Int J Adv Manuf Technol (2010) 46:465–490
DOI 10.1007/s00170-009-2107-7

ORIGINAL ARTICLE

Octree-based NC simulation system for optimization of feed rate in milling using instantaneous force model

K. P. Karunakaran · Rohitashwa Shringi · Deepak Ramamurthi · C. Hariharan

Received: 2 July 2008 / Accepted: 11 May 2009 / Published online: 13 June 2009

Abstract An octree-based (numerical control) NC simulation (Oct-OAC) system developed for end milling has two major applications: (1) NC verification and (2) optimization of the cutting parameters, viz., spindle speed, N (s^{-1}), and feed rate, f (ms^{-1}). Oct-OAC has a geometric modeling module to simulate the geometry of material removal process. Every object in the machining environment such as cutter, instantaneous workpiece, swept volume, etc. is stored as octree, an inexact representation of solid. Using this module, one can predict the geometry of the material removed at any instant of time and update the geometry of the blank subsequently. Optimization of cutting parameters using Oct-OAC is achieved through optimization module using a mechanistic model for computation and prediction of the cutting forces at any instant. The basic input for this module is the geometry of the contact surface between the cutter and workpiece which comes from the geometric modeling module using an octree-based solid modeler. It is through this contact surface that the cutting forces are passed from the workpiece onto the cutter and vice versa. The mechanistic modeling module can predict the instantaneous cutting forces from the instantaneous contact geometry and other process parameters like material combination of cutter–workpiece, parameters defining cutter geometry, and current cutting parameters such as N and f. Using this prediction, it will modify the cutting parameters for maximizing the material removal rate. This way, the mechanistic modeling module does what an adaptive controller will do with the help of force sensing. Therefore, the NC program optimization done using the Oct-OAC system is actually off-line adaptive control.

Keywords NC verification · Image space simulation · Octree representation · CSG · Boundary representation · Voxel representation · Feed rate optimization

Nomenclature

Ø	Radial immersion angle
θ	Rotation angle
δ_θ	Angular increment
β	Helix angle
ψ	Lag angle
κ	Axial immersion angle
a	Angle of segment OP measured counter clockwise from the X-axis
b	Angle of segment QS measured clockwise from the Z-axis
BRep	Boundary representation
CSG	Constructive solid geometry
CL	Cutter location
CL_i	ith cutter location data giving the next position (x_i, y_i, z_i) and orientation (i_i ,j_i, k_i) of cutter
d	Diameter of cutter measured from point R
dF_t	Elemental tangential force
dF_r	Elemental radial force
dF_a	Elemental axial force
(e,f)	coordinates of point A, the center of fillet arc

K. P. Karunakaran · R. Shringi (✉)
Computer Graphics Laboratory,
Department of Mechanical Engineering,
Indian Institute of Technology Bombay,
Powai,
Mumbai 400076, India
e-mail: rshringi@iitb.ac.in

D. Ramamurthi · C. Hariharan
Department of Computer Science and Engineering,
National Institute of Technology,
Tiruchirappalli 620016, India

f_t	Feed/tooth
F_t	Tangential cutting force
F_{th}	Thrust force
F_R	Resultant force
f_{b_cur}	Bending stress current
f_{r_opt}	Optimum feed rate
f_{r_cur}	Current feed rate
f_{s_cur}	Shear stress current
f_{s_a}	Allowable shear stress
F_{max_limit}	Upper force limit
$F_{predicted}$	Predicted force
$g(u)$	Parametric definition of 2D profile of cutter in terms of u
h	Non-parallel height of cutter
HSD	Hierarchical space decomposition
h_1	Shank length of cutter
MRR	Material removal rate
N_θ	No. of angular increments
N_f	Number of flutes
n	Number of tool paths
N	Spindle rotating speed
ORep	Octree representation
OP	First straight line segment corresponding to the bottom cutting edge of cutter
P_{avg}	Average power
PQ	Fillet arc of cutter
$p(u,v)$	Biparametric definition of the surface of the cutter
QS	Second straight line segment corresponding to the side cutting edge of the cutter
r	Radius of circular segment PQ of cutter
R	Point of intersection of first line segment OP and third line segment QS
s_1	Length of first straight line segment OP
s_2	Length of arc segment PQ
s_3	Length of second straight line segment QS
s_4	Length of third straight line segment ST
ST	Straight line shank corresponding to the shank of the cutter
t_c	Uncut chip thickness
t_d	Disk thickness
USD	Uniform spatial decomposition
v	Parameter defining the rotation of $g(u)$ about the Z-axis
VRep	Voxel representation
V_c	Cutting speed

1 Introduction

Present computer-aided manufacturing (CAM) systems which can determine cutting tool locations and generate numerical control (NC) programs for machining are indispensable for operating computer numerical controlled (CNC) machine tools. However, the current technology does not consider important physical behaviors of real machining process such as cutting forces, etc. Thus, the generated NC programs may produce a part which fails to meet quality requirements or cause damage such as cutter breakage. Further evaluation of physical cutting performance is not possible with present systems which are essential for the optimization of NC programs.

Most CAM systems only allow one to set machining parameters (spindle speed, feed rate, etc.) once for an operation (for example, to mill a cavity). To obtain a stable process and avoid cutter damage, accuracy violation, excessive deformation, vibration, or failure of fixing, the selected parameters are often so conservative that efficiency is very low during a large part of the process. It is always difficult to modify the NC program manually either because it requires complicated calculation or the program is fairly long. Hence, there is need for automated optimization of parameters.

Existing CAM software such as CGTech's OptiPath [1] and Mastercam's Hi-feed [2] use volume of material being removed for the feed rate planning; however, using the material removal rate (MRR) model is inherently incapable of determining the instantaneous cutting force direction which is essential for optimized cutting and for arriving at optimal values of cutting parameter such as feed rate. Hence, there is a need to develop such a system that considers the physical behavior of the machining process. In this paper, the authors have presented an octree-based NC simulation system (Oct-OAC) for optimization of feed rate in milling using instantaneous force model. With this system, the physical model of machining can be integrated with the geometric modeling methods to determine if the cutting conditions are safe. The optimization of cutting parameters during NC machining using developed system can improve the productivity and efficiency of CNC machines.

2 Literature review of existing systems of NC simulation

Research efforts [3-26] have concentrated on (1) image-space-based, (2) vector-based, and (3) object-space-based methods using solid modeling representation for NC simulation and verification, as shown in Fig. 1.

2.1 Image-space-based NC simulation

Image space (also called as z-map) approach is the most widely used for NC simulation. Anderson [3] developed a 3D histogram method to simulate three-axis machining for

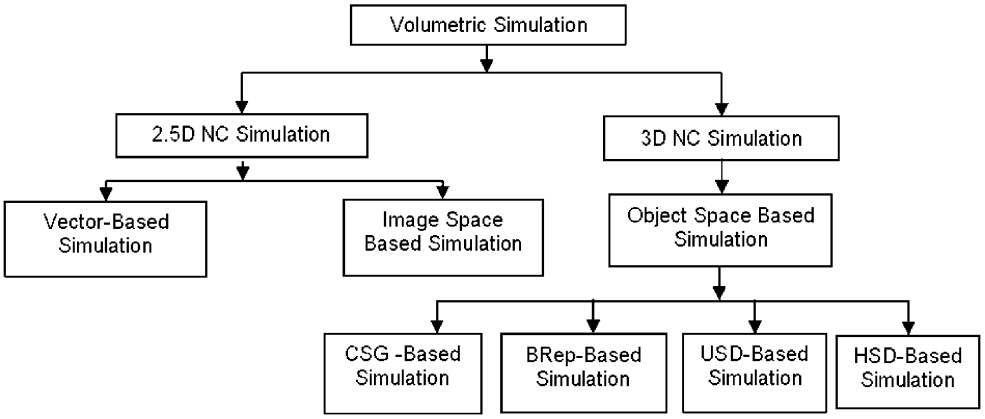


Fig. 1 Classification of volumetric NC simulation systems

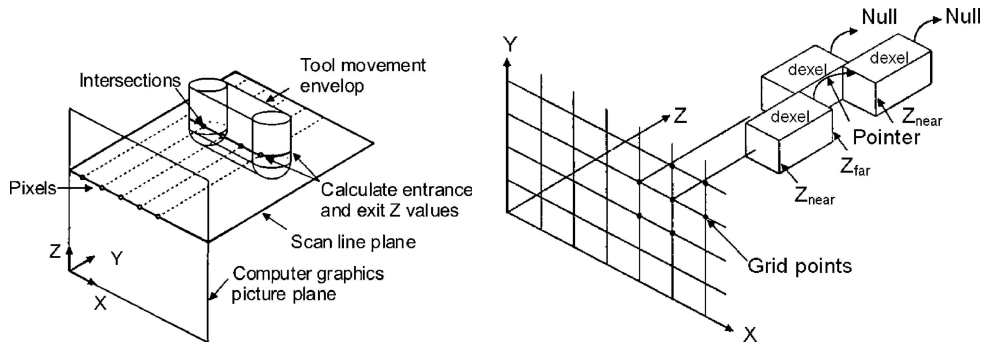
collision detection. The limitation of a histogram approximation for 3D shapes is that it cannot represent shapes that have undercutting, since there must be a unique z value for a given x, y value. Wang and Wang [4] suggested image space approach using a variation of the standard Z-buffer hidden surface algorithm. A vector is drawn at each pixel that is normal to the plane of the screen. Intersections of these vectors with tool path envelopes are calculated with a scan-line algorithm. The buffer is provided for each pixel in a given graphical view of the part, and each tool movement updates the pixel information that it passes over. The workpiece Z-buffer is modified by comparing it with the swept volume Z-buffer and performing Boolean operation during simulation. Each tool movement changes the graphic image of the workpiece to show the cutting action. Van Hook [5] developed NC milling display using an extended Z-buffer data structure. His method differs from Wang's in that instead of intersecting scan lines with swept volume envelopes, pixel image of the cutting tool is pre-computed and Boolean subtractions of the cutter from the workpiece is done along a tool path. Atherton et al. [6] extended Van Hook's approach to handle five-axis machining.

In the image space methods (Fig. 2) of Wang, Van Hook, and Atherton, errors not visible in the chosen viewing direction are undetected, and generating another view of the part requires rerunning the entire simulation, i.e., reconstruction of the entire data structure is needed [7]. The image-based systems can give a 3D view of the cutting process and the final part from the original viewing direction, but since only a Z-buffer image is maintained, a true solid model is not available. It also allows for fast computation and approximate calculation of volume removal rates, but its computation time and memory consumption increases drastically if the accuracy is to be enhanced. Commercially reported NC simulator VeriCUT (CGTech, USA) uses image space approach [1].

2.2 Vector-based NC simulation

Chappel [8] proposed NC simulation method based on the vector clipping approach also known as the "point-vector" technique. In this method, the blank can be visualized as a bundle of discrete outward normal vectors or grass of the component buried inside the blank. An analogy can be made to mowing a field of grass. Each vector in the simulation corresponds to a blade of grass "growing" from the desired object. As the simulation progresses, the blades are "mowed down." The length of the final vectors correspond to the amount of excess material (if above the surface) or the depth of the gouge (if below the surface) at

Fig. 2 Image-space-based simulation



Intersection of z-map of image with tool movement [4]

Extended z-map as linked list of dexels [5]

that point. In contrast to image space approach, this technique allows for toleranced dimensional verification, as discrepancies can be measured relative to the surface normal vectors of the designed part. Oliver and Goodman [9] developed a system similar to Chappel's, which uses a computer graphics image of the desired surface to select the points. This image space is then used as the basis for the simulation. Jerard et al. [7, 10] and Drysdale et al. [11] used an approach that shares characteristics of the methods of Chappel and Oliver, but it also contains features that improve efficiency and allow the user to make trade-offs between the accuracy of the approximate simulation and the CPU time.

Discrete vector methods (Fig. 3) have the advantage of being able to handle five-axis machining, but the limitation is that it does not directly generate a solid model for the machined part in the simulation process and is not suitable for cases when the normal vector directions of a surface at some locations on the part model dramatically change during the simulation process.

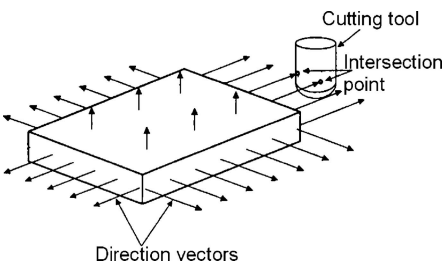
2.3 Object-space-based NC simulation

There are various schemes of representing an object, each with its advantages, limitations, and hence specific applications. Some represent the object exactly and some are only inexact. Some representations are user-friendly, while the others are system-friendly. Several investigators [12-26] have used 3D object space solid modeling for NC simulation. These can further be classified into (1) constructive solid geometry (CSG)-based, (2) boundary representation (BRep)-based, (3) uniform spatial decomposition (USD)-based, and (4) hierarchical space decomposition (HSD)-based (Fig. 1).

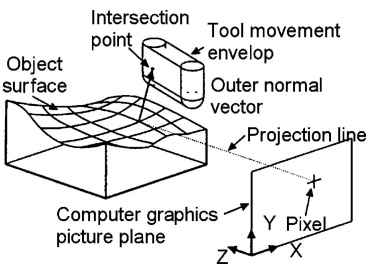
Voelcker and Hunt did an exploratory study of the feasibility of using part and assembly description language (PADL) CSG modeling system for simulation and verification of NC programs [7]. Sungurtekin and Voelcker [12] developed a simulation system for milling machines that uses the PADL-2 CSG modeling system for maintaining a 3D model of the stock-in-progress. Spence and Altintas [13, 14] proposed a 2.5 axis milling process simulation system using CSG for part representation. CSG approach is popular because this method can complete Boolean operation of any 3D part model relatively easily and accurately. The CSG has very high user-friendliness but is extremely poor in system-friendliness. Furthermore, the limitation of CSG-based approach is its high computational expense. The cost of simulation is reported to be $O(n^4)$, where n is the number of tool movements [15].

Boundary representation (BRrep) is the dominant choice for commercial modelers. Currently, a number of systems based on the spatial Technology Inc. ACIS solid modeler kernel have been marketed (AutoCAD, SolidEdge). Feng et al. [16] used commercial computer-assisted design (CAD)/CAM software CATIA to determine the boundary of engaged surface between tool and workpiece for three-axis chip geometry calculation. The in-cut segments were computed using the boundary curves. However, the geometric calculations were not based on an informationally complete BRep model of in-process part. Using a BRep polyhedral-based solid modeler, ball end milling simulation was reported in Mounayri et al. [17]. The removed volume was computed and the cutting edge, modeled with a cubic Bezier curve, was intersected with the volume to find the in-cut segments.

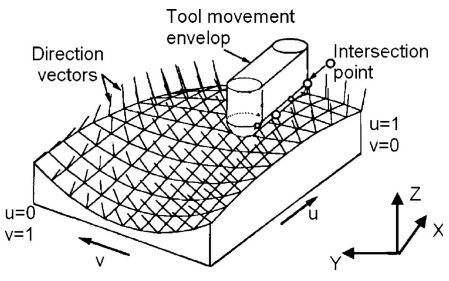
A BRep-based machining simulation eliminates the need to convert the model to another representation. As BRep is an evaluated representation with suitable redundant data stored, algorithms work very fast. However, BRep models are huge and it is extremely difficult for a user to directly input the data required to form the BRep model. Hence, BRep has very high system-friendliness but is extremely poor in user-friendliness. Furthermore, the disadvantage of BRep modelers for machining simulation is long running time. Parallel processing and BRep topology to reduce



Discrete vectors normal to surface of part to simulate NC machining, [8]



Discrete normal vectors projected on z map of part surface [9]



Tool path envelopes intersected with normal vectors [10]

Fig. 3 Vector-based simulation

overall wall clock running time of BRep-based machining simulations for rough machining of the 2.5D pockets is reported by Spence et al. [13] and Fleisig et al. [18]. It was shown that a BRep modeler more directly supports extraction of the tool immersion angle intervals required for machining process simulation. It was estimated that the BRep growth rate is $O(n^{1.5})$ for a total of n tool paths.

USD-based scheme is most suitable when the solid is not amenable to representation through a set of inequalities. Examples are clouds, geological and geographic objects, biomedical objects, etc. They are also useful to represent objects in multi-resolution useful in finite element analysis, volumetric NC simulation, etc. There are several solid representation schemes based on spatial decomposition. All of them are invariably *inexact* representation. Any spatial decomposition method is associated with a *universe* and *resolution*. The universe is expected to contain all objects of interest. The spatial decomposition can be *uniform* or *adaptive* (also known as *hierarchical*). When the object is represented as a collection of simple shapes of uniform size, it is known as USD or exhaustive enumeration or voxel representation (VRep).

Vergeest et al. [19] developed a simulation system based on the VRep. In VRep, the set operations can very easily be executed which enables to generate fast updating of a part model. However, to increase the accuracy of cell decomposition in the model, the size of the voxels needs to be reduced, resulting in large memory space requirement for storing the model.

A HSD-based model is similar to a USD in that it represents a solid as an aggregate of hexahedra, but it reduces the memory requirement considerably by dividing the space adaptively. HSD models may be of different types, such as octree, bintree or polytree, etc. In an octree representation, the universe (a cube that contains the object) is subdivided into eight parts recursively. Each cube is one eighth of its parent cube in size and is called an octant. Furthermore, all the octants can be represented as the nodes of a tree in which every node has eight branches (Fig. 4). This tree is called octree [20].

Several octree-based modeling methods and 3D decomposition approaches have been reported in literature [21-24]. Ayala et al. [21] developed a polytree data structure by adding surfaces to octree's node in order to solve the major limitations of the classical octrees. A similar modeling scheme is also reported by Carlbom et al. [23]. Their research focused on the application of object representation with the quadtree technique, which is a special case of octree in 2D space. Brunet et al. [22] developed an extended octree representation for solid modeling. This method incorporates terminal nodes with higher geometric complexity; hence, the calculation for certain tool shapes and tool paths can be complex within this model. Leu et al. [25] implemented extended quadtrees for dimensional verification of NC machining profiles. However, their method can handle only prismatic or 2.5D object. Roy and Xu [26] applied the extended octree modeling technique to machining simulation. However, Roy's model search cutting area uses quadtree in an envelope projected on an xy plane. It is thus difficult to say when an octree structure was used for machining simulation (Table 1).

Although octree representation (ORep) is an inexact representation, one can choose any desired resolution. There are other advantages which make ORep ideally suited for NC simulation. Memory required by octree is independent of the number of NC blocks (primitives and operators). For a given resolution, memory required depends only on the surface area of the component. Frequently used operators (such as Boolean operations in NC simulation) rendered display of any isometric view trivial since they involve only tree traversals performing binary Boolean operations on the corresponding leaf nodes. Furthermore, all the computations take place in binary or unsigned integer modes and this minimizes space and time complexity of the algorithms. It inherently lends itself to parallelization and is able to handle non-manifold and self-intersecting objects. No commercial HSD-based NC simulation system is available, though these have been reported in literature [26].

2.4 Scheduling of feed rate in machining

Research work on process planning by Sungurtekin and Voelcker [12] indicate the importance of using the desired cutting force in determining the machining feed rate. Using the solid-modeler-based milling simulation, he addressed geometric cutter path verification, but the process mechanics such as cutting forces and deflections were ignored. Some of the first work on feed rate planning was by done by Wang et al. [4]. He used a Z-buffer representation (image-based scheme) of the workpiece and a simple volumetric model to relate cutting force to the metal removal rate. However, the MRR model based on the empirical equation representing the cutting force as proportional to the volume of removed metal (overall chip volume) can only estimate the average cutting forces and does not provide the instantaneous force vector, i.e., force direction cannot be determined which is necessary for estimation of the tool deflection. Instantaneous cutting forces and torque are proportional to the cross-sectional area chip thickness and not the overall chip volume. During semi-finish and finish cutting, it is important to know the magnitude and direction of the cutting force. Feed rate scheduling within the NC Program block is also not considered.

Takata et al. [27] utilized a Z-buffer approach for the workpiece/cutter geometry description, a swept volume

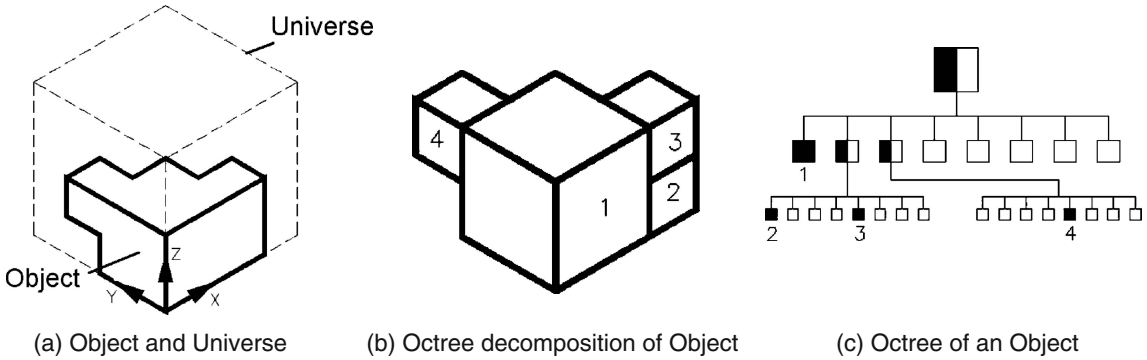


Fig. 4 Octree representation of an object

generation technique [4], and combined this with the improved mechanistic model of Kline et al. [28] to investigate cutting forces and tool deflection. Boundary representation scheme was used to describe the part. The tool workpiece zone was divided into a large number of elements and a milling process model used to estimate the forces for each element. But the resolution of the geometric simulation was coarse to consider the required accuracy of modern parts. Further work by Takata [29] resulted in a process planner for 2.5 axis flat end milling using a solid modeler for intersection calculations.

Spence and Altintas [14] developed a 2.5 axis process simulation and planning systems that utilize solid modelers (CSG-based) for the workpiece geometry description. This provides a method for determining the volumetric intersection of the tool with the workpiece [30]. Feed velocities are scheduled through use of the tool/workpiece intersection data provided by the solid modeler and a ball end mill mechanistic model. This work was later transferred to the ACIS BRep modeler. Mounayri et al. [31] extended the force and torque simulation capability of a solid modeler system (BRep-based) to three-axis milling of complex parts. They use a cubic Bezier representation of the cutting edges and intersect this with the swept tool volume. Experimental verification of the system is shown for 2.5 axis semi-finish ball end milling of a die. Research work is under progress for the optimization of cutting conditions [31].

2.5 Existing commercial systems

There are several commercial systems that target the optimization of tool paths which modify the CAD/CAM-generated NC programs by post-processing each NC block.

OptiPath® The most widely used commercially available verification system, VeriCUT marketed by CGTech, USA [1], uses this method. The commercial OptiPath product uses the MRR at the current point along the tool motion to

Table 1 Comparison of various techniques for NC simulation

	Investigator	Method	Workpiece	Cutter	Swept volume
Image space pixel-based	Anderson, 1978	3D histogram	Image-space-based	Depth element	Calculated in image space
	Van Hook, 1986	Extended Z-buffer (dexel)	Depth element	Depth element	Image-space-based
	Wang and Wang, 1986	*z*-map	Image-space-based	CSG model	CSG model
	Atherton et al., 1987	*z*-map	Image-space-based	Image-space-based	calculated in image space
Vector-based	Chappel, 1983	Lawn mowing approach	Shape vectors	Cylinder	Not used
	Oliver and Goodman, 1990	Normal Vector	Rational B-spline	BRep	BRep
	Huang and Oliver, 1994	Dexel approach	Dexel-based	Dexel-based	Calculated in image space
	Jerard et al. 1986, 1988, 1989; Drysdale and Jerard 1987; Drysdale et al. 1989	*Z*-map	Surface point sets	Polygons	Explicit as polygons
Object-space-based	Voelcker and Hunt, 1981, 1982	CSG-based (PADL-1)	CSG	CSG	CSG
	Kawashima et al., 1991	Graftree (HSD)	Octree + CSG	Octree + CSG	Explicit as CSG
	Veergeest et al. 1994	Voxel (USD)	Voxel buffer	Voxel buffer	3D voxel space
	Jaehyun Kim,1998	Octree (HSD)	Octree	Array of points	Using interpolation
	Roy and Xu, 1999	Extended octree	Extended octree	Octree	Polyhedral BRep

adjust the feed rate. Product literature of CGTech claims that cycle times can be reduced by "as much as 50% or more" (http://www.cgtech.com). Although easy to implement, particularly using the Z-buffer method, this solution provides no guarantee that tooth breakage, which is caused by an excessive uncut chip thickness, will be avoided. As with all Z-buffer methods, the initial choice of discretization determines the resolution. Extraction of the immersion intervals is more complex. Because it is computationally fast, this approach has received wide acceptance. VeriCUT is the most popular and mature among all commercial systems.

Hi-Feed It has been reported that the existing Mastercam's [2] automated feed rate optimization reduces cycle time by optimizing feed rates based on the volume of the material being removed. Where there is more material, the feed rate decreases; less material and the feed rate increases. This lets the machine run at continuously optimized feed rates and keeps a more constant chip load on the cutter for faster job completion and more efficient tool use.

FeatureCam, developed and marketed by Engineering Geometry Systems [32], also includes built-in tables for a wide variety of materials and cutting tools and allows automatic calculation of the correct spindle speed and feed rate.

PS-Optifeed is an optimization module developed by DelCAM [33] offered at a prohibitive price without revealing its basic principles. PS-Optifeed is a feed rate optimization module for PowerMILL. This module analyzes the tool path generated within PowerMILL and automatically adjusts the feed rate to give a constant rate of material removal. With PS-Optifeed, a higher feed rate is set for lighter cuts and air moves, enabling faster machining, particularly when using high-speed mills. All these software are proprietary in nature and probably use optimization based on estimation of average forces and not peak or instantaneous forces. Hence, it is simpler to implement but less accurate. However, forces are proportional to the instantaneous volumetric removal rate and identical volumetric removal can have very different instantaneous peak values [30]. The model used for optimization must take into account the instantaneous forces.

The authors have developed the octree-based NC simulation system for off-line adaptive control of machining [34]. In this paper, the authors have presented the instantaneous force model for optimization of cutting parameters in end milling

3 Architecture of octree-based NC simulation system

In order to optimize the cutting parameters, the stock being removed at any time is required. Rather than sensing it through power consumption or tool deflection, it is inferred from the geometric modeling of machining. The information of the stock removed currently is processed along with the other details of machining by the optimization modeler to arrive at the appropriate values of the cutting parameters. The architecture of the octree-based NC simulation system (Oct-OAC) developed by the authors is shown in Fig. 5. It has the eight internal modules: (1) CSG modeler, (2) generic cutter modeler, (3) NC program file manager and translator, (4) Octree modeler, (5) Octree to BRep converter, (6) display, (7) module for swept volume calculation, and (8) optimization module.

CSG modeler A CSG modeler is required to construct the elements in the machining environment such as blank, fixture, clamps, and tool holders. It has the basic primitives of block, sphere, cone, cylinder and torus, and the Boolean operations of union, subtraction, and intersection. The CSG models created using this module are immediately converted into octree models. This modeler also has a STL interface for accepting geometries from other CAD systems

Generic cutter modeler The generic cutter defined in the NC literature [35] has been adopted here. The generic cutter is a surface of revolution of a cross-section consisting of three straight line segments and a circular fillet between the root flank and the side flank (Fig. 6). This cross-section is defined by means of eight parameters, viz., d, r, e, f, a, b, h, and h_1. By appropriately choosing these parameters, several cutter shapes such as flat end mill, dome end mill, ball end mill, angle cutters, face mill, etc. can be instantiated from the unified model of the generic cutter. Further details of this generic cutter formulation may be seen in [36]. The control flow diagram of generic cutter modeling is shown in Fig. 7.

NC program file manager and translator Oct-OAC accepts the NC programs either in CL file format or directly in the NC format (G and M codes). The CL file contains not only the path but also the information of machine kinematics and cutter details. The machine tool file contains the details about the process, machine kinematics, and the formats of the CNC machine. As the NC program does not contain the machine tool data and it only refers to a cutter by a number, the user has to input the relevant machine tool data and the cutter library. When NC program is given as input, a generic inverse processor (complement of a NC post-processor) collates the input NC block with the machine tool data and the details of the current cutter and obtains the equivalent CL data. These CL data are interpreted by a CL file interpreter and the necessary actions are carried out.

CL file contains cutter motions and auxiliary machine control information. Based on the type of movement (rapid, linear or circular) and the number of simultaneous axes moving, it calls the appropriate routine for

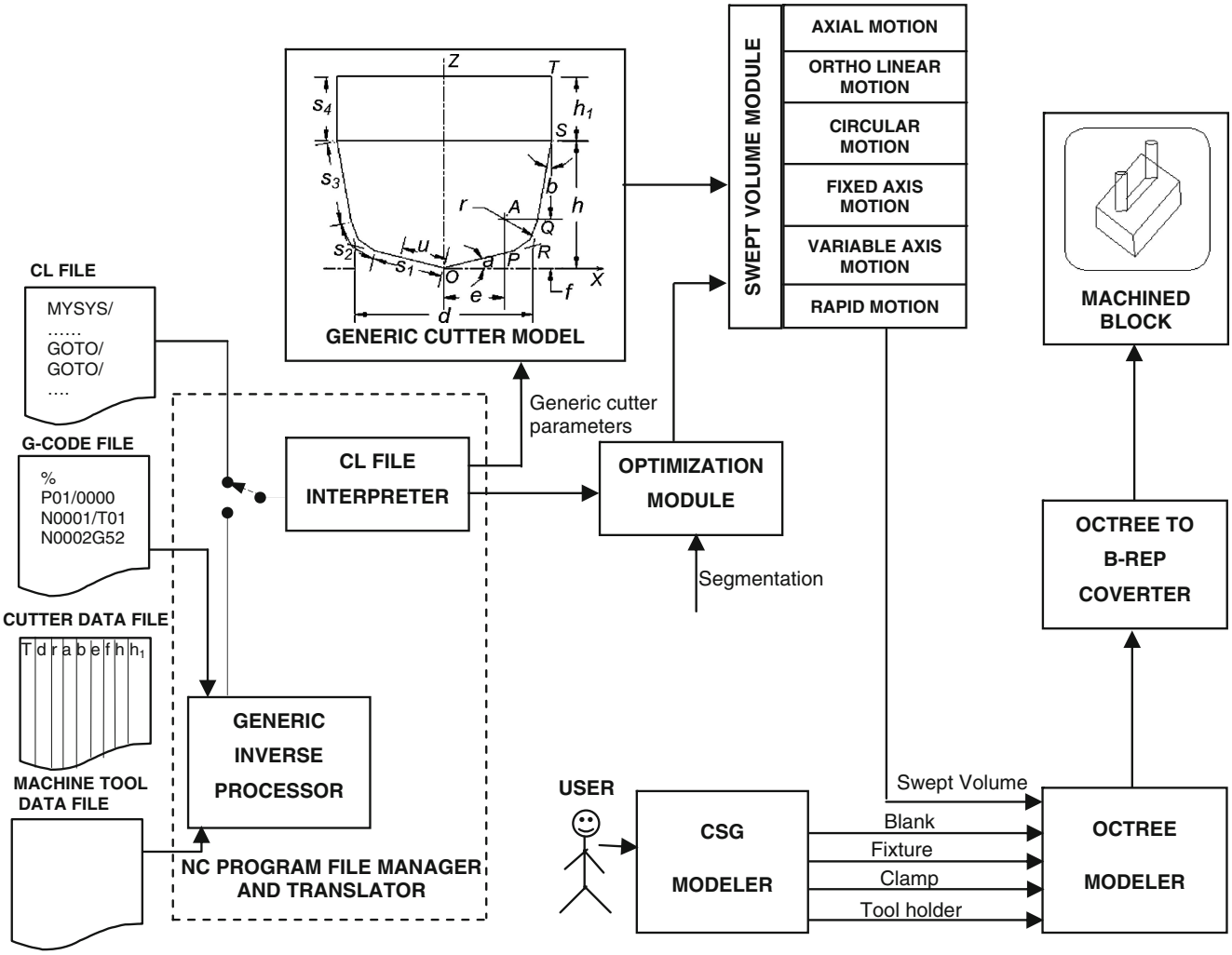


Fig. 5 Architecture of octree-based volumetric NC simulation system

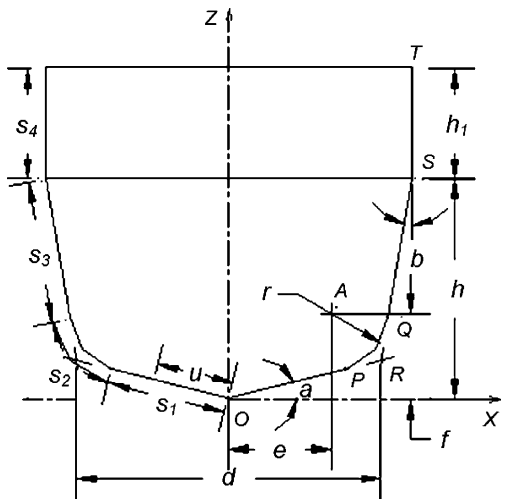


Fig. 6 Parametric profile of generic cutter

calculating the swept volume. This module first opens the given CL file and reads it into a buffer. A line at a time is scanned from this buffer and stored in a character array called current block. The *keyword* is then extracted from this current block. The keyword may be MSYS, TLDATA, FEDRAT, RAPID, GOTO, GODLTA, CIRCLE, etc. Separate routines handle different keywords. The position of the cutter is stored in a static variable after each block. This is required to know the start point of the next statement. Sometimes, it may be necessary to read the next line for the current line to make sense, for example the RAPID and the CIRCLE statements. In such cases, the next line specifies the target point of the cutter motion.

Octree modeler Octree has been chosen as the 3D representation scheme for Oct-OAC. Octree modeler has

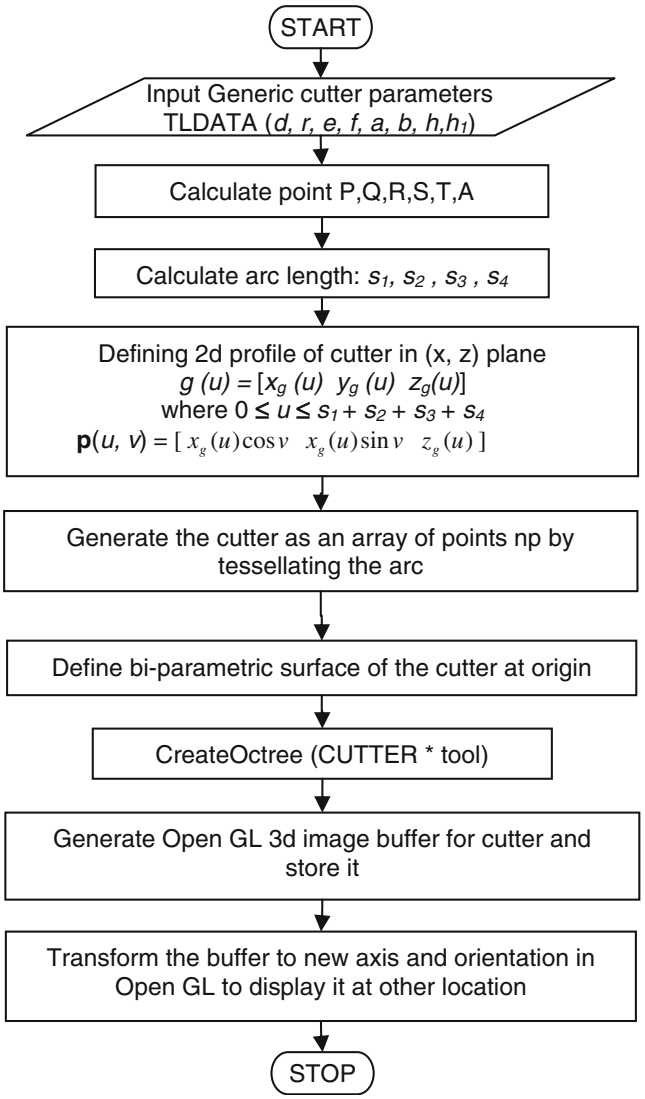


Fig. 7 Control flow diagram of generic cutter modeling

the algorithms for creating the octree model of the blank, clamps, fixtures, cutter, and swept volume generated by the cutter motions. It also provides Boolean operators (union, subtraction, and intersection) to operate on different solids. The geometric information contained in an octree is implicit and can be retrieved by use of procedures. The following simpler data structure for octree modeling is used as given below:

```
class Octree
{
    unsigned long fill_status;
    unsigned char is_point_inside;
    Octree *sons;
};
```

The proposed class design for octree node is based on using CPU memory efficiently. The following internal object representation of a typical octree object is used and is explained below:

fill_Status: fill_status stores the statuses of the eight siblings of the node. This variable is 4 byte in length on a 32-bit system. Each 4 bit contains information for the current nodes' children. In the present system, 8 (1000) and 9 (1001), respectively, denote EMPTY and FULL siblings and 0 (0000) to 7 (0111) denote the PARTIAL node. This value for a PARTIAL sibling refers to its index in the sons array where its pointer can be found. fill_status is of type unsigned long.

is_point_inside: The octree node has eight vertices, which may be inside or outside the object. is_point_inside is unsigned char and hence has 8 bits, each bit representing 1 or 0 (true or false) depending upon whether the corresponding corner of the node is inside or outside the object. All these 8 bits are initialized to 0 or False. is_point_inside is manipulated using bit-level operators.

Sons: The sons array is represented as linked list. This is an array of octree and holds the pointers of children of current octree object. This array does not hold any information about an empty or full child. From the 4 bits decimal value in fill_status variable, the index can be determined for the particular child. The sons array of a PARTIAL node is illustrated in Fig. 8.

Although bool variable stores only binary information, it actually consumes a byte (i.e., 8 bits); in other words, using bool for storing binary information is eight times less efficient. Therefore, bit operators are used extensively to manage the memory efficiently in the octree data structure by tightly packing the bits in the variables fill_status and is_point_inside.

The object is represented as a tree of the data type Octree. The root of the tree always represents a PARTIAL node. (x, y, z) is the reference point of the node which is the minimum point of the corresponding octant; s is its size. A node is always subdivided into eight octants or siblings; the nature of all these is stored in fill_status. The fill_status is declared as "unsigned long" and hence has 4 bytes or 32 bits. Therefore, 4 bits are available to describe the fill status of each of the eight siblings. All siblings are initialized to 10. If any sibling is PARTIAL, its fill status will be denoted by number in the range of 0–7, which refers to its spatial relation with the parent. If the sibling is EMPTY or FULL, its fill status will be denoted by 8 or 9, respectively. The values 11–15 are not used. The rightmost 4 bits refers to the fill status of the first sibling. Each sibling has a spatial relation with the node, as shown in Fig. 9, and hence, it is required to correlate the index of siblings and son, each having a range of 0 to 7 and 0 to (number of PARTIAL octants—1), respectively.

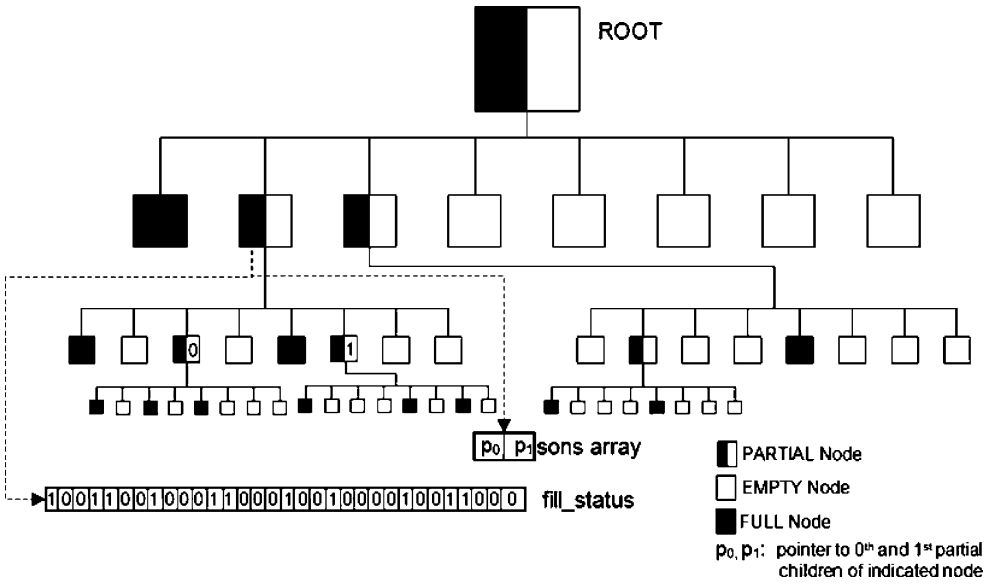


Fig. 8 Octree data structure

The procedure for generating an octree is as follows. First, a universe (cube with 2^n edge) is created to enclose completely (minimal bounding cube) the solid to be represented (Fig. 4). The length of the universe is bounded by value n where $2^{n-1} < \text{length} < 2^n$. This universe is called the root octant. Second, the root is divided into eight identical octants with an edge length of 2^{n-1} and the spatial relation of each octant with respect to the solid is investigated. If an octant is completely inside the solid, it is marked as FULL. If it is completely outside, it is marked EMPTY. If it is partially inside and partially outside, then it is marked PARTIAL. All PARTIAL octants are further subdivided into eight octants. The octants marked FULL and EMPTY are not subdivided further. Finally, the second step is repeated until the current octants being generated by the subdivision are as small as the pre-specified resolution value. The octants marked PARTIAL at the terminal nodes (octants have a minimum edge length called resolution) can be marked as FULL or EMPTY depending upon the strategy (upper limit of volume or lower limit of volume) used. Then, the collection of all octants marked by FULL will represent the solid of interest. The total number of octants to be stored in an octree is much less than that of voxel representation because the octants marked FULL or EMPTY do not take part in the subdivision. In case of octree, the number of octants needed is nearly proportional to the surface area of the object [37]. The control flow diagram of creating the octree of an object is given in Fig. 10.

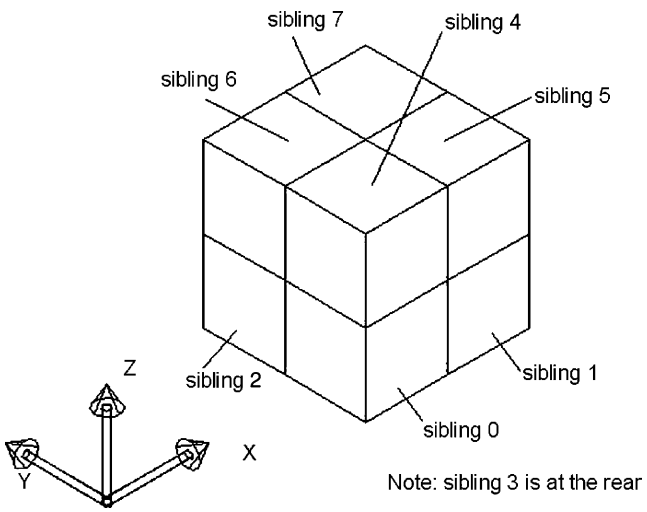


Fig. 9 Spatial relationship between the siblings and the parent node

Octree to BRep converter The designed component is generally available as a BRep model, and in order to identify the dimensional deviations and depict them, it is required to convert octree of the machined component to BRep. The conversion of octree of the machined component into BRep is much simpler and faster than the reverse. Hence, in the NC simulation system being developed, the blank is always stored as octree and the user can convert it into BRep for visual verification (zoom, rotate, and rendering features) as often as required and for dimensional verification at the end. The authors have presented detail algorithm for Octree to BRep conversion in [38]

Graphic display Two types of displays are possible in Oct-OAC. One is a quasi-rendered isometric display in one of the eight views and the other is a true 3D fully rendered display using OpenGL. A FULL node is a cube with its faces parallel to the principal planes. When it is seen in isometric view, it appears to be a hexagon. This hexagon

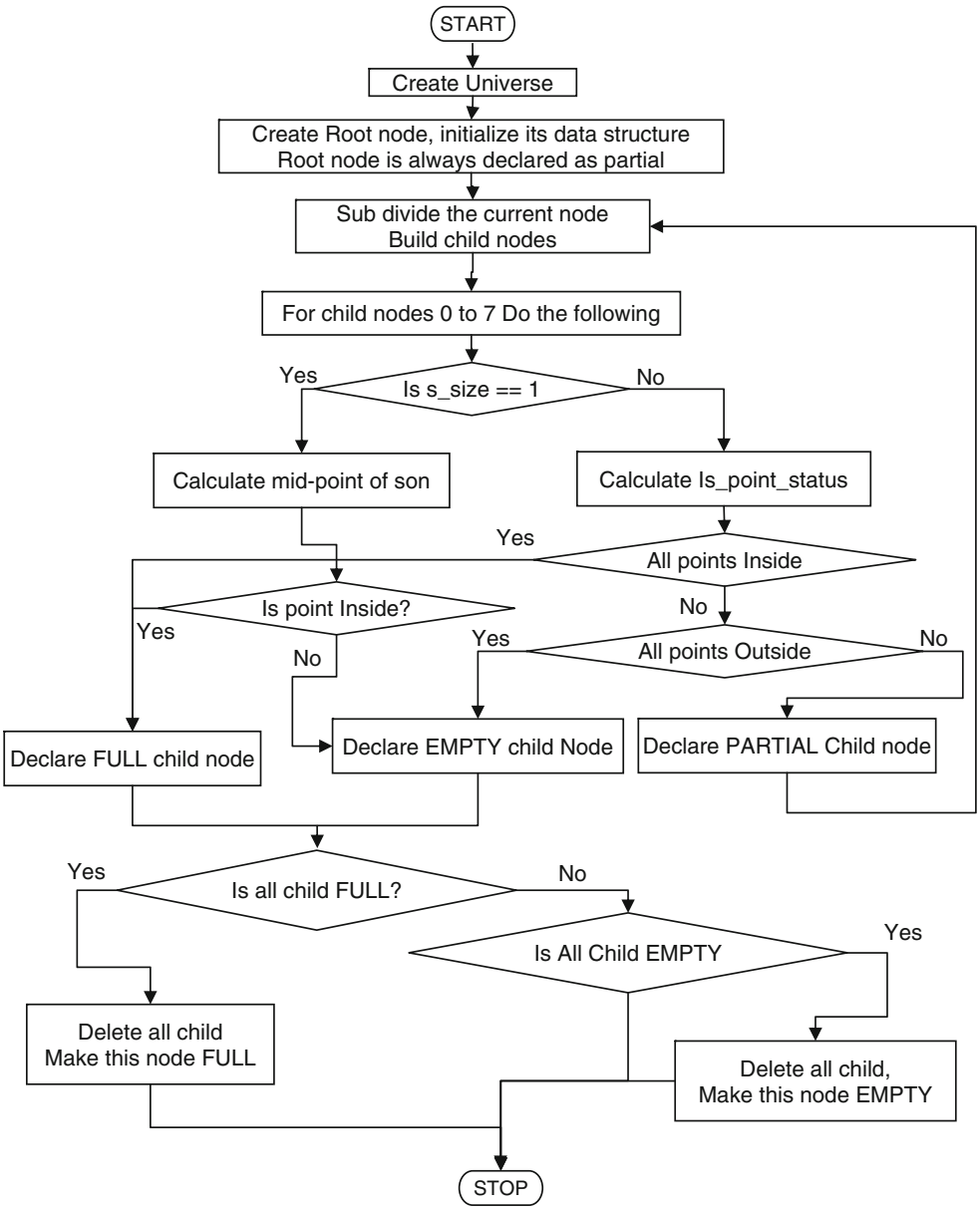


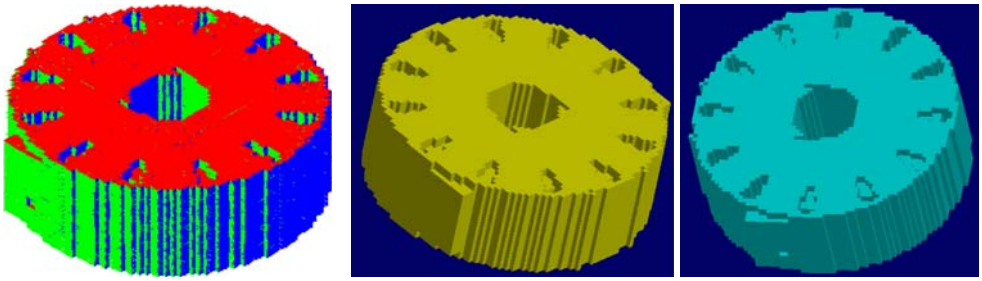
Fig. 10 Control flow diagram of create octree of an object

can be divided into six triangles. Thus, an isometric projection of any object encoded in the form of an octree can be obtained by having each of the FULL nodes drawn as a set of six triangles. The triangles corresponding to the FULL nodes are first gathered in a triangular quad-tree and then dumped onto the screen. This display is very fast as it involves only a simple tree traversal. The quasi-rendered isometric display is extremely fast but makes use of only three colors and, hence, is unable to provide adequate contrast required to depict edges. It is not amenable for viewing transformations. Therefore, the quasi-rendered isometric display is limited to only quick visualization.

For detailed visual verification of the machined component, one requires to rotate and zoom the object at arbitrary angles and scales. The designed shape is normally available in BRep. In order to assess the conformance of the machined model, it also has to be in BRep form. In order to achieve these goals, first, the octree is converted into BRep. The BRep model can be subsequently displayed using OpenGL. OpenGL inherently supports zooming, rotation, and translation of the model for easy viewing. The same conversion also can be used for outputting the octree into STL format.

Figure 11a shows the octree model of a rotor displayed in the static isometric view. After converting it into BRep, the same display was created, as shown in Fig. 11b. When it was displayed from BRep model, it could be rendered realistically in any desired view point and viewing direction with a rich choice of light and color settings using OpenGL.

Fig. 11 Display using Oct-OAC



(a) Static isometric view (b) Display after converting to B-Rep

Swept volume module The fundamental requirement of any volumetric NC simulation system is the calculation of the volume swept by the cutter along its path called *swept volume*. A typical cutter used for NC machining has a variety of shapes. Flat end mills, dome end mills, ball nose end mills, angle cutters, face mills, and side-and-face cutters are some of the commonly used cutter shapes. The paths followed by the cutter also can be many. CNC machines can drive the cutter along certain types of paths, each type being controlled by an *interpolator*. Among them, four types of motions are the most common, namely, (1) rapid motion (G00), (2) linear motion (G01), (3) CW circular motion (G02), and (4) CCW circular motion (G03). However, the classification for calculating swept volume is driven by the geometry of the cutter path and its relation with the cutter axis [39]. For this purpose, the various motions possible on a CNC machine are classified into the following six groups to enable the development of efficient algorithms to calculate the swept volume: (1) axial motion, (2) orthogonal linear motion, (3) circular motion, (4) fixed-axis motion, (5) variable-axis motion, (6) rapid motion. The closed swept surface SV for any motion will have three distinct portions: (1) SV_1: portion of the cutter positioned at CL_{i-1}; (2) SV_2: middle portion of the swept volume; and (3) SV_3: portion of the cutter positioned at CL_i, as shown in Fig. 12. SV_2 is the envelope of all the cutters placed in the intermediate positions along the path at infinitesimal intervals. The ultimate purpose of calculating the swept volume is to subtract it from the workpiece. Whenever a tool change takes place, CL_{i-1} corresponds to the position given by "FROM/..." statement of the CL file. Therefore, after every tool change, the cutter is positioned at CL_{i-1} and subtracted from the blank. Subsequently, for any "GOTO/..." statement, it is enough to subtract SV_2 and SV_3 from the blank since SV_1 of this motion has already been subtracted from the blank, as is the SV_3 of the previous motion. Therefore, only for the first motion of the cutter, all the three portions, SV_1, SV_2, and SV_3, need to be calculated; for the subsequent motions, portions SV_2 and SV_3 alone need to be calculated. The algorithm for swept volume creation form CL file data is given in Fig. 13.

Optimization module Relative hardness and interfering relative motion between tool and workpiece are essential for material removal. The cutting takes place because the cutter is in contact with the workpiece over a common surface and they both have relative motion. The energy required for material removal is transferred to the workpiece through the application of cutting forces by the tool. The instantaneous forces, torque, and the bending moments experienced by the cutter at any time in this process are dependent upon the following two things, one being geometric and the other kinematic: (1) The geometry of the contact surface (area, shape, orientation w.r.t. cutting direction, distance of its CG from the support, etc.) at any time. (2) The instantaneous relative velocity vector between the cutter and workpiece. Two other factors influence significantly the cutting forces, which are constant for a given cutting process: (1) the geometry of the cutter which includes number of flutes, helix angle, various rake and clearance angle, form of the cutter, etc. and (2) the material combination of cutter and workpiece. Since these factors are responsible for the cutting forces, torque, and bending moments, it is possible to calculate them from the geometry and the kinematics of the machining process. Further using

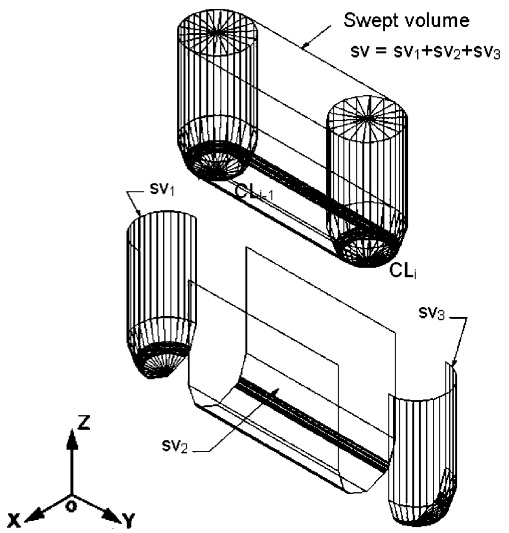


Fig. 12 Three portions of the swept volume

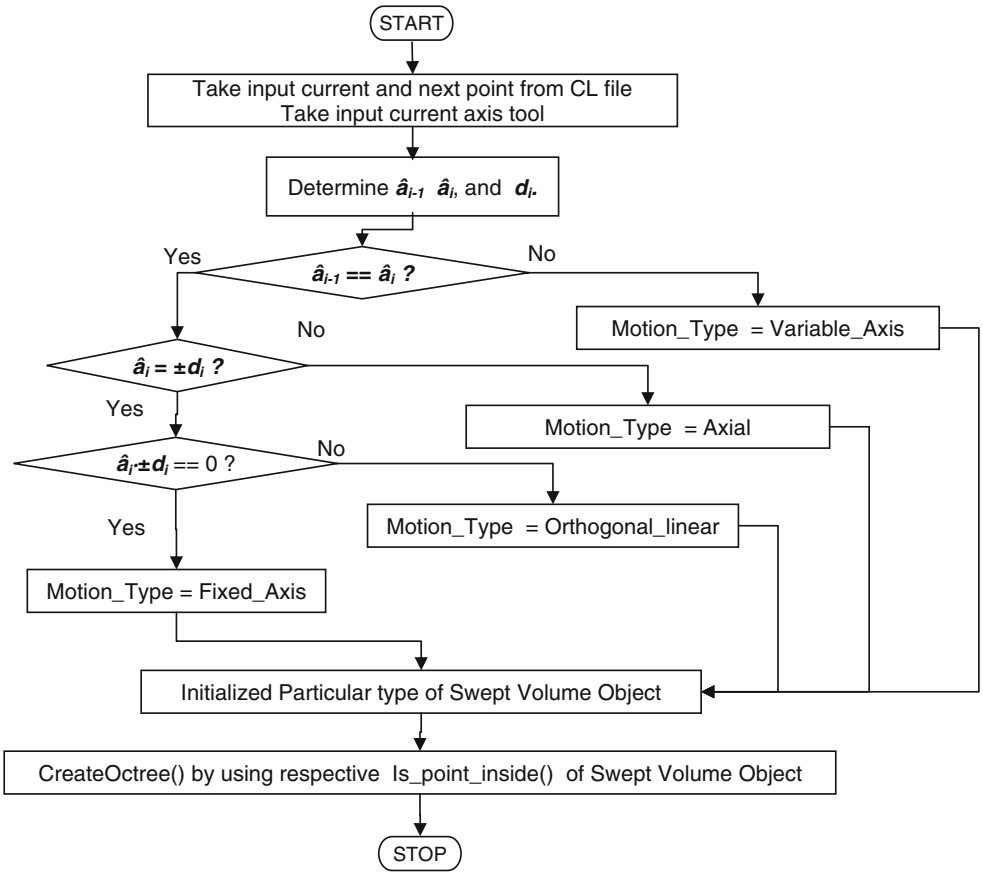


Fig. 13 Control flow diagram for creating swept volume

predicted forces experienced by cutter at any time by the developed system, it is possible to optimize the feed rate to satisfy optimization criterion.

The optimization module reads the NC tool path file and divides the motion into a number of smaller segments based on user-specified sampling interval. The geometric modeler passes on the contact geometry and the pairs of entry and exit angles at different slices of the cutter to the optimization module at regular intervals of the motion. The optimization module can predict the cutting forces from this information. If the calculated force is away from the threshold force value specified for a given cutter, the cutting parameters are so adjusted to bring it around this value. In other words, it is an off-line ACC based on the threshold forces.

As an example, consider a wedge-shaped blank in which a slot is milled along the *X*-axis as illustrated in Fig. 14a. The end position of the cutter is shown in Fig. 14b, c which shows its swept volume. The shape of the material removed is shown in Fig. 14d. The contact geometry between the cutter and the workpiece is shown at regular intervals along the cutter path, referred to as sampling intervals (Fig. 14e). From the contact area, the geometrical parameters necessary for the physical simulation are calculated. For example, to calculate the cutting force, the geometrical parameters needed may include chip volume, axial and radial depth of cut, cutting thickness, etc. These data are extracted from the octree of the contact area. By analyzing the geometry of this contact area and by calculating the cutting forces, the optimal feed rate for every sampling interval can be found out and output.

4 Illustrations of NC simulation using octree

The octree-based NC simulation system developed is able to simulate large files up to five-axis motions. The examples of output generated using the developed system for 2.5-axis, three-axis, and five-axis machining performed is given in Fig. 15. For 2.5-axis machining, a grove-milling program is chosen for simulation. Figure 15b shows a hair drier, idler arm, and iron after machining. This is an example of three-axis machining. The simulation of five-axis machining of an impeller in five-axis mode is shown in Fig. 15c.

5 Simulation of cutting process

The simulation of physical cutting process is needed to be done for selecting cutting conditions that are both safe and

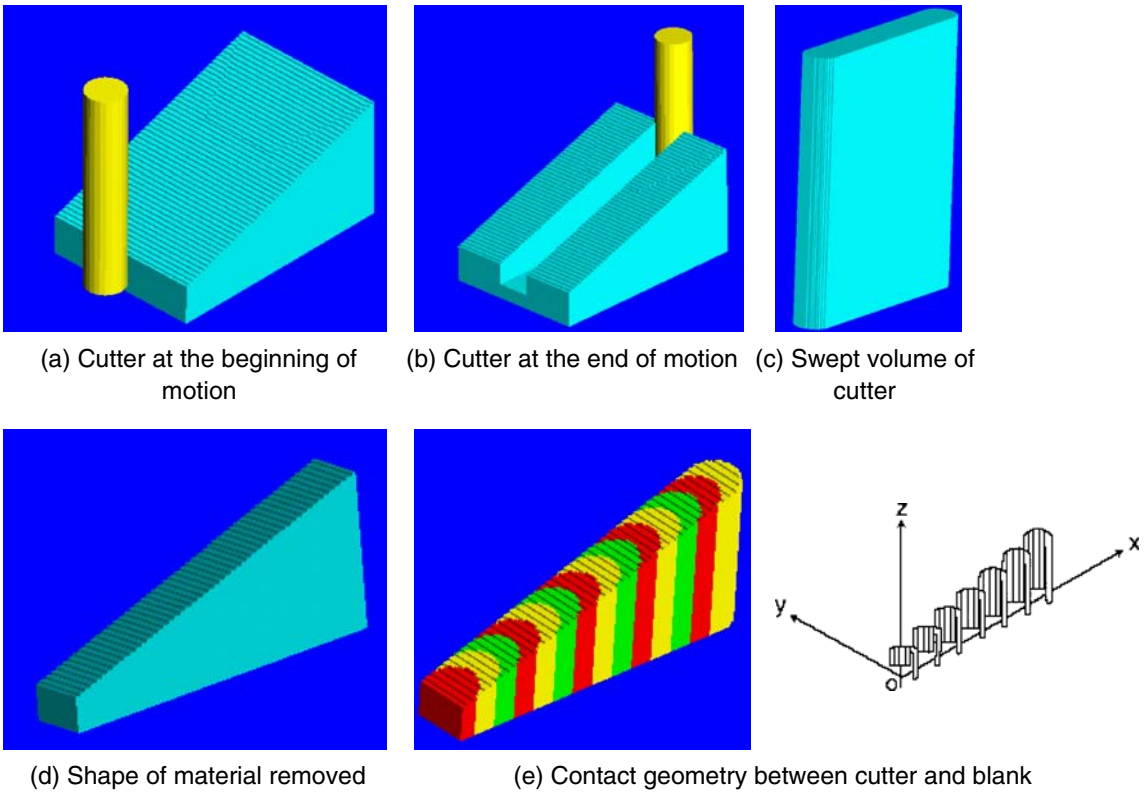


Fig. 14 Principle of operation for optimization

efficient. This step deals with the simulation of the end milling process for prediction of the cutting forces during the cutting process. A number of different methods to predict cutting forces have been developed over the last years. Based on the review of different force models proposed by various researchers [28, 40-48], these can be classified into four categories according to the method of force computation and of the deflection feedback of the force in the increasing order of sophistication and accuracy: (1) material removal rate model (MRRM), i.e., average force-rigid cutter static model; (2) instantaneous force model (IFM), i.e., distributed force-rigid cutter static model; (3) distributed force deflection feedback static model; and (4) regenerative force deflection feedback dynamic model.

The MRRM relies on data available in the literature [49] to empirically relate the cutting conditions to the process responses of interest. However, such information is generally in the form of average measured forces and associated empirical predicting equations for specific cutter geometry and workpiece pairs. From this information, it is difficult to generalize and mechanistically explain the force system and its specific impact on cutter deflection and breakage. The authors have presented the implementation of MRRM model in [34]. The IFM model explains the variation of the cutting forces during end milling process as a function of cutting conditions and provides a more realistic computation of the cutting force as it computes the instantaneous force on incremental sections of the helical cutting edge. Distributed force deflection feedback static model can be considered as extension of IFM model wherein deflection of the cutter is not only computed based on force but it is also feedback to have influence on the

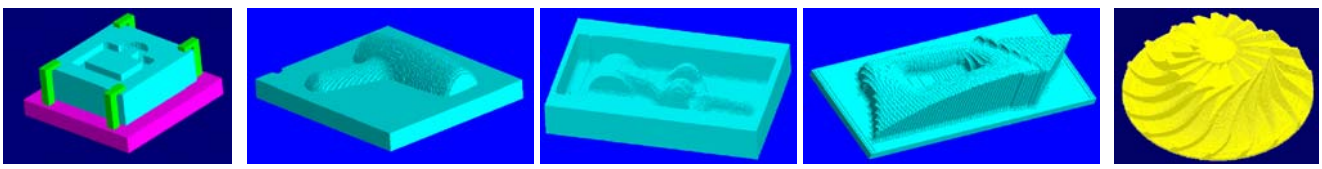


Fig. 15 Illustrative examples of simulation using Oct-OAC

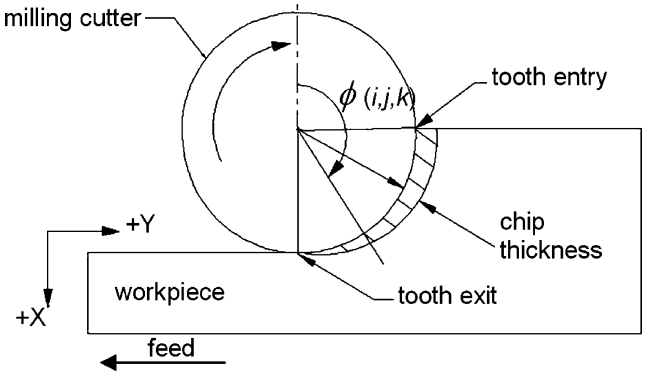


Fig. 16 Variation of chip thickness

force. Regenerative force deflection feedback dynamic model is of sufficient complexity to handle time domain simulation. This model takes into consideration the inertia of the system.

Based on the review undertaken, the IFM, i.e., distributed force-rigid cutter static model, has been adopted in Oct-OAC. The procedure of model building of the milling process can be divided into three parts: (1) modeling of the cutter geometry, (2) determination of cutter entry and exit angle using octree, and (4) development of the force relations.

5.1 Modeling the cutter geometry

The basic input to the prediction of the cutting forces is the chip load on the cutter. At any instant, the uncut chip thickness (t_c) is (Fig. 16) [50]:

$$t_c = f_t \sin\phi(i,j,k) \tag{1}$$

The complete representation of the chip load on the end mill at any instant is obtained by considering thin disk-like sections along the axis of the cutter .The cutter is sliced at several places perpendicular to the axial direction to form number of small disks of thickness t_d. The flute on the disk forms the smallest cutting edge which can be termed as *elemental cutting edge*. Just as the cutter is discretized into disks, its angular rotation θ also is considered at discrete intervals of δ_θ. If N_θ is the number of angular increments in one rotation of the cutter, then $\delta_\theta = 360/N_\theta$. Any elemental cutting edge can be located from the indices i, j, and k, where i refers to the orientation of the first cutting edge of the bottom most disk with respect to the X-axis, j refers to the jth rotational increment, and k refers to the kth flute. Since there are multiple cutting edges, one of them is taken as reference and the others are distributed circumferentially by angular increments of δ_f (δ_f=360/N_f) where N_f is the number of flutes.

The location of each flute on each disk is determined, and for each flute engaged in the cut, the chip thickness multiplied by thickness of the disk yields chip load. The geometry of general milling cutter (flat, ball, and taper end mills) is shown in Fig. 17. The flutes are right-handed and the cutter is assumed to be rotating clockwise (case of down milling is considered). Feed is along −ve Y-axis (and hence, the cutter is moving along +ve Y-axis).

Due to helix angle β, a point on the cutting edge lags at amount of angle Ψ with respect to its flute tip. Ψ=0 at the bottom of the cutter, i.e., at z=0. The lag angle for the elemental cutting edge of the first flute at any axial location z is given by:

$$\psi(z) = z\frac{\tan\beta(\psi)}{R(\psi)} \tag{2}$$

At any axial location z ($z = i \times t_d$), the radial immersion angle $\emptyset$ (i,j,k) corresponding to the cutting edge at jth angular position of the kth flute of the ith disk at axial location z is:

$$\phi(i,j,k) = (j)\delta_\theta + (k)\delta_f - \psi(z) \tag{3}$$

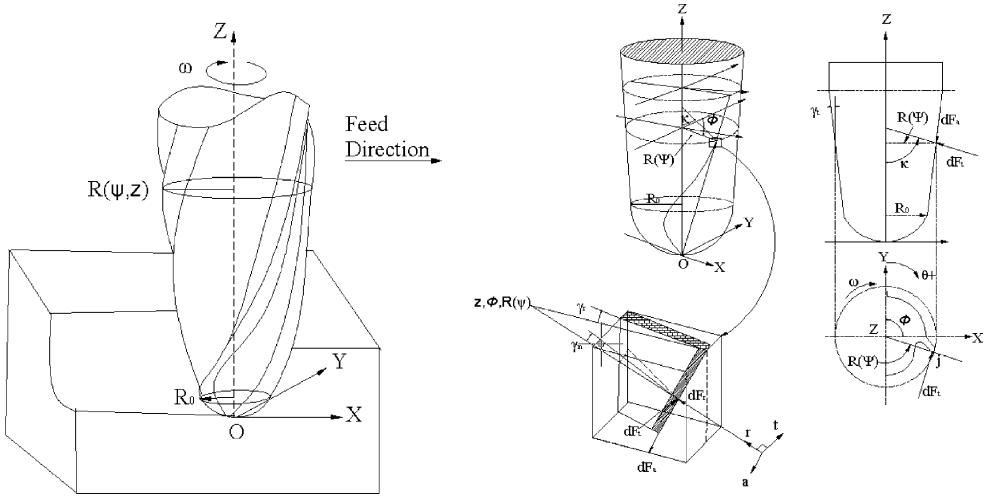


Fig. 17 Geometry of end milling cutter

This is the angle made by the elemental cutting edge with the +*Y*-axis measured in anticlockwise direction where $1 \le i \le N_d$ and $1 \le j \le N_\theta$, $1 \le k \le N_f$. The general formulation given above can be applied to any helical end mill geometry such as ball end, taper end, and cylindrical end mill. Helical cylindrical end mill have constant radius $R(\Psi) = R_0$ and constant helix angle ($\beta(\Psi) = \beta_0$) and zero taper angles (γ_t=0), as shown in Fig. 17. The lag angle produced by the constant helix angle between the tip and the upper points of the flute is given by Eq. 2 as $\Psi(z) = z \tan\beta_0/R_0$.

The axial immersion angle κ depends on the cutter geometry and elevation z_k of the axial element (Fig. 17) for the different types of cutter geometry [44]. For cylindrical end mill, κ is equal to $\pi/2$. From the disk thickness t_d and the immersion angle κ, the length of the elemental flute on the disk can be found out as:

$$t_s = t_d / \sin\kappa. \tag{4}$$

The chip thickness t_c changes with both radial immersion angle $\emptyset(i,j,k)$, and axial immersion κ and is given by:

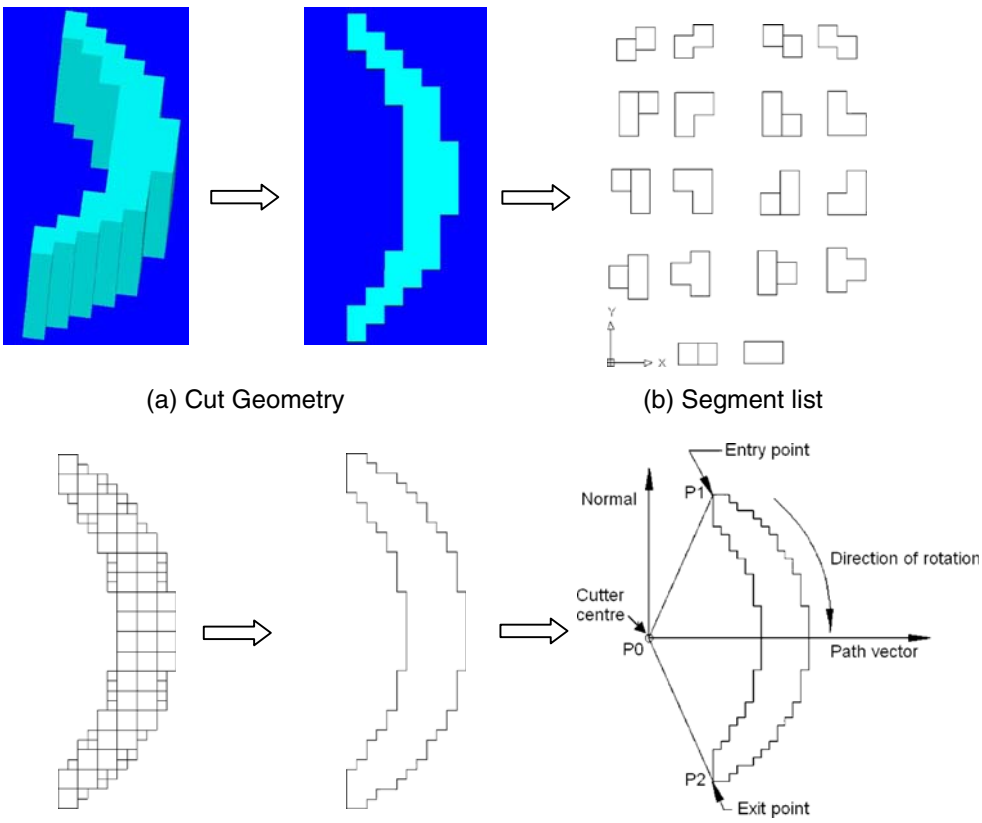
$$t_c = f_t \sin\phi \sin\kappa. \tag{5}$$

In Eq. 5 of the uncut chip thickness, angle $\emptyset(i,j,k)$ represents the angular position of the cutter at any instant. Its value is compared with the entry and exit angles obtained from the contact geometry. Entry angle θ_{st} is the angle at which the cutting edge enters the workpiece, and exit angle θ_{ex} is the angle at which the cutting edge leaves the workpiece. The chip thickness is calculated as:

If $\emptyset(i,j,k) \ge \theta_{st}$ and $\emptyset(i,j,k) \le \theta_{ex}$
$t_c = f_t \sin\emptyset(i,j,k) \sin\kappa$
else
t_c = 0.0, i.e., cutting edge is not engaged in the cut.

5.2 Determination of cutter entry and exit angle using octree

In Oct-OAC, the entry and exit angles for the cutter teeth are computed directly from the in-process part geometry. This information is available for subsequent use in finding the cutting forces and torque needed for optimization module. For a given NC Block and user specified sampling interval, next point is calculated which is end point of the current segment and the swept volume is created for current



(a) Cut Geometry (b) Segment list

(c) Extraction of 2D loop from Cut Geometry using Octree (d) Entry and Exit angle calculation using 2D loop

Fig. 18 Extraction of geometric information from simulated cut geometry

segment. Boolean subtraction is carried out of the cutter swept volume from the octree of the blank during the simulation run. The updated blank is stored by declaring locally a new octree material_removed. Finally, Boolean intersection of the octree of the cutter is done from the material_removed to generate the cut geometry (Fig. 18a). The cut geometry gives the solid representing the amount of material removed.

The contact point of the octants of cut geometry in octree form gives the contact point of the cutter and the workpiece. This cut geometry is stored in the octree form for further analysis, i.e., calculation of the exit entry angle. The function GetEntryExitAngle() is the main routine that computes the key entry and exit angles for the cutter teeth using the octree model of the material removed, as shown in Fig. 19. The subroutine CalculateZMinZMax()calculates the maximum and minimum coordinates of the cut geometry (Fig. 20). The axial depth of cut (ADoC) is then given by z_max–z_min.

The simulation gives the contact points in the machine coordinate system. These are transformed to the cutter coordinate system with the origin at the bottommost center point or the tip of the cutter. Corresponding to the elemental disk of the cutter, a *z*_plane is defined at the center of the disk. This *z*_plane is passed through the cut geometry, and full nodes of this solid are checked for interference with this *z*_plane. If *z*_plane interferes with a particular full node of the solid then for all such nodes, the bottom face is

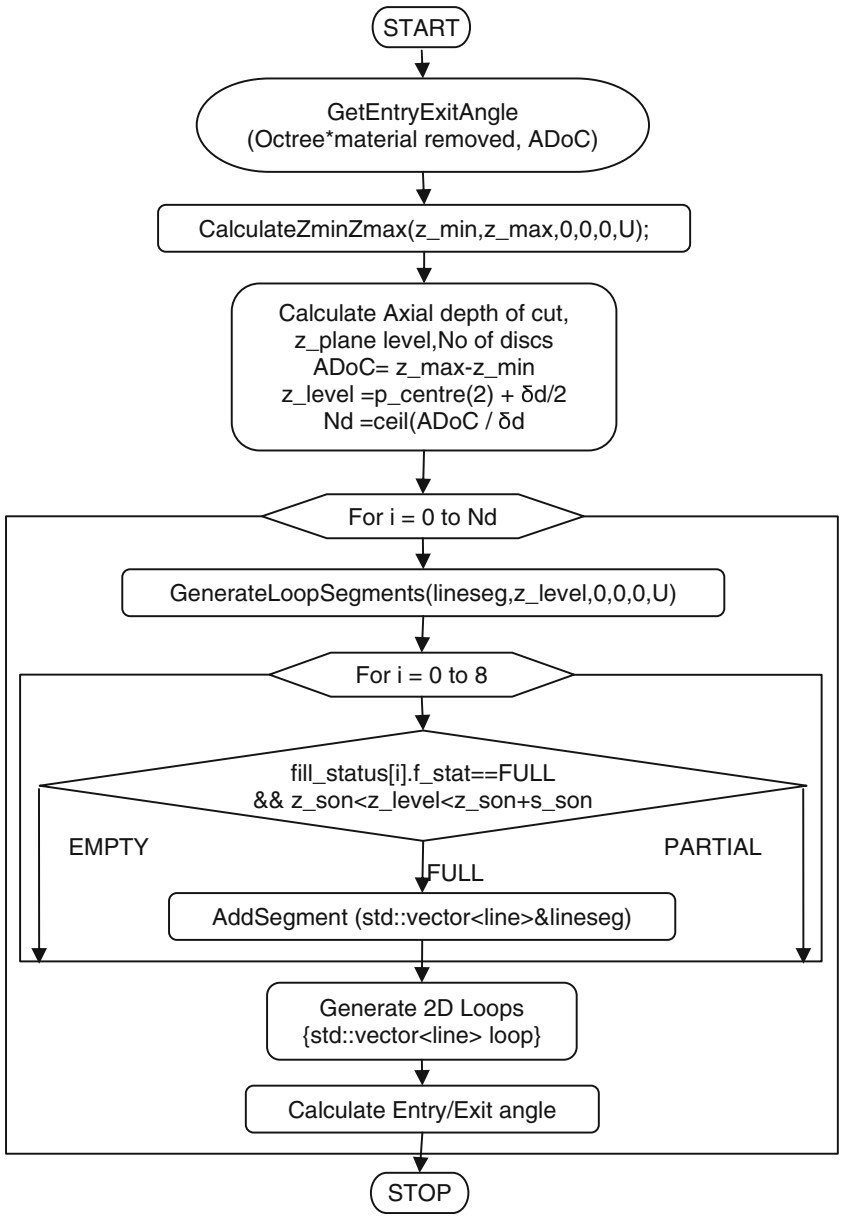


Fig. 19 Control flow diagram of GetEntryExitAngle

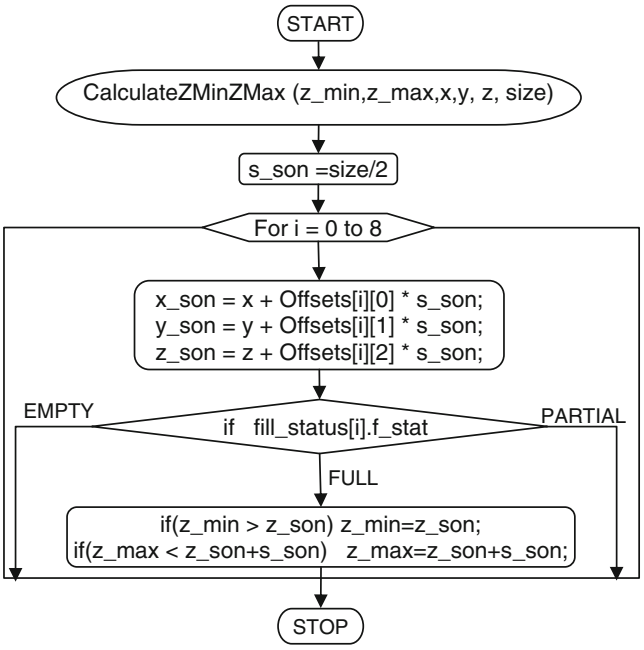


Fig. 20 Control flow diagram of CalculateZMinZMax

projected on the z_plane by collecting the line segments in a standard vector array consisting of segment list *seg*, as shown in Fig. 18b, using Generateloopsegment() routine for further processing .For the interfering node of the octree, there will be four such line segments which will be added to the segment list.

This segment list is further processed to generate the 2D loop, as shown in Fig. 18c. The points of 2D loop are used to calculate the entry/exit angles. Figure 18d shows the cluster of points representing the contact area at particular z_plane for the disk. For the first segment, the entry and exit points are extracted as points P_1 and P_2. Point P_0 is the cutter center. The entry and exit angles can be derived as (Fig. 18d);

$$\theta_{en} = \arccos((Y_1 - Y_0)/r) \tag{6}$$

$$\theta_{ex} = \arccos((Y_2 - Y_0)/r) \tag{7}$$

For each segment of the loop, two angles will be obtained. A standard vector array is EEAngle, which is used to hold the values of entry or exit angle. Initially, the entry angle is given as the default value of 2π and exit as -2π. Now, if the value of the angle calculated comes greater than the default value, it is updated and stored as entry angle, and if it is less, default value for the exit angle it is again stored as exit angle. In this way, the angles are found out for all the points of segments in the loop and compared with the already stored values in the EEAngle, and updating of the angle values is done. For each individual 2D loop, the entry and exit angles are determined and are used for further calculation.

5.3 Modeling of the cutting forces

The cutter is discretized by slicing along its axis. Each slice acts as an elemental cutting edge. The elemental tangential, dF_t, radial, dF_r, and axial, dF_a, cutting forces acting on an oblique cutting element (Fig. 17) with height t_d can be expressed as a function of varying uncut chip area ($t_c \times t_s$) and edge contact length t_s [44] as:

$$\left.\begin{aligned} dF_t &= K_{tc} t_c(\phi,\kappa) t_s + K_{te}\, t_s \\ dF_r &= K_{rc} t_c(\phi,\kappa) t_s + K_{re}\, t_s \\ dF_a &= K_{ac} t_c(\phi,\kappa) t_s + K_{ae}\, t_s \end{aligned}\right\} \tag{8}$$

where $t_s = t_d/\sin\kappa$ is the projected length of the infinite small flute segment in the direction along the cutting velocity. This is consistent with the chip width defined in the classical oblique cutting theory. The uncut chip thickness t_c normal to the cutting edge is evaluated using the true kinematics of milling [50] and varies with the position of the cutting point and cutter rotation as given by Eq. 1. The cutting forces are separated as edge and cutting components. Subindices (c) and (e) represent shear and edge force components, respectively. The edge cutting coefficients K_{te}, K_{re}, and K_{ae} are constants in Newton per millimeter and related to the cutting edge length t_s given in Eq. 4 The shear force coefficients K_{tc}, K_{rc}, and K_{ac} represent cutting force per unit chip area and can be identified either mechanistically from milling tests conducted at a range of feed rate [51] or a set of orthogonal cutting tests using an oblique transformation method presented by Budak et al. [52]. The elemental

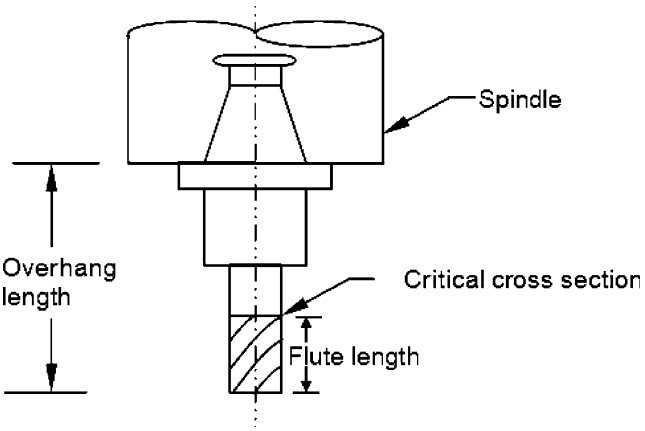


Fig. 21 Milling cutter with critical cross-section

radial, tangential, and axial forces are projected x, y, and z Cartesian coordinate directions as:

$$\{dF_x\, dF_y\, dF_z\}' = [T]\{dF_r\, dF_t\, dF_a\}' \tag{9}$$

where the transformation matrix [**T**] is given by:

$$\begin{pmatrix} -\sin(\kappa)\sin(\phi) & -\cos(\phi) & -\cos(\kappa)\sin(\phi) \\ -\sin(\kappa)\cos(\phi) & \sin(\phi) & -\cos(\kappa)\cos(\phi) \\ -\cos(\kappa) & 0 & -\sin(\kappa) \end{pmatrix} \tag{10}$$

As the cutter is an axially digitized cutter with small disk elements with uniform differential height of t_d, the elemental forces for all the disks for each flute which is in-cut, and thus for all flutes, at every angular position of the cutter are summed up to get the total forces in X, Y, and Z directions for one complete revolution of the cutter.

5.3.1 Failure analysis

Once the forces are predicted, the force analysis is taken up at the critical cross-section of milling cutter, i.e., at the end of flutes where the flutes end, as there is abrupt change in the cross-sectional area (Fig. 21), to verify whether the cutter will be able to withstand load or not. The failure at

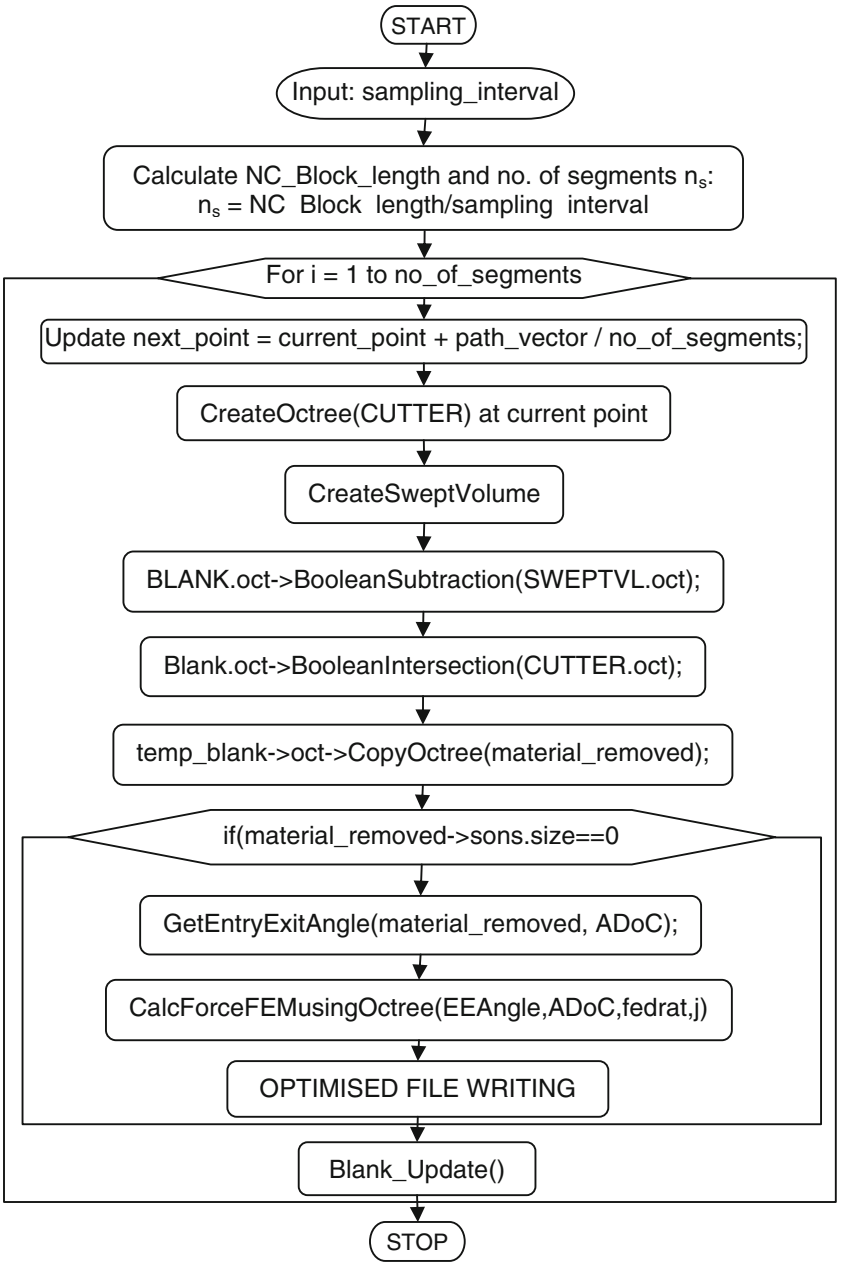


Fig. 22 Control flow diagram of control flow diagram of segmentation

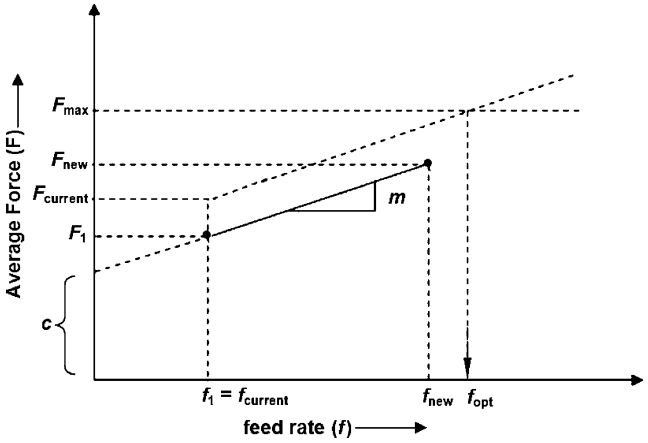


Fig. 23 Optimum feed rate calculation

this section is either due to bending or torsion. The cutter cross-section is approximated due to the flutes. In case of a four-fluted cutter, the cross-sectional area can be approximated as $\pi(0.807R)^2$ [53]. The tangential force encountered by the cutter causes twisting of the cutter, thereby generating shear stress in the critical cross-section, leading to shear failure. The total torque on the cutter for a complete rotation is calculated as:

$$T = \sum_{j=1}^{N_\theta} \sum_{k=1}^{N_f} \sum_{i=1}^{N_d} dF_t(i,j,k) \times R \tag{11}$$

Corresponding shear stress acting on the cutter at the critical cross-section is:

$$f_{s\_cur} = \frac{16T}{\pi D_{eff}^2} \tag{12}$$

This value of the shear stress is compared with the allowable shear stress, f_{s_a}, for the given cutter geometry and cutter material. If the current stress value is greater than the allowable value, current feed rate, f_{r_cur}, has to be reduced up to the value for which the current induced shear stress is comparable with the allowable value and vice versa. The feed rate can be modified assuming a linear relationship [54]. The optimum feed rate is decided as:

$$f_{r\_opt} = f_{r\_cur}\left(\frac{f_{s\_a}}{f_{s\_cur}}\right) \tag{13}$$

The cutter undergoes bending about the support as the radial and tangential forces are acting on it. The forces acting on each disk contribute to the total moment acting on the cutter. As the focus of the stress analysis is at the critical cross-section where shank ends and flutes start, the bending stress is also calculated at this cross-section. Moments in the X and Y directions are calculated for all i, j, and k values. The resultant value of moments is given by:

$$M = \sum_{i=1}^{N_d} \left\{ \sum_{j=1}^{N_\theta} \sum_{k=1}^{N_f} \sqrt{[dF_x(i,j,k)]^2 + [dF_y(i,j,k)]^2} \right\} l_c(i) \tag{14}$$

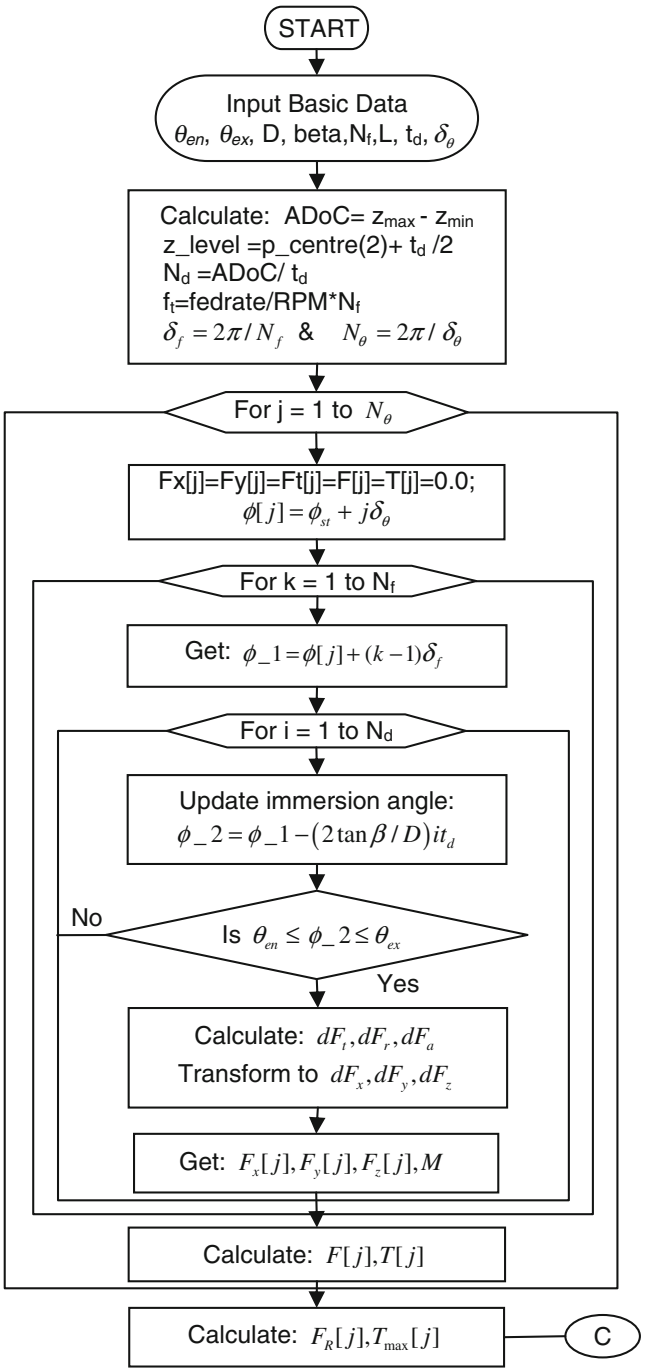


Fig. 24 Control flow diagram of force calculation

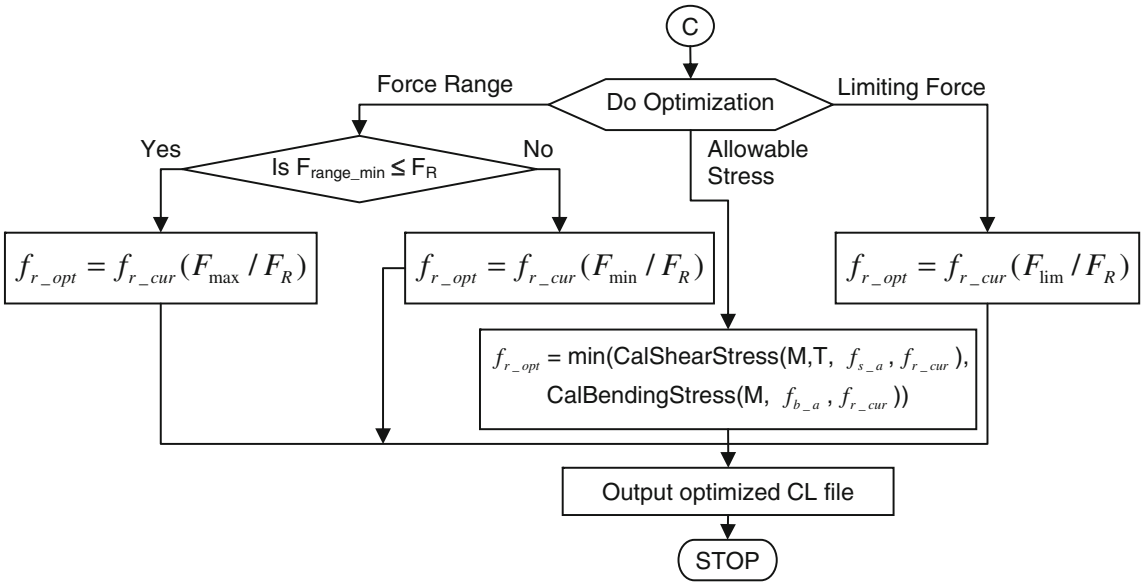


Fig. 25 Control flow diagram of feed rate optimization

where $l_c(i)$ is the length of ith axial disk from the critical cross-section. Hence, the bending stress f_{b_cur} is:

$$f_{b\_cur} = \frac{32\,M}{\pi D_{eff}^3} \tag{15}$$

In a similar manner as in Eq. 13, the allowable bending stress (which is the function of the cutter geometry and the cutter material) and actual bending stresses are compared and the optimum feed rate is decided assuming a linear relationship as:

$$f_{r\_opt} = f_{r\_cur}\left(\frac{f_{b\_a}}{f_{b\_cur}}\right) \tag{16}$$

6 Feed rate optimization using octree-based NC simulation system

The main inputs for feed rate scheduling are: (1) resolution parameters: disk thickness (t_d), angular resolution (δ_θ); (2) tool geometry: radius (r), number of flutes (N_f), flute length, tool length, helix angle (β_0), rake angle (γ_r); (3) feed, cutting speed, and axis of the tool which is obtained from the input CL file to the system; (4) contact geometry: obtained from geometric simulation; and (5) tool and workpiece material.

Optimization uses the calculated cutting force as feedback from the machining process to adjust the feed rate. The feed rate programmed by the NC programmer is the reference input to the system. The cutting force is feedback-generated using the workpiece geometry (contact area of the cutter with the workpiece) obtained from segmentation and force equations (8, 9, and 10). The program starts processing the cutter paths after positioning moves of the cutter in RAPID mode (G00 in NC data). It checks whether initial tool, spindle speed, and feed rate are defined. Then, for each NC pass, i.e., APT GOTO statement, the cutting force is calculated using the IFM model discussed in Section 5.3 at the sampling intervals specified by the user. This is done using segmentation routine (Fig. 22). It calculates the length of the NC block, i.e., path between two end points of block. Then, for the user-defined sampling

Fig. 26 View of the circular hole milling

interval, the number of segments for the given NC block is calculated.

For ∀ segments:

- Calculate locally next point which is the end point of the current segment and create *s*wept volume for this segment.
- Store the initial shape of the blank and call it *temp_blank*
- Carry out the Boolean subtraction of the swept volume from the octree of the blank; also do the Boolean intersection of the cutter from the *temp_blank* at the current point to generate the *cut_volume*.
- Save the swept volume for the current segment for further analysis. Also save the octree of the Boolean intersection of the cutter from *temp_blank* for generating contact area.
- Generate the boundary loop from the contact area.

If loop segment size > 0

  - Use the boundary loop to find the *z* range.
  - Get entry and exit angles from the loop segment
  - Perform force analysis to determine the optimized feed rate for current segment from the mechanistic modeling module.
  - Write the NC block corresponding to the current segment of the cutter path as optimized NC block into user-specified CL file

- Delete the stored *temp_blank*.
- Carry out the Boolean subtraction of the cutter from the octree of the blank and update the blank.
- Update the current point with the next point locally and repeat iteration for all segments, similarly as shown in Fig. 22.

Table 2 Comparison of unoptimized and optimized CL files

Unoptimized CL file
MSYS/75.0000,76.500,50.0000,1.0000000,0.0000000,0.0000000,0.0000000,1.0000000,0.000000
TLDATA/MILL,20.000,0.000,0.000,0.000,0.000,10.000,0.00
FEDRAT/MMPM,50
GOTO/75,76.5,22
GOTO/65,76.5,22
CIRCLE/75,76.5,22,0.00,0.00,1.00,10.00
GOTO/85,76.5,22
CIRCLE/75,76.5,22,0.00,0.00,1.00,10.00
GOTO/65,76.5,22
GOTO/55,76.5,22
CIRCLE/75,76.5,22,0.00,0.00,1.00,20.00
GOTO/95,76.5,22
CIRCLE/75,76.5,22,0.00,0.00,1.00,20.00
GOTO/55,76.5,22
GOTO/55,76.5,50
SPINDL/OFF
FINI
Instantaneous force model (optimized CL file)
MSYS/75.000000,76.500000,50.000000,1.000000,0.000000,0.000000,0.000000,1.000000,0.000000
TLDATA/MILL,20.000000,0.000000,0.000000,0.000000,0.000000,10.000000,0.000000
FEDRAT/MMPM,50.00
GOTO/75.000000,76.500000,22.000000
FEDRAT/MMPM,176.86
GOTO/72.500000,76.500000,22.000000
FEDRAT/MMPM,178.28
GOTO/70.000000,76.500000,22.000000
FEDRAT/MMPM,176.86
GOTO/67.500000,76.500000,22.000000
FEDRAT/MMPM,178.28
GOTO/65.000000,76.500000,22.000000
FEDRAT/MMPM,214.29
CIRCLE/75.000000,76.500000,22.000000,0.000000,0.000000,1.000000,10.000000
GOTO/65.405070,79.317326,22.000000…

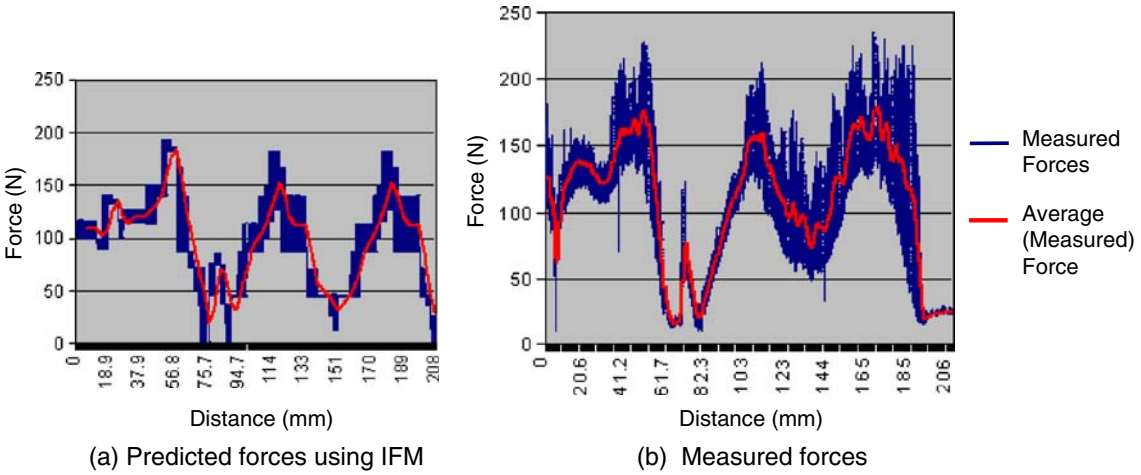


Fig. 27 Resultant force plot in unoptimized cutting using constant feed rate

For optimization, the developed octree-based NC simulation system adopts the MRR based on average force calculations [34] and IFM model based on instantaneous forces as the basic physical model for physical simulation. The predicted cutting force using the model is used as feedback from the machining process to optimize the feed rate. Three approaches for optimization are used: (1) maximum force limitation, (2) force range limitation, and (3) force limitation based on allowable stress. The maximum force limitation method is used to adjust cutting force to be as large as possible but less than the upper specified limit, and the force range limitation method is to make the force fall into the desired specified range. Force limitation based on allowable stress is used to adjust cutting force to be as large as possible so that maximum stress induced in the cutter are limited to allowable values depending upon the cutter material. The feed rate is adjusted to reach optimization objectives according to the predicted cutting force. The developed system is validated using the IFM model-based predicted force with maximum force limitation approach.

For feed rate optimization, calculation of edge force is done first. For calculating the edge force for a particular NC block, the average force of all the segments is calculated at the given feed rate. The same procedure is repeated for the same block at some higher feed rate value. From the plot of feed rate versus average force, edge force is calculated as shown in Fig. 23. This edge force, when subtracted from the resultant force, gives the value of cutting force at each segment of the NC block. Then, for each segment of the given NC block, the feed rate value is adjusted so that the resultant force becomes equal to the maximum force specified by the user. As shown in Fig. 23, f_1 or f_{current} is the feed rate specified in the NC program and f_{new} is the new value of the feed rate greater than the specified value. F_1 and F_{new} are the values of average forces at those federate. The value of edge force (c) and slope (m) is obtained from the plot of average force versus federate

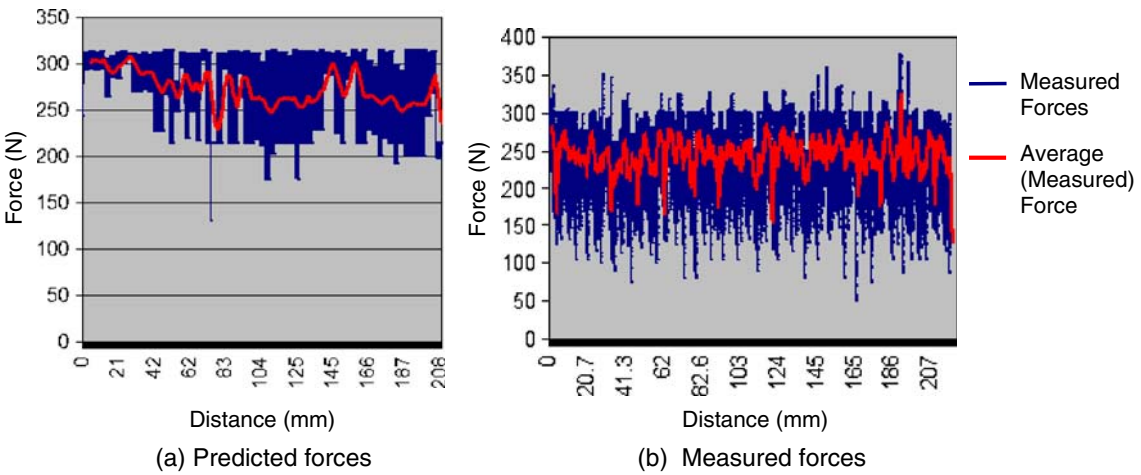


Fig. 28 Resultant force plot in optimized cutting using variable feed rate

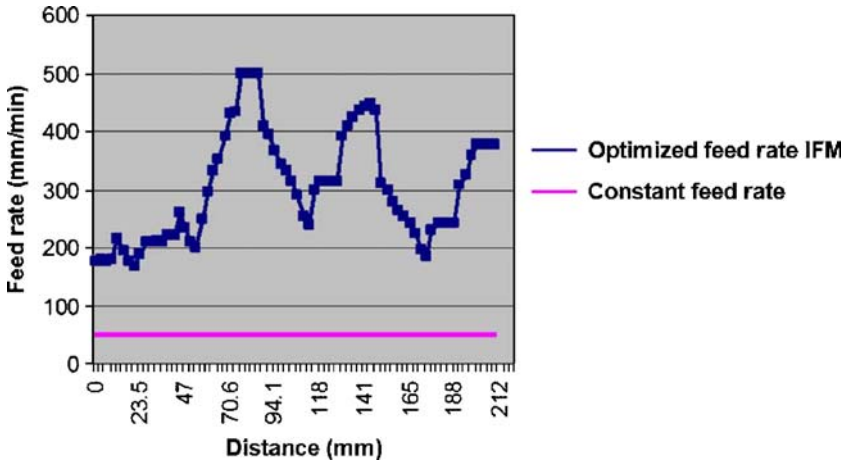


Fig. 29 Variation of feed rate along the cutter path

curve. Then, if $F_{current}$ is the value of force at $f_{current}$ feed rate for the first segment of the given NC block, a line with slope m is drawn through the point ($f_{current}$, $F_{current}$). The line is extended until it touches the F_{max} line, as shown in Fig. 23. The value of federate corresponding to F_{max} is the optimum value for that segment of the given NC block. The procedure is repeated for all the segments of the block. The procedure for force calculation and feed rate optimization is shown in Figs. 24 and 25, respectively.

7 Experimental verification

For validation of the octree-based NC simulation system, cutting experiments were performed on 2.5 axis DECKEL FP4A Maho CNC milling machine. The cutting forces were measured using KISTLER 9257A milling dynamometer with charge amplifier (5051A) and data acquisition system. The charge amplifier which has RS232C port is used to transfer the data to PC. The voltage output of the charge amplifier is connected to a data logger which acquires data by converting analog data into digital signals. A Windows-based program was used to read these digital data and store these into a file and simultaneously display these on screen.

To evaluate the validity of the cutting force prediction and optimization, circular milling is undertaken. As shown in Fig. 26, a circular hole is to be milled on a cylindrical surface. The cutting path includes a series of concentric circles and several lines. Along the path in NC block of the cutter movement, the cutter comes across variable axial depth of cut; hence, the feed rate optimization becomes advantageous. The blank size was 180 × 153 × 25mm (aluminum, AL6061 grade); HSS cutter, 20-mm diameter; six-fluted flat end mill with helix angle of 30º was used for machining. The spindle speed was kept at 500 rpm.

The simulation of the forces is performed using a 512Mhz Intel PIV-based microcomputer. Using the unoptimized CL file given in Table 2, the forces from the IFM model are predicted and compared with the forces measured. The values along the X-axis show the linear distance of cutter movement. For the circular portion of the cutter path, arc length is taken. The force values predicted from the IFM model and the measured force values are in close agreement. The cutting time from unoptimized file was found to be 251 s.

During the experiments, the force data are collected at intervals of 0.01 s. When milling, the circular profile along the cutting path, the material to be removed, and the geometrical cutting parameters change with the cutter positions, and therefore, the cutting forces also change with the cutter positions. It can be seen (Fig. 27) that when IFM model is used to predict the cutting forces, the force increases when the metal removal rises and decreases when the removed material volume declines. The optimized CL file given in Table 2 is generated by the segmentation of NC block at 3-mm interval using IFM. Figure 28 shows the forces predicted by the instantaneous force model using octree-based NC simulation system and experimentally obtained resultant forces for optimized cutting. Here, the upper bound of 300-N maximum force values is used for adjusting the feed rate. It can be seen (Fig. 28b) that the average force values move close to maximum force mark

Table 3 Comparison of model predicted cutting time, maximum, and average forces for unoptimized and optimized feed rates

	Unoptimized case		Optimized case	
	Measured	Predicted IF model	Measured	Predicted IF model
Cutting time (s)	251	250	43	43
Max. force (N)	234.9	191.8	375.2	300.0
Average force (N)	105.8	97.3	243.3	248.2

(300 N). The cutting time for optimized cutting (variable feed rate cutting) came out to be 43 s from instantaneous force model.

Figure 29 shows the comparison of constant and adjustable feed rates from IFM model in circular profile milling. Table 3 shows the model predicted cutting time, average, and maximum forces for unoptimized and optimized cuts. The predicted force shows fairly good agreement with the measured force. The improvements can be seen from the contrast of the cutting force curves of the unoptimized and optimized cases. In maximum force optimization using the instantaneous force model, the cutting time is shortened to almost one fifth of the cutting time without optimization.

8 Conclusions

In this paper, an Octree-based volumetric NC simulation system (Oct-OAC) for simulation and optimization of milling has been presented. This is achieved by using a hybrid representation of Octree and BRep. The present implementation is able to simulate fairly complex NC programs up to five-axis machining. The architecture of this Oct-OAC was presented in this paper along with the illustrations of 2.5-axis, 3-axis, and 5-axis simulations. Using the geometric information made available by the simulation system, the optimization module is able to predict the cutting forces in milling. Furthermore, it can be seen from the experimental results that such a system can be effectively used for optimization of feed rate using instantaneous force model.

References

1. CGTech, USA, the developers of image-based NC simulation package VeriCut. http://www.cgtech.com
2. CNC Software Inc., the developer of Mastercam software. http://www.mastercam.com
3. Anderson AO (1978) Detecting and elimination collisions in NC machining. Comput-Aided Des 10(4):231–237
4. Wang WP, Wang KK (1986) Real time verification of multiaxis NC programs with raster graphics. Proceedings of IEEE, International Conference on Robotics and Automation, San Francisco, 7–10 April, pp 166–171
5. Van Hook T (1986) Real time shaded NC milling display. Comput Graph (Proc. SIGGRAPH) 20(4):15–20
6. Atherton PR, Earl C, Fred C (1987) A graphical simulation system for dynamic five-axis NC verification. Proc Autofact, SME, Dearborn, MI, pp 2.1–2.12
7. Jerard RB, Angleton JM, Drysdale RL, Su P (1990) The use of surface points sets for generation, simulation, verification and automatic correction of NC machining programs. Proceedings of NSF Design and Manufacturing Systems Conference, Tempe, AZ, Society of Manufacturing Engineers, pp 143–1, Jan. 8–12
8. Chappel IT (1983) Use of vectors to simulate material removed by NC milling. Comput-Aided Des 15(3):156–158
9. Oliver JH, Goodman ED (1990) Direct dimensional NC verification. Comput-Aided Des 22(1):3–10. doi:10.1016/0010-4485(90)90023-6
10. Jerard RB, Hussaini SZ, Drysdale RL, Schaudt B (1989) Approximate methods for simulation and verification of numerically controlled machining programs. Vis Comput 5:329–348. doi:10.1007/BF01999101
11. Drysdale RL, Jerard RB, Schaudt B, Hauck K (1989) Discrete simulation of NC machining. Algorithmica 4(1):33–60. doi:10.1007/BF01553878
12. Sungertekin VA, Voelcker H (1986) Graphical simulation and automatic verification of NC machining programmes. International Conference on Robotics and Automation, pp 156–165
13. Spence A, Altinas Y (1991) End milling force algorithms for CAD systems. CIRP Ann 40:31–34. doi:10.1016/S0007-8506(07)61927-1
14. Spence AD, Altinas Y (1994) A solid modeler based milling process simulation and planning system. Trans ASME 116:61–69. doi:10.1115/1.2919377
15. Roth D, Ismail F, Bedi S (2003) Mechanistic modeling of the milling process using an adaptive depth buffer. Comput-Aided Des 30(8):1–17
16. Feng H-Y, Menq LM, Chai H, Hang ZL (1995) The prediction of dimensional error for sculptured surface production using the ball-end milling process. Int J Mach Tools Manuf 35(8):1149–1169. doi:10.1016/0890-6955(94)00044-K
17. Mounayri HEI, Spence AD, Elbestawi MA (1998) Milling process simulation—a generic solid modeller based paradigm, journal of manufacturing science and engineering. Trans ASME 120:213–221
18. Fleisig RV, Spence AD (2005) Techniques for accelerating B-Rep based parallel machining simulation. Comput-Aided Des 37:1229–1240. doi:10.1016/j.cad.2004.11.008
19. Vergeest JSM, Walstra WH, Bronsvoort WF (1994) Interactive simulation of robot milling for rapid shape prototyping. Comput Graph (ACM) 18(6):861–871. doi:10.1016/0097-8493(94)90013-2
20. Liu C, Esterling DM, Fontdecaba J, Mosel E (1996) Dimensional verification of NC machining profiles using extended quadtrees. Comput-Aided Des 28(11):845–85. doi:10.1016/0010-4485(95)00077-1
21. Ayala D, Brunet P, Juan R, Navazo I (1985) Object representation by means of nonminimal division quadtrees and octrees. ACM Trans Graph 4(1):41–59. doi:10.1145/3973.3975
22. Brunet P, Navazo I (1990) Solid representation and operation using extended octrees. ACM Trans Graph 9(2):170–197. doi:10.1145/78956.78959
23. Carlbom I, Chakravarty I, Vanderschel D (1985) A hierarchical data structure for representing the spatial decomposition of 3D Object. IEEE Comput Graph Appl 5:24–31
24. Kondo M (1994) Decomposition of complex geometry for a manufacturing application. Comput-Aided Des 26(3):244–252. doi:10.1016/0010-4485(94)90047-7
25. Leu MC, Park SH, Wang KK (1986) Representation of swept volumes and its applications. J Eng Ind Trans ASME 108(2):113–119
26. Roy U, Xu Y (1999) Computation of geometric model of a machined part from its NC machining program. Comput-Aided Des 31:401–411. doi:10.1016/S0010-4485(99)00039-1
27. Takata S, Tsai MD, Sata T (1989) A cutting simulation system for machinability evaluation using a work piece model. CIRP Ann 38(1):417–420
28. Kline WA, Devor RE, Lindberg JR (1982) The prediction of the cutting forces in end milling with application to cornering cuts. Int J Mach Tools Manuf 22(1):7–22
29. Takata S (1993) Generation of a machining scenario and its applications to intelligent machining operations. CIRP Ann 42(1):531–534. doi:10.1016/S0007-8506(07)62502-5

30. Jerad RB, Fussel BK, Hemmett (2001) Robust feedrate selection for 3 axis NC machining using discrete models, ASME. J Manuf Sci Eng 123:214–223. doi:10.1115/1.1365398
31. Mounayri HEI, Spence AD, Elbestawi MA (1998) Milling process simulation—a generic solid modeller based paradigm. J Manuf Sci Eng 120:213–221
32. FeatureCAM. http://www.featurecam.com
33. Website of Delcam, the developer of Powermill. http://www.delcam.com
34. Karunakaran KP, Shringi R (2008) A solid model-based off-line adaptive controller for feed rate scheduling for milling process. J Mater Process Technol 204:384–396. doi:10.1016/j.jmatprotec.2007.11.092
35. Pressman RS, William JE (1977) Numerical control and computer-aided manufacturing. Wiley, USA
36. Dhande SG, Karunakaran KP (1994) Symbolic and computational conjugate geometry of a generic cutter and machine tool assembly. J Des Manuf 4:167–186
37. Meghar D (1983) Geometric modeling using octree encoding. Comput Graph Image Process 24:160–181. doi:10.1016/0734-189X(83)90041-5
38. Karunakaran KP, Shringi R (2006) Octree-to-BRep conversion for volumetric NC simulation. Int J Adv Manuf Technol 32(1–2):116–131. doi:10.1007/s00170-005-0310-8
39. Karunakaran KP (2000) Swept volume of a generic cutter. Proc Inst Mech Eng B, J Eng Manuf 214:915–938. doi:10.1243/0954405001517991
40. Koenigsberger F, Sabberwal AJP (1961) An investigation into the cutting force pulsations during milling operations. Int J Mach Tool Des Res 1:15–33. doi:10.1016/0020-7357(61)90041-5
41. Armarego EJA, Despande NP (1989) Computerized predictive cutting models for forces in end milling including eccentricity effects. CIRP Ann 38:45–49. doi:10.1016/S0007-8506(07)62649-3
42. Armarego EJA, Whitefield RC (1985) Computer based modeling of popular machining operations for force and power prediction. CIRP Ann 34(1):65–69. doi:10.1016/S0007-8506(07)61725-9
43. Yang M, Park H (1991) The prediction of cutting force in ball end milling. Int J Mach Tools Manuf 31(1):45–54. doi:10.1016/0890-6955(91)90050-D
44. Altintas Y, Lee P (1996) A general mechanics and dynamics model for helical end mills. CIRP Ann 45(1):59–64. doi:10.1016/S0007-8506(07)63017-0
45. Lazoglu I, Lyang SY (2000) Modeling of ball end milling forces with cutter axis inclination. J Manuf Sci Eng 122(1):3–11. doi:10.1115/1.533540
46. Tlusty J, Smith S (1991) An overview of modeling and simulation of the milling process. Trans ASME 113:169–175
47. Sabberwal AJP (1962) Cutting forces in down milling. Int J Mach Tool Des Res 2:27–41
48. Sutherland JW, Devor RE (1986) An improved method for cutting force and surface error prediction in flexible end milling systems. J Eng Ind Trans ASME 108:269–279
49. Metal Carboloy Systems Department (1980) Milling handbook of high efficiency metal cutting. General Electric Company, Detroit
50. Martellotti ME (1941) An analysis of the milling process. Trans ASME 63:677–700
51. Yucesan G, Altintas Y (1996) Prediction of ball end milling forces. Trans ASME 118:95–103
52. Budak E, Altintas Y, Armarego EJA (1996) Prediction of milling forces from orthogonal cutting data. Trans ASME 118:261–224. doi:10.1115/1.2824057
53. Liao CL, Tsai JS (1994) Dynamic response analysis in end milling using pre-twisted beam finite elements. J Mater Process Technol 40:407–432. doi:10.1016/0924-0136(94)90465-0
54. Tarng YS, Shyur YY (1993) Identification of radial depth of cut in numerical control pocketing routines. Int J Mach Tools Manuf 33(1):1–11. doi:10.1016/0890-6955(93)90059-4