NC machine tool path generation from CSG part representations

James E Bobrow

Recent improvements in geometric modelling systems have led to the need for more reliable and highly automated software for machine tool path generation. Current machining algorithms require that any part geometric information which cannot be determined from the modelling system be supplied by the user. Much geometric information is needed if the model used to represent the part is incomplete. This is the case with many conventional boundary representation systems. However, this information can easily be determined automatically if a solid modelling system is used. This paper presents a method for generating numerically-controlled milling machine tool paths directly from constructive solid geometry part representations. The algorithm requires less user interaction than APT boundary representation methods. A wide variety of parts may be machined using standard torus (bull) ended milling cutters. The algorithm is computationally efficient, and requires iteration only on portions of the part where gouging may occur.

geometric modelling, algorithms, NC machine tool path generation

The use of computer-based systems for modelling the geometry of solid objects is becoming increasingly more prominent in various manufacturing applications. The geometric modelling systems used for these applications are becoming more reliable, and are able to automatically perform many operations that required human interaction in the past. One operation which requires considerable interaction is the generation of numerically controlled (NC) machine tool paths. In this paper, we discuss the widely used APT[1,2] (automatically programmed tools) algorithm for the generation of NC tool paths, and we present an alternative method that requires less user interaction to obtain the tool path provided that a good geometric modelling system for the part description is available.

The most common types of geometric modelling systems used for machine tool path generation are boundary representation systems. Most machining algorithms which use boundary representation systems, including APT, do not require the system to possess all the information necessary to represent the true solid model, ie, to contain all the necessary face, edge, and vertex data relationships needed to represent the solid[3]. This is because the machining algorithms require the user to supply any information that cannot be obtained from the part geometric representation. Hence, most of the burden is placed on the user instead of on the modelling system. APT part programming has become a skilful trade in itself. The part programmer must be well trained in using the APT language, and must know the various techniques to use when the algorithm does not converge (as is frequently the case for irregularly curved surfaces).

Boundary-representation based systems that have machine tool path generation capabilities usually allow for the description of general curved object surfaces. These surfaces consist of arrays of parametric patches[4] which are specified by a finite number of points. General parametric surfaces are used extensively to model objects to be machined that have irregularly contoured surfaces (eg many injection-moulded objects). The main disadvantage of using parametric surfaces in a modelling system is that it is difficult to represent accurately a surface with non-rectangular parametric boundaries, and it is difficult to represent arbitrary holes through a surface. The accurate representation of these boundaries requires that a large amount of data be stored with the surface in order to approximate the curves.

The machine tool path generation algorithm presented here can be used with any modelling system capable of representing the true solid object. We used the University of Rochester's PADL-2[5] constructive solid geometry (CSG) system since much of the software required for our algorithm is available with PADL-2, and McAuto has the interactive Unisolids/PADL-2[6] system. With Unisolids, an accurate description of the part can be created interactively very quickly, provided that the part can be represented with the limited number of solid primitives available for the construction.

Current CSG based systems including PADL-2 do not have a primitive which can be used to represent a general surface. This limits the number of realistically shaped objects that the system can represent. However, a wide variety of typical objects to be machined can be constructed using the few primitives (block, sphere, cone, and wedge) of PADL-2. The algorithm presented here can generate machine tool paths for these objects more automatically than APT because it exploits the complete solid representation available with the PADL-2 CSG system. No attempt was made to retain any of the APT commands or language (unlike Chan[7]) with this system, since many of these commands become unnecessary.

Several researchers have addressed the general problem of machine tool path generation and verification using constructive solid geometry or similar representation schemes. Hunt and Voelcker[8,9] have studied extensively the problem of NC machine tool path verification. Their work provides a general scheme for uncovering various problems

Department of Mechanical Engineering, University of California, Irvine CA 92717, USA

This research was conducted at McDonnell Douglas Automation, Cypress, CA 90630, USA

 0010—4485/85/020069—08 $03.00

which may arise from an incomplete or inaccurate NC tool path. In their method, first the individual commands are examined to determine whether the tool will gouge the part or clamps, and then the actual finished part is compared to the desired finished part to determine the amount of material remaining and the undercut regions. Arbab[10] has proposed a similar technique using his 'Deforming Solid Geometry' and 'Realizable Shape Calculus' solid modelling methods. Armstrong[11] has developed a method for tool path generation and process planning for parts with planar surfaces of different heights. This is a somewhat limited class of parts, but the ability to combine tool path generation and process planning is highly desirable.

The tool paths considered in the studies mentioned above were only those which would give cutter swept volumes that could be represented using PADL-2 primitives. This is a strict limitation since these primitives are not general enough to represent five axis rotations as well as non-planar three axis translations of bull (torus) ended cutting tools. The main reason for requiring a PADL-2 representation of the cutter swept volumes was to ensure that an exact representation of the final part produced by the NC tool path could be computed and compared to the desired final part. The ability to compare the actual final part machined to the desired final part is important, but it is not a necessary requirement for producing tool paths.

For the work presented here we take a different point of view than was used in the studies mentioned previously. Given a description of the final part to be machined and the cutting tool parameters, we find a tool path which will yield a part which is close to the desired one, regardless of whether it is possible to verify the path using PADL-2 primitives. In the near future we may have the ability to verify tool paths to within normal machining tolerances using a secondary octree spacial enumeration representation[12,13] and high performance hardware. For simplicity, the method presented here is for 3-axis machining. However, it may easily be extended to 5-axis machining, and the required modifications to the algorithm will be noted at the appropriate instances.

The algorithm presented in this paper is for one continuous motion (pass) of a tool across a part. To machine an entire part, multiple passes or applications of this algorithm must be used. The algorithm generates parametric curves on the surface to be machined which represent tool contact points. Roughly speaking, these curves are obtained by intersecting the part surfaces with an infinite plane, and determining the portions of the intersection curves that lie on the object. The parametric curves are then shortened to produce non-gouging tool contact intervals.

The key to obtaining each pass is the tool contact curve-trimming algorithm which produces non-gouging tool positions along any curve on the surface of a part. The trimming algorithm uses an iterative APT-like procedure which positions the tool on the contact curve at the intersection of the surface to be machined and the surface to be avoided. The position found corresponds to a parameter value on the tool contact curve which gives the last good (non-gouging) tool position along that curve.

SURFACE DEFINITION

The first problem that we must address is that of determining which regions on the part are to be cut and which regions on the part must be avoided. In APT, this problem

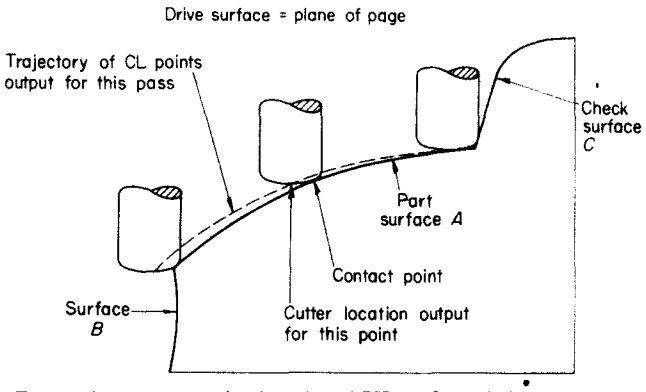


Figure 1. An example showing APT surface definitions

is handled by the part programmer for each continuous tool motion (pass) as shown in the following example.

In Figure 1 let surface *A* be the surface to be machined (part surface), surface *B* be the left side of the part, and surface *C* represent a boundary of the part which must not be gouged (check surface). In addition to the part surface *A*, a second surface on which the tool motion takes place must also be specified. This is called the drive surface. In this case let the drive surface *D* be the plane of the page. After defining these surfaces using the APT programming language, the part programmer instructs the APT module to position the tool at the intersections of surfaces *B* and *D*, and tangent to the part surface *A*. The APT module is then ready to generate the cutter location file (CL file) by starting from the current tool position and remaining on the drive surface *D* tangent to the part surface *A* until the check surface *C* is reached. The next tool motion can be generated after new drive, part, and check surfaces have been specified.

In order to generate a tool path, we must have some method for interactively selecting the surfaces to be processed. With the PADL-2 CSG system, each surface is actually a halfspace boundary of a solid primitive, and we know only the location of the infinite surface. We may evaluate the bounding edges of the surface by traversing the CSG tree[14]. One method of selecting a surface is to cast a ray along a user-defined vector or along the view direction from the screen cursor location. The surface selected would be the first one the ray pierces. A second more automatic method is to machine any surface on the 'top' of the part; the top being defined as any surface whose normal has an upward component. This method will lead to problems if there is a surface with an upward normal that the machine tool cannot reach. For instance, the surface may lie beneath an overhanging portion of the part. The method we implemented was to have the user select a surface by choosing any two of its edges. This made it easy to choose all the surfaces to be machined and all the surfaces to be avoided with the same interactive routine. Automatic part surface selection and path planning techniques were not considered in this study. However, this method provides a strong basis for such investigation, and could be used with systems such as that of Choi *et al*[15].

In our version of PADL-2, all the surfaces of a part which intersect to give an edge have their pointers stored with the boundary file data for that edge. Once the user has selected at least two edges of all the surfaces to be machined, the pointers to the surfaces that give rise to each edge are retrieved. These pointers are then processed in a routine that saves all non-unique numbers from a list of numbers in order to determine the appropriate surfaces to be machined.

CUTTER MOTION GENERATION

Once the surfaces to be machined and the surfaces to be avoided have been determined, we are ready to generate cutter motions. There are two different approaches that are commonly used to generate tool paths with the parametric surfaces found in boundary representation systems. These are the APT approach and the surface parameter tool contact curve approach. Both of these approaches produces a cutter location file, which is a list of cutting tool endpoint positions and tool axes that will be input to a numerically controlled machine tool. The NC machine will then move the cutting tool to each one of the CL points in the sequence that they are input. The motion of the cutting tool in between the CL points is usually given by linear interpolation. However, if the input points form a circular arc, then a circular motion in between these points is sometimes used. Note that this method of motion may produce some error in the actual part machined, but the CL points can be spaced together as closely as is needed for any required accuracy.

Before discussing machining algorithms, we will demonstrate the difference between the tool contact point and the actual point that is output to the CL file. For the case of three-axis machining as in Figure 1, the tool axis vector remains constant throughout the motion, and the three coordinates of the tool end point are specified as a sequence of points to be connected by straight line moves of the cutting tool. For a given contact point and tool axis, there is only one tool end point position which causes the tool to be tangent to the part surface at the contact point, and it is this endpoint location that is output to the CL file. The method used to compute the endpoint for a known contact point and tool axis is given in the Appendix. The tool position in the centre of Figure 1 shows the difference between the CL point and the contact point. Notice that for the tool position on the right side, the CL point and contact point are virtually identical. For five-axis machining, the tool axis is usually kept normal to the part surface, so the tool endpoint will coincide with the contact point throughout the motion.

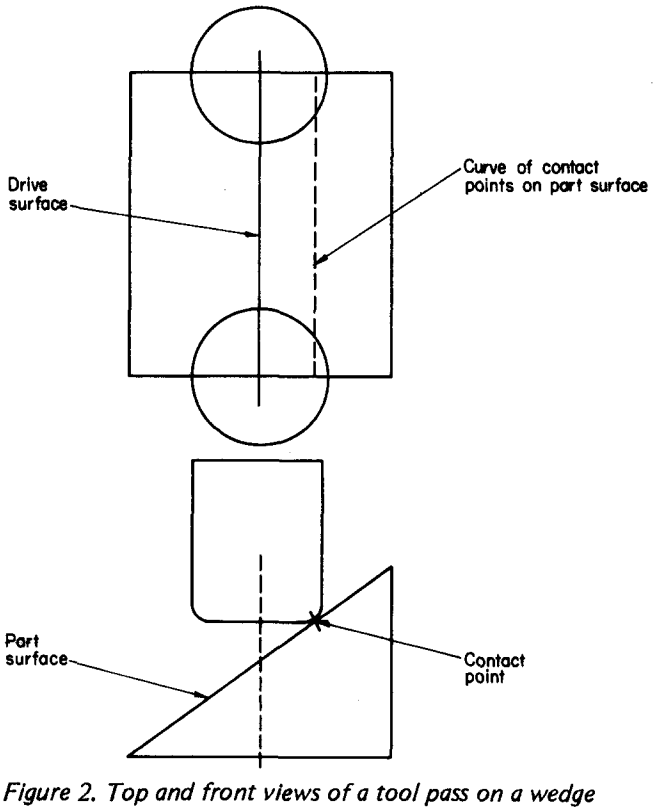


Figure 2. Top and front views of a tool pass on a wedge

As another example, consider the simple case of generating one pass along the surface of the wedge shown in Figure 2. In APT, the part programmer specifies the drive surface and part surface as shown, and generates the tool motion such that the tool axis lies on the drive surface, and the tool end is tangent to the part surface. To generate the tool motion, the APT algorithm begins with the tool raised off the part surface, and iteratively searches for the location of the tool end point that causes the tool to be tangent to the part surface. For a planar part surface as in this case, only one iteration is required. For a curved part surface, several iterations are needed, depending on the surface curvature. The heart of the iteration is a routine that computes the minimum distance between the tool and part surface. After the point of tangency is found, the corresponding tool endpoint is output to the CL file, and the tool is moved a small step along the drive surface in a direction that is tangent to the part surface. The minimum distance iteration for the tool endpoint location is then repeated. The points of tangency on the part surface that are converged upon for each step of the tool's motion form a locus of tool contact points. These points lie in a straight line for the case of a wedge as is shown in Figure 2.

This example demonstrates that the APT algorithm will cause the tool to cut a portion of the surface that may not have been expected, since the user-specified drive surface is not close to the contact points for this case. This is a problem, as the user usually knows the area on the part surface that should be cut and not where the input drive surface should be located.

For general curved surfaces and tool paths, the APT tool positioning algorithm requires iterative computation of the minimum distance between the tool and both the drive and part surfaces at every output point of the motion. Each minimum distance computation is also iterative, so the entire procedure is often computationally time-consuming for curved parametric surfaces. Also there is no guarantee that the iterations will converge for irregularly curved sculptured surfaces. Apart from these disadvantages, the two strongest points of the APT approach are:

- The tool motion may be in any direction specified regardless of the surface parameterization; the motion is defined entirely by the drive and part surfaces.
- The algorithm ensures that the tool will not gouge any specified surface because the endpoint is found by using a minimum-distance computation.

An equivalent tool motion to that shown in Figure 2 could have been produced directly by specifying the trajectory of the tool contact points obtained from the APT algorithm in the form of a parameter curve on the part surface. Given any point on the parametric curve and its corresponding surface normal, we can position the cutting tool tangent to the surface at that point by using the method in the Appendix. As shown in the Appendix, the point of tangency and the tool axis vector uniquely determine the location of the tool endpoint. The APT method produced the same points of tangency on the part surface for any position on the drive surface. Hence, the cutter location files produced by both methods give the same contact points and represent identical tool paths. The obvious advantage of using parametric tool contact curves is that no iterations are required to generate the CL file. Only a simple algorithm for stepping the tool along the contact curve is required.

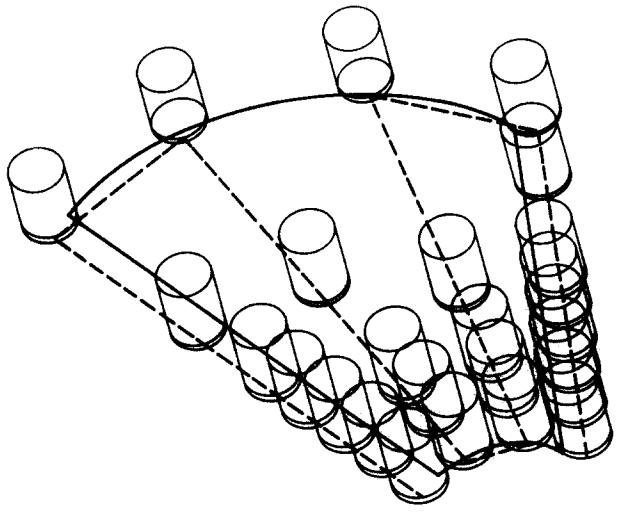


Figure 3. Machining along constant parameter curves

Typically, parameter curve machining algorithms produce tool motions in either constant u or v surface parameter directions (surface coordinates = $S(u,v)$ where $S \epsilon R^3$). There are two main problems with this approach; the first is due to the surface parameterization, and the second is due to the method of tool placement as described in the Appendix. The first problem is demonstrated in the ruled surface of Figure 3. It is difficult to machine this surface along the constant u or v directions since the actual stepover distance is much greater on the top than it is on the bottom, even though the parameter increment on the top is the same as it is on the bottom.

The second problem is that the method for placing the tool given in the Appendix sometimes causes the tool to gouge the part surface. The gouging occurs either on the part surface being machined as in Figure 4(a), or at the transition from one surface to another as in Figure 4(b). The problem in Figure 4(a) occurs when the radius of the cutting tool is large compared to the curvature of the surface being machined. The part programmer may correct it by selecting a smaller cutting tool, or it can be corrected approximately with a routine which searches the curve for points which cause the tool to gouge, such as point *C* in Figure 4(a), and eliminates these bad points from the tool path (as in the path from point *A* to point *B* in Figure 4(a)).

The problem shown in Figure 4(b) occurs when multiple surfaces are to be machined consecutively in one pass. The tool position at the end of parametric curve 1 causes the tool to gouge the surface containing curve 2 and *vice versa.* This problem can be avoided by decreasing the parameter limit of curve 1 to end at point *A*, and changing the starting parameter limit of curve 2 to begin at point *B*. This would eliminate gouging of either one of the part surfaces, but would not remove the material beneath the tool near the intersection of the surfaces. This implies that the part that would be produced by this tool path is not identical to the part desired. However it is as close as can be obtained without gouging the part surfaces. The part programmer can obtain a more accurate tool path for this example by selecting a smaller cutting tool.

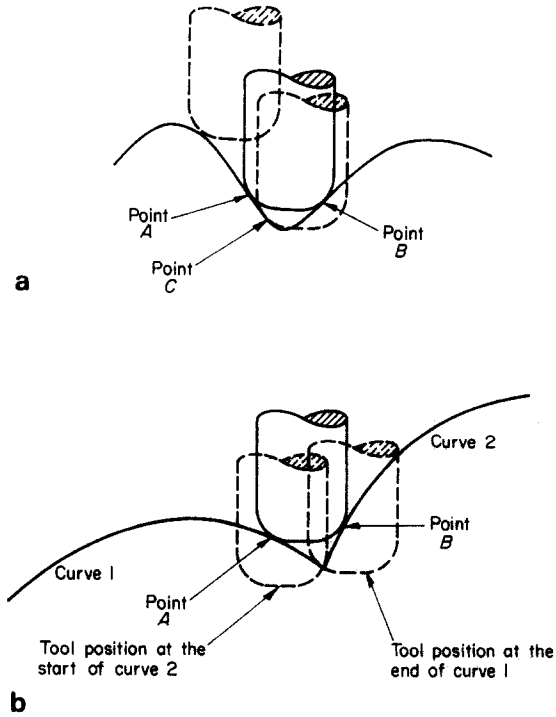
The method used for finding the surface parameter curve limits *A* and *B* in Figure 4(b) is presented in the next section of this paper, and is a key requirement for our tool path generation algorithm. With APT, the problem of gouging for this part would not have occurred as the part programmer would have generated the two portions of the tool path separately. The surface containing curve 2 would be specified as a check surface for the first part of the tool's motion, and the surface containing curve 1 would be specified as a start-up surface for the second part of the tools motion. This technique requires much more user interaction than that of the surface parameter curve approach. Note that the APT method also would not produce a part identical to the part desired.

METHOD OF PATH GENERATION IMPLEMENTED

The approach that we used to generate tool paths on PADL-2 parts was to use parametric tool contact curves to define the tool paths as discussed above, instead of using the APT method. The method presented here can generate passes which traverse a complex array of surfaces with no constraints on the direction of cut. To generate one pass of the cutting tool across an array of surfaces as shown in Figure 5, the user defines an infinite plane which will intersect the part surfaces that he has chosen to be machined. To generate a tool path which will cover the entire part, a family of planes parallel to the initial one must be specified. Let us emphasize that the surfaces selected to be machined are CSG solid primitive faces which are represented as infinite surfaces (halfspaces) internally.

The algorithm first generates the infinite intersection curves that the plane makes with the part surfaces using standard routines available in PADL-2. Next, each intersection curve is classified against the object to determine the portions of the curve which lie on the object (point classification is a process used in PADL-2 to determine if points lie inside, outside, or on an object). Each curve segment is then stored as a simple geometric entity using the method described in Check *et al*[16,17].

Once the pointers to each analytic curve segment have been stored, they must be ordered in the same sequence that they will be traversed by the cutting tool. This is done by having the user select the starting point, and by computing which curve has an end point closest to the starting



Figures 4(a) and (b). Two types of gouging that can occur

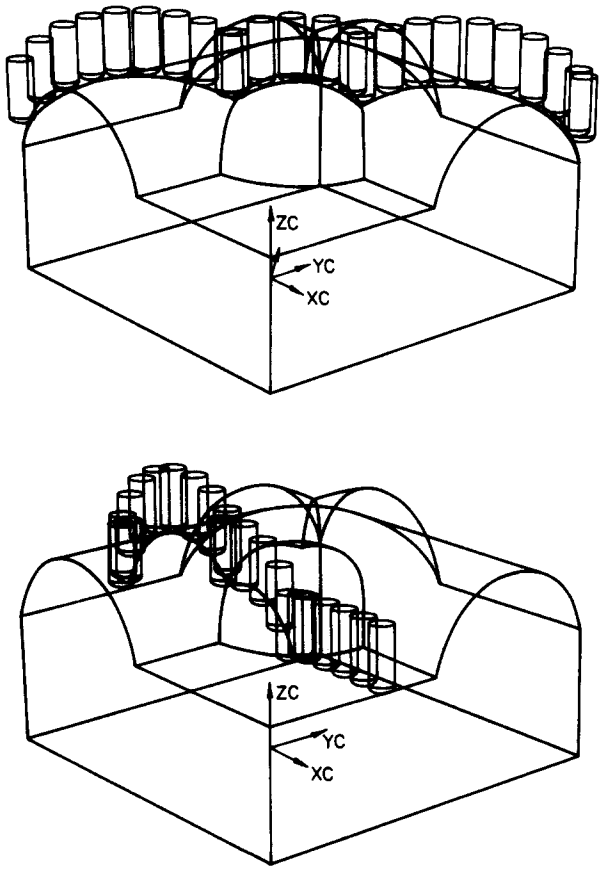


Figure 5. Two sample tool passes over a Unisolids part

point. The starting point is then reset to be the opposite end of this curve, and the closeest end of the remaining curves to this point is then found. This process is repeated until all the curves have been ordered.

In addition to ordering the curves, a flag must be set for each curve to denote the positive direction of cut relative to the curve's parameterization. This is easily accomplished by comparing the desired tool motion to the increasing parameter direction. Also, an outward normal direction flag is needed for each surface containing a contact curve and for each check surface. Every PADL-2 surface has an outward normal direction associated with it that is not necessarily the direction that points to the outside of the solid. To determine if the outward normal of a surface is the outward normal of the solid, a point must be offset from the surface in the direction of its normal. This point is then classified against the solid object to determine if it lies inside or outside of the solid. Note that this would not be possible without a true solid representation.

TRANSITIONS BETWEEN CONTACT CURVES

In order to machine along analytic tool contact curves, we must develop a method that eliminates gouging at the transitions between surfaces as in Figure 4(b), and at the check surfaces. To do this we need to find the last parameter value on the curve that gives a tool position that does not gouge the check surface. The approach we used is similar to that of the APT ARLEM III[2] processor. It requires that the derivative of the contact curve with respect to the parameter can be computed at any point on the curve, and that the minimum (perpendicular) distance between the cutting tool and the check surface can be computed for any position and orientation of the cutting tool.

The algorithm is described, with reference to Figure 6 and using the notation defined in the Appendix, as follows:

(a) At some point t on the contact curve, evaluate the derivative of the curve with respect to the parameter at t.
(b) At this point, position the tool as in the Appendix.
(c) From this tool position, compute the minimum distance to the check surface, along with the corresponding point on the tool Pt, the point on the surface Ps, and the unit surface Snl.
(d) Check if $\| Pt - Ps \| < \delta$, for some small $\delta > 0$. If it is, then the iteration is complete.
(e) Compute the parameter increment Δt such that the distance that the tool moves in the direction normal to the planar approximation of the surface at Ps is equal to the perpendicular distance between the tool and the check surface. This parameter increment is given by:

$$< ds/dt\ \Delta t,\ Snl > = < Pt - Ps,\ Snl > \qquad (1)$$

or

$$\Delta t = < Pt - Ps,\ Snl > / < ds/dt,\ Snl > \qquad (2)$$

These five steps are repeated until the convergence criterion in step d is satisfied.

As with APT, it is difficult to prove that this algorithm will converge for the most general combination of contact curve and check surface. However for a straight line and planar check surface, one iteration gives the exact solution. Thus we can expect that the algorithm will converge when there is a smooth curve and smooth check surface, and when the iteration is started with a good initial guess for the parameter. We have tested this algorithm with many combinations of curves and surfaces, and in most cases the algorithm converged in less than five iterations. Cases were encountered that did not converge only when the intersecting surfaces were very curved in comparison with the tool dimensions. In these cases, use of a smaller cutting tool easily corrected the problem.

As stated earlier, the algorithm requires both the derivative of the contact curve with respect to the parameter, and the minimum distance from the tool to the check surface. Since each contact curve has an analytic description, the derivatives are readily available. The minimum distance computation, however, requires more effort. The calculations depend strongly on the fact that the normal to the tool at its minimum point Pt aligns with the normal to the surface at its minimum point Ps. This can be shown by first noting that the minimum distance vector from a point in space to a surface is normal to the surface at the minimum. Then we can use this fact twice to show that the

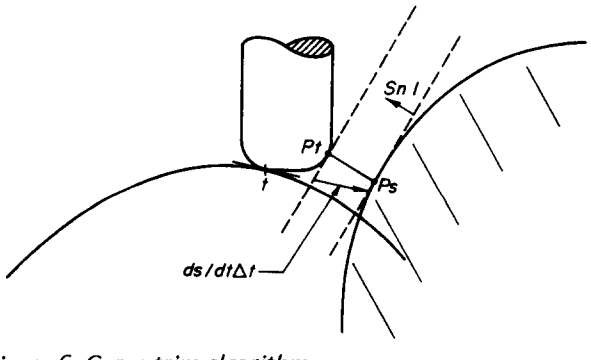


Figure 6. Curve trim algorithm

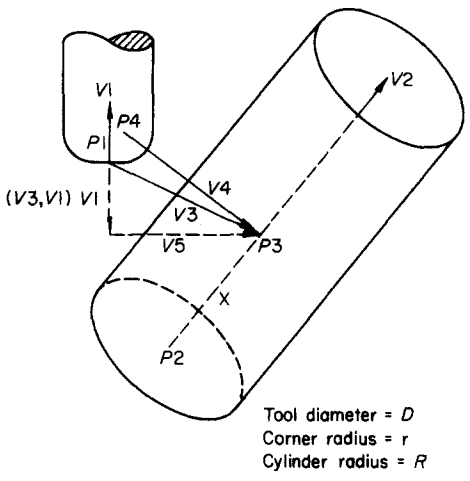


Figure 7. Minimum distance between a cylinder and a tool

minimum distance vector between two surfaces is normal to both surfaces.

For a plane and a sphere, the minimum distance can be computed without iteration. For other surfaces in PADL-2 we found it necessary to use a simple first order Newton iteration on one variable. There are several special cases which must be considered, but the following is a representative procedure for computing the minimum distance between a cylinder and the cutting tool.

It is assumed that we know the location of the tool end $P1$, the tool axis unit vector V1, the tool diameter D, the tool corner radius r, a point on the cylinder $P2$, the cylinder axis unit vector V2, and the cylinder radius R as shown in Figure 7. Let $P3$ be a point on the cylinder axis given by $P3 = P2 + x\mathrm{V2}$ where x is a scalar. For any value of x, we can construct a vector V4 normal to the tool which passes through $P3$ by the following vector computations (see Figure 7):

$$P3 = P2 + x\mathrm{V2} \tag{3}$$

$$\mathrm{V3} = P3 - P1 \tag{4}$$

$$\mathrm{V5} = \mathrm{V3} - <\mathrm{V3}, \mathrm{V1}>\mathrm{V1} \text{ (orthogonalization)} \tag{5}$$

$$U = <\mathrm{V5}, \mathrm{V5}>^{-1/2}\mathrm{V5} \text{ (a unit vector perpendicular to the tool axis)} \tag{6}$$

$$P4 = P1 + (D/2 - r)\,U + r\mathrm{V1} \tag{7}$$

$$\mathrm{V4} = P3 - P4 \tag{8}$$

What remains to be found is a value for x that gives the vector V4 which is also perpendicular to the cylinder axis V2. If this is found, we have our minimum distance points on the tool and cylinder simply by placing them the appropriate distance along the vector V4. To find x, we use a first order Newton's method as follows. For any x, compute V4 as in equations (3) to (8) and let:

$$<\mathrm{V4}, \mathrm{V2}> = e(x) \tag{9}$$

We want $e(x) = 0$, so for the next iteration, solve equation (10) for the value of Δx that gives the best linear approximation of this condition.

$$e(x) \simeq e(x_o) + \frac{de}{dx}\,\Delta x = 0 \tag{10}$$

$$\Delta x = -e(x_o)/\frac{de}{dx} \tag{11}$$

All that remains is to solve for de/dx, which is done as follows:

$$\frac{de}{dx} = <\frac{d\mathrm{V4}}{dx}, \mathrm{V2}> \tag{12}$$

$$\frac{d\mathrm{V4}}{dx} = \frac{d}{dx}\,(P2 + x\mathrm{V2} - ((D/2 - r)\,U + r\mathrm{V1} + P1)) \tag{13}$$

$$\frac{d\mathrm{V4}}{dx} = \mathrm{V2} - (D/2 - r)\,\frac{dU}{dx} \tag{14}$$

Substituting equation (14) back into equation (12) gives:

$$\frac{de}{dx} = 1 - (D/2 - r) <\frac{dU}{dx}, \mathrm{V2}> \tag{15}$$

So we need dU/dx, which is given by

$$\frac{dU}{dx} = <\mathrm{V5}, \mathrm{V5}>^{-1/2}\,\frac{d\mathrm{V5}}{dx} + \frac{d}{dx}\,(<\mathrm{V5}, \mathrm{V5}>^{-1/2})\,\mathrm{V5} \tag{16}$$

where

$$\frac{d\mathrm{V5}}{dx} = \frac{d}{dx}\,(\mathrm{V3} - <\mathrm{V3}, \mathrm{V1}>\mathrm{V1}) \tag{17}$$

$$\frac{d\mathrm{V5}}{dx} = \frac{d}{dx}\,(P2 + x\mathrm{V2} - P1 - <P2 + x\mathrm{V2} - P1, \mathrm{V1}>\mathrm{V1}) \tag{18}$$

$$\frac{d\mathrm{V5}}{dx} = \mathrm{V2} - <\mathrm{V2}, \mathrm{V1}>\mathrm{V1} \tag{19}$$

and

$$\frac{d}{dx}\,(<\mathrm{V5}, \mathrm{V5}>^{-1/2}) = -<\mathrm{V5}, \mathrm{V5}>^{-3/2} <\frac{d\mathrm{V5}}{dx}, \mathrm{V5}> \tag{20}$$

This iteration on the value of x is repeated by evaluating equations (11) to (20) until $e(x) = 0$. When we have found the correct value for x, the vector V4 will be perpendicular to both the cutting tool and the cylinder. The minimum distance points Pt and Ps required by the curve trimming algorithm lie along V4.

As stated earlier, there are several special cases which must be considered when performing the minimum distance computation. The most common case (which was just presented) is when the minimum distance is between the toroidal part of the tool and the cylinder. However, the minimum distance vector may intersect the bottom flat portion of the tool, or it may intersect the side cylindrical portion of the tool. The presence of either of these situations can be ascertained by some initial computations regarding the angle between V1 and V2, and the location of the tool end $P1$ relative to the cylinder.

SUMMARY OF THE METHOD

All the steps that are needed to compute tool paths have now been defined. After the part surfaces and check surfaces have been selected as in the second section, we compute tool motions as follows:

- slice the part surfaces with a planar surface and classify the intersection curves with respect to the object to find those portions of the curve which lie on the object
- order the curves, set positive cut direction flags and set outward surface normal flags
- trim the ordered curves against all part surfaces and all check surfaces to obtain nongouging contact curves
- output the cutter location data using a suitable stepping algorithm and the tool positioning procedure in the Appendix

This procedure will produce one pass of the cutting tool across the part surfaces. To machine the entire part, multiple applications of this algorithm must be performed. For each pass, a new planar surface must be offset by an appropriate lateral step-over distance from the one previously defined.

CONCLUSIONS

The research presented here has exploited the complete solid representation available with constructive solid geometry to develop a tool path generation algorithm that requires less user interaction than APT. The APT method requires the user to supply information that can easily be determined automatically if the true solid model of the part being machined is available. In addition to being more automatic, the algorithm presented here is computationally efficient because the minimum distance routine, which is at the lowest level for this algorithm and for APT, is only invoked on portions of the part where gouging may occur.

To generate tool paths, parametric surface curves produced from standard PADL-2 surface intersection routines are used to create a locus of candidate tool contact points. These parametric tool contact curves are then shortened using a minimum distance routine to produce non-gouging tool motions. The method presented here has been successfully used for a wide variety of objects and complex arrays of surfaces.

APPENDIX

The following procedure is used to find the location of the tool end point which will cause the tool to be tangent to a surface, given the unit tool axis vector, the surface point, and the unit surface normal vector. As is shown in Figure 8, let *Tend* be the tool end point, *Tax* be the tool axis vector, *Spt* be the surface point, and *Snl* be the surface normal.

Using the following notation:

$$< a, b > = a_1 b_1 + a_2 b_2 + a_3 b_3 \text{ (dot product)} \quad (21)$$

$$\| a \| = < a, a >^{1/2} \text{ (Euclidean norm)} \quad (22)$$

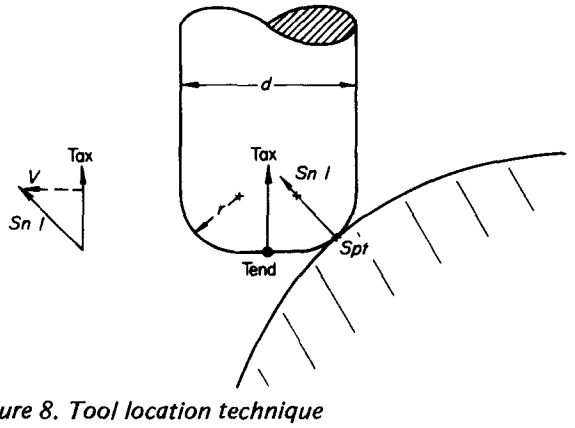


Figure 8. Tool location technique

Construct a vector perpendicular to the tool axis in the plane containing *Tax*, *Tend* and *Snl*.

$$V = Snl - < Snl, Tax > Tax \quad (23)$$

If $\| V \| = 0$, then the tool end *Tend* is equal to the surface point *Spt*. Otherwise, unitize this vector and compute the tool end point *Tend*.

$$U = V / \| V \| \quad (24)$$

$$Tend = Spt + rSnl + (d/2 - r)\, U - rTax \quad (25)$$

REFERENCES

1 *APT Part Programming* IIT Research Institute, McGraw-Hill (1967)

2 *APT IV Computer System Manual, Volume 2, Subroutine Library A4V3* Computer Aided Manufacturing-International, TX, USA

3 **Requicha, A A G** 'Representations of Rigid Solid Objects' in *Computer-Aided Design* Encarnacao, J (ed) *Lecture notes in computer science* (1980)

4 **Faux, I D and Pratt, M J** *Computational geometry for design and manufacture* John Wiley (1979)

5 **Voelcker, H, Requicha, A, Hartquist, E, Fisher, W, Metzger, J, Tilove, R, Burrell, N, Hunt, W, Armstrong, G, Check, T, Moote, R and McSweeney, J** 'The PADL-1.0/2 system for defining and displaying solid objects' *ACM Comput. Graphics* Vol 12 No 3 (August 1978)

6 *Unisolids operational description* Version 2.0, McDonnell Douglas Automation Company, CA, USA (August 1984)

7 **Chan, B T F** 'ROMAPT: a new link between CAD and CAM' *Comput.-Aided Des.* Vol 14 No 5 (September 1982) pp 261–266

8 **Hunt, W A and Voelcker, H B** *An exploratory study of automatic verification of programs for numerically controlled machine tools* Production Automation Project TM-34, University of Rochester, NY, USA (January 1982)

9 **Voelcker, H B and Hunt, W A** 'The role of solid modelling in machining – process modelling and NC verification' *Proc. SAE Int. Congr. Exposition*, MI, USA (February 1981)

10 **Arbab, F** 'Requirements and architecture of CAM-oriented CAD systems for design and manufacture of mechanical parts' *Ph D Dissertation* University of California at Los Angeles, LA, USA (1982)

11 **Armstrong, C A** 'A Study of automatic generation of non-invasive NC machine paths from geometric models' *Ph D Dissertation* The University of Leeds, UK (October 1982)

12 **Meagher, D J** *The octree encoding method for efficient solid modelling* Image Processing Laboratory IPL-TR-032, Rensselaer Polytechnic Institute, USA (August 1982)

13 **Meagher, D J** *Octree generation, analysis and manipulation* Image Processing Laboratory IPL-TR-027, Rensselaer Polytechnic Institute, USA (April 1982)

14 **Voelcker, H B, Requicha, A A G** Boundary evaluation procedures for objects defined via constructive solid geometry *Tech. Memo. 26,* Production Automation Project, University of Rochester, NY, USA (1980)

15 **Choi, B K, Barash, M M and Anderson, D C** 'Automatic recognition of machined surfaces from 3D solid model' *Comput.-Aided Des.* Vol 16 No 2 (March 1984) pp 81–86

16 **Check, T F, Dodsworth, J, Hartquist, E E and Tilove, R B** *CGPAK: a computational geometry group memo No 17* Production Automation Project, University of Rochester, NY, USA (June 1982)

17 **Check, T F, Dodsworth, J, Hartquist, E E and Tilove, R B** 'Representations in the PADL-2.0/N Processor: Low Level Geometric Entities' *Computational Geometry Group Memo. No 12,* Production Automation Project, University of Rochester (June 1982)