PERGAMON

Computers & Industrial Engineering 43 (2002) 455–472

computers & industrial engineering

www.elsevier.com/locate/dsw

On the tool-path optimization of a milling robot

S.S. Makhanov[a,*], D. Batanov[b], E. Bohez[b], K. Sonthipaumpoon[c], W. Anotaipaiboon[a], M. Tabucanon[d]

[a]*Department of Information Technology, Sirindhorn International Institute of Technology Tammasat University, Pathum Thani 12121, Thailand*

[b]*School of Advanced Technologies, Asian Institute of Technology, Pathum Thani 12120, Thailand*

[c]*Industrial Engineering Department, Naresuan University, Phitsanulok 65000, Thailand*

[d]*Industrial Systems Engineering Program, School of Advanced Technologies, Asian Institute of Technology, Pathum Thani 12120, Thailand*

Accepted 20 March 2002

Abstract

We present a new approach to tool-path optimization of milling robots based on a global interpolation of the required surface by a virtual surface composed from tool trajectories. The procedure combines inverse kinematics techniques and a variational gridding method endowed with constraints related to the required scallop height. We demonstrate the capability of the proposed technique to generate a tool-path for workpieces with complex geometries comprising 'islands' or boundaries with sharp edges requiring a combined spiral–zigzag pattern.

Our technique provides a significant increase in the accuracy of milling. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Milling machine; Tool-path optimization; Grid generation

1. Introduction

Innovations in the field of mechanical engineering have enhanced the involvement of milling robots in various manufacturing processes. Nowadays, milling machines, guided by a computer system, are employed to produce free-shape surfaces to supply the need of mass manufacturing plants, such as: the automobile, airplane and ship-building industry. Milling machines are becoming more and more popular due to their ability to handle geometrically complex workpieces composed of raw material (rubber,

* Corresponding author.

PII: S0360-8352(02)00118-3

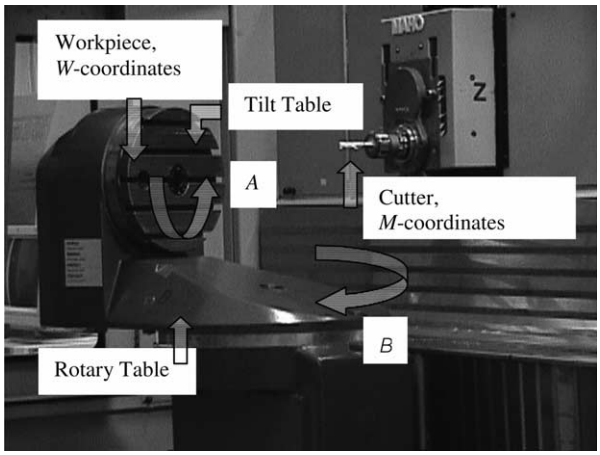

Fig. 1. Five-axis milling.

wood, stone, metal, plastic, etc.). Moreover, up-to-date milling robots are characterized by a high material removal rate and by an efficient surface finish up.

Unfortunately, it is not all rosy. Several physical phenomena, such as machine kinematics, thermal effects, static and dynamic loading, common-cause failures, clamping and spindling of the cutting device, often affect the quality of the desired surface. However, the particular impact of the machine kinematic–geometric errors, which seems to be the most significant 'bug-bear' (Srivastava, Veldhuis, & Elbestawit, 1995), is the main subject of our paper.

Let us first introduce a typical configuration of the prototype five-axis milling robot with the rotary axes on the table depicted in Fig. 1. The cutting tool (which has 5 degrees of freedom) is guided by axial commands carrying the 3 spatial (Cartesian) coordinates of the tool-tip in the machine coordinate system M and the two rotation angles (Fig. 1). The supporting CAM software generates a successive set of coordinates (usually called cutter location points or CL-points) in the workpiece coordinate system P. Typically, the CAM software distributes the CL-points along a set of curves which constitutes the so-called zigzag or spiral tool-path (Fig. 2). An appropriate transformation into the M-system generates the set of the machine axial commands providing reference inputs for the servo-controllers of the milling robot.

In order to guide the tool between the prescribed CL-points, an axial command translates the centers of rotation (Fig. 1) and simultaneously rotates the P-coordinates. Consequently, the tool-tip (affected by the machine kinematics) travels in the P-system along a non-linear trajectory. Fig. 3 illustrates the phenomenon showing a trajectory obtained by our simulation program. Note that the trajectory is purely illustrative. Practical machining requires the CL-points positioned much closer. In order to ensure a

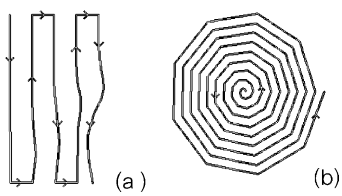

Fig. 2. Zigzag (a) and spiral tool-path.

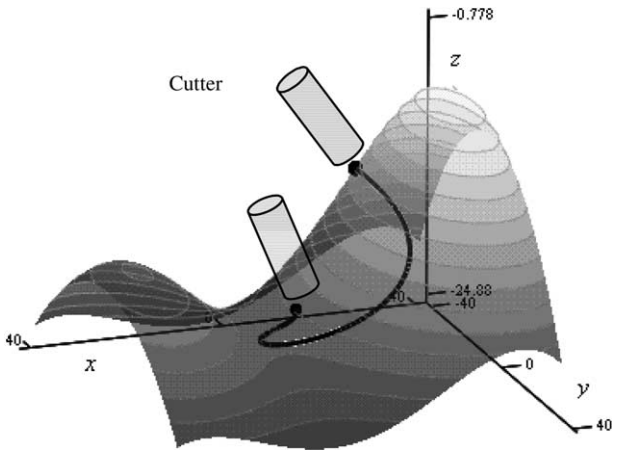

Fig. 3. Illustration of non-linearity of the tool-path in the workpeice coordinates.

prescribed tolerance, the standard CAM software estimates the local errors and incorporates additional cuts (if applicable) into a single output-block. However, such a strategy invokes a substantial increase of the CL-points and consequently a substantial increase of the machining time.

Recent papers have displayed a number of sophisticated methods to optimize a zigzag, spiral or window frame pattern (Srivastava et al., 1995; Kamien & Li, 1990; Chou & Yang, 1992) often combined with techniques dealing with the geometric complexity of the workpiece (Koren, 1995; Kim & Jeong, 1996). Moreover, a variety of off-line methods are available to generate a suitable non-uniform tool-path, for instance: the neural network modeling approach (Suh & Shin, 1996), the Voronoi diagram technique (Jeong & Kim, 1999a), the monotone chain method (Park & Choi, 2000), the distance map method (Jeong & Kim, 1999b), space filling curves (Sarma, 2000), etc. However, a *robust* algorithm to generate such complicated patterns is still an open problem. On the other hand, the 'structured' zigzag/spiral approach is simple and robust and therefore suitable to be embedded into conventional CAM-applications.

Our survey clearly indicates the need for further research in the tool-path planning of milling robots. In the meantime, we propose a tangible contribution based on grid generation methods, initially developed for CFD-applications (Huang & Oliver, 1995; Liseikin, 1996). It should be noted that the grid generation technique is surprisingly well-adapted to tool-path optimizations. As a matter of fact, the concept of a grid refinement contains almost all the main ingredients for tool-path planning, such as: grid adaptation to the regions of large milling errors, conventional zigzag(spiral) patterns and constraints related to the tool diameter and the scallop height. Moreover, in contrast to the standard techniques which are characterized by a *local* error estimate, a grid generator deals with a *global* spatial error and consequently adapts *all* the CL-points simultaneously. However, to our best knowledge, the advantages of an appropriate modification to tool-path planning seem to be overlooked.

In order to construct a suitable grid generator, we first define the global spatial error by the difference (in an appropriate normed space) between the required surface and the surface generated by non-linear trajectories. Next, we present a modification of the variational gridding techniques (Thompson & Weatherill, 1993; Ivanenko, 1993; Daniel, 1995). Our modification is based on the optimization of the smoothness of the grid, the adaptivity to the regions of large milling errors and on a particular constraint

related to the prescribed scallop height. Then, we present the numerical solution of our proposed constrained minimization problem by means of a penalty-type iterative algorithm.

Finally, we show(by means of some lucid numerical simulations) that our contribution provides a significant increase in the accuracy of milling.

2. Non-linear kinematics

In order to calculate the tool-path between successive positions p and $p+1$, we first transform the part-surface coordinates into the machine coordinates. Secondly, the rotation angles, denoted by a and b, and the corresponding machine coordinates $M \equiv (X, Y, Z)$ are assumed to change linearly between the prescribed points, namely,

$$M(t) = t\,M_{p+1} + (1-t)M_p, \quad a(t) = t\,a_{p+1} + (1-t)a_p, \quad b(t) = t\,b_{p+1} + (1-t)b_p,$$

where t is a fictitious time coordinate ($0 \le t \le 1$) and M_p the machine coordinate at the point p. The corresponding rotation angles are given by

$$a_p = \begin{cases} \tan^{-1}\left(\dfrac{T_x^{\mathrm{o}}}{T_y^{\mathrm{o}}}\right), & \text{if } T_x^{\mathrm{o}} \ge 0 \text{ and } T_y^{\mathrm{o}} \ge 0, \\ \pi + \tan^{-1}\left(\dfrac{T_x^{\mathrm{o}}}{T_y^{\mathrm{o}}}\right), & \text{if } (T_x^{\mathrm{o}} < 0 \text{ and } T_y > 0) \text{ or } (T_x^{\mathrm{o}} < 0 \text{ and } T_y^{\mathrm{o}} < 0), \\ 2\pi + \tan^{-1}\left(\dfrac{T_x^{\mathrm{o}}}{T_y^{\mathrm{o}}}\right), & \text{otherwise,} \end{cases}$$

$$b_p = -\sin^{-1}\{T_z^{\mathrm{o}}\},$$

where $\{T_x^{\mathrm{o}}, T_y^{\mathrm{o}}, T_z^{\mathrm{o}}\}$ is the orientation of the tool.

Finally, we invoke an inverse kinematics equation technique e.g. (Chou & Yang, 1992; Lin & Koren, 1994). Observe that the inverse kinematics is depending upon the structure of a machine. For our machine configuration the resulting inverse kinematics equations, involving two rotations and three translations, are given by

$$P = A^{-1}(B^{-1}(M - C) - T) - R,$$

where $M \equiv M(t)$, $P \equiv P(t)$, $R \equiv R(t)$, $T \equiv T(t)$, $C \equiv C(t)$ are respectively the machine, the part-surface, the rotary table, the tilt table and the cutter center coordinates. $A \equiv A(a(t))$, $B \equiv B(b(t))$ are respectively the matrices of rotation around the primary and secondary axes.

Note that the procedure to compute the rotation angles becomes ill-conditioned in the regions $|i + j| < \epsilon$ or $|k| < \epsilon$(ϵ is a small number), for instance for 'flat' surfaces (Daniel, 1995). As a matter of fact, small variations of the normal vector in the above mentioned regions could produce sharp variations of the rotation angles and numerical instabilities. Therefore, in order to provide a 'continuous' change of the rotation angles, we implement a special interpolation procedure in the regions where the normal vector degenerates. Also note that if the a-angle jumps from a small value to 360° or vice versa, then the

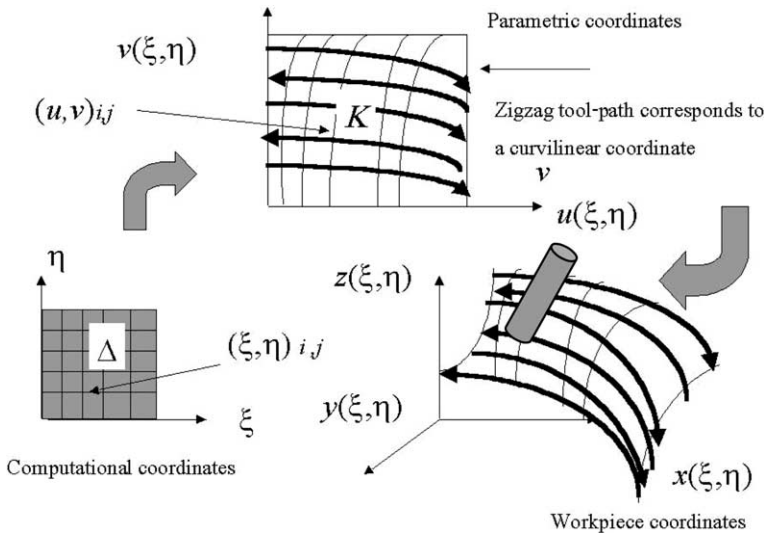

Fig. 4. Coordinate transformation.

sequence $\{a_p\}$ should be adjusted and smoothed in order to minimize the difference between the successive values.

3. Optimization problem

Let $S \equiv S(u,v) \equiv (x(u,v),y(u,v),z(u,v))$ be a machined surface, where u and v are parametric variables. Consider a set of CL-points $\{(u,v)_{i,j}\}$, $0 \le i \le N_\xi$, $0 \le j \le N_\eta$, being a discrete analogy of a mapping from the 'computational region' $\Delta = \{0 \le \xi \le N_\xi, 0 \le \eta \le N_\eta, \}$ into the 'physical region' K defined in the workpiece coordinate system (Fig.4).

A virtual surface $T(\xi,\eta)$ is composed from subsurfaces $T_{i+1/2,j+1/2}(\xi,\eta)$ spanned onto a grid-cell $\{(u,v)_{i,j},(u,v)_{i+1,j}\ (u,v)_{i,j},\ (u,v)_{i,j+1}\}$. Note that T depends on the direction of the zigzag motion (along the curves $\xi = \text{const}$ or the curves $\eta = \text{const}$). However, without a loss of generality, we shall assume that the tool path is constructed along the ξ-lines.

A quality criterion for a global tool-path is then formulated in terms of the spatial estimate $\omega \equiv |S(\xi,\eta) - T(\xi,\eta)|$.

First of all, note that the error estimate could be modified in order to involve some additional technical effects such as machining techniques, tool inclination, etc. (Lin & Koren, 1994). For instance, the tool path may depend on a prescribed policy to calculate the orientation of the tool. Such a policy could then be formulated in terms of some supplementary relations between components of the normal vector and the rotation angles. However, such modifications do not affect the main results presented in our paper. Secondly, an evaluation of the actual surface constructed in the frame-work of rigid body dynamics is quite an intricate procedure (Huang & Oliver, 1995). Moreover, concatenation of trajectories of the points belonging to the cutter, does not necessarily constitute a surface. Besides, a trajectory depends on a fictitious time variable t, whereas the required surface is characterized by the parametric representation $S(u,v)$. Therefore, one has to find (numerically) the parametric coordinates corresponding to the

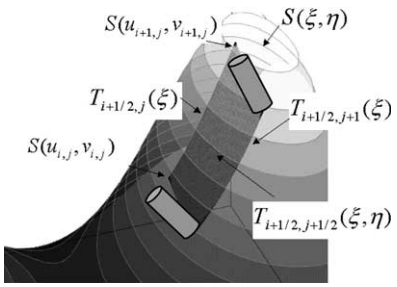

Fig. 5. Calculation of the error due to the non-linearity.

trajectories *or* to approximate $S(u,v)$ within a grid cell to obtain a local explicit representation of the surface. However, an appropriate approximation of $T(\xi,\eta)$ could be constructed by solely employing trajectories of the tool-tip between the successive CL-points. The actual surface is then given by $\bigcup_{i,j} T_{i+1/2,j+1/2}$, where

$$T_{i+1/2,j+1/2}(\xi,\eta) = L(T_{i+0.5,j}, T_{i+0.5,j+1}),\ \textit{if}\ i \le \xi \le i+1 \text{ and } j \le \eta \le j+1,$$

and where $T_{i+1/2,j}(\xi)$, $T_{i+1/2,j+1}(\xi)$ are the tool-paths between the points $(u,v)_{i,j}$, $(u,v)_{i+1,j}$ and $(u,v)_{i,j+1}$, $(u,v)_{i+1,j+1}$, respectively and L denotes the linear interpolation in the η-direction. Fig. 5 obtained by our simulation program illustrates the procedure of the error evaluation. Let $N_\xi \times N_\eta$ be an arbitrary but fixed number of grid points. We call an optimal tool-path, a solution of the following problem:

$$\underset{u_{i,j},v_{i,j}}{\text{minimize}}(E),$$

where $E = \|\omega\| \equiv \|S(\xi,\eta) - T(\xi,\eta)\|$ and $\|\ \|$ is an appropriate norm (such as the max norm or the L_2 norm). Therefore, we formulated the problem of tool-path planning in terms of a grid adaptation. As a matter of fact, considering a virtual surface T as a special interpolation of the required surface, we re-distribute a given number of the CL-points belonging to a structured curvilinear grid in order to find the best interpolation with regard to the specified norm.

4. Grid generator

Next, we introduce a modification of the variational gridding techniques (Thompson & Weatherill, 1993; Ivanenko, 1993; Ivanenko, 1992; Brackbill & Saltzman, 1982; Batanov, Bohez, Makhanov, Sonthipaumpoon, & Tabucanon, 1997) based on the optimization of several desirable properties of the mapping $(u(\xi,\eta), v(\xi,\eta))$.

The required grid is a discrete version of a solution of the minimization problem

$$\underset{(u,v)}{\text{minimize}}\,(F_s + \lambda_v F_v),$$

where

$$F_s \equiv \int\int_\Delta [(\nabla\xi)^2 + (\nabla\eta)^2]\mathrm{d}u\ \mathrm{d}v,$$

$$F_v \equiv \int\int_{\Delta} J\omega \, \mathrm{d}u \, \mathrm{d}v,$$

$$\nabla \equiv \left(\frac{\partial}{\partial u}, \frac{\partial}{\partial v} \right),$$

J denotes the Jacobian of the mapping and λ_v the calibration parameter. Note that F_s is related to the smoothness of the grid (the Winslow-functional (Winslow, 1967)), whereas F_v is related to an adaptation of the grid to the regions where ω is large (regions characterized by large milling errors).

The heart of the algorithm is a procedure to generate the tool-path by minimizing the weighted sum of the functionals. Changing variables in F_s and F_v yields

$$F_s = \int_{\Delta} J^{-1} \left[\left(\frac{\partial u}{\partial \xi} \right)^2 + \left(\frac{\partial u}{\partial \eta} \right)^2 + \left(\frac{\partial v}{\partial \xi} \right)^2 + \left(\frac{\partial v}{\partial \eta} \right)^2 \right] \mathrm{d}\eta \mathrm{d}\xi, \qquad F_v = \int_{\Delta} J^2 \omega \mathrm{d}\xi \mathrm{d}\eta.$$

Denote approximations of F_s, F_ν by the trapezoidal rule ($\Delta\xi = \Delta\eta = 1$) by I_s and I_v respectively. Differentiating $I_s + \lambda_v I_v$ with regard to $u_{i,j}$, $v_{i,j}$ yields a discrete analogy of the Euler equations given by

$$\hat{R}_u \equiv \frac{\partial (I_s + \lambda_v I_v)}{\partial u_{i,j}} = 0, \quad \hat{R}_\nu \equiv \frac{\partial (I_s + \lambda_v I_v)}{\partial v_{i,j}} = 0.$$

It is not hard to demonstrate that

$$\hat{R}_u \equiv a_1 u_{\xi\xi} + a_2 u_{\xi\eta} + a_3 u_{\eta\eta} + c_1 v_{\xi\xi} + c_2 v_{\xi\eta} + c_3 v_{\eta\eta} + J^2 \omega_v,$$

$$\hat{R}_v \equiv b_1 u_{\xi\xi} + b_2 u_{\xi\eta} + b_3 u_{\eta\eta} + a_1 v_{\xi\xi} + a_2 v_{\xi\eta} + a_3 v_{\eta\eta} + J^2 \omega_u$$

where the subscripts ξ, η, u and v correspond to the finite difference derivatives at the points (ξ_i, η_j) and $(u_{i,j}, v_{i,j})$ respectively, for instance, $u_{\xi\xi} = u_{i+1,j} - 2u_{i,j} + u_{i-1,j}$.

The coefficients at the partial derivatives of the second order are given by

$$a_k = a_{s,k} + \lambda_v a_{v,k}, \; b_k = b_{s,k} + \lambda_v b_{v,k}, \; c_k = c_{s,k} + \lambda_v c_{v,k}, \; k = 1, 2, 3.$$

$$a_{s,1} = -A\alpha, b_{s,1} = B\alpha, \; c_{s,1} = C\alpha, \; a_{s,2} = 2A\beta, \; b_{s,2} = -2A\beta, \; c_{s,2} = -2C\beta,$$

$$a_{s,3} = -A\gamma, b_{s,3} = B\gamma, \; c_{s,3} = C\gamma, \; A = u_\xi v_\xi + u_\eta v_\eta, \; B = v_\xi^2 + v_\eta^2$$

$$a_{v,1} = -u_\eta v_\eta, \; b_{v,1} = v_\xi^2, \; c_{v,1} = u_\eta^2,$$

$$a_{v,3} = -u_\xi v_\eta, b_{v,3} = v_\xi^2, c_{v,3} = u_\xi^2$$

See further details in Brackbill & Saltzman (1982). See also our Pascal type pseudo code (Appendix A, procedure Coefficients_ABC).

In order to construct a tool-path simultaneously adapted to large milling errors and to the constraints related to the prescribed scallop height, we endow the variational method with the discrete constraints

$$D_{i,j+1/2} \equiv (d_{i,j+1/2})^2 - (\rho_{i,j+1/2})^2 \geq 0,$$

where ρ denotes the distance between the points on the adjustent curves η_j and η_{j+1} given by $\rho_{i,j+1/2} \equiv [(x_{i,j+1} - x_{i,j})^2 + (y_{i,j+1} - y_{i,j})^2 + (z_{i,j+1} - z_{i,j})^2]^{1/2}$, $d(x, y, z) \equiv d(u, v)$ the maximum allowed distance

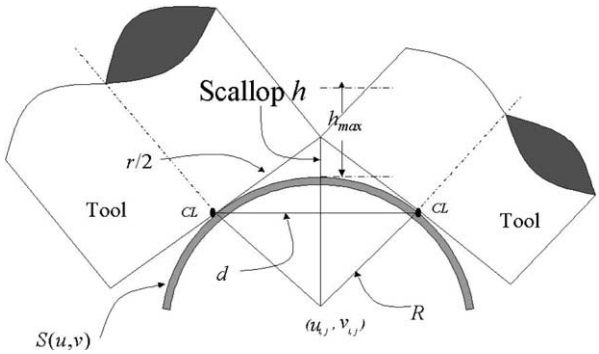

Fig. 6. Evaluation of the scallop height.

evaluated by means of the tool size and the curvature of the required surface (see Appendix A, function **Max_Allowable_Dist**).

Example. Consider a flat-end cutter with the radius r (Fig. 6) we have

$$d = \frac{2|R|\sqrt{2|R|h + h^2}}{|R| + h},$$

where h denotes the prescribed maximum scallop height and R the radius of the surface curvature in the η-direction.
In order to solve the constraint minimization problem, we define the penalty function

$$I_p = \sum_{i,j} \lambda_{i,j} p(D_{i,j+1/2}),$$

and the grid-function

$$I \equiv I_s + \lambda_v I_v + \lambda_p I_p,$$

where $\lambda_{i,j}$ are the penalty coefficients and λ_p the weight coefficient. $p(D)$, $D \in (-\infty,0)$, is a convex decreasing function, $p(D) \to \infty$ if $D \to -\infty$.
The derivatives of I_p with regard to $u_{i,j}, v_{i,j}$ are added to the left hand side of the above finite difference equations, namely,

$$R_u = \hat{R}_u + \lambda_p \frac{\partial I_p}{\partial u_{i,j}} = 0,\ R_v \equiv \hat{R}_v + \lambda_p \frac{\partial I_p}{\partial v_{i,j}}.$$

Example. Consider a penalty function given by $p(D_{i,j+1/2}) = [\min(0, D_{i,j+1/2})]^2$. Since the required surface $S(u,v)$ is represented in terms of (u,v) by an appropriate approximation (such as the Bezier polynomial, the NURBS, etc.) we substitute $u_{i,j}, v_{i,j}$ into $x_{i,j}, y_{i,j}$ in $\rho_{i,j+1/2}$. Differentiation with regard to $u_{i,j}$ and $v_{i,j}$ yields the supplementary penalty terms (see Appendix A, **function Penalty**). For instance, let $S(u,v) \equiv (x,y,z) = (u,v,u^2+v^2)$. For $D < 0$ we have

$$p(D_{i,j+1/2}) = D^2_{i,j+1/2} = [(u_{i,j+1} - u_{i,j})^2 + (v_{i,j+1} - v_{i,j})^2 + (u^2_{i,j+1} + v^2_{i,j+1} - u^2_{i,j} - v^2_{i,j})^2 - d^2]^2.$$

Consequently the required derivative is given by

$$\frac{\partial I_p}{\partial u_{i,j}} = 2D_{i,j+1/2}[-2(u_{i,j+1} - u_{i,j}) - 4(u_{i,j+1}^2 + v_{i,j+1}^2 - u_{i,j}^2 - v_{i,j}^2)u_{i,j}]\lambda_{i,j+1/2} +$$

$$2D_{i,j-1/2}[2(u_{i,j} - u_{i,j-1}) + 4(u_{i,j}^2 + v_{i,j}^2 - u_{i,j-1}^2 - v_{i,j-1}^2)u_{i,j}]\lambda_{i,j-1/2}.$$

Remark. $p(D)$ is not differentiable at $D = 0$. Therefore, one has to perform an appropriate regularization near $D = 0$. For instance,

$$p_{reg}(D) = \begin{cases} p(D), & \text{if } D \geq \epsilon, \\ H(D), & \text{if } 0 < D < \epsilon, \\ 0, & \text{otherwise,} \end{cases}$$

where $H(D)$ is a Hermite polynomial satisfying

$$H(\epsilon) = p(\epsilon), \quad \frac{\mathrm{d}H(\epsilon)}{\mathrm{d}D} = \frac{\mathrm{d}p(\epsilon)}{\mathrm{d}D}, \quad H(0) = 0.$$

We solve the corresponding algebraic system by the Jacobi iterations given by

$$u_{i,j}^{n+1} = u_{i,j}^n - [\gamma(R_u R_{vv} - R_v R_{uv})(R_{uu}R_{vv} - R_{vu}R_{uv})^{-1}]_{i,j},$$

$$v_{i,j}^{n+1} = v_{i,j}^n - [\gamma(R_v R_{uu} - R_u R_{vu})(R_{uu}R_{vv} - R_{vu}R_{uv})^{-1}]_{i,j},$$

where n is the iteration index, R_{vv}, R_{uu}, $R_{vu}R_{uv}$, the corresponding partial derivatives, γ the iteration parameter. The iteration parameter is chosen so that the the discrete Jacobian calculated by $J_{i,j} = (u_{i+1,j} - u_{i,j})(v_{i,j+1} - v_{i,j}) - (v_{i+1,j} - v_{i,j})(u_{i,j+1} - u_{i,j})$ is positive. If $J_{i,j} \leq 0$ for some i, j the parameter γ is multiplied by 0.5 (see Appendix A, **function Coeff_R** and **function Jacobi**).

The penalty coefficients are updated by the iterative procedure

$$\lambda_{i,j+1/2}^{l+1} = \begin{cases} \lambda_{i,j}^l + \delta\lambda_{i,j}^l, & \text{if } D_{i,j+1/2} < 0, \\ \lambda_{i,j}^l, & \text{otherwise,} \end{cases}$$

where l is the index of the iteration and where $\delta\lambda_{ij}^l$ denotes the corresponding increment (see Appendix A, **function La_New**). In order to improve the stability of the algorithm, we use the smoothing procedure (see Appendix A, **function Smooth**)

$$(u_{i,j}^{n+1})_{\text{new}} = u_{i,j}^{n+1}(1 - \theta) + u_{i,j}^n\theta, \quad (0 \leq \theta \leq 1).$$

$$(v_{i,j}^{n+1})_{\text{new}} = u_{i,j}^{n+1}(1 - \theta) + u_{i,j}^n\theta.$$

Performance of the algorithm

Next, we illustrate performance of the proposed procedure by means of the numerical examples.

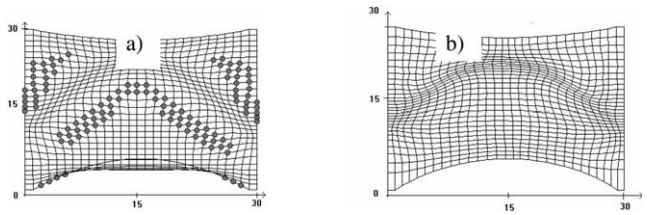

Fig. 7. Adaptation to the curvlinear zone of large milling errors (a) 20 iterations, (b) 47 iterations.

Example 1. Adaptation of the zigzag tool-path to a sinus-shaped zone of large milling errors.
This example illustrates convergence of the tool-path generation method for a workpiece having curvilinear boundaries. Consider a flat end cutter, let $R_{\min} = 45$, $h = 0.01$. The feasible region is then bounded by $d < d' = 1.25$. The milling errors are *emulated* by

$$\omega = \exp\Big(- |y - 0.5(L_y - 6)\sin\Big(\frac{\pi x}{L_x - 0.5}\Big) + 0.5|\Big).$$

Fig. 7(a) and (b) are displaying generation of the curvilinear tool-path satisfying the prescribed constraints generated by $p(D) = [\min(D, 0)]^2$, $\delta\lambda = 1$, $\lambda_v = 10$, $\lambda_p = 0.9$. Fig. 8 illustrates the convergence of the method, i^* denotes the total number of iterations, curve (1) corresponds to the error E, curve (2) displays $\max|\min(D_{i,j}, 0)|$ regarded as the 'distance' to the feasible region, curve (3) corresponds to the number of 'bad' points i.e. the points such that $D_{i,j} < 0$. (Note that the curves are normalized to 3). Finally, Table 1 reveals that $\lambda_p = 0.9$, $v = 0.25$ are optimal. The symbol # indicates divergence.

Example 2. Adaptation of the spiral tool-path.
Fig. 9 displays a grid and the corresponding spiral tool-path adapted to a circular zone belonging to a circular workpiece. Note that the center of the zone does not coincide with the center of the workpiece. Consequently, the proposed procedure generates a sophisticated tool-path which can not be produced manually or by a conventional software.

Example 3. A spiral tool-path embedded into a zigzag tool-path.
This example demonstrates how to produce a tool-path composed from segments corresponding to different types of motion. The generated tool-path (Fig.10(a)) is simultaneously adapted to the curvilinear boundaries and to the three zones of large milling errors located inside the circular region and

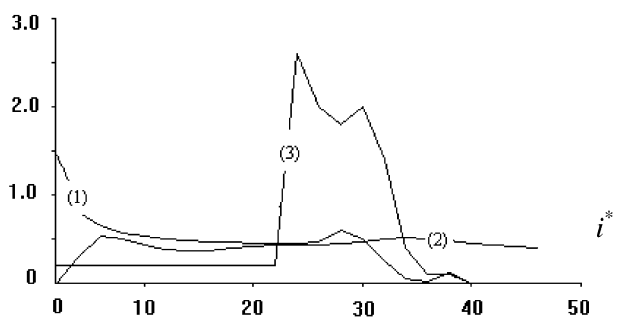

Fig. 8. Convergence of the method.

Table 1
Convergence of the method

$\lambda_p \backslash \theta$	0.1	0.25	0.50	0.75
0.01	836	900	1001	1200
0.10	301	320	380	402
0.25	115	54	331	380
0.50	201	56	256	371
0.90	90	52	210	330
1.00	82	58	201	301
1.25	78	63	180	270
1.50	#	#	#	265
1.75	#	#	#	#

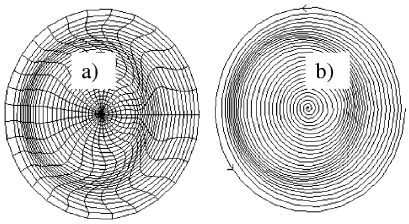

Fig. 9. Adaptation of the spiral tool path.

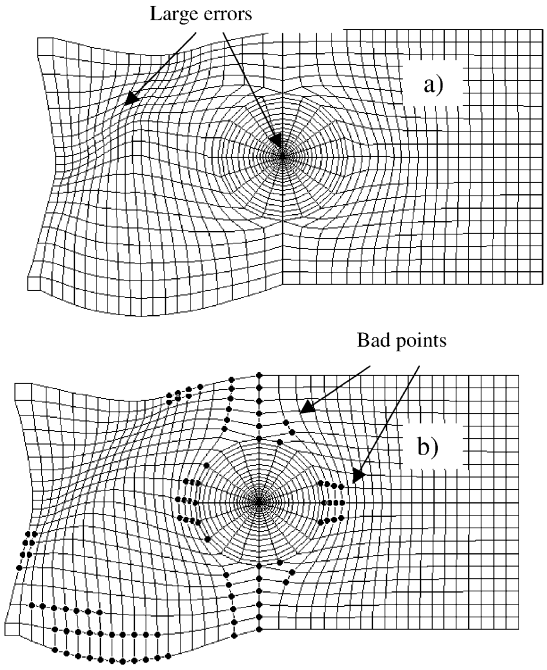

Fig. 10. Adaptaion of the composed zigzag/spiral tool path. (a) constraint minimization (b) unconstraint minimization.

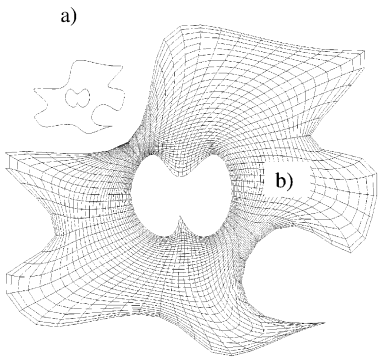

Fig. 11. Adaptation to the boundary. Complex pocket milling.

at the left part of the workpiece. Let $R_{\min} = 63$, $h = 0.01$. Observe that the constraint related to the scallop height is quite significant. Fig. 10(b) displays an 'over-adapted' grid (147 iterations) constructed by an unconstrained minimization. Clearly, the grid is unacceptable. The maximum scallop height for the over-adapted grid is about 50 times more than the prescribed value. Furthermore, in order to prevent a too small distance between the *CL*-point located in the irrelevant regions (i.e. regions where the milling errors are small) we impose the lower bound

$$D^- \equiv \rho - d'' \geq 0, (u, v) \in \Omega \equiv \{(u, v) : \omega(u, v) < \epsilon_1\},$$

where ϵ_1 is a small positive constant.
The composed tool-path(175 iterations) generated with $\epsilon_1 = 10^{-5}$, $d' = 1.6$, $d'' = 0.6$. (Fig.10(a)) is adapted to both prescribed constraints with $p(D) = [\min(D, 0)]^2$, $\delta\lambda = 1$, $\lambda_v = 10$, $\lambda_p = 0.25$.
Clearly, the second tool-path is well adapted to the region adjacent to the large milling errors. The tool-path has been generated by a re-distribution of the points belonging to the irrelevant regions. This application clearly demonstrates that too small (too large) steps are often side effects produced by an adaptation to the zones of large milling errors. However, our method allows to satisfy the prescribed constraints by modifying the space steps in the irrelevant regions. The number of the required steps does not increase significantly. Therefore, the penalty functions technique constitutes an essential supplementary measure to improve the properties of the tool-path.

Example 4. Complex pocket milling
Furthermore, the algorithm is applicable to generate a tool path in the case of a complex boundary. Consider an example of a complex shaped domain depicted in Fig. 11(a). First of all, note that such regions are not likely to often appear in the practice of conventional manufacturing. However, the authors of Jeong and Kim (1999a) address this domain as a methodological example of complex pocket milling which may not be solved by means of a regular zigzag pattern. The solution proposed in Jeong and Kim (1999a) is based on the distance map algorithm. However, the grid generation technique enables us to simultaneously generate an appropriate zigzag and spiral tool paths depicted in Fig. 11(b). The grid is well adapted to the internal and external boundary. Besides, (and it is much more important!) the flexibility of the grid generation approach allows adaptation to the regions where high quality milling is required. Also observe that, in contrast to a solution based on the neural network modeling principle

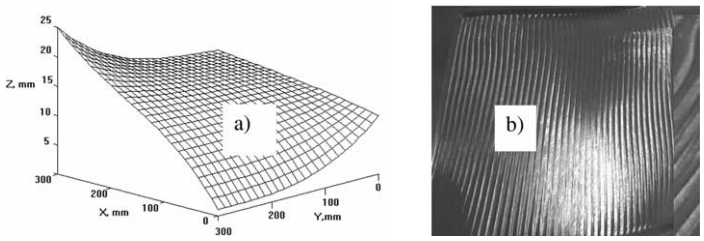

Fig. 12. Typical Bezier surface. (a) simulation (b) machined surface(steel).

applied to a similar case (Suh & Shin, 1996), the grid generator produces very smooth paths without intersections. Consequently, a feed rate adjustment is totally superfluous.
Finally, we have used the following simplifications. First, the maximum allowable distance d has been replaced by d' such that if $D \equiv d' - \rho \geq 0$ then $d - \rho \geq 0$ for any d. However, such a replacement could be rather restrictive in some applications. Secondly, the milling errors were emulated.

In Section 5, we present an application without the above drawbacks.

5. Application

Consider a set of grids $\Im \neq \varnothing$ (tool-paths) comprising an arbitrary but fixed number of the CL-points and satisfying a prescribed scallop height constraint.

Let $G_c \in \Im$ be a conventional tool-path and G_1 be a minimizer of F. If $G_1 \in \Im$ and $\Im$ is a compact set. Then our algorithm converges to G_1 given any initial grid G (for instance $G = G_c$). Moreover, $E_1 < E_c$. Now consider $G_c, G_1 \notin \Im$. In this case, the constraint minimization techniques could produce a grid G_2 which actually does not decrease E (even increase E with regard to E_c!). Such a 'paradox' has a very simple explanation. The accuracy of milling is determined by the pair (E,h) rather than solely by E. Therefore, only the CL-points located in the intersection of the zones with a small curvature and small milling errors are admissible for the re-distribution.

Consequently, if the minimization leads to an increase of the error, the optimization requires a 'trial-and-error compromise' between E and h. In other words, the set $\Im$ has to be expanded by increasing the maximum allowed scallop height. Alternatively (if the scallop height can not be modified) the set of the CL-points has to be increased.

The following example illustrates a trial-and-error procedure. Fig.12(a) displays a typical required surface whereas Fig.12(b) the machined part. A conventional tool-path (20×20) is shown in Fig.13(a). Bad points are indicated by circles sized proportionally to D, whereas points with large ω-values are indicated by boxes with the size proportional to ω. The corresponding $E_c = 0.08$.

A grid generated by the unconstrained minimization (Fig.13(b)) yields $E_1 = 0.04$. Finally, the constrained minimization (Fig.13(c)) produces $E_2 = 0.085 > E_c$(!). Therefore in order to construct an appropriate tool-path, the maximum allowed scallop height or the number of CL-points has to be increased.

We analyze both proposed options. Increasing h to 0.004 yields an adapted tool-path satisfying the prescribed scallop constraint. In this case $E_2 = 0.041$, i.e. we obtain an accuracy increase of about 50%. On the other hand, an adapted tool-path having (35×35) CL-points Fig.13(d) yields $E_2 = 0.021$,

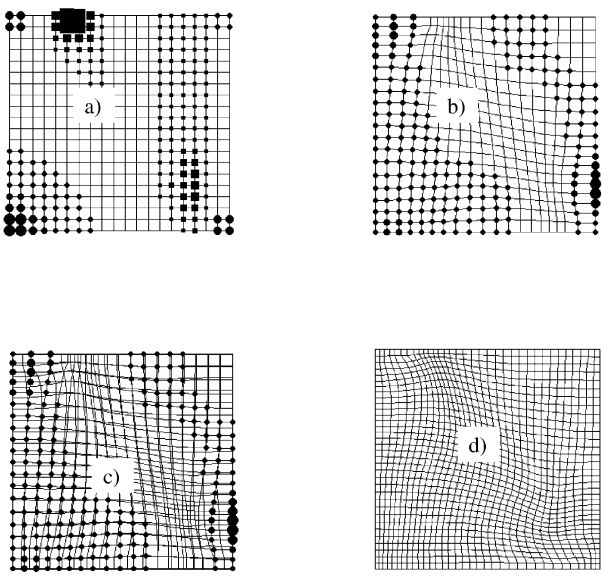

Fig. 13. The Bezier surface, (a) conventional tool path, 400 points, (b) unconstrained optimization with regard to the error, 400 points, (c) constrained optimization with regard to the error and the scallop height compared with (b), 400 points, (d) constrained optimization, 1225 points.

whereas a conventional (35×35) tool-path still does not satisfy the condition $h = 0.003$. Moreover $E_c = 0.045$.

Observe the possibility to adapt a tool-path if only a cross-feed adjustment is required. A suitable adaptation could then be solely performed by penalty functions, i.e. by a discrete functional given by $I_s + \lambda_p I_p$.

A large series of surfaces has been tested by means of the proposed algorithm. A typical testing surface sequence is depicted in Fig. 12. The corresponding accuracy increase produced by the adapted tool-path presented by Table 2, demonstrates an average accuracy increase δ_A ranging between 32 and 43%. R_C denotes the roughness after the conventional machining whereas R_A roughness after the adaptive method. The symbol $*$ indicates that the real machining has not been performed.

Finally, we remark that orthogonality measures (Brackbill & Saltzman, 1982) could be easily included into the algorithm. However, (in contrast to the CFD-applications) angles between the grid lines do not have a significant impact on the accuracy.

Table 2
Accuracy and roughness of the machined surface

$N_\xi N_\eta$	h (mm)	δ_A (%/mm)	R_C (μm)	R_A (μm)
100	3.60	34/0.2600	*	*
400	1.80	41/0.0930	*	*
900	1.20	34/0.0580	34.8	17.3
1600	0.90	36/0.0410	14.3	6.6
3600	0.60	32/0.0260	5.9	4.3
6400	0.45	40/0.0180	2.6	2.1

6. Conclusion

Our proposed numerical algorithm has the following notable features.

1. Two-dimensional adaptation to the regions of large milling errors.
2. Straightforward control of the constraints related to a scallop height.
3. Capability to generate a tool-path for workpieces with complex geometries.
4. Significant increase in the accuracy of milling.

Finally, the proposed approach is particularly suitable to be embedded in a conventional CAD/CAM software in so far that the algorithm deals with a curvilinear version of the standard zigzag/spiral pattern. Therefore, such modifications require only an additional mapping obtained by means of the grid generation procedures.

Acknowledgments

This research is sponsored by the National Science and Technology Development Agency (NSTDA) and the National Electronics and Computer Technology Center (NECTEC) of Thailand.

Appendix A. The algorithm

The algorithm comprises the global iterations based on incrementing the adaptation coefficient λ_v, the local Jacobi iterations and smoothing. At each λ_v-iteration we adapt the tool path to the error E while E exceeds the prescribed tolerance. In turn, the local iterations generate the tool path for a fixed λ_v satisfying the prescribed constraints.

Our Pascal style pseudo code specifies four basic data types: **Basic_Array** corresponds to the tool path with regard the parametric coordinates. **Tool_Path** denotes the entire tool path in the parametric domain, Surface the required surface at the point (u,v) and **Tool_Path_Loc** an array representing the tool path between (u_1,v_1) and (u_2,v_2). The basic variables are: **Path_W, Path_J:Tool_Path** employed by the λ_v-iterations and the Jacobi iterations respectively. The error ω is declared as **Basic_Array**. The magnitude of the difference between the preceding and the updated tool path is declared by **Tool_Path_Diff: Tool_Path**.

The basic functions and procedures are

procedure Input_Data;{the procedure inputs the following data
$S(u,v)$ the required surface (control points),
r a radius of the tool,
N_ξ and N_η a number of the CL points,
Tolerance the max allowable deviation between the required and the actual surface,
h the maximum allowable scallop height,
λ_v the initial value of the adaptation coefficient,
eps1, **eps2** the stopping constants for the external and internal iterations}

procedure Output_Data; {performs a numerical and graphic output}
function S(u,v:real):Surface; {returns the required surface at the point (u,v)}
function ToolPathLoc(u1,v1,u2,v2:real):Tool_Path_Loc; {returns the actual 3D tool path between (u_1,v_1) and (u_2,v_2)}
function Error_Loc(u1,v1,u2,v2:real):Tool_Path_Loc;{returns the error along the path between (u_1,v_1) and (u_2,v_2)}
function Error_W(Path:Tool_Path):BasicArray;{returns the error ω}
function Generate_New_Tool_Path(ω:Basic_Array;Path: Tool_Path): Tool_Path {performs the Jacobi and penalty iterations, returns the adapted tool path satisfying the prescribed constraints};
function Smooth(Path:Tool_Path):Tool_Path;{returns the smoothed tool path}
function Lv_New:real;{updates the coefficient λ_v}

The program body is given by

begin
Input_Data;
Path_W $\leftarrow$ a uniform tool path; {initialize the tool path}
$\omega \leftarrow$ **Error_W(Path_W)**;{calculate the error on Path_W}
while ($E \geq$ **Tolerance) and (Tool_Path_Diff <eps1) do**
begin
Path_W $\leftarrow$ **Generate_New_Tool_Path(Path_W)**; {adapt the tool path}
$\omega \leftarrow$ **Error_W(Path_W)**;{calculate the error on the Path_W};
$\lambda_v \leftarrow$ **Lv_New**;{increment λ_v}
end
Output_Data;
end.

Next, we present a pseudo code of the function **Generate_New_Tool_Path**; We declare $d_{i,j+1/2}$, $P(D_{i,j+1/2})$,$\lambda_{i,j}$, a_k, b_k, c_k and $R_u, R_v, R_{uu}, R_{vv}, R_{uv}$ as **Basic_Array**. The Jacobi iterations are interrupted whenever the residual of the Jacobi equations (**Jacobi_Res**) is less then eps2.
Functions declared by the procedure are

function Curvature(u,v:real):real; {returns the curvature of the surface at (u,v)}
function Max_Allowable_Dist(Path_J: Tool_Path, *r*, *h*: real): Basic_Array; { returns the array of $d_{i,j+1/2}$}
function Penalty(Path_J:Tool_Path,d:Basic_Array):Basic_Array; {returns $P(D_{i,j+1/2})$}
function L_New(Path_J:Tool_Path):Basic_Array; {updates the penalty coefficients $\lambda_{i,j}$}
procedure Coefficients_ABC;{calculates a_k, b_k, c_k passed as the global variables}
procedure Coefficients_Jacob;{calculates$R_u, R_v, R_{uu}, R_{vv}, R_{uv}$ passed as the global variables}
function Jacobi(Path:Tool_Path):Tool_Path;{performs one step of the Jacobi iterations}

The body of the procedure is given by

begin

Path_J ← Path_W; {Path_W, supplied by the main program}
while (there exists i, j **such that** $P(D_{i,j}) \neq 0$) **do** {penalty iterations}
begin
$\lambda \leftarrow$ **L_New(Path_J)**; {updates the penalty coefficients}
$d \leftarrow$ **Max_Allowable_Dist(Path_J,*r*,*h*)**; {updates the penalty related variables}
$P(D) \leftarrow$ **Penalty(Path_J,d)**; {updates the penalty function}
while (Jacobi_Res > eps2) do {Jacobi iterations}
begin
a_k, b_k, $c_k \leftarrow$ **Coefficients_ABC**;{updates the coefficients of the Euler equations}
R_u, R_v, R_{uu}, $R_{uv} \leftarrow$ **Coefficients_Jacobi**;{updates the Jacobi equations}
Path_J ← Jacobi(Path_J);{performs one step of the Jacobi iterations}
end {end of the Jacobi iterations}
Path_J ← Smooth(Path_J);
end {end of the penalty iterations}
Path_W ← Path_J; {passes the result to the main program}
end;

It should be noted that neither the λ_v-iterations nor the penalty iterations are guaranteed to converge. Therefore, the code is endowed with the graphic interface displaying the evolution of the tool path. Besides the code displays a warning message in the case when a possibility of divergence is detected. The user may exit the program or continue the iterations. (see the discussion regarding the convergence presented in Section 6). Finally, in practice the error ω is often proportional to the curvature. In this case, ω in I_v could be replaced by the curvature multiplied by a calibrating parameter. The numerical experiments show that the algorithm generates a tool path close that produced by means of ω.

References

Batanov, D., Bohez, E., Makhanov, S. S., Sonthipaumpoon, K., & Tabucanon, M. T. (1997). A curvilinear grid adaptation technique applied to tool-path planning of a five-axis milling machine (*21–24 October*) (pp. 1351–1358). Proceedings of Fourth International Conference On Computer Integrated Manufacturing, Singapore: Springer.

Brackbill, J. U., & Saltzman, J. S. (1982). Adaptive zoning for singular problems in two dimensions. *Journal of Computational Physics*, *46*(3), 342–368.

Chou, J., & Yang, D. (1992). On the generation of coordinate motion of five-axis CNC/CMM machines. *Journal of Engineering for Industry*, *114*, 15–22.

Daniel, M. (1995). A note on degenerate normal vectors. *Computer Aided Geometric Design*, *12*, 857–860.

Huang, Y., & Oliver, J. H. (1995). Integrated simulation, error assessment, and tool path correction for five-axis NC milling. *Journal of Manufacturing*, *14*(5), 331–344.

Ivanenko, S. A. (1992). *Construction of Curvilinear Meshes for Computing Flows in Basins*. Modern Problems of Computational Aerodynamics, London: CRC Press. pp. 211–250..

Ivanenko, S. A. (1993). Adaptive Grids and Grids on Surfaces. *Journal of Computational Mathematics and Mathematical Physics*, *33*(9), 1179–1194.

Jeong, J., & Kim, K. (1999). Generating tool paths for free-form pocket machining using z-buffer-based Voronoi diagrams. *Advanced Manufacturing Technology*, *15*, 182–187.

Jeong, J., & Kim, K. (1999). Generating tool paths for machining free-form pockets with Islands using distance maps. *Advanced Manufacturing Technology*, *15*, 311–316.

Kamien, M. I., & Li, L. (1990). Subcontracting, coordination, flexibility and production smoothing in aggregate planning. *Management Science*, *36*(11), 1352–1363.

Kim, K., & Jeong, J. (1996). Finding feasible tool-approach directions for sculptured surface manufacture. *IIE Transactions*, *28*(10), 829–836.

Koren, Y. (1995). Five-axis interpolators. *Annals of CIPR*, *44*(1), 379–382.

Lin, R., & Koren, Y. (1994). Real time interpolator for machining ruled surfaces. *Proceedings of ASME Annual Meeting*, *DSC-55*(2), 951–960.

Liseikin, V. D. (1996). The construction of structured adaptive grids—a review. *Journal of Computational Mathematics and Mathematical Physics*, *36*(1), 1–32.

Park, S. C., & Choi, B. K. (2000). Tool-path planning for directional parallel milling. *Computer Aided Design*, *32*, 17–25.

Sarma, R. (2000). An assessment of geometric methods in trajectory synthesis for shape creating manufacturing operations. *Journal of Manufacturing Systems*, *19*, 59–72.

Srivastava, A. K., Veldhuis, S., & Elbestawit, A. (1995). Modeling geometric and thermal errors in a five-axis CNC machine tool. *International Journal of Machine Tools and Manufacturing*, *35*(9), 1321–1337.

Suh, S.-H., & Shin, Y.-S. (1996). Neural network modeling for tool path planning of the rough cut in complex pocket milling. *Journal of Manufacturing Systems*, *15*, 295–324.

Thompson, J. F., & Weatherill, N. P. (1993). *Aspects of Numerical Grid Generation: Current Science and Art, AIAA-93-3539-CP*. pp. 1029–1070..

Winslow, A. M. (1967). Numerical solution of quasilinear Poisson equation on non-uniform triangle mesh. *Journal of Computational Physics*, *1*(2), 149–172.