$\text{MRR} = D \times W \times F \tag{1}$$

where D is the cutting depth, W is the cutting width, and F is the feedrate.

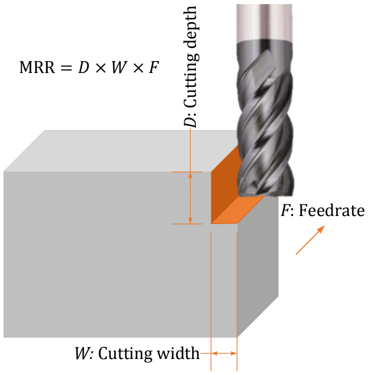


Fig. 3. The mathematical expression of MRR in the flat-end milling process.

More generally, as shown in Fig. 4, when the tool moves along a three-dimensional curve, the removed material volume is equal to the swept volume generated by the tool-material contact area ψ along the tool trajectory. A generalized MRR expression can be calculated as follows:

$$\text{MRR} = \int_{\psi} d\vec{s} \cdot \vec{F} \tag{2}$$

where ψ is the tool-material contact surface during machining, $\vec{F}$ is the federate of the tool at any point on the contact surface ψ, and $d\vec{s}$ is the area vector of the finite element at the point. The inner production $d\vec{s} \cdot \vec{F}$ is the finite MRR on the infinitesimal area near the point. For 5-axis milling, federate $\vec{F}$ is usually not a constant on surface ψ, but a vector function in three-dimensional space. What needs to be pointed out is that the tool outline surface is the enveloping surface of the cutting edge due to its spinning.

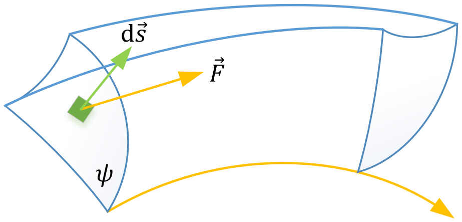


Fig. 4. Schematic diagram of the generalized MRR definition. Removed material volume is equal to the tool-material contact area moving along the tool trajectory.

Floating Point Representation

CUDA is a parallel computing platform and application programming interface (API) model created by NVIDIA that enables a GPU to be used for general purpose processing – an approach termed GPGPU. The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements for the execution of compute kernels. The creation of GPU-based algorithms for toolpath generation and analysis in this work relies on CUDA. Sanders and Kandrot (2010) describe that CUDA is designed as a parallel platform and programming model to work with programming languages such as C, C++, and FORTRAN. CUDA accepts both single and double precision numbers as determined by specific GPU architecture that is targeted during application development. The single and double precision floating point formats are described as shown in Fig. 5 (IEEE, 2018).

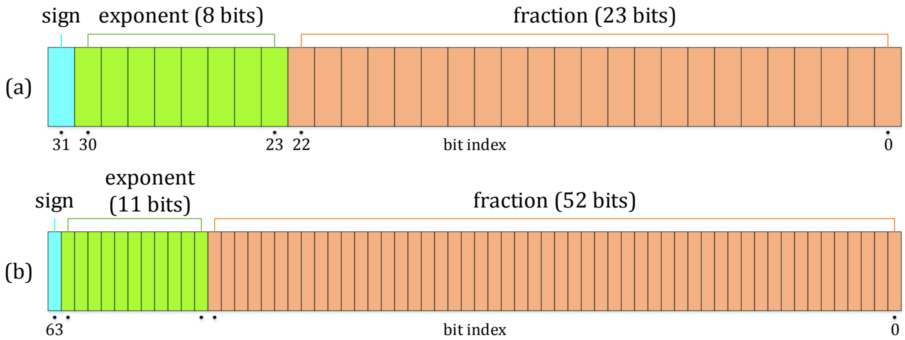


Fig. 5. IEEE754 binary floating-point format. (a): single precision, (b): double precision.

Conversion of a single precision binary floating-point number to decimal can be performed using Equation (3) and (4) (IEEE, 2018):

$$\text{Value} = (-1)^{b_{31}} \times \left(1 + \sum_{i=1}^{23} b_{23-i} 2^{-i}\right) \times 2^{(e-127)} \tag{3}$$

$$e = \sum_{i=0}^{7} b_{23+i}\, 2^{+i} \tag{4}$$

where $b_0,\ b_1, \cdots,\ b_{31}$ are the binary digits. Double precision also has similar expressions as shown in equation (5) and (6) (IEEE, 2018). After transformation from binary to decimal, the number has only six or seven significant decimal digits with the single precision ($\log_{10}(2^{23}) \approx 6.92$), while about fifteen or sixteen with the double precision ($\log_{10}(2^{52}) \approx 15.65$).

$$\text{Value} = (-1)^{b_{63}} \times \left(1 + \sum_{i=1}^{52} b_{52-i} 2^{-i}\right) \times 2^{(e-1023)} \tag{5}$$

$$e = \sum_{i=0}^{10} b_{52+i}\, 2^{+i} \tag{6}$$

Non-Uniform Rational B-Splines (NURBS) interpolator

Non-Uniform Rational B-Splines (NURBS) interpolator was used in computer-aided design (CAD) and manufacturing (CAM) to represent a wide range of curves and surfaces due to its flexibility and precision. NURBS are parametric curves and surfaces, which is usually described by standard STEP files in CAD/CAM systems. The parametric form is very convenient for controlling multi-axis machine tools and robotics, where each axis is driven individually (Zhang and Greenway, 1998). A three-dimensional curve is expressed in parametric form as follows:

$$C(u) = [x(u), y(u), z(u)] = \sum_{i=0}^{n} f_i(u) P_i \tag{7}$$

where u is an arbitrary parameter and usually normalized to [0,1], $\{P_i\}$are control points, and $\{f_i(u)\}$ are piecewise polynomial basis functions forming a basis for the vector space of all piecewise polynomial functions of the desired degree and continuity . Similarly, A three-dimensional surface is expressed by two parameters (u and v), as shown in Equation (8).

$$S(u) = [x(u,v), y(u,v), z(u,v)] = \sum_{i=0}^{n} \sum_{j=0}^{m} f_i(u) g_j(v) P_{i,j} \tag{8}$$

$C(u)$ and $S(u)$ are called basis spline (B-spline) functions, were first used to define curves and surfaces for CAD by Gordon and Riesenfeld (1974). Many mainstream CAM software, e.g. Mastercam, CAMWorks, CATIA, Siemens NX CAM, etc., all adapt NURBS interpolator to represent the geometry.

For a steady linear milling process (cutting depth: 0.1 inches, milling tool feedrate: 0.5 inch/s, the radius of the ball-end tool: 0.1 inches.), MRR are computed respectively using the voxel model and the NURBS interpolator. For the voxel model, MRR is calculated on the different voxel sizes (or the different scale factor M). For NURBS interpolator, an open-source package NURBS-Python (Bingol and Krishnamurthy, 2019) is used to simulate the milling process and compute the change in MRR. As shown in Fig. 6, the truth MRR of the milling is 2.936E-4 inch^3/s, and the MRR curve of NURBS is close to the truth value with a relative error of less than 1%. However, with an increase of M (or decrease of the voxel size), the MRR curves of the voxel mode draw closer to the truth-value, and the oscillation is getting smaller. In other words, fine voxel can obtain a smaller bias and variance in MRR computation. Nevertheless, it is attention that the voxel mode has higher accuracy and lower oscillation than NURBS when M is roughly equal to or greater than 1,000. This means we can get high computation precision of MRR by means of reducing the voxel size.

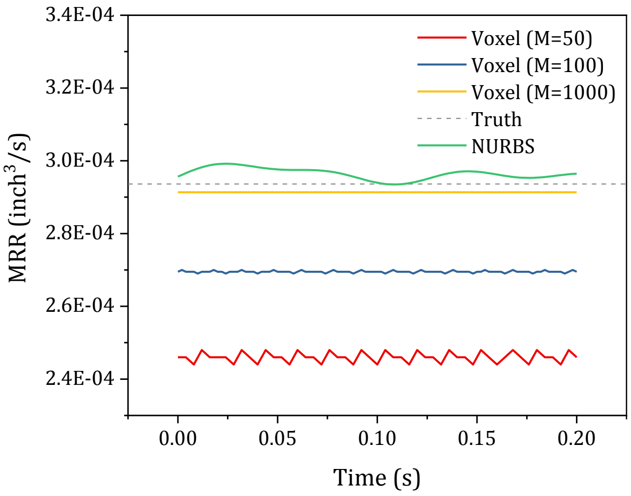


Fig. 6. MRR computational results of steady linear milling.

3. Voxel-based MRR expression of the milling process

Fig. 7 shows some common types of end mills. Flat-end milling and ball-end milling are the two most common milling processes. Flat-end milling with a cylindrical tool is mainly used to produce a flat surface, while ball-end milling is widely used in the multi-axis NC machines for manufacturing complicated curved surface. In this section, voxel-based MRR expression of both the flat-end milling and the ball-end milling are derived respectively.

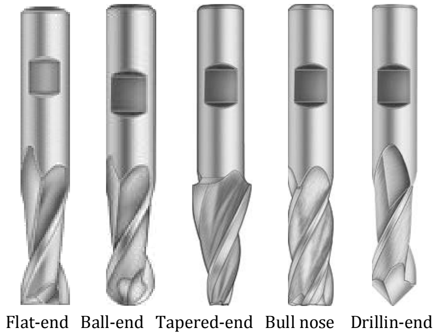


Fig. 7. Common types of end mills

3.1 Voxel-based MRR expression of the flat-end milling

As shown in Fig. 8, the flat-end milling process is first discussed to deduce the voxel-based MRR expression. Flat-end milling is, in essence, a two-dimensional problem. Top view of the flat-end milling process is shown in Fig. 9. The red circular arc is the external surface of the cylindrical tool in top view, the squares are voxels that make up the part, the bold dots are the center points of voxels, and the distance of feed step is chosen the same value as the voxel size (a). Once a bold dot intrudes into the tool boundary, the corresponding voxel will be removed by the tool. It can be seen that there is one and only one voxel removed in each row along the moving direction. All the voxels to be removed are marked as green within the current step. Fig. 10 shows the relationship of the real removed shape (which is a crescent shape in the physical milling process, domain A) and the discrete voxels removed (which is made up of the entire removed voxels in one-step, domain B) when the tool moves just one-step. According to the Cavalieri's principle (Wikipedia, 2018), the area of the domain A is equal to the total area of the domain B between the upper and lower limits. Above all, the quantity of voxels removed (N) in one feed step is equal the number of voxels in the projected plane (marked with brown in Fig. 8) for three-dimensional flat-end milling.

To quantitatively describe the relative size of the voxel to the milling tool, a scale factor M is defined in Equation (9) as a ratio of the tool radius to the voxel size.

$$M = \frac{R}{a} \tag{9}$$

where R is the radius of the milling tool, and a is the voxel size. The scale factor M illustrates the relative size of the tool radius to the voxel size. For a fixed tool, a larger M means fine voxel, while a smaller M means rough voxel.

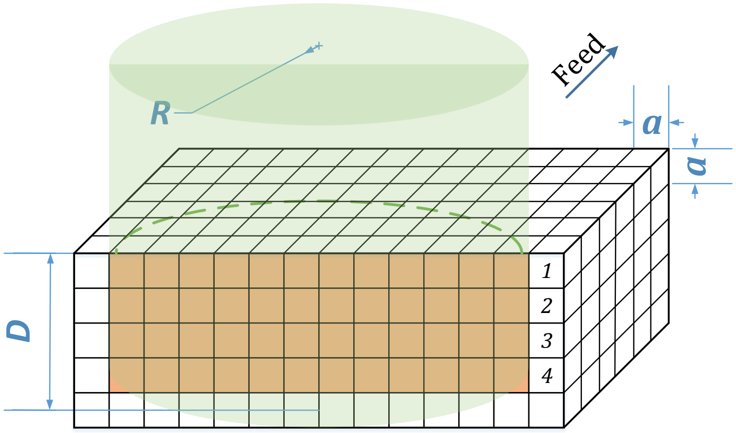


Fig. 8. Voxelization of the workpiece-tool system in flat-end milling

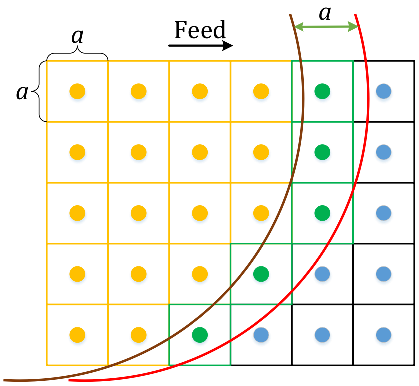


Fig. 9. Top view of the flat-end milling.

As shown in Fig. 10, it can be seen that the maximum error between domain A and domain B is two voxels that are located on the head and end of the domain B. When the scale factor M is large enough, the residual area outside the two limits of domain B can be ignored. Therefore, the area of domain A can be substituted by the area of domain B in MRR computation. The relative error satisfies the inequality (10). This inequality illustrates and guarantees the bias of the voxel-based MRR results in Fig. 6.

$$\varepsilon_{A|B} < \frac{2}{2R/a} = \frac{1}{M} \tag{10}$$

where $\varepsilon_{A|B}$ is the relative error between domain A and domain B.

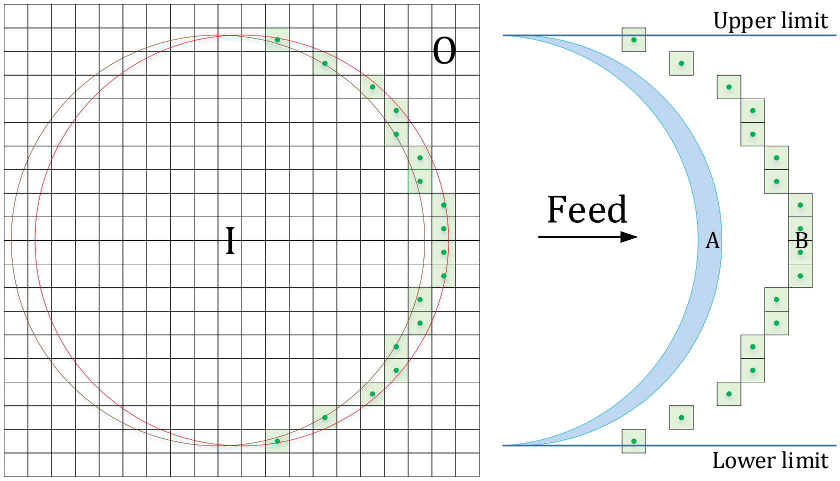


Fig. 10. Area equivalence principle. According to Cavalieri's principle, the area of domain A is equal to the area of domain B between the upper and lower limits. N is the number of voxels in domain B.

Based on the definition, MRR is the ratio of volume removed to traveling time as expressed in Equation(11):

$$\mathrm{MRR} = \frac{Na^3}{a/F} = Na^2F \tag{11}$$

where N is the variable quantity of removal voxels in one feed step, and F is the feedrate. It is visible that MRR is proportional to N. Therefore, the analysis of MRR can be equivalent to the analysis of N as other conditions keep invariable.

In the flat-end milling process, N can be expressed in a discrete form as shown in Equation (12), which is a product of voxel quantity in the top view and cutting depth. Then the expression of MRR of flat-end milling is represented as Equation (13).

$$N_{flat} = \left\lceil \frac{2R}{a} \right\rceil \left\lceil \frac{D}{a} \right\rceil = \lceil 2M \rceil \left\lceil \frac{MD}{R} \right\rceil \tag{12}$$

$$\mathrm{MRR}_{flat} = N_{flat}a^2F = \lceil 2M \rceil \left\lceil \frac{MD}{R} \right\rceil a^2F \tag{13}$$

where a is the voxel size, R is the radius of the tool, D is the depth of cut, and ⌈⌉ is the ceiling function. MRR would be a constant value in the face milling with fixed a, F, M, D, and R. The ceiling function can be neglected when M is large enough, and N can be expressed as a quadratic function of M in Equation (14).

$$N_{flat} \simeq \frac{2D}{R} M^2 \tag{14}$$

3.2 Voxel-based MRR expression of the ball-end milling

The ball-end milling process can be regarded as flat-end milling with a variable radius. As shown in Fig. 11, the cross-section of the intersection between a sphere and cuboid is voxelized by cubes. The sphere is the milling tool, and voxels inside the sphere are removal material. The number of voxels removed (N) in one feed step is equal to the number of voxels in the projected plane (marked with brown in Fig. 11).

In order to express MRR, each voxel layer is simplified into a flat-end milling process with a cutting depth of a. Taking the i^{th} voxel layer as the research object, N_{ball}^{i} is the variable number of voxel removed in one feed step and can be computed as in Equation (15), where r_i is the radius of the circle on the i^{th} voxel layer and can be calculated by Equation (16). The total number of voxel removed (N_{ball}) in one feed step is the sum of N_{ball}^{i} with i from 1 to $[D/a]$. $[D/a]$ is the number of layers within the scope of the ball-end tool. In conclusion, MRR in ball-end milling process can be expressed in Equation (18).

$$N_{ball}^{i} = \left[\frac{2r_i}{a}\right] \tag{15}$$

$$r_i = \sqrt{(D - ia)(2R - D + ia)} \tag{16}$$

$$N_{ball} = \sum_{i=1}^{[D/a]} N_{ball}^{i} \tag{17}$$

$$\mathrm{MRR}_{ball} = N_{ball} a^2 F \tag{18}$$

Ball-end milling is a nonlinear sum of a series of variable-radius flat-end millings. Due to its nonlinearity, it is difficult to get a concise equation for MRR in the ball-end milling process. The flat-end milling process with a cylindrical tool has a constant radius along the rotational axis. The research on flat-end milling would reveal more general laws than the ball-end milling or some other formed millings.

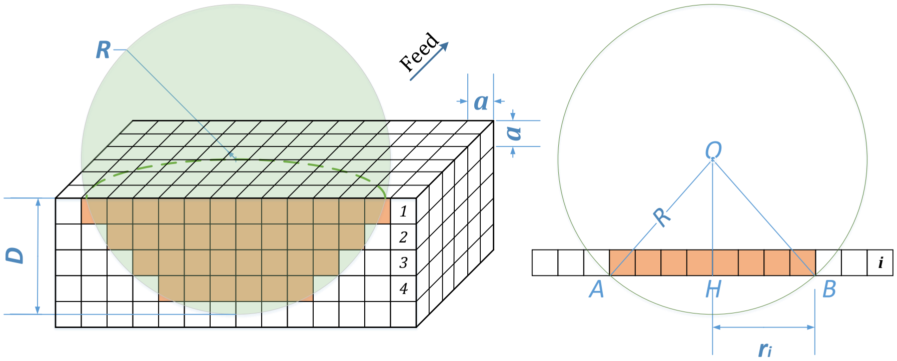


Fig. 11. Voxelization of the workpiece-tool system in ball-end milling

4. MRR Computational results with GPGPU

Studies on flat-end and ball-end millings were respectively performed to determine the accuracy of material removal calculation from voxel models. The radius of the cutting tool is R, and the depth of cut is D. As shown in Fig. 10, the internal domain of the tool boundary is defined as domain I and the outside is domain O. Any voxel in domain I is considered to be removed when its center point crosses from domain O to domain I when the tool is moving through the workpiece.

N is the number of voxels removed in one-step, and a function of tool shape, tool size, and voxel size. First, according to Equation (11), MRR is proportional to N. MRR and N have the same variation trend in the same cutting parameters. Second, N can be computed by counting the number of voxels crossing into the tool boundary as the center of the tool moves from one voxel center to the next. Third, N is a time-independent variable, which can just focus on the tool and voxel size. Therefore, for better understanding the cause of MRR oscillation during the milling process, N is studied as the research objective instead of MRR. Theoretically, for a steady-state linear milling process, the mill tool keeps the fixed cutting depth and feedrate. N should be a constant; however, due to misjudgment of voxel status caused by discretization, N has oscillation in numerical computation. A series of simulation experiments were conducted using computer programs that are developed to simulate the two millings as shown in Fig. 8 and Fig. 11. The cutting process contains three stages: cut-in, steady cutting, and cut-out. All voxel removal calculations were conducted using CUDA on an NVidia Quadro M4000 GPU, which uses Maxwell architecture and supports both the single and double precisions.

4.1 Flat-end milling process with single precision computation

For a steady-state linear milling process, the tool, cutting depth, and feedrate are all fixed. It means theoretical MRRs should be the same in different scale factors (M). Fig. 12 shows simulated results of N with M. All four figures show that the curves have three stages: cut-in (ascent stage), steady cutting (dynamic stability stage), and cut-out (descent stage). Unlike the theoretical constant value of N in the steady cutting stage, the curves exhibit oscillation. For each curve during the steady cutting stage, the average value $\overline{N}_{flat}$ and standard deviation σ are calculated in

Table 1. The theoretical value of N in the stable cutting stage ($\widehat{N}$) can also be calculated by Equation (12). All the data points of N in the stable cutting stage are plotted into frequency histograms as shown in Fig. 13. The distribution of N conforms to the law of normal distribution, where the parameters of the mean value ($\overline{N}_{flat}$) and standard deviation (σ_{flat}) are shown in

Table 1. Fig. 14 shows scatter points of $\left(M, \overline{N}_{flat}\right)$, and quadratic fit curve through points of $\left(M, \widehat{N}\right)$.

Table 1. Computational results with the single precision of flat-end milling in the steady cutting stage. $R = 0.1, D = 0.01$.

M	a	$[D/a]$	$\overline{N}_{flat}$	σ_{flat}	$\widehat{N}$
10	0.01	1	20.95	1.31525	20
100	0.001	10	1808	27.58856	2000
200	0.0005	20	7619	61.68451	8000
500	0.0002	50	50049	130.24701	50000
1000	0.0001	100	200099	429.43052	200000

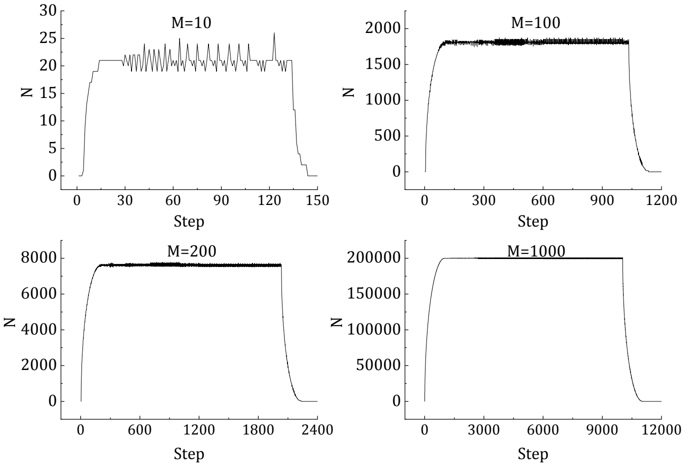


Fig. 12. Computed results of N by GPGPU with single precision ($R = 0.1, D = 0.01$), and the theoretical MRR is identical in four cases.

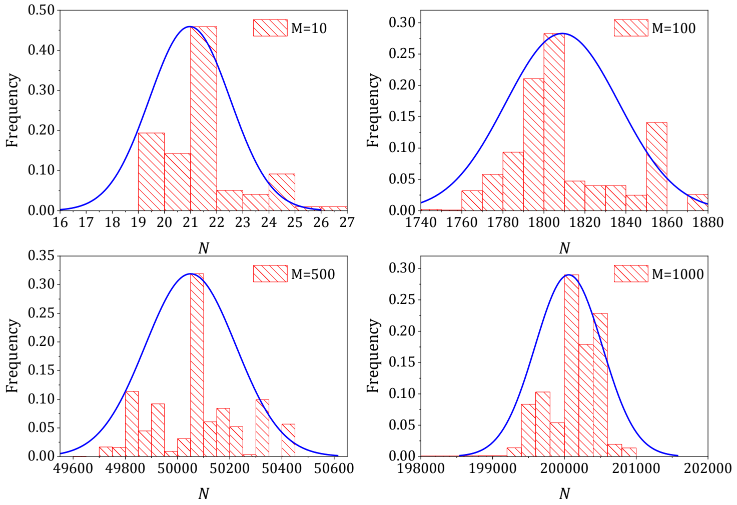


Fig. 13. Frequency histogram of N with single precision ($R = 0.1, D = 0.01$).

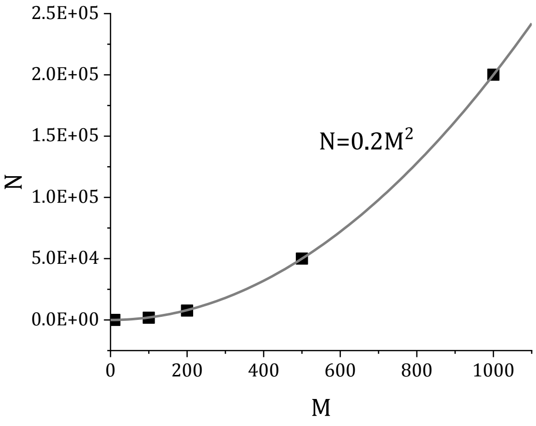


Fig. 14. The quadratic increase of $\overline{N}$ changes against the increasing M.

4.2 Flat-end milling process with double precision computation

The simulation experiments with double precision were conducted on the same GPGPU. The comparison results are shown in Fig. 15. Diagrams (a), (b) and (c) show the variation of N along the cutting path with both single and double precisions, and diagram (d) is an enlarged portion of the curve which is depicted by the rectangle in diagram (c). The average value ($\overline{N}_{flat}$) and standard deviation (σ) of N during the stable cutting stage with double precision are presented in Table 2. It is obvious that double precision computation has a lower standard deviation than single precision. Lower standard deviation shows up as smaller oscillation on the curves of N.

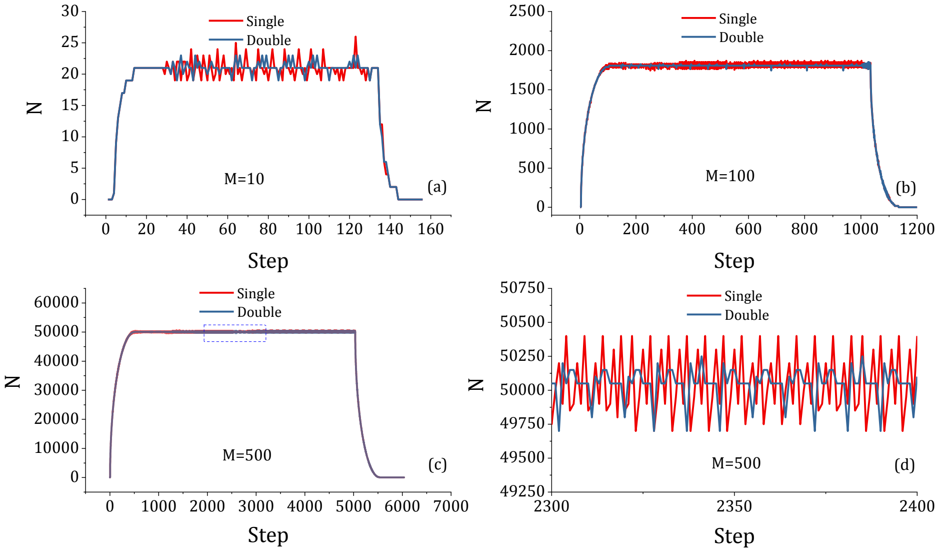


Fig. 15. Comparison using of N single and double precisions in flat-end milling ($R = 0.1$, $D = 0.01$).

Table 2. Computational results with the double precision of flat-end milling in the steady cutting stage. $R = 0.1$, $D = 0.01$.

M	a	$[D/a]$	$\overline{N}_{flat}$	σ_{flat}
10	0.01	1	21	1.26352
100	0.001	10	1809	6.17655
200	0.0005	20	8020	9.64606
500	0.0002	50	50050	72.40815
1000	0.0001	100	200099	336.11364

4.3 Ball-end milling process

Both single precision and double precision computations were conducted for ball-end milling process with different voxel sizes. Fig. 17 shows the comparison results of N between ball-end and flat-end millings with single precision using the same values of tool radius ($R = 0.1$) and cutting depth ($D = 0.01$). It can be seen that ball-end milling has a lower value and smaller oscillation than flat-end milling. The penetration shape of the cutting tool for flat-end milling is a cylinder, while it is a spherical crown for ball-end milling. Based on the analysis before, N is proportional to the projection area of the active part of the tool along the direction of motion (brown voxels in Fig. 8 and Fig. 11). As shown in Fig. 16, for ball-end milling, the projection area is a circular segment (MHN); for flat-end milling, the projection area is a rectangle ($CDEF$). With the same tool radius (R)

and cutting depth (D), the ratio of the two millings on N in the stable cutting stage is equal to the ratio of the two projection areas. When $R = 0.1$ and $D = 0.01$, the theoretical ratio is about 3.41. From the computed results in Table 3, the computed ratio agrees well with the theoretical one when the values of M is larger than 100. However, the ratio of $\overline{N}_{flat}/\overline{N}_{ball}$ deviates drastically from the theoretical ratio when $M = 10$. In other words, coarse voxel size and shallow cutting depth (technologically speaking, the smaller number of layers $[D/a]$) will cause a severe computational error for ball-end milling in voxel-based MRR calculation.

Compared with the flat-end milling, a smaller oscillation of ball-end milling manifests lower standard deviation (σ_{ball}), which is shown in Table 3. The reason for the lower standard deviation of ball-end milling will be analyzed in the following section.

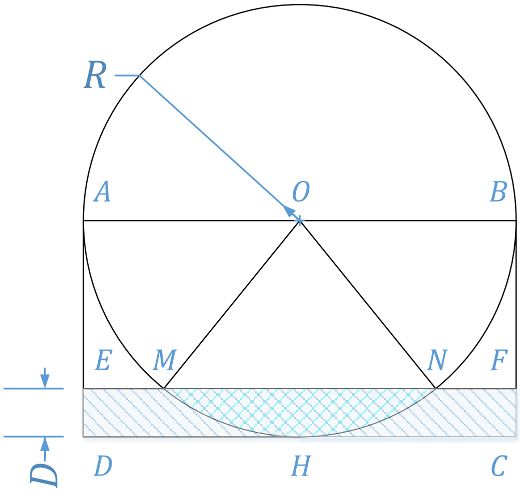


Fig. 16. The projection areas of the active part of tools for flat-end and ball-end mills.

Table 3. Computational results with the single precision of ball-end milling in the steady cutting stage. $R = 0.1$, $D = 0.01$.

M	a	$[D/a]$	$\overline{N}_{ball}$	σ_{ball}	$\overline{N}_{flat}/\overline{N}_{ball}$
10	0.01	1	1	0	20.95
100	0.001	10	539	0.55164	3.35436
200	0.0005	20	2254	0.80397	3.38021
500	0.0002	50	14454	1.43342	3.46264
1000	0.0001	100	58271	2.64711	3.43394

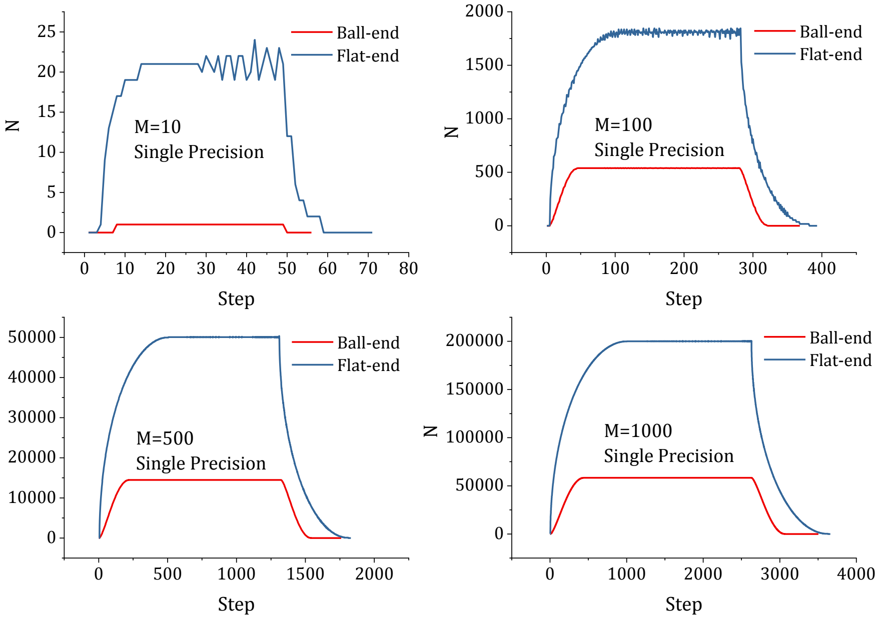


Fig. 17. Comparison of N curves in ball-end and flat-end millings with single precision ($R = 0.1$, $D = 0.01$).

Just as flat-end milling, the ball-end milling simulation is also affected by the computational accuracy. As shown in Fig. 18, the curves with double precision have almost zero oscillation in the stable cutting stage. The markedly statistical result of N in Table 4 is that all the standard deviations are zeros. It means the ball-end milling with double precision can effectively avoid voxel misjudgment.

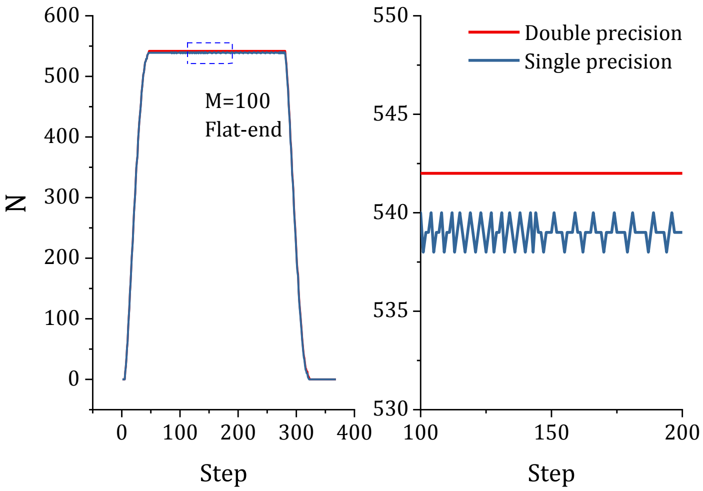


Fig. 18. The comparison result of N with both single and double precisions in ball-end milling ($R = 0.1$, $D = 0.01$).

Table 4. Computational results with the double precision of ball-end milling in the steady cutting stage ($R = 0.1, D = 0.01$).

M	a	$[D/a]$	$\bar{N}_{ball}$	σ_{ball}
10	0.01	1	1	0
100	0.001	10	542	0
200	0.0005	20	2254	0
500	0.0002	50	14454	0
1000	0.0001	100	200099	0

5. Error analysis of voxel-based MRR

The amplitude and oscillation of N curves in the stable cutting stage change with M, computational accuracy, and tool end shape. Error analysis is essential to understand such behavior of MRR calculation error variations. Conclusions of this section are helpful to set up the criterion for voxel size limit in the voxel-based CAM to guarantee the MRR computation accuracy.

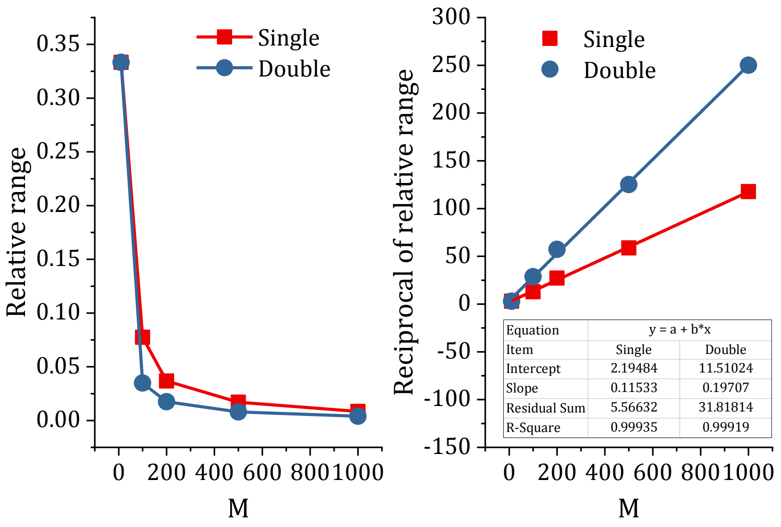
5.1 Relative range analysis of N

The relative range is defined as the ratio of the range to mean value, as shown in Equation (19),

$$R_r = \frac{N_{max} - N_{min}}{\widehat{N}} \tag{19}$$

where N_{max} is the maximum number of the sample, N_{min} is the minimum number of the sample, and $\widehat{N}$ is the predicted outcome which can be calculated by equation (12). The relative ranges of N in the steady cutting stage of flat-end milling are shown in Fig. 19. It can be seen that the reciprocal of the relative range has a linear relationship with M. The fitting equation is expressed in Equation (20). For single precision, $a = 2.195, b = 0.1153$; while for double precision, $a = 11.51, b = 0.1971$.

$$\frac{1}{R_r} = a + bM \tag{20}$$



Equation	y = a + b*x	
Item	Single	Double
Intercept	2.19484	11.51024
Slope	0.11533	0.19707
Residual Sum	5.56632	31.81814
R-Square	0.99935	0.99919

Fig. 19. The relative range changes against M in the flat-end milling.

5.2 Mean absolute error analysis of N

The mean absolute error (MAE) is a quantity used to measure how close predictions are to the eventual outcomes. The MAE is given by equation (21):

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n}\left|N_i - \widehat{N}\right|}{n} \tag{21}$$

where N_i is the computed data points, $\widehat{N}$ is the predicted outcome which can be calculated by equation (12), and n is the sample capacity. The computed values for MAE with both single and double precisions in flat-end milling are shown in Fig. 20. MAE increases with the increasing M. Double precision has a smaller MAE than single precision.

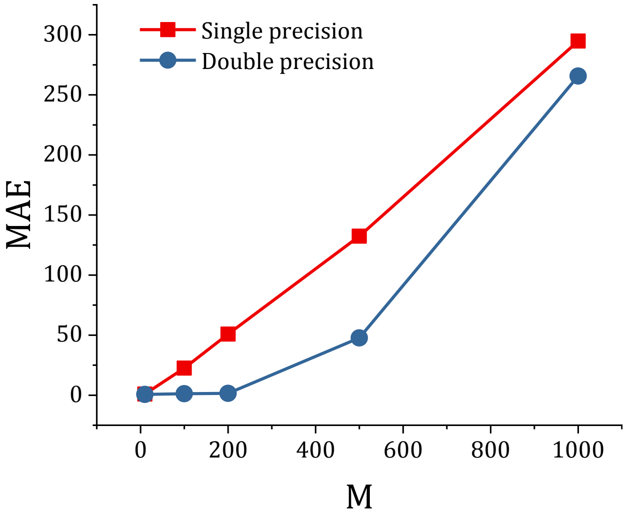


Fig. 20. MAE against M on both single and double precision in flat-end milling.

5.3 Variance analysis

From the previous discussion, double precision can improve the ability of judgment on voxel removal, and decrease the standard deviation of N curves. Besides, ball-end milling has good computational accuracy with a smaller variance than flat-end milling. According to Fig. 21, standard deviations of flat-end milling increase with M for both single and double precision, while standard deviations of ball-end milling are approximate horizontal lines due to their small values.

To explain this phenomenon, we can establish a simple mathematical model on the variance of random variables. Based on the previous analysis, N_{flat} is accumulated by many the same single-layers and expressed in Equation (22), but N_{ball} is accumulated by descending series of single-layers and expressed in Equation (23). Mathematical expectation and variance of N for flat-end and ball-end are respectively expressed in Equation (25)-(28). $E(X_1) = \left[\frac{2R}{a}\right]$ is the mathematical expectation of voxels removed on the first layer in Fig. 8. $E(Y_1)$ is the expected quantity of voxels removed on the first layer in Fig. 11, and equal to $\left[\frac{2r_1}{a}\right] < \left[\frac{2R}{a}\right] = E(X_1)$. $E(Y_i)$ is a decreasing sequence, so $E(N_{ball}) = \sum_{i=1}^{n} E(Y_i) < nE(Y_1) < nE(X_1) = E\left(N_{flat}\right)$. In addition, it can be seen that $\sigma^2(x)$ is a monotonously increasing function from Fig. 21. So $\sigma^2(Y_i) = \sigma^2\left(\frac{2r_i}{a}\right) < \sigma^2\left(\frac{2r_1}{a}\right) < \sigma^2\left(\frac{2R}{a}\right) = \sigma^2(X_1)$, we get $\sigma^2(N_{ball}) = \sum_{i=1}^{n} \sigma^2(Y_i) < \sum_{i=1}^{n} \sigma^2(X_1) = n\sigma^2(X_1) \ll n^2\sigma^2(X_1) = \sigma^2\left(N_{flat}\right)$. It means the variance (standard deviation) of the ball-end milling is far less than the variance (standard deviation) of the flat-end milling. Base on the above analysis, we get

$E(N_{ball})/E(N_{flat}) < 1$, and $\sigma^2(N_{ball})/\sigma^2(N_{flat}) < 1/n = [a/D]$. This conclusion is perfectly compatible with the computed results on the difference of the ball-end and the flat-end millings.

$$N_{flat} = X_1 + X_2 + \cdots + X_n = nX_1 \tag{22}$$

$$N_{ball} = Y_1 + Y_2 + \cdots + Y_n = \sum_{i=1}^{n} Y_i \tag{23}$$

$$n = \left[\frac{D}{a}\right] \tag{24}$$

$$E(N_{flat}) = nE(X_1) \tag{25}$$

$$\sigma^2(N_{flat}) = n^2\sigma^2(X_1) \tag{26}$$

$$E(N_{ball}) = \sum_{i=1}^{n} E(Y_i) < nE(Y_1) < E(N_{flat}) \tag{27}$$

$$\sigma^2(N_{ball}) = \sum_{i=1}^{n} \sigma^2(Y_i) < n\sigma^2(Y_1) \ll \sigma^2(N_{flat}) \tag{28}$$

where X_i is the number of voxels removed on the i^{th} layer as shown in Fig. 8, Y_i is the number of voxels removed on the i^{th} layer as shown in Fig. 11, and n is the total number of layers.

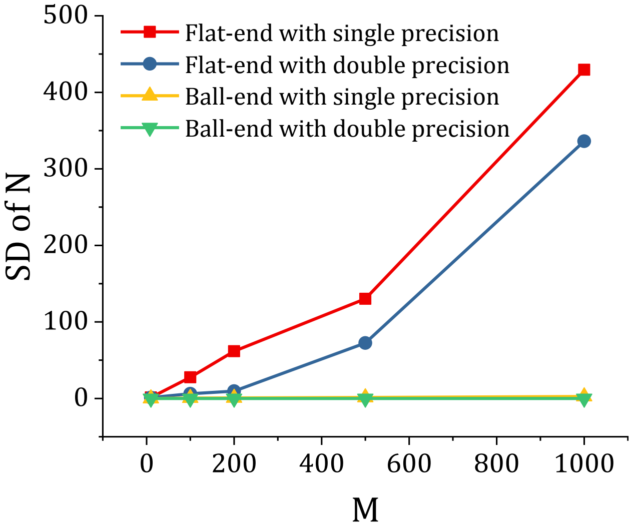


Fig. 21. The standard deviation of N

5.4 Analysis of periodic characteristics

The curves of N show periodic characteristics. The minimal positive period of N changes slightly with M in flat-end milling is shown in Fig. 22. With the increase of M, the minimal positive period just decreases a little, but with no obvious variation trend. The average value of the minimal positive period of N is about 2.9, which is quite close to 3.

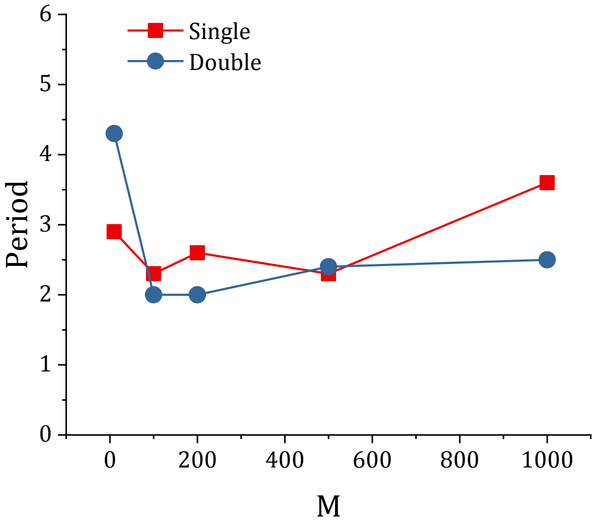


Fig. 22. The minimal positive period of N in flat-end milling.

6. The probability distribution of normal distance from voxel to tool

According to the analysis above, the cause of oscillation in voxel-based MRR computation is the misjudgment of voxel removal due to the discrete nature of the voxel model. The basic judging criterion of voxel removal is the distance from the voxel center to tool boundary. Once the distance changes from positive to negative, the voxel is removed. The probability distribution of normal distance from the voxel center to tool boundary is studied to understand the error distribution when the voxel model makes a decision for voxel removal or not.

6.1 Definition of distance from the voxel center to tool boundary

The centers of voxels that comprise the workpiece from a point grid that the cutting tool travels through as it removes material. Fig. 23 shows a graphical depiction of distances from various voxel centers to the boundary of the cutting tool, which is shown as a red circle. In this figure, the tool moves from left to right, therefore, the material is only removed on the right side of the tool. Some points near the right-hand side of tool boundary are efficient points, which are probable to be removed in the current or next step with tool traveling. Distances from these efficient points to the tool boundary form a mathematical set X, which can be normalized using Equation (30), to form set Y as expressed in Equation (31).

$$X = \{x_1, x_2, x_3, \cdots, x_i, \cdots x_{N^*}\} \tag{29}$$

$$y_i = \frac{x_i}{a} \in \left[-\sqrt{2}, \sqrt{2}\right] \tag{30}$$

$$Y = \{y_1, y_2, y_3, \cdots, y_i, \cdots y_{N^*}\} \tag{31}$$

where N^* is the total amount of the efficient points, x_i is the normal distance from the i^{th} voxel center to the tool boundary, y_i is the normalized distance of x_i.

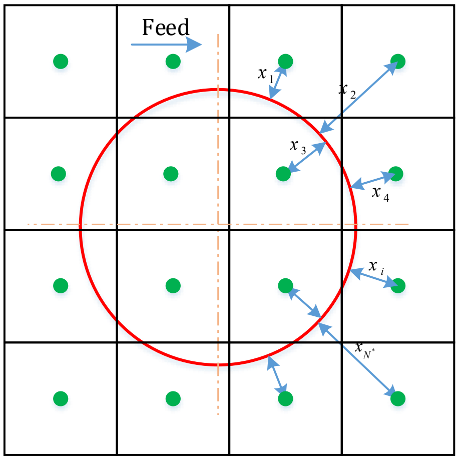


Fig. 23. Distance from voxel centers to tool boundary.

6.2 Probability distribution results

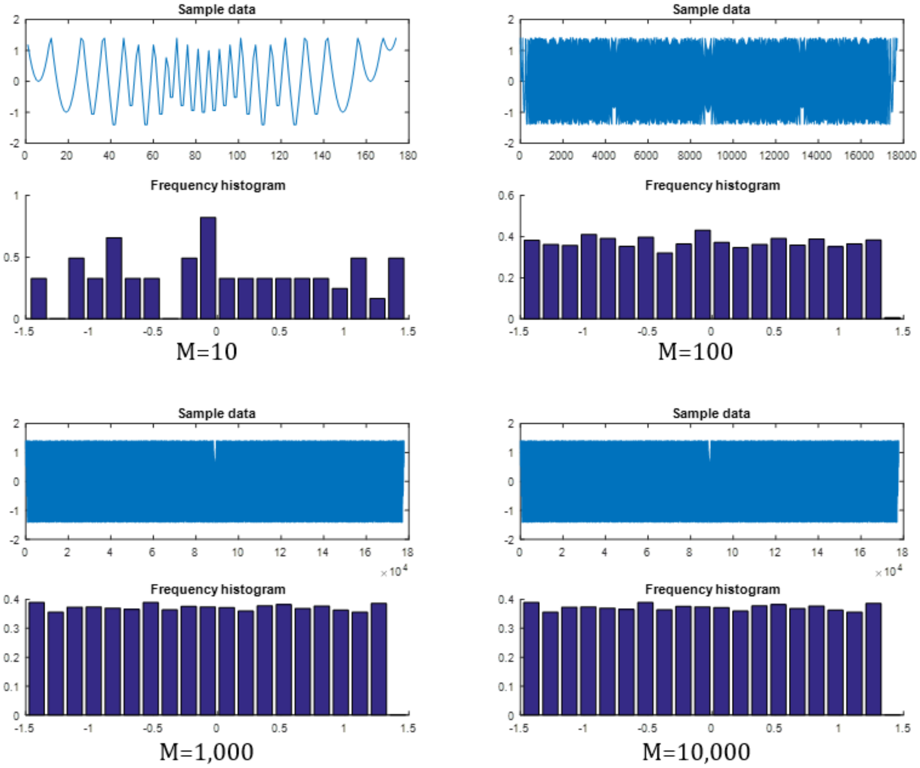
The probability distribution of elements in set Y when the tool center locates coincides with a voxel center is shown in Fig. 24, and the probability distribution of elements in set Y when the tool center coincides with a voxel vertex is shown in Fig. 25. As M increases, the histogram tends to be a uniform distribution. As M tends to infinity, the distribution will be a uniform distribution even when the tool center is located randomly on the voxel grid; this case is shown in Fig. 26. Based on the analysis above, all the elements in set X obey uniform distribution when M tends to infinity; the probability density function of set X is expressed in Equation (32).

$$f(x) = \begin{cases} \dfrac{\sqrt{2}}{4a} & , -\sqrt{2}a \le x \le \sqrt{2}a \\ 0 & , x \le -\sqrt{2}a \text{ or } x \ge \sqrt{2}a \end{cases} \tag{32}$$

The uniform distribution indicates that all the voxels have the same probability, regardless of the distance to the tool boundary; in other words, the misjudgment in voxel removal has a linear relationship with the computational error. The misjudgment in voxel removal causes the oscillation on the curve of N, which should be a constant in the steady cutting stage. The relative range R_r can be used to characterize the metric of misjudgment; the ratio of misjudgment between single and double precision in voxel removed is presented in Equation (33),

$$\left.\frac{(R_r)_{\mathrm{s}}}{(R_r)_{\mathrm{d}}}\right|_{M\to\infty} = \left.\frac{(a+bM)_{\mathrm{d}}}{(a+bM)_{\mathrm{s}}}\right|_{M\to\infty} = \frac{(b)_{\mathrm{d}}}{(b)_{\mathrm{s}}} = \frac{0.1971}{0.1153} = 1.709 \tag{33}$$

where subscript s is single precision, and subscript d is double precision. As mentioned above, the data precision with single precision computation has six or seven significant decimal digits; with double precision computation, it has and fifteen or sixteen significant digits. The ratio in Equation (33) reveals that the possibility of misjudgment in voxel removal with single precision computation is about 1.709 times that of the probability when using double precision computation.



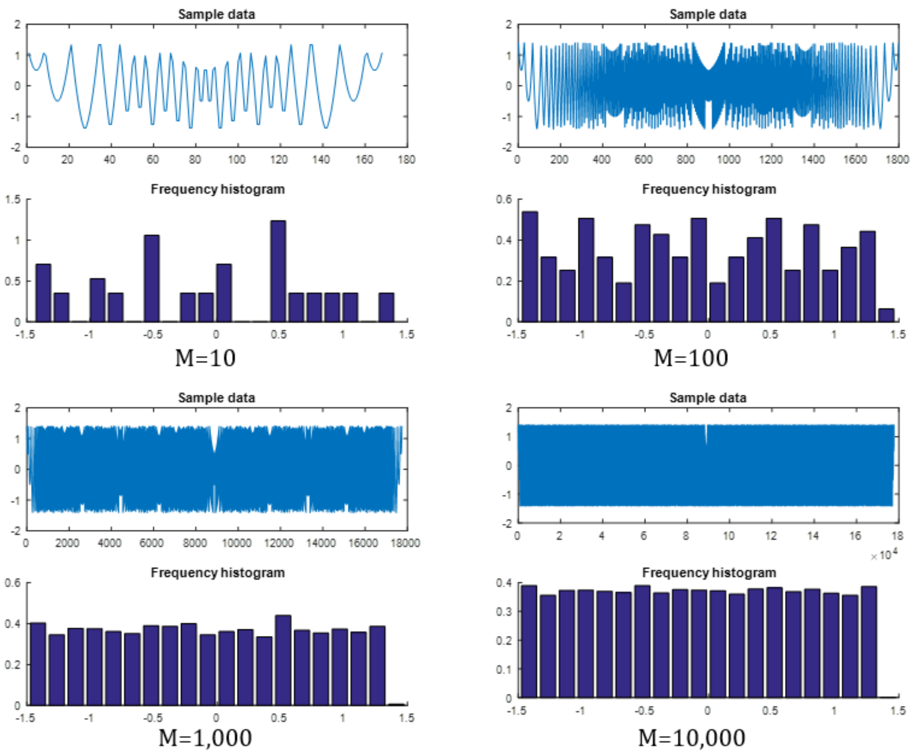
M=10

M=100

M=1,000

M=10,000

Fig. 24. The probability distribution of elements in set Y when the tool center coincides with a voxel center.



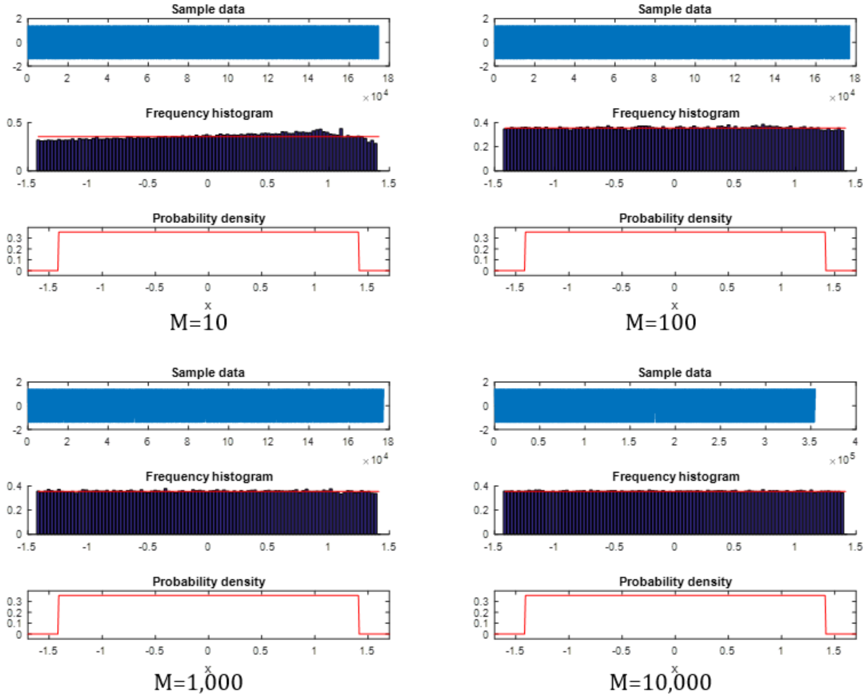
M=10

M=100

M=1,000

M=10,000

Fig. 25. The probability distribution of elements in set Y when the tool center coincides with a voxel vertex.



M=10

M=100

M=1,000

M=10,000

Fig. 26. The probability distribution of elements in set Y when the tool center is located randomly on the voxel grid.

6.3 Frequency distribution of the length of circular segments

Fig. 27 shows how the tool boundary was broken into many segments by the voxel grid. When the M is large enough, the length of each circular segment tends to be infinitely small, and the length of a circular segment can be substituted by the length of the chord drawn between the ends of the segment.

The probability distribution of the chord lengths is shown in Fig. 28, where two distinct portions are visible: the first portion is a uniform distribution located on $[0, a]$, and the second portion is a monotonically decreasing curve on the interval $[a, \sqrt{2}a]$. The probability density approaches infinity at a. It means the greatest possibility of the chord length is equal to the voxel length a.

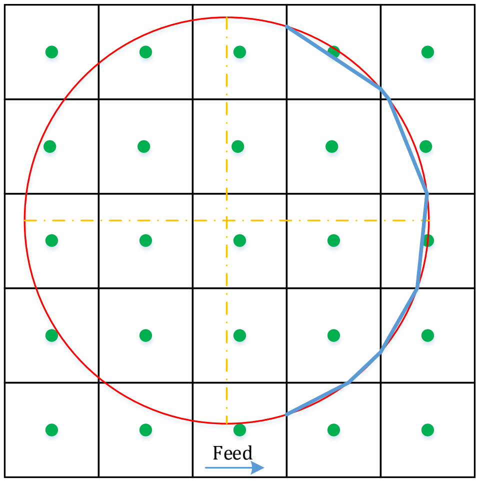


Fig. 27. The tool boundary is broken into many segments by the voxel grid.

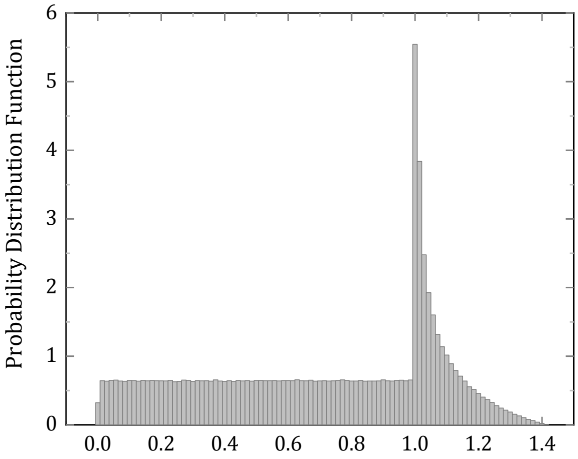


Fig. 28. The probability distribution function of the chord lengths (unit of x-axis: a; unit of y-axis: $1/a$).

7. Error Modeling of MRR computation

Modeling and simulation are two powerful tools in scientific research and engineering application (Allen, 2004, Nie et al. , 2015, Nie et al. , 2018, Stavropoulos and Chryssolouris, 2007). Due to the discreteness of the voxel model, a computer cannot make an accurate judgment of voxel removal when a voxel center is located close to the tool boundary. Thus, an error band should be considered when performing analysis and modeling the number of voxels removed in one-step.

As shown in Fig. 29, voxels to be removed is located in one of three domains: E, O, and I. Domain E is the error band, which is an annulus centered around the tool boundary with radius interval $[R-\varepsilon, R+\varepsilon]$; domain O is an annulus that is completely external to the tool boundary, concentric with domain E, with radius interval $\left[R+\varepsilon, R+\sqrt{2}a+\varepsilon\right]$; and domain I is an annulus that is completely internal to the tool boundary, concentric with domain E, with radius interval $\left[R-\sqrt{2}a-\varepsilon, R-\varepsilon,\right]$. When the step length is just equal to the voxel size, the six possible domain changes are shown in Fig. 30: (1): $O \rightarrow O$, (2): $O \rightarrow E$, (3): $O \rightarrow I$, (4): $E \rightarrow I$, (5): $E \rightarrow E$, (6): $I \rightarrow I$.

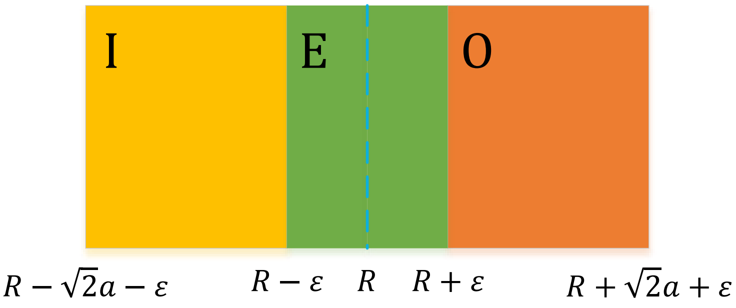


Fig. 29. Three domains: E, O and I.

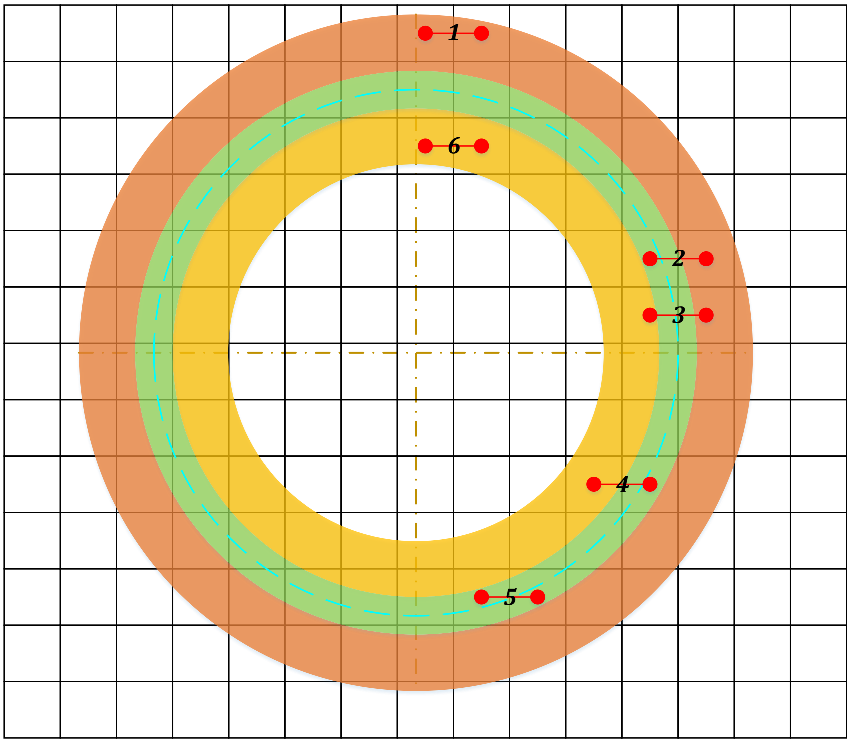


Fig. 30. Possible domain changes:

(1): $O \rightarrow O$, (2): $O \rightarrow E$, (3): $O \rightarrow I$, (4): $E \rightarrow I$, (5): $E \rightarrow E$, (6): $I \rightarrow I$.

The definition of the contribution function S_i^j of the i^{th} voxel in j^{th} step is presented in Equation (34). The contribution function can be used to assist in voxel judgment as follows: when the domain changes clearly from O to I ($O \rightarrow I$), the value of the contribution function is 1; when the domain remains unchanged ($O \rightarrow O, E \rightarrow E, I \rightarrow I$), the value of the contribution function is 0; when the domain changes into or out of the error range ($O \rightarrow E, E \rightarrow I$), the voxel status cannot be judged accurately and the contribution function can be either 1 or 0 with the same probability of 0.5. Usually, $O \rightarrow E$ is prior to $E \rightarrow I$ when the tool moves through the voxel grid; however, there exist several distinct paths that a voxel center may take as it transitions through the $O - E - I$ domains. The possible state changes are as follows: (a): one or more $E \rightarrow E$ transitions follow an $O \rightarrow E$ transition and the voxel center never reaches domain I; (b): a single $E \rightarrow I$ transition follows the $O \rightarrow E$ transition; (c): one or more $E \rightarrow E$ transitions occur between the $O \rightarrow E$ transition and the $E \rightarrow I$ transition. All the algorithms are shown in Table 5.

$$S_i^j = \begin{cases} 0 & O \rightarrow O \\ 1|0 & O \rightarrow E \\ 1 & O \rightarrow I \\ 1|0 & E \rightarrow I \\ 0 & E \rightarrow E \\ 0 & I \rightarrow I \end{cases} \tag{34}$$

$$N^j = \sum_{i=1}^{N^*} S_i^j \tag{35}$$

Table 5. The algorithm of contribution function on the error domain.

$(\boldsymbol{j}-\mathbf{1})^{\text{th}}$ step	$\boldsymbol{j}^{\text{th}}$ step
$S_i^{j-1}(O \rightarrow E) = 1, (50\%)$	$S_i^j(E \rightarrow I) = 0$
$S_i^{j-1}(O \rightarrow E) = 0, (50\%)$	$S_i^j(E \rightarrow I) = 1$

As computational error ε is quite small with respect to the voxel size a, most of the voxels in domain O will directly experience $O \rightarrow I$, and only a few voxels experience other domain transitions. The modeling results of the flat-end milling with single precision are shown in Fig. 31. It can be seen that the two curves are basically the same in the amplitude and oscillation, the period of the modeling curve is 3, which is close to the computed curve. One distinction between the two curves is the symmetry of amplitude: the modeling curve is transversely zygomorphic, and the previously computed curve not. The reason is complex, and one possible explanation is the probability is not 0.5 in $O \rightarrow E$ judgment.

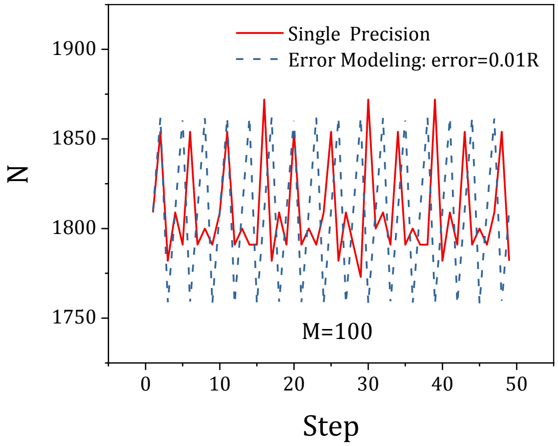


Fig. 31. Comparison between error modeling prediction and previously computed data.

8. Conclusions

This research presented an analysis and modeling of voxel-based MRR computation of the milling process. The cause of oscillation in voxel-based MRR computation is the misjudgment of voxel removal due to the discrete nature of the voxel model. Discrepancies between theoretical MRR and computed MRR were investigated using both single and double precision computation of the number of voxels removed in one feed step (N) of milling processes. By the Cavalieri's principle, discrete mathematical expressions of MRR were respectively derived for flat-end milling and ball-end milling. MRR has a linear relationship with the variable quantity of voxels removed (N) in one feed step, so the analysis of MRR can be equivalent to the analysis of N as other conditions keep invariable.

A series of simulation experiments were conducted on GPGPU by using computer programs we developed to simulate the two millings. Discrepancies between these two values were investigated of N. The dependence with respect to the ratio between the radius of the tool and the voxels size, and the use of single or double precision floating-point computations were studied. Results indicate that the cause of oscillation in discrete MRR computation is the misjudgment of voxel removal due to computational error. Comparisons on N-curves indicate that double precision can improve the ability of judgment on voxel removal, and decrease the standard deviation of N curves. Besides, ball-end milling has good computational accuracy with a lower standard deviation than flat-end milling.

Error analysis conducted on the curves of N revealed that the relative range decreased with the increasing scale factor M, and the mean absolute error increased with the increasing M. Standard deviation analysis was conducted through a simple mathematical model on the variance of random variables. It turns out that the variance (standard deviation) of ball-end milling is far less than the variance of flat-end milling. This conclusion is perfectly compatible with the computed results on the difference of ball-end and flat-end millings. Finally, analysis of the periodic characteristics reveals that all the curves approximately have a minimal period of 2.9. The probability distribution of normal distance from the voxel center to tool boundary is studied to understand the error distribution when the voxel model makes a decision for voxel removal or not. It turns out that the normal distance obeys a uniform distribution when M is large enough.

For better understanding and improving MRR computation precision in voxel-based CAM, a computational error model was proposed to describe the behavior of N-curves. The error domain was embedded into the model, and the contribution function was used to count the valid number of voxels removed. Comparison between the modeling result and the previously computed result shows that the model can well predict the periodic variation of N curves in the milling process.

The error model demonstrates the error source of MRR computation and offers a promising method to accuracy control in CAM. In our future work, the criterion for voxel size limit will be set up in the voxel-based CAM to guarantee the MRR computation accuracy.

Acknowledgments

This work was supported by NSF grants IIP-1631803, CMMI-1646013, CMMI-129742, and DGE-1650044.

Reference

Abi-Chahla F. Nvidia's CUDA: The End of the CPU?'. June; 2008.
Allen MP. Introduction to molecular dynamics simulation. Computational soft matter: from synthetic polymers to proteins. 2004;23:1-28.
Bingol OR, Krishnamurthy A. NURBS-Python: An open-source object-oriented NURBS modeling framework in Python. SoftwareX. 2019;9:85-94.
Choudhury SK, Appa Rao IVK. Optimization of cutting parameters for maximizing tool life. International Journal of Machine Tools and Manufacture. 1999;39:343-53.
Gordon WJ, Riesenfeld RF. B-spline curves and surfaces. Computer aided geometric design: Elsevier; 1974. p. 95-126.
Hossain MM, Tucker TM, Kurfess TR, Vuduc RW. Hybrid Dynamic Trees for Extreme-Resolution 3D Sparse Data Modeling. Parallel and Distributed Processing Symposium, 2016 IEEE International: IEEE; 2016a. p. 132-41.
Hossain MM, Vuduc RW, Nath C, Kurfess TR, Tucker TM. A graphical approach for freeform surface offsetting with GPU acceleration for subtractive 3D printing. ASME 2016 11th International Manufacturing Science and Engineering Conference: American Society of Mechanical Engineers; 2016b. p. V002T04A31-VT04A31.
IEEE. IEEE Standard for Floating-Point Arithmetic. IEEE Computer Society; 2018.
Jang D, Kim K, Jung J. Voxel-based virtual multi-axis machining. The International Journal of Advanced Manufacturing Technology. 2000;16:709-13.
Kara S, Li W. Unit process energy consumption models for material removal processes. CIRP Annals-Manufacturing Technology. 2011;60:37-40.
Ko TJ, Kim HS, Park SH. Machineability in NURBS interpolator considering constant material removal rate. International Journal of Machine Tools and Manufacture. 2005;45:665-71.
Konobrytskyi D. Automated CNC Tool Path Planning and Machining Simulation on Highly Parallel Computing Architectures. 2013.
Krishnamurthy A, Khardekar R, McMains S. Optimized GPU evaluation of arbitrary degree NURBS curves and surfaces. Computer-Aided Design. 2009;41:971-80.
Kurfess T. Multi-Axis Voxel-Based CNC Machining of Centrifugal Compressor Assemblies. American Helicopter Society Forum 742018.

Kurfess TR, Tucker TM, Aravalli K, Meghashyam P. GPU for CAD. Computer-Aided Design and Applications. 2007;4:853-62.
Lynn R, Contis D, Hossain M, Huang N, Tucker T, Kurfess T. Extending access to HPC manufacturability feedback software through hardware-accelerated virtualized workstations. Flexible Automation (ISFA), International Symposium on: IEEE; 2016. p. 271-7.
Lynn R, Contis D, Hossain M, Huang N, Tucker T, Kurfess T. Voxel model surface offsetting for computer-aided manufacturing using virtualized high-performance computing. Journal of Manufacturing Systems. 2017;43:296-304.
Lynn R, Dinar M, Huang N, Collins J, Yu J, Greer C, et al. Direct Digital Subtractive Manufacturing of a Functional Assembly Using Voxel-Based Models. Journal of Manufacturing Science and Engineering. 2018a;140:021006.
Lynn R, Medrano RL, Contis D, Tucker T, Kurfess T. Automated Multi-User Analysis of Virtualized Voxel-Based Cam on Shared GPUs. 2018b:V002T07A1.
Nie Z, Wang G, Lin Y, Rong YJJoME, Performance. Precision Measurement and Modeling of Quenching-Tempering Distortion in Low-Alloy Steel Components with Internal Threads. 2015;24:4878-89.
Nie Z, Wang G, Liu D, Rong Y. A Statistical Model of Equivalent Grinding Heat Source Based on Random Distributed Grains. Journal of Manufacturing Science and Engineering. 2018;140:051016--13.
Sanders J, Kandrot E. CUDA by Example: An Introduction to General-Purpose GPU Programming, Portable Documents: Addison-Wesley Professional; 2010.
Stavropoulos P, Chryssolouris G. Molecular dynamics simulations of laser ablation: the Morse potential function approach. International Journal of Nanomanufacturing. 2007;1:736-50.
Tarbutton JA, Kurfess TR, Tucker TM. Graphics based path planning for multi-axis machine tools. Computer-Aided Design and Applications. 2010;7:835-45.
Tikhon M, Ko TJ, Lee SH, Kim HS. NURBS interpolator for constant material removal rate in open NC machine tools. International Journal of Machine Tools and Manufacture. 2004;44:237-45.
Tsai Y-F, Farouki RT, Feldman B. Performance analysis of CNC interpolators for time-dependent feedrates along PH curves. Computer Aided Geometric Design. 2001;18:245-65.
Wikipedia. Cavalieri's principle. 2018.
Zhang QG, Greenway RB. Development and implementation of a NURBS curve motion interpolator. Robotics and Computer-Integrated Manufacturing. 1998;14:27-36.