Optimal grids for five-axis machining

W. Anotaipaiboon\textsuperscript{a}, S.S. Makhanov\textsuperscript{b,*}

\textsuperscript{a} Department of Electrical and Computer Engineering, Faculty of Engineering, Thammasat University, Thailand
\textsuperscript{b} School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University, 135 Moo 5 Tiwanont Road, Bangkadi, Muang, Pathum Thani 12000, Thailand

Received 23 February 2009; received in revised form 3 March 2010; accepted 5 May 2010
Available online 31 May 2010

Abstract

This paper presents a new grid generation method for tool path planning for five-axis machining based on minimization of the kinematics error. First, the procedure constructs a space-filling curve so that the scallops between the resulting tracks of the tool do not exceed a prescribed tolerance. At the second stage (which is the subject of this paper) a one-dimensional grid along the space-filling curve is generated using direct minimization of the kinematics error. A closed form representation of the kinematics error as a function of locations cutter contact points is derived from the inverse kinematics transformations associated with a particular five-axis machine and obtained through automatic symbolic calculations. The grid of cutter location points is generated so that it minimizes the kinematics error. Numerical and cutting experiments demonstrate that the proposed procedure outperforms distribution of points based on the equi-arc-length principle. Finally, we show that our optimization routine requires less points than that based on sequential point insertion and numerical evaluation of the cost function, such as bisection.

© 2010 Published by Elsevier B.V. on behalf of IMACS.

Keywords: Grids; CNC optimization; Milling machines

1. Introduction

Milling machines (Figs. 1 and 2) are programmable mechanisms for cutting complex industrial parts. The machines are designed in such a way that the cutting tool (cutter) is capable of approaching the desired surface at a given point with a required orientation.

The machine consists of several moving parts designed to establish the required coordinates and orientations of the tool during the cutting process. The movements of the machine parts are guided by a controller which is fed with the so-called NC program or G-code comprising commands carrying three spatial coordinates of the tool tip and a pair of rotation angles needed to rotate the machine parts to establish the orientation of the tool. The tool path \( \Pi = \{ \Pi_0, \Pi_1, \ldots, \Pi_m \} \) is a sequence of positions \( \Pi_p = \{ M_p, I_p \} \), where \( M_p = \{ x_p, y_p, z_p \} \) are the Cartesian coordinates of the tip of the tool in the machine coordinate system (cutter location points) and \( I_p = \{ I_x, I_y, I_z \} \) the orientation vector.

* Corresponding author. Tel.: +66 89 488 17 96; fax: +66 25 013 524.
E-mail addresses: makhanov@siit.tu.ac.th, smakhanov@yahoo.com (S.S. Makhanov).

0378-4754/$36.00 © 2010 Published by Elsevier B.V. on behalf of IMACS.
doi:10.1016/j.matcom.2010.05.004
The rotation angles $\mathcal{R}_p = \{a_p, b_p\}$ are represented in terms of the components of $I_p$ given a configuration of the machine. Such machining is called Computer Numerically Controlled (CNC) machining. The configuration of the five-axis machine is characterized by (see Fig. 3):

- two rotation matrices $A, B$ corresponding to the two rotary axes;
- two translations associated with the position of the workpiece and the design of the five-axis machine $T_{23}, T_{24}$;
- the length of the tool $L$. Since we consider the tool aligned with the $Z_4$-axis (Fig. 3), $L$ is treated as an additional translation $T_4$ ($T_4$ is either $(0, 0, L)$ or $(0, 0, -L)$ depending on the direction of the tool tip in the spindle coordinate system).
Furthermore, let \( S(u, v) = (x(u, v), y(u, v), z(u, v)) \) be the required surface. The tool path optimization problem is formulated as follows [25]:

\[
\text{minimize}(C), \quad \Pi
\]

where the criterion vector \( C \) includes one or several estimates of the difference between the required and the actual surface and minimization applies to each component of \( C \) independently.

The vector may also include the length of the path, the negative of the machining strip (strip maximization), the machining time, etc., see, for instance, [19,21,28]. Besides, the optimization is often subjected to constraints, the most important of which are the scallop height constraints, the local and the global accessibility constraints [5,13,20,31]. A popular tool path minimization is performed with regard to the length of the tool path \( L \) as follows:

\[
\text{minimize}(L), \quad \Pi
\]

subject to

\[
\begin{align*}
h &< h_{\text{max}}, \\
\varepsilon &< \varepsilon_{\text{max}}
\end{align*}
\]

where \( h \) is the scallop height, measured between each pair of the adjusted tracks, and \( h_{\text{max}} \) is the maximum allowed scallop height, \( \varepsilon \) is a difference between the actual and the prescribed trajectory and \( \varepsilon_{\text{max}} \) the prescribed tolerance.

The two constraints are important. As a matter of fact they are nothing else than a numerical approximation of a general constraint that the machined surface does not deviate from the actual surface by more than \( T_{\text{max}} = \max(h_{\text{max}}, \varepsilon_{\text{max}}) \). Although, \( T_{\text{max}} \) characterizes only an approximation to the actual cut. Suffice to say that \( h_{\text{max}}, \varepsilon_{\text{max}} \) are measured at different points. Still the actual milling error might exceed \( T_{\text{max}}, \) however, \( h_{\text{max}} \) and \( \varepsilon_{\text{max}} \) are the error estimates which should be considered first. Note that, in five-axis machining the shortest spatial length \( L \) does not necessarily mean the shortest time when the tool path has many turns. In this case the definition of the distance should include the angular path as well (see a relevant discussion in [6]). In the meantime, this paper is focused on the second constrain whereas the scallop optimization is performed by an adaptive space-filling curve technology proposed in [3,4]. However, the space-filling curve part of the algorithm is replaceable. As a matter of fact, the proposed minimization applies to an arbitrary set of (possibly disjointed) curves representing the tool path.

This paper presents new techniques to minimize \( \varepsilon \) by employing a one-dimensional adaptive grid. A one-dimensional grid is a structured set of points distributed along the desired curve in such a way that a certain optimization criterion is minimized or at least reduced. The grid can be treated as a discrete analogy of a coordinate transformation from the physical coordinate (for instance, the curve length) into so-called computational coordinates, where the grid points are equally spaced (see a variety definitions in [30]). In this sense the proposed method is similar to conventional grid generation. However, as opposed to the majority of grid generation problems occurring in mechanical engineering when only a numerical evaluation of the error is possible, our technique is based on a closed form representation of the kinematics error.

Fig. 3. (a) 2-0 machine and (b) the reference coordinate systems.
It should be noted that the preceding papers on grid generation for five-axis machining use either a numerically evaluated error or an assumption that the error is a monotone decreasing function of a certain parameter, such as the surface curvature [9,24,26]. A more general approach originally proposed in [7,23] includes the kinematics error as a weighting function for grid generation whereas excessive scallops are treated by a penalty function. However, these techniques have several major drawbacks. Chief among them is slow convergence for complex constraints and possible divergence when the kinematics error is evaluated using iterations “grid–error–grid” (a process when, first, a grid adapted to the error is constructed, then a new error is evaluated on that grid and a new grid is generated again). Besides, the approach requires an equal number of the cutter contact points along each track of the tool which is often impractical for the CNC machining.

Adaptive space-filling curves composed of zigzag patches based on [12] have been proposed by Anotaipaiboon and Makhanov [4]. A rectangular grid in the parametric space created by overlaying two zigzag tool paths is constructed so that the step between two adjacent machining strips overlap and the scallops are within the allowable margins. The evaluation step requires the surface curvature, the cutter shape and size, and the maximum allowable scallop height. The output of the procedure is a minimal tool inclination angle and a size of the maximum possible machining strip which defines offsets between parallel lines of the zigzag. As opposed to the tool paths based on the classic Hilbert curves [11,14,15,18] the adaptive space-filling curve multiplies only where necessary and does not involve many sharp turns.

However, the grid is not designed to minimize or to reduce the kinematics error along the tracks as in [24]. The error is ignored and the grid is generated with regard to the scallop height constraints. Furthermore, space-filling curve generation step treats the cells of the grid as nodes and connects them by arcs. Finally, the kinematics error is reduced in a post-processing step by bisection or a Newton–Raphson technique, inserting additional points along the resulting space-filling curve.

Curvilinear space-filling curves have been proposed in [6]. A curvilinear grid is adapted with the use of the harmonic functional [8,10,16,17] using the scallop height as the control function. If the scallop is large the grid moves into this area decreasing the scallops, so that the resulting tool path satisfies the scallop height constraints. The algorithm automatically evaluates the number of the required grid lines and the resulting grid is converted into a curvilinear space-filling curve. However, [6] does not present a method to place additional cutter location (CL) points along the resulting space-filling curves.

This paper presents and analyzes a grid generation method to assign the CL points along the resulting space-filling or zigzag curve. The method provides a considerably better rate of the decrease of the kinematics error with the increase of the number of the CL points as compared with the equi-parametric or equi-arc-length distribution of the points along the desired curve. The comparison is performed in terms of the $L_2$ and the Hausdorff distance.

In CNC programming it is often necessary to differentiate between the cutter location points (the tip of the tool) and the cutter contact points (CC points), which define a point of the actual contact between the tool and the desired surface. This is in particular important for the case of the so-called flat-end cutter. In this case, in order to ensure against the curvature interference, cutting is performed by the edge of the tool, whereas the CL point is often in the air. Our proposed procedure applies to the case of CC points as well. A closed form representation of the error is obtained from the inverse kinematics transformations corresponding to a particular five-axis machine. A code for automatic symbolic calculation to evaluate the error and its derivatives explicitly, given arbitrary machine kinematics is presented.

The advantage of the proposed techniques comes from representing the error as a function of coordinates of the inserted points along a desired curve on the surface. Theoretically the procedure applies to the entire surface, however, this could be computationally intractable task since the tool path may require tens or even hundreds of thousands of points.

Therefore, we define “control points” on the tool path and generate sub grids using a direct evaluation of the kinematics error along each segment. The control points can be constructed by a simple approach, for instance, by using turning points.

Unfortunately, the resulting equations are long and inconvenient to deal with. For example, the first derivative of the kinematics error may occupy about 100 text pages. Therefore, the proposed symbolic evaluation of the error followed by generation of the corresponding C or MATLAB functions using Maple-12 is indispensable.

The procedures for symbolic evaluation are general and apply to any configuration of the five-axis machine and to an arbitrary surface. The codes generated by the symbolic engine of Maple 10 are used for direct constraint minimization of the error, thus, generating an optimal or at least suboptimal grid.
2. Kinematics errors of the five-axis milling machine

Let \( R \equiv R[\text{parameters}] / [\text{arguments}] = R[\Re] / [M] \) be a kinematics transformation from the machine coordinates to the workpiece coordinates. For simplicity we denote the transformations by \( R[M] \) (when possible) keeping in mind the dependence on \( \Re \).

Let \( R^{-1}[W] \) be an inverse transformation such that \( \forall W, M, \Re, R^{-1}[R[M]] = M \) and \( R[R^{-1}[W]] = W \). Let \( \Pi_p \equiv (W_p, \Re_p), \Pi_{p+1} \equiv (W_{p+1}, \Re_{p+1}) \) be two successive coordinates of the tool path in \( \Re^5 \). \( W_p \) and \( W_{p+1} \) denote two successive spatial positions of the tool path, \( \Re_p \) and \( \Re_{p+1} \) the corresponding rotation angles. In order to calculate the tool trajectory between \( W_p \) and \( W_{p+1} \), we first invoke the inverse kinematics to transform the part-surface coordinates into the machine coordinates \( M_p \equiv (x_p, y_p, z_p) \) and \( M_{p+1} \equiv (x_{p+1}, y_{p+1}, z_{p+1}) \). Namely, \( M_p \equiv R^{-1} [\Re_p][W_p] \). Second, the rotation angles \( \Re \equiv \Re(t) = (a(t), b(t)) \) and the machine coordinates \( M = M(t) \equiv (x(t), y(t), z(t)) \) are assumed to change linearly between the prescribed points, namely, \( M(t) = t M_{p+1} + (1 - t) M_p \). \( \Re(t) = t \Re_{p+1} + (1 - t) \Re_p \), where \( t \) is the fictitious time coordinate (\( 0 \leq t \leq 1 \)). Finally, transforming \( M \) back to \( W \) for every \( t \) yields a trajectory of the tool tip in the workpiece coordinates given by

\[
W_{p+1}(t) = R[\Re(t)][M(t)] = R[\Re_p + (1 - t) \Re_p] [t M_{p+1} + (1 - t) M_p].
\]

(2)

In order to represent the tool path in terms of the workpiece coordinates, we eliminate \( M_p, M_{p+1} \) by using the inverse transformation \( M_p = R^{-1} [\Re_p][W_p] \). Substituting, \( M_p, M_{p+1} \) yields

\[
W_{p+1}(t) = R[\Re_p + (1 - t) \Re_p] [t R^{-1} [\Re_p + (1 - t) \Re_p][W_{p+1}] + (1 - t) R^{-1} [\Re_p][W_p]].
\]

(3)

Introduce the following coordinate systems: the workpiece coordinate system \( O_1 \), a coordinate system of the first rotary part \( O_2 \), a coordinate system of the second rotary part \( O_3 \) and a coordinate system of the spindle \( O_4 \). We shall call the first rotary axis the \( A \)-axis and the second rotary axis the \( B \)-axis. Next, we consider three important machine kinematics categorized by the positions of rotary axes at tool-workpiece frames of serial kinematics.

2.1. The 2–0 machine

Two rotary axes on the table (see Fig. 3). In this case

\[
M \equiv R^{-1} \equiv R^{-1} [\Re_p][W] = G_a B[b](A[a](W + T_{12}) + T_{23}) + T_{34} - T_4,
\]

\[
a = \begin{cases} \arctan\left( \frac{I_x}{T_c} \right) & \text{if } I_x > 0 \text{ and } I_y > 0, \\ \arctan\left( \frac{I_y}{T_c} \right) + \pi & \text{if } I_x < 0, \\ \arctan\left( \frac{I_x}{T_c} \right) + 2\pi & \text{otherwise,} \end{cases}
\]

(4)

\[
b = -\arcsin I_z,
\]

where \( G_a = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \) and \( T_4 = (0, 0, -L) \).

The angles should be in the limits specified by a particular machine. For instance, MAHO600E admits any \( a \), however, \( b \) must belong to \([-105^\circ, 105^\circ]\).

Furthermore, (4) is not a unique solution. First of all, there is a \( 2\pi \) invariance. Second, if \( a, b \) satisfy (4) then \((a - \pi, -b - \pi), (a + \pi, -b - \pi)\) are also solutions.
The fourth non-periodic solution is given by \( (a - 2\pi, b), \text{ if } a > 0, \)
\( (a + 2\pi, b), \text{ otherwise} \).

Further details and several versions of the shortest path optimization with regard to the multiple solutions above are given in [26,27]. Similar multiple solutions can be derived for the forthcoming configurations 1–1 and 0–2.

Finally, note, a singularity \( I_x/I_y \) at stationary points characterized by \( I_x = I_y = 0 \). In this case (4) is not applicable. Angle \( a \) can be defined and found using interpolation in a neighborhood of the stationary point. Alternatively, the tool path can be modified so that it avoids the stationary positions [1].

2.2. The 1–1 machine

One rotary axis on the table and one on the tool (see Fig. 4). In this case
\[
M \equiv G_a A[a](W + T_{12}) + T_{23} + B^{-1}[b](T_{34} - T_4)
\]
\[
a = \begin{cases} 
\arctan \left( \frac{I_y}{I_x} \right) & \text{if } I_x > 0 \text{ and } I_y < 0, \\
\arctan \left( \frac{I_y}{I_x} \right) + \pi & \text{if } I_x < 0, \\
-\arctan \left( \frac{I_y}{I_x} \right) + 2\pi & \text{otherwise},
\end{cases}
\]
\[
b = \arccos I_z,
\]
where \( G_a = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \) and \( T_4 = (0, 0, L) \).

2.3. The 0–2 machine

Two rotary axes on the tool (see Fig. 5). In this case
\[
M \equiv G_a W + T_{12} + A^{-1}[a](T_{23} + B^{-1}[b](T_{34} - T_4)),
\]
\[
a = \begin{cases} 
\arctan \left( \frac{I_y}{I_z} \right) & \text{if } I_y < 0 \text{ and } I_z > 0, \\
\arctan \left( \frac{I_y}{I_z} \right) + \pi & \text{if } I_z < 0, \\
\arctan \left( \frac{I_y}{I_z} \right) + 2\pi & \text{otherwise},
\end{cases}
\]
\[
b = -\arccos I_x,
\]
where \( G_a = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{pmatrix} \) and where \( T_4 = (0, 0, L) \).

3. Kinematics errors and measures

The tool path is defined by a sequence of cutter contact points and orientation vectors. The positions of translation and rotary axes are linearly interpolated. Due to the prescribed transformations, the machine tool tip follows curves which represent a continuous piecewise-smooth non-linear 3D interpolation of the desired trajectories. The measure of a deviation of the interpolating function from the actual trajectory is called the kinematics error. A number of
Let \( W^D_{p,p+1}(t) = (x^D_{p,p+1}(t), y^D_{p,p+1}(t), z^D_{p,p+1}(t)) \) be a curve between two tool positions \( W_p \) and \( W_{p+1} \) extracted from the machined surface \( S(u,v) \), where \( t \) is a parametric coordinate along the curve. The curve is extracted in such a way that it represents the desired tool trajectory. The kinematics error is defined as a distance between the desired trajectories \( W^D_{p,p+1}(t) \) and the actual trajectories \( W_{p,p+1}(t) = (x_{p,p+1}(t), y_{p,p+1}(t), z_{p,p+1}(t)) \) generated by the machine kinematics, namely,

\[
\varepsilon = \sum_p \text{dist}(W^D_{p,p+1}, W_{p,p+1}),
\]

where \( \text{dist}(A, B) \) denotes an appropriate distance between space curves \( A(t) \) and \( B(t) \).

The difference between the space curves can be evaluated by the generic Hausdorff or Fréchet distance. However, these measures are computationally expensive and may lead to intractable distance optimization problems.

Some computationally simple choices are the root-mean-square (rms) given by

\[
\text{dist}_2(A, B) = \| |A(t) - B(t)|_E\|_2
\]

and the max distance given by \( \text{dist}_\infty(A, B) = \| |A(t) - B(t)|_E\|_\infty \), where \( |E|_\infty \) is the Euclidian distance and \( A(t) \) and \( B(t) \) are parameterized with regard to \( t \in [0, 1] \).

Although these distances depend on parameterization, they often produce good results when the actual trajectory is parameterized with regard to the fictitious time \( t \) (see Section 4) and the surface curve with regard to a line segment.
between \((u_p, v_p)\) and \((u_{p+1}, v_{p+1})\) in the parametric plane \((u, v)\). This is because when the number of inserted points is large enough the compared curves are similar, arc-like segments.

The parameterization-invariant Hausdorff distance is given by

\[
dist_H(A, B) = \max_a \min_b |a - b|_E.
\]

Another option is the Fréchet distance defined by

\[
dist_F(A, B) = \min_{t \in [0, 1]} |A(t) - B(t)|_E.
\]

where minimum is considered over all continuous and increasing functions \(\alpha(t)\) and \(\beta(t)\).

However, \(dist_H\) and \(dist_F\) are computationally expensive.

Note that \(dist_H(A, B) \leq dist_\infty(A, B)\). Therefore, minimization with regard to \(dist_\infty\) may reduce the error measured by \(dist_H\). Recall summed Hausdorff metrics (such as the Lindstrom-Turk’s mean geometric distance \(dist_{\text{HS}}\)) obtained by replacing “max” in (8) by summation or integration \([22]\). Since \(dist_{\text{HS}}(A, B) \leq dist_2(A, B)\) the minimization with regard to \(dist_2\) may reduce \(dist_{\text{HS}}\), but, of course, it does not guarantee a minimum in \(dist_{\text{HS}}\).

Finally, the compared curves can be approximated by piecewise linear functions. In this case the Fréchet distance (which is essentially the minimum equal-parameter distance between \(A\) and all possible reparametrizations of \(B\)) can be evaluated explicitly \([2]\). However, the algorithm is computationally expensive and may lead to hard optimization problems.

A good option is an rms-distance based on a natural parameterization given by

\[
dist_N(A, B) = \sqrt{\int_0^1 |A(l_A(t)) - B(l_B(t))|^2 \, dt},
\]

where \(A(l_A(t))\) and \(B(l_B(t))\) denote the corresponding arc-length parameterizations.

Unfortunately, it not possible to find a closed form parameterization for real rational curves (such as NURBS) by rational functions of its arc length \([29]\). Therefore, such parameterizations are evaluated numerically.

Finally, finding a distance between curve \(W_{p, p+1}(t)\) and the entire part surface \(S(u, v)\) (rather than between \(W_{p, p+1}(t)\) and \(W_{p, p+1}(t)\)) is certainly a better option than evaluation of the distance between the two curves. However, \(W_{p, p+1}(t)\) is unknown. Therefore, this computationally hard option is often impractical.

Although, finding the best distance for this application lies out of the scope of this paper our rule of thumb is that if the number of inserted points is large enough, a simple \(dist_2\) or \(dist_N\) produce approximately the same results as parameterization-invariant metrics.

Alternatively, \(dist_N(A, B)\) or \(dist_2(A, B)\) can be used for the first several iterations after which the optimization is performed with regard to the Hausdorff distance.

Finally, the tool trajectory is compared with the desired trajectory which is in some way extracted from the machined part. In engineering practice the parts are defined by standard formats such STEP, IGES, etc. For instance, the IGES represents curvilinear NURBS faces glued together along the boundary edges. Therefore, the method of extracting the trajectory should include the case of the multi-patch surfaces when the curve crosses the boundary or even several boundaries.

4. Optimization

Consider minimization of kinematics error (7):

\[
\begin{align*}
\text{minimize} & \quad \epsilon, \\
\text{subject to} & \quad \Pi \in \Pi',
\end{align*}
\]

where \(\Pi'\) is a set of curves belonging to the surface and \(\Pi\) is a set of CC points. The curves pass through a certain number of fixed points such as the turning points at the boundary of the zigzag pattern.
The optimization procedure inserts a certain number of CC points between each pair of the fixed points in such a way that the kinematics error is minimized. Let \((u_p, v_p)\) and \((u_{p+1}, v_{p+1})\) be parametric coordinates corresponding to control points \((x_p, y_p, z_p)\) and \((x_{p+1}, y_{p+1}, z_{p+1})\) respectively. Denote the desired curve by \(W^D_{p,p+1}(t)\), for instance, \(W^D_{p,p+1}(t) = S((1 - t)u_p + tu_{p+1}, (1 - t)v_p + tv_{p+1}), 0 \leq t \leq 1\).

Furthermore, points \(s_0, s_1, \ldots s_m\) are inserted between \(t = 0\) and \(t = 1\) and the corresponding trajectories are denoted by \(W^D_{p,p+1}(s_k, s_{k+1}, t)\).

Let \(s_0 = 0, s_m = 1, s_0 < s_1 < \cdots < s_{m-1} < s_m\).

We will obtain these points by solving the following minimization problem:

\[
\operatorname{argmin}(\varepsilon_{p,p+1}(s_0, s_1, \ldots s_m)) = \operatorname{argmin}\left(\sum_{k \in [s_k, s_{k+1}]} \text{dist}(W^D_{p,p+1}(s_k, s_{k+1}, t), W^D_{p,p+1}(s_{k+1}, s_{k+1}, t))\right),
\]

where \(\text{dist}\) is an appropriate distance (see discussion in Section 3).

4.1. Symbolic evaluation

For simplicity, consider inserting only one point. Omitting index \(p, p + 1\) and substituting \(s_0, s_1, s_2\) into (11), yields

\[
\varepsilon(s_1) = \text{dist}(W^D(0, s_1, t), W(0, s_1, t)) + \text{dist}(W^D(s_1, 1, t), W(s_1, 1, t))
\]

Note that \(W^D(0, s_1, t)\) and \(W^D(s_1, 1, t)\) can be easily evaluated, however, \(W(0, s_1, t)\) and \(W(s_1, 1, t)\) invoke inverse kinematics (3)–(6).

Therefore, the following symbolic calculations are performed:

1. The normal to the surface is evaluated as a function of the derivatives of \(S(u, v)\).
2. The rotation angles in (4)–(6) are evaluated using normal \(n(u, v) = (I_x(u, v), I_y(u, v), I_z(u, v))\) represented parametrically along the curves corresponding to \([0, s_1]\) and \([s_1, 1]\). Furthermore, \(I_x, I_y, I_z\) are evaluated symbolically at \(t = s_1\) and used to parameterize the rotation angles as follows:

\[
a(0, s_1, t) = a(0) \left(1 - \frac{t}{s_1}\right) + a(s_1) \frac{t}{s_1}, \quad b(0, s_1, t) = b(0) \left(1 - \frac{t}{s_1}\right) + b(s_1) \frac{t}{s_1},
\]

\[
a(s_1, 1, t) = a(s_1) \left(1 - \frac{t - s_1}{1 - s_1}\right) + a(1) \frac{t - s_1}{1 - s_1}, \quad b(s_1, 1, t) = b(s_1) \left(1 - \frac{t - s_1}{1 - s_1}\right) + b(1) \frac{t - s_1}{1 - s_1}.
\]

Note that the tool orientation is not necessarily along the normal. The tool can be inclined. In that case the tool vector is evaluated using the normal vector and the inclination angles (see Section 4.2).

3. The tool trajectory in the machine coordinates is generated using (2) (Section 2):

\[
M(0, s_1, t) = M(0) \left(1 - \frac{t}{s_1}\right) + M(s_1) \frac{t}{s_1}, \quad M(s_1, 1, t) = M(s_1) \left(1 - \frac{t - s_1}{1 - s_1}\right) + M(1) \frac{t - s_1}{1 - s_1}.
\]

4. Next, we apply the kinematics transformation \(\mathfrak{R}\) as follows \(W^D(0, s_1, t) = \mathfrak{R}[M(0, s_1, t)], W^D(s_1, 1, t) = \mathfrak{R}[M(s_1, 1, t)]\) (see Section 2). Note that the transformation includes two rotation matrices employing rotation angles (13).

5. The resulting \(W^D(0, s_1, t)\) and \(W^D(s_1, 1, t)\) are substituted into (12) and the error is evaluated symbolically using Maple 12. In the case of integral norms such as \(\text{dist}_2\) or \(\text{dist}_N\) we use a quadrature such as

\[
\int_0^{s_1} |W^D(0, s_1, t) - W(0, s_1, t)|^2 dt \approx \sum_i |W^D(0, s_1, t_i) - W(0, s_1, t_i)|^2 \Delta t.
\]
It should be noted that the symbolic result may be very lengthy. If the minimization procedure needs the derivatives, it further increases the size of the required formulas. Finally, the actual minimization may include hundreds of points \( s_1, \ldots, s_{m-1} \) being inserted between the fixed points. This also contributes to the complexity of the symbolic result.

In order to avoid such lengthy results we use the symbolic engine of Maple 12 capable of generating a C or MATLAB code given symbolic expressions as the input.

Consider a vector function \( \text{ErrPath}(t,s_0,s_1,up,vp,upp_1,vpp_1) \) designed to generate the tool trajectory on the curve between \((up,vp)\) and \((upp_1,vpp_1)\) parameterized with regard to \( t \) between unknown points \( t = s_0 \) and \( t = s_1 \) (the bold face is used here for the pseudo-codes, program variables and functions):

\[
\text{ErrPath}(t,s_0,s_1,up,vp,upp_1,vpp_1) =
\{ \text{us}_0 := \text{linterp}(up,upp_1,s_0); \\
\text{vs}_0 := \text{linterp}(vp,vpp_1,s_0); \\
\text{us}_1 := \text{linterp}(up,upp_1,s_1); \\
\text{vs}_1 := \text{linterp}(vp,vpp_1,s_1); \\
\} // evaluates the workpiece coordinates and the rotation angles at W0 and W1
\]

\[
W_0 := \text{S}(\text{us}_0,\text{vs}_0); \\
\text{a}_0 := \text{UV2a}(\text{us}_0,\text{vs}_0); \\
\text{b}_0 := \text{UV2b}(\text{us}_0,\text{vs}_0); \\
W_1 := \text{S}(\text{us}_1,\text{vs}_1); \\
\text{a}_1 := \text{UV2a}(\text{us}_1,\text{vs}_1); \\
\text{b}_1 := \text{UV2b}(\text{us}_1,\text{vs}_1); // the corresponding machine coordinates M0 and M1
\]

\[
M_0 := \text{W2M}(W_0,a_0,b_0); \\
M_1 := \text{W2M}(W_1,a_1,b_1); \\
\text{return}\ \text{ErrPath2}(t,s_0,s_1,M_0,M_1); \}
\]

where \( \text{ErrPath2}(t,s_0,s_1,M_0,M_1) \) interpolates linearly and converts into the workpiece coordinates as follows:

\[
\text{ErrPath2}(t,s_0,s_1,M_0,M_1) =
\{ \text{tp} := (t-s_0)/(s_1-s_0); \\
\text{Mt} := \text{linterp}(M_0,M_1,tp); \\
\text{at} := \text{linterp}(a_0,a_1,tp); \\
\text{bt} := \text{linterp}(b_0,b_1,tp); \\
\text{Wt} := \text{M2W}(\text{Mt},\text{at},\text{bt}); \\
\} \]

\( S(u,v) \) denotes the required surface. \( \text{UV2a}(u,v) \) and \( \text{UV2a}(u,v) \) are functions to compute the required rotations angles corresponding to the normal vector at the point \( S(u,v) \). \( \text{W2M}(W,a,b) \) and \( \text{M2W}(M,a,b) \) convert points from the workpiece to machine coordinates and vise versa.

The derivatives

\[
\frac{\partial \text{ErrPath}(t, s_0, s_1, up, vp, upp_1, vpp_1)}{\partial s_0}, \frac{\partial \text{ErrPath}(t, s_0, s_1, up, vp, upp_1, vpp_1)}{\partial s_1}
\]

are evaluated using the chain rule as follows:

\[
\frac{\partial \text{ErrPath}}{\partial s_0} = \sum_{M_i \in \{M_0x_0,M_0y_0,M_0z_0\}} \sum_{W_j \in \{W_0x,W_0y,W_0z,a_0,b_0\}} \sum_{U_k \in \{us_0,vs_0\}} \frac{\partial \text{ErrPath2}}{\partial M_i} \frac{\partial M_i}{\partial W_j} \frac{\partial W_j}{\partial U_k} \frac{\partial U_k}{\partial s_0} + \frac{\partial \text{ErrPath}}{\partial s_0},
\]

\[
\frac{\partial \text{ErrPath}}{\partial s_1} = \sum_{M_i \in \{M_1x_1,M_1y_1,M_1z_1\}} \sum_{W_j \in \{W_1x,W_1y,W_1z,a_1,b_1\}} \sum_{U_k \in \{us_1,vs_1\}} \frac{\partial \text{ErrPath2}}{\partial M_i} \frac{\partial M_i}{\partial W_j} \frac{\partial W_j}{\partial U_k} \frac{\partial U_k}{\partial s_1} + \frac{\partial \text{ErrPath}}{\partial s_1}.
\]

Note that \( \text{ErrPath2} \) is a symbolically evaluated tool trajectory represented in terms of \( t, s_0, s_1, M_0, \) and \( M_1. \) \( M \) is represented as a function of \( W, a \) and \( b. \) In turn, \( W, a \) and \( b \) are functions of \( ut \) and \( vt. \) Finally, \( ut \) and \( vt \) are obtained by interpolating linearly between \( up, upp_1 \) and \( vp, vpp_1. \) Using the chain rule above, the symbolic derivatives can be reduced to a few pages. The symbolic results apply to an arbitrary surface and to a particular machine. The user simply writes functions to evaluate \( S(u,v) \) and its derivatives. The derivatives of the desired curve \( WD(s_k, s_{k+1}, t) \) are computed using the chain rule with \( WD \) expressed in terms of \( ut \) and \( vt. \)

In case of a non-differentiable objective function such as the one based on the Hausdorff distance, a non-gradient minimization procedure must be used. However, when the number of inserted points is large it may significantly increase the computational time. In order to combine the advantages of the gradient methods with the theoretical
rigor of the parameterization-invariant distance the following heuristic algorithm can be used. First, a gradient based minimization is performed with regard to a differentiable norm such as dist$_2$. After a certain numbers of steps the procedure switches to the generic Hausdorff distance.

A further simplification is based on the following heuristic approximation. A pair of points $t_{k,1}$ and $t_{k,2}$ corresponding to the Hausdorff max–min between the discretized curves $W^D(s_k, s_{k+1}, t)$ and $W(s_k, s_{k+1}, t)$ is fixed for several iterations. This generates an auxiliary objective function given by:

$$\sum_k |W^D(s_k, s_{k+1}, t_{k,1}) - W(s_k, s_{k+1}, t_{k,2})|.$$  

(14)

After several steps the procedure switches back to the generic Hausdorff distance and a new set of the Hausdorff pairs is selected in (14).

4.2. Inclination of the tool

The above algorithm applies only to the so-called ball-nose tool (see Fig. 6). Such a tool is oriented along the normal to the surface or inclined with regard to the normal by a certain recommended fixed angle (usually about 5°). This is merely done to prevent the radial velocity of the rotating tool to be equal to zero at the center of the cutting area. However, when the radius of the tool is larger than the radius of the curvature in the direction perpendicular to the cutting curve, the tool radius should be adjusted. In other words the tool must be replaced by a smaller one. As opposed to that, a flat-end tool (Fig. 7(a)) can be inclined dynamically in order to avoid the curvature interference [28,31] as shown in Fig. 7(b) and (c).

In this case the inclination angle $\lambda$ first must be represented as a continuous function along the desired curve. Next, the components of the tool vector $T$ must be modified at $s_0, s_1, ..., s_m$. Once the tool vector has been modified, steps 2–5 of our symbolic evaluation algorithm are performed similarly to the ball-nose case. Note, that actually the tool orientation is controlled by the inclination angle and a yaw angle $w$ (Fig. 8(b)).
Fig. 8. Comparison of the point insertion methods. Example 1. (a) Part surface, (b) inserted points (optimal), (c) tool trajectories (optimal points), (d) arc-length equally spaced points, (e) tool trajectory arc-length equally spaced points, (f) virtual cutting (optimal), (g) virtual cutting arc-length equally spaced points, (h) real cutting (optimal) and (i) real cutting (equally spaced points).
However, this paper considers \( w = 0 \). In this case a minimal inclination angle to avoid the curvature interference is given by [3]:

\[
\lambda = \arcsin(Rk_{\text{max}}),
\]

where \( R \) is the tool radius and \( k_{\text{max}} \) the maximum curvature at a cutter contact point. Let us parameterize \( k_{\text{max}} \) along the desired curve \( W^D \). This is done by finding \( k_{\text{max}}(u, v) \) for the entire surface using the relationship

\[
k_{\text{max}} = H + \sqrt{H^2 - K},
\]

where \( K = (eg - f^2)/(EG - F^2) \) and \( H = (eG - 2fF + gE)/(2(EG - F^2)) \) are the Gaussian and the mean curvatures respectively and

The subscripts \( u, v \) denote partial derivatives. The inclination angle along the prescribed curve \( W^D_{p,p+1}(t) \) is given by

\[
\lambda(t) = \arcsin(Rk_{\text{max}}((1-t)u_p + tu_{p+1}, (1-t)v_p + tv_{p+1})).
\]

The tool vector \( T(t) \) is obtained by rotating the normal \( n(t) \) towards \( r(t) \) (the feed direction), where \( r(t) = (dW^D_{p,p+1}(t)/dt)/|dW^D_{p,p+1}(t)/dt| \) is the tangent to \( W^D_{p,p+1}(t) \). Using the corresponding transformation matrices yields \( T(t) = \cos(\lambda)n(t) + \sin(\lambda)r(t) \) (see Fig. 8(b)). The above manipulations are performed symbolically given the external functions to evaluate \( S(u, v) \) and its derivatives up to the second order. Finally, step 2 (Section 4.1) employs rotation
Table 1a
Optimal insertion vs. equally arc-length spaced points. Example 1.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Number of inserted points</th>
<th>Optimal insertion (mm)</th>
<th>Equi-parameter (mm)</th>
<th>Reduction percentage</th>
<th>Equi-arc length (mm)</th>
<th>Reduction percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1 mm</td>
<td>7590</td>
<td>0.0423</td>
<td>0.1001</td>
<td>57.74</td>
<td>0.0998</td>
<td>57.62</td>
</tr>
<tr>
<td>0.5 mm</td>
<td>3083</td>
<td>0.2143</td>
<td>0.3732</td>
<td>42.58</td>
<td>0.3717</td>
<td>42.35</td>
</tr>
<tr>
<td>1.0 mm</td>
<td>2101</td>
<td>0.4103</td>
<td>0.6459</td>
<td>36.48</td>
<td>0.6438</td>
<td>36.27</td>
</tr>
</tbody>
</table>

...angles in (4)–(6) evaluated using the components of \( T(t) \) instead of the components of \( n(t) \). Below is a pseudo-code to compute the required rotation angles:

\[
\text{UV2a}(u,v) \\
\begin{align*}
N &: \text{CrossProduct}(Su,Sv); \\
N &: N / \text{Norm}(N); \\
kmax &: \text{ComputeKmax}(Su,Sv,Suu,Svv,Su); \\
lambda &: \text{asin}(R*kmax); \\
R &: Su*du/dt + Sv*dv/dt; \\
R &: R / \text{Norm}(R); \\
I &: \cos(\lambda) * N + \sin(\lambda) * R; \\
\text{return } I2a(I); 
\end{align*}
\]

The symbolic expressions are converted into C or MATLAB codes. The subsequent minimization presented in the forthcoming section is performed using the MATLAB optimization toolbox.

5. Numerical examples and cutting experiments

In this section the proposed optimization techniques are compared with equi-parametric and equi-arc-length grids for two convex–concave surfaces. The ball-nose and the flat-end tool have been examined for the zigzag and the space-filling tool paths. The numerical experiments verified by virtual and real cutting demonstrate that the proposed procedure consistently outperforms the equi-parametric and equi-arc-length techniques. In order to show the advantage of our approach we also compare our results with a gradient-free bisection method which generates a sequence of nested grids. The proposed procedure is applied with objective functions based on \( \text{dist}_2 \) (the root-mean-square), the Hausdorff distance, or a combination of rms and the Hausdorff distance based on heuristic evaluation (14).

5.1. Example 1. Zigzag tool path. The ball-nose tool

Consider a surface depicted in Fig. 8(a). The surface is produced by a ball-nose tool with the radius 5 mm using a zigzag tool path. Control points are specified at the turns and the optimization algorithm is applied to construct grids between them. Fig. 8(b) and (c) show optimally inserted points and the resulting tool tip trajectories obtained by the proposed optimization algorithm. The points are inserted so that the maximum error (dist\(_H\)) is less than a prescribed tolerance 1 mm.

Next, the same number of points is used to generate equally spaced grid in the parametric domain and an equally arc-length spaced grid (Fig. 8(d)). Poor surface quality obtained with the equal arc-length spaced grid is shown in Fig. 8(e). The virtual and the real cuts shown in Fig. 8(f)–(i) respectively validate the numerical results. Note that the error nearby the center of symmetry of the surface (Fig. 8(e)) is not clearly seen on the virtual and real cuts. This is because the trajectories are above the workpiece. The resulting scallops are eliminated by the cutting along the nearby tracks.

Table 1a compares the optimal and the equi-arc-length grids. The cost function is based on the rms norm but the error is measured in terms of the Hausdorff distance. The proposed optimization leads to a significant accuracy increase with the reference to the equally spaced and equally arc-spaced grids (36–57%). The comparison with the parametrically and arc-length equally spaced grids shows slightly different results.
Table 1b
Optimal point insertion method vs. the bisection. Example 1.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Number of inserted points</th>
<th>Reduction percentage</th>
<th>Max dist&lt;sub&gt;H&lt;/sub&gt; (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bisection</td>
<td>Optimal insertion</td>
<td></td>
</tr>
<tr>
<td>0.1 mm</td>
<td>7951</td>
<td>7950</td>
<td>4.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.0989</td>
</tr>
<tr>
<td>0.5 mm</td>
<td>3453</td>
<td>3083</td>
<td>10.72</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.4948</td>
</tr>
<tr>
<td>1.0 mm</td>
<td>2397</td>
<td>2101</td>
<td>12.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.9928</td>
</tr>
</tbody>
</table>

Table 1c
Optimal insertion with different norms. Example 1.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Number of inserted points</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bisection</td>
</tr>
<tr>
<td></td>
<td>rms</td>
</tr>
<tr>
<td>0.1 mm</td>
<td>7951</td>
</tr>
<tr>
<td>0.5 mm</td>
<td>3453</td>
</tr>
<tr>
<td>1.0 mm</td>
<td>2397</td>
</tr>
</tbody>
</table>

Interestingly enough, in this range the accuracy improvement does not decrease as the number of points increases. This is because even though the cutter location points get closer, the corresponding trajectories may still produce larger error. Of course, eventually, the errors on the equally spaced and the optimal grids converge.

Table 1b compares the grid with an insertion based on bisection. The bisection inserts an additional mid point between each two cutter location points if the maximum kinematics error (dist<sub>H</sub>) along the corresponding trajectory exceeds a specified tolerance.

The advantage with regard to the bisection leads to a less impressive but still valuable 4–12% decrease of the number of required points. Note that cutting complex industrial parts may employ tens, hundreds of thousands and even millions of points. Therefore, the 12% improvement is still significant and may amount to long hours of machining if the machining time at programmed feed is less than the sampling time of the controller.

Finally, Table 1c compares the optimal point insertion method applied with different metrics. The advantage of the proposed global minimization method increases when the objective function is based on the Hausdorff distance. Interestingly enough the combination rms with the Hausdorff distance leads to slightly better results. Most likely this is because the Hausdorff pairs (14) simplify the objective function which eventually leads to better convergence.

### 5.2. Example 2. Zigzag tool path. Flat-end cutter

The proposed optimization technique is applied to a zigzag tool path designed for the surface in Fig. 9(a) manufactured with a 10 mm flat-end mill. The tool trajectories for the optimal and the equally arc-length spaced insertion methods are shown in Fig. 9. Clearly, the loop-like trajectories on the top of the surface (b) lead to large cutting errors and consequently to poor surface quality. The numerical comparison of the errors is shown in Tables 2a–2c.

Table 2a
Optimal point insertion method vs. bisection. Example 2.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Number of inserted points</th>
<th>Reduction percentage</th>
<th>Actual dist&lt;sub&gt;H&lt;/sub&gt; (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bisection</td>
<td>Optimal insertion</td>
<td></td>
</tr>
<tr>
<td>0.1 mm</td>
<td>2273</td>
<td>2238</td>
<td>1.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.1000</td>
</tr>
<tr>
<td>0.5 mm</td>
<td>971</td>
<td>925</td>
<td>4.74</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.4998</td>
</tr>
<tr>
<td>1.0 mm</td>
<td>687</td>
<td>634</td>
<td>7.71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.9987</td>
</tr>
</tbody>
</table>
5.3. Example 3. Space-filling tool path. Flat-end cutter

A space-filling curve (SFC) tool path is constructed to manufacture surface in Fig. 9(a) employing a 10 mm flat-end mill. The example validates our symbolic evaluation for the flat-end cutter geometry given in Section 5.2. The resulting tool trajectories for optimal insertion and equal distribution methods are shown in Fig. 10. The errors are compared in Tables 3a–3c.

5.4. Example 4. A two-bell surface. Space-filling path with a flat-end cutter

An SFC tool path is constructed for a two-bell surface shown in Fig. 10(a) for a cut employing a 10 mm flat-end mill. The resulting tool trajectories constructed for optimal insertion and equal distribution methods are shown in Fig. 11. The errors are compared in Tables 4a–4c.
Fig. 10. Comparison of the point insertion methods. Example 3. (a) Tool trajectories, CC points, optimal algorithm, (b) tool trajectories, CC points, equal distribution, (c) real cutting, optimal algorithm and (d) real cutting, equal arc-length method.

<table>
<thead>
<tr>
<th>Threshold dist_{th} (mm)</th>
<th>Number of inserted points</th>
<th>Reduction percentage</th>
<th>Actual dist_{th} (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bisection</td>
<td>Optimal insertion</td>
<td></td>
</tr>
<tr>
<td>0.1 mm</td>
<td>3079</td>
<td>2981</td>
<td>3.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.1000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.0999</td>
</tr>
<tr>
<td>0.5 mm</td>
<td>1322</td>
<td>1230</td>
<td>6.96</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.4997</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.4981</td>
</tr>
<tr>
<td>1.0 mm</td>
<td>876</td>
<td>750</td>
<td>14.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.9980</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.9980</td>
</tr>
</tbody>
</table>

Table 3b
Optimal point insertion vs. equally spaced points. Example 3.

<table>
<thead>
<tr>
<th>Threshold dist_{th} (mm)</th>
<th>Number of inserted points</th>
<th>rms error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Optimal insertion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Equi-parameter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reduction percentage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Equi-arc length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reduction percentage</td>
</tr>
<tr>
<td>0.1 mm</td>
<td>2981</td>
<td>0.0502</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5599</td>
</tr>
<tr>
<td></td>
<td></td>
<td>91.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5834</td>
</tr>
<tr>
<td></td>
<td></td>
<td>91.40</td>
</tr>
<tr>
<td>0.5 mm</td>
<td>1230</td>
<td>0.2297</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.7952</td>
</tr>
<tr>
<td></td>
<td></td>
<td>71.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.8418</td>
</tr>
<tr>
<td></td>
<td></td>
<td>72.71</td>
</tr>
<tr>
<td>1.0 mm</td>
<td>750</td>
<td>0.4632</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.2015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>61.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.1769</td>
</tr>
<tr>
<td></td>
<td></td>
<td>60.64</td>
</tr>
</tbody>
</table>
Table 3c
Optimal point insertion method with different norms. Example 3.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Number of inserted points</th>
<th>Bisection</th>
<th>Optimal insertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>distH</td>
<td></td>
<td>rms</td>
<td>Reduction percentage</td>
</tr>
<tr>
<td>0.1 mm</td>
<td>3079</td>
<td>2981</td>
<td>3.18</td>
</tr>
<tr>
<td>0.5 mm</td>
<td>1322</td>
<td>1230</td>
<td>6.96</td>
</tr>
<tr>
<td>1.0 mm</td>
<td>876</td>
<td>750</td>
<td>14.38</td>
</tr>
</tbody>
</table>

Fig. 11. Comparison of the point insertion methods. Example 4. (a) Tool trajectories, CC points, optimal algorithm, (b) tool trajectories, CC points, equal distribution, (c) real cutting, optimal algorithm and (d) real cutting, equal distribution.

Table 4a
Comparison of optimal point insertion method with bisection for Example 4.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Number of inserted points</th>
<th>Reduction percentage</th>
<th>Actual distH (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>distH</td>
<td>Bisection</td>
<td>Optimal insertion</td>
<td>Bisection</td>
</tr>
<tr>
<td>0.1 mm</td>
<td>2398</td>
<td>2238</td>
<td>6.67</td>
</tr>
<tr>
<td>0.5 mm</td>
<td>941</td>
<td>819</td>
<td>12.96</td>
</tr>
<tr>
<td>1.0 mm</td>
<td>567</td>
<td>475</td>
<td>16.23</td>
</tr>
</tbody>
</table>

Table 4b
Comparison of optimal point insertion method with equal arc-spaced tool path for Example 4.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Number of inserted points</th>
<th>rms error</th>
</tr>
</thead>
<tbody>
<tr>
<td>distH</td>
<td>Optimal insertion</td>
<td>Equi-parameter</td>
</tr>
<tr>
<td>0.1 mm</td>
<td>2238</td>
<td>0.0687</td>
</tr>
<tr>
<td>0.5 mm</td>
<td>819</td>
<td>0.3539</td>
</tr>
<tr>
<td>1.0 mm</td>
<td>475</td>
<td>0.6589</td>
</tr>
</tbody>
</table>
Table 4c
Comparison of optimal point insertion method with different norms. Example 4.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Number of inserted points</th>
</tr>
</thead>
<tbody>
<tr>
<td>distH</td>
<td>Bisection Minimization</td>
</tr>
<tr>
<td></td>
<td>rms Reduction percentage</td>
</tr>
<tr>
<td></td>
<td>Hausdorff Reduction percentage</td>
</tr>
<tr>
<td></td>
<td>rms + Hausdorff pair Reduction percentage</td>
</tr>
<tr>
<td>0.1 mm</td>
<td>2398 2238 6.67 2189 8.72 2134 11.01</td>
</tr>
<tr>
<td>0.5 mm</td>
<td>941 819 12.96 797 15.30 794 15.62</td>
</tr>
<tr>
<td>1.0 mm</td>
<td>567 475 16.23 451 20.46 453 20.11</td>
</tr>
</tbody>
</table>

The flat-end tool cutting combined with the optimal insertion method provides a good accuracy increase reaching 73% and an relatively small but still valuable decrease in terms of the required cutter location points with the maximum advantage of about 20%.

6. Conclusions

A new method for generation of optimal grids of cutter location points for five-axis machining has been presented and verified. The method is based on a direct evaluation of the kinematics error by the symbolic engine of Maple-12 and build-in minimization procedures of MATLAB. The symbolic calculations generate an explicit formula for the error and its derivatives (if necessary). The evaluations are performed automatically and only once for a particular five-axis machine. The results are converted into C-codes and can be used to optimize the tool path for any given surface.

For particular cuts the method provides up to 89% decrease of the kinematics error with regard to iso-parametric and equi-arc-length point method. On the other hand, the method provides less CL points for the same error tolerance, which is in particular important for high speed milling when an increase in number of the points leads to a substantial increase in the machining time. The advantage with regard to the non-gradient technique such as bisection exceeds 10%. For many industrial setups it amounts to hundreds of hours of machining.

Finally, the procedure can be coupled with optimization schemes based on space-filling curve technologies serving as an efficient tool for rough machining in the five-axis mode.

Acknowledgement

We acknowledge a sponsorship of Thailand Research Fund, grant number MRG5380227.

References