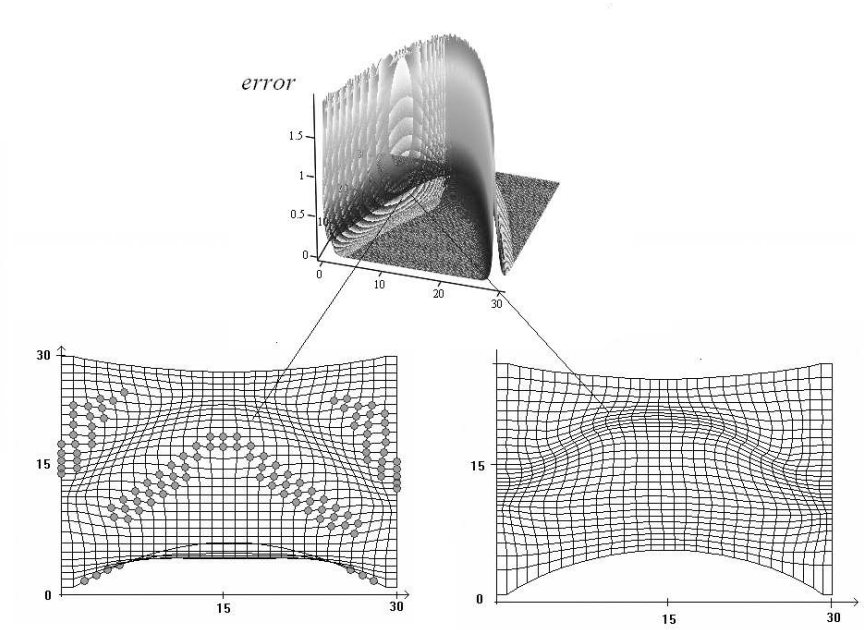


Fig. 5.7. Tool path generation with adaptation to a weight function and the boundary

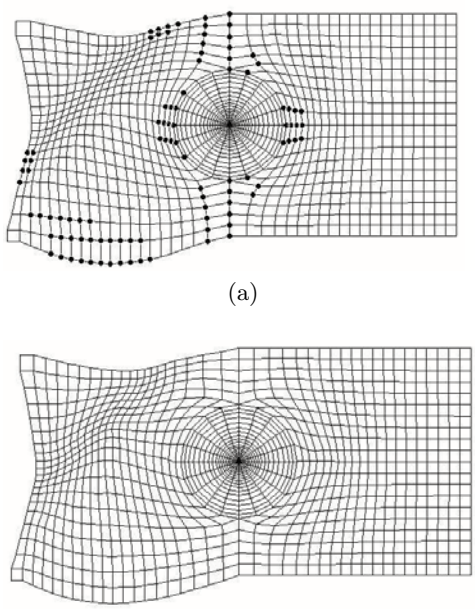
Table 5.1 illustrates the convergence of the iterations and reveals that $\lambda_p = 0.9, v = 0.25$ in this particular case lead to the best convergence rate. The symbol # indicates a situation when the iterations do not have a limit and either fluctuate or reach the computer infinity. Such situation is called divergence.

Table 5.1. Convergence of the method λ_p versus θ

$\lambda_p \backslash \theta$	0.10	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	#	#	#	#

Example 5.3. **A spiral tool path embedded into a zigzag tool path**

This example demonstrates how to produce a tool path composed from patches corresponding to different types of motion. The generated tool path in Fig. 5.8 has been adapted to the curvilinear boundaries and to three zones of large milling errors located inside the circular region and at the left part of the domain. The surface curvature and the scallop height have been set up as in Example 5.2, *artificially*, that is, without employing an actual surface.



(a)

(b)

Fig. 5.8. Grids obtained by constraint minimization (**a**) and unconstrained minimization (**b**)

The grid in Fig. 5.8(a) is constructed by an unconstrained minimization. Clearly, the grid is unacceptable since it has many *bad points*, where the constraints are not satisfied. The maximum scallop height is about 50 times more than the prescribed value. Furthermore, in order to prevent a too small distance between grid points located in the irrelevant regions it is often useful to impose a lower bound by adding a second constraint given by

$$d - w' \geq 0, \text{if } \epsilon < \epsilon_1,$$

where ϵ_1 is a small positive constant. This helps to avoid degenerated grids.

The composed tool path generated with $\epsilon_1 = 10^{-5}, w = 1.6, w' = 0.6$ shown in Fig. 5.8(b) is adapted subject to the both prescribed constraints with $p(D) = [\min(D, 0)]^2, \delta\lambda = 1, \lambda_v = 10, \lambda_p = 0.25$.

Example 5.4. **A Bezier surface**

The constraint minimization techniques could produce a grid which actually does not decrease ϵ (even increase it). This is because our model implies that the accuracy is determined by the pair (ϵ, h) rather than solely by ϵ. Consequently, if the minimization does not decrease of the error, the user should either increase the number of the tool tracks or apply a trial-and-error compromise between ϵ and h. The following example illustrates the trial-and-error procedure. We consider a Bezier surface (Fig. 5.9) characterized by the following control points.

$$B_x = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 12 & 12 & 12 & 12 \\ 24 & 24 & 24 & 24 \\ 36 & 36 & 36 & 36 \end{pmatrix}, B_y = B_x^T, B_z = \begin{pmatrix} -16 & -8 & -6 & 0 \\ -8 & -6 & 0 & -2 \\ -6 & 0 & -2 & 0 \\ 0 & -2 & 0 & 0 \end{pmatrix}.$$

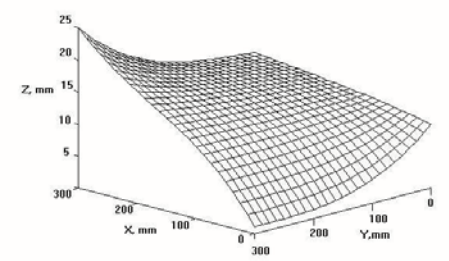


Fig. 5.9. A test Bezier surface

Fig. 5.10. The machined Bezier surface

Figure 5.10 displays an actual machining. A conventional tool path is shown in Fig. 5.11(a). The bad points are indicated by circles sized proportionally to the distance from the constraints, whereas points with large ϵ-values are indicated by boxes with the size proportional to ϵ. The resulting $\epsilon = 0.08$.

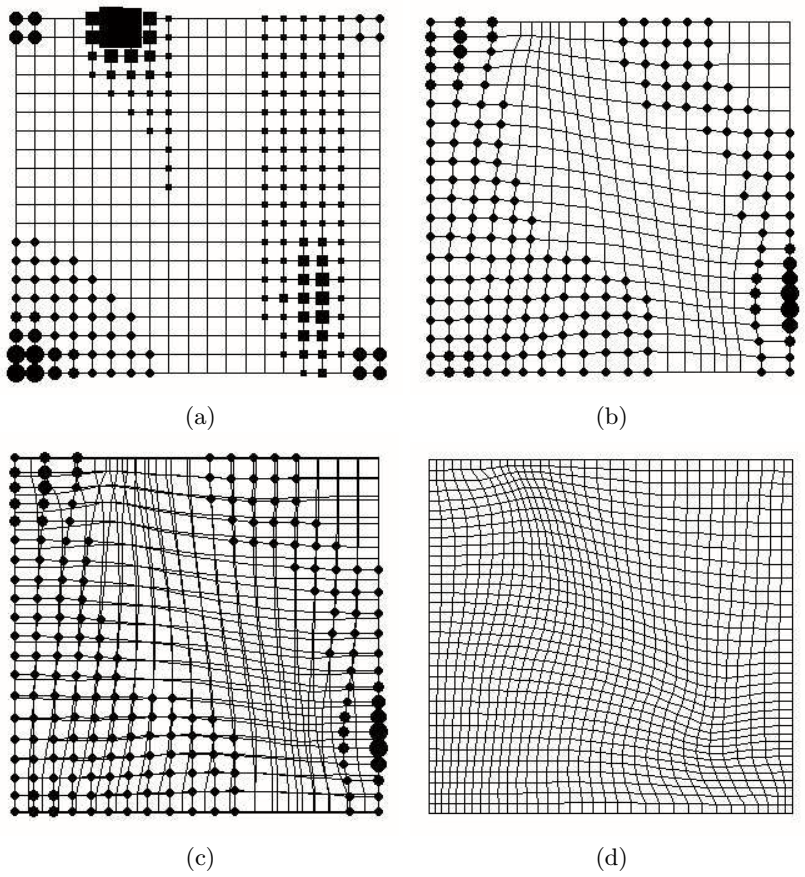


Fig. 5.11. (**a**) a conventional tool path, (**b**) unconstrained minimization, (**c**) constrained minimization, (**d**) constraint minimization on a grid 35×35

A grid generated by the unconstrained minimization (Fig. 5.11(b)) yields $\epsilon = 0.04$. Finally, the constrained minimization Fig. 5.11(c) produces $\epsilon = 0.085$. In order to construct an appropriate tool path, either the maximum allowed scallop height or the number of the grid points has to be increased.

We analyze the both proposed options. Increasing h from 0.01 to 0.04 yields an adapted tool path satisfying the prescribed scallop constraints $\epsilon = 0.041$. On the other hand, an adapted tool path having 35×35 CL points (Fig.

5.11(d)) yields $\epsilon = 0.021$, whereas a conventional 35×35 tool path still does not satisfy the scallop constraints. 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 $I_s + \lambda_p I_p$. Alternatively the scallop constraints may be included into F_v or even into the harmonic functional (5.12). The last possibility will be discussed in the next section. The orthogonality measure could also be included into the algorithm to ensure against degenerate grids and divergence.

A series of test surfaces has been tested by means of the proposed algorithm. Table 5.2, demonstrates an average accuracy increase ranging between 32 and 43%.

Table 5.2. Accuracy of the machined surface

Grid size	Scallop (mm)	improvement % of conventional
10×10	3.60	34 / 0.2600
20×20	1.80	41 / 0.0930
30×30	1.20	34 / 0.0580
40×40	0.90	36 / 0.0410
60×60	0.60	32 / 0.0260
80×80	0.45	40 / 0.0180

It is plain that the constraints related to scallops may substantially affect the solution. Besides, the solution to the minimization problem is not unique or may not exist. In practice, the initial grid typically does not satisfy the constraints and there is no prior knowledge whether such a grid exists. Consequently, the convergence is analyzed by means of numerical experiments. Note, that the adaptive approach allows nor a closed form estimate of the number of required tracks neither an estimate of the number of the CL points belonging to one track. Therefore, we developed a realistic rule applicable to the majority of practical situations. If a number of the CL points not satisfying the scallop constrains is more than 2/3 of the total number of points than the adaptation is not possible or requires a very large number of iterations. In other words, the computational efficiency of the algorithm depends on the number of points admissible for re-distribution. If this number is small than, first of all, the error can not be substantially reduced, secondly, the optimal grid may not exist at all. Even if such a grid exists, the iterations may fail to converge. Table 5.3 illustrates computational complexity of the algorithm for a 40×40 zigzag tool path constructed for the Bezier surface. The accuracy increase characterized by the ratio $\epsilon_R = 100\epsilon/\epsilon_C$, where ϵ_C is the error on the rectangular pattern, τ_C the computational time on a PC (Pentium 4), I' the number of the required iterations, N_B the number of the *bad points*, where the constraint $h = 0.01$ is violated, from the initial rectangular grid, # indicates divergence. Observe that $h = 0.015$ mm leads to a substantial accuracy increase of about 40 % with regard to the rectangular tool path, whereas

$h = 0.013$ mm yields only a 22% increase. $h = 0.001$ mm and $h = 0.012$ mm actually increase the error. It should be noted that $h = 0.015$ mm seems to be the most suitable since the scallop height and the error are approximately in the same magnitude. The calculations reveal that h and ϵ varying in the same magnitude often lead to the minimum number of iterations. It means that appropriate scallop heights could be taken as fractions of ϵ corresponding to the conventional tool path.

Table 5.3. Convergence of the algorithm h versus ϵ

Scallop h (mm)	Computational time τ_C (min)	Iterations I'	Error ϵ	Error relative to conventional ϵ_R	Number of bad points N_B
0.0001	#	#	#	#	1132
0.0010	32	1710	0.094	200	500
0.0012	11	550	0.047	105	92
0.0130	9	421	0.032	78	61
0.0150	4	250	0.024	59	25
0.0170	3	200	0.020	55	0

The grid generation algorithms applied to tool path generation offer a number of benefits such as the global two-dimensional adaptation to the regions of large milling errors. As opposed to many tool path generation methods (see Introduction) the grid searches for a global minimum adapting all the CL points simultaneously. Of course such an adaptation leads to certain disadvantages such as possible divergence, large computational time, etc. However, solutions obtained by the grid generation technologies can hardly be obtained by the local methods which may require astronomical times to reach the same optimum. Furthermore, the grid generation provides a straightforward control of the constraints related to a scallop height and capabilities to generate a tool path for workpieces with complex shaped boundaries for trimmed surfaces

Finally, the grid generation technologies are particularly suitable to be integrated with the conventional CAD/CAM software in so far that the algorithms deal with a curvilinear versions of standard zigzag or spiral patterns. Therefore, such modifications require only an additional mapping obtained by means of the grid generation procedures.

5.4 Application of Harmonic Functional to Tool Path Generation

Recall that $S \equiv S(u, v) \equiv (x(u, v), y(u, v), z(u, v)$ denotes the required surface given in the workpiece coordinates, where u and v are parametric variables. Consider a set of CC points, $\{(u, v)_n\}, 1 \leq n \leq N_n$, being a discrete analogy of a mapping from the computational region on the plane (ξ, η) into the physical region defined in the parametric coordinate system (u, v) (see Fig. 5.4).

A more general case is a regular or structured grid which consists of curvilinear quadrilaterals such as the grid shown in Fig. 5.12 or a block-structured grid in Fig. 5.13. In both cases, the grid is not topologically equivalent to a rectangular Cartesian grid. These two cases invoke mapping of the unit square in the plane (ξ, η) onto each quadrilateral in the (u, v) plane individually. The block-structured grids can also be used to construct the SFC whereas the use of a general irregular grid to construct an appropriate SFC lies out of the scope of this book.

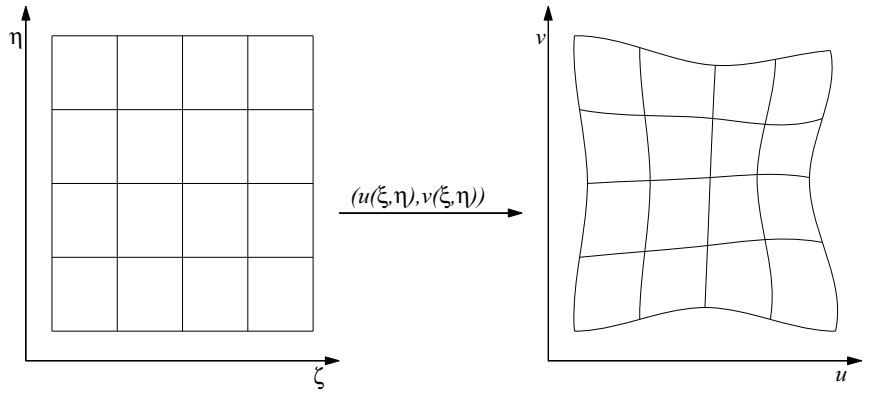


Fig. 5.12. Example of regular grid

The problem of grid generation can be treated as a discrete analog of the problem of finding functions $u(\xi, \eta)$ and $v(\xi, \eta)$. It can be shown that if a smooth mapping of one domain onto another with one-to-one mapping between boundaries possesses a positive Jacobian, then such a mapping will be one-to-one [1]. Hence, the curvilinear coordinate system constructed in (u, v) domain will be non-degenerate if the Jacobian of the mapping $u(\xi, \eta)$, $v(\xi, \eta)$ is positive, i.e.,

$$J = u_\xi v_\eta - u_\eta v_\xi > 0. \tag{5.13}$$

A number of techniques for grid generation have been developed [47]. In general, the methods of grid generation for structured grid can be classified into three basic groups, namely, the algebraic methods, the differential methods and the variational methods. In the algebraic approach the interior points of the grid are computed through various forms of interpolation or special functions. Differential methods generate grid points based on the solution of elliptic and parabolic equations. In variational methods, the grid is generated by optimization of grid quality properties such as non-degeneracy, smoothness, uniformity, near-orthogonality, or adaptivity (see, for examples, [15, 19, 37, 38]).

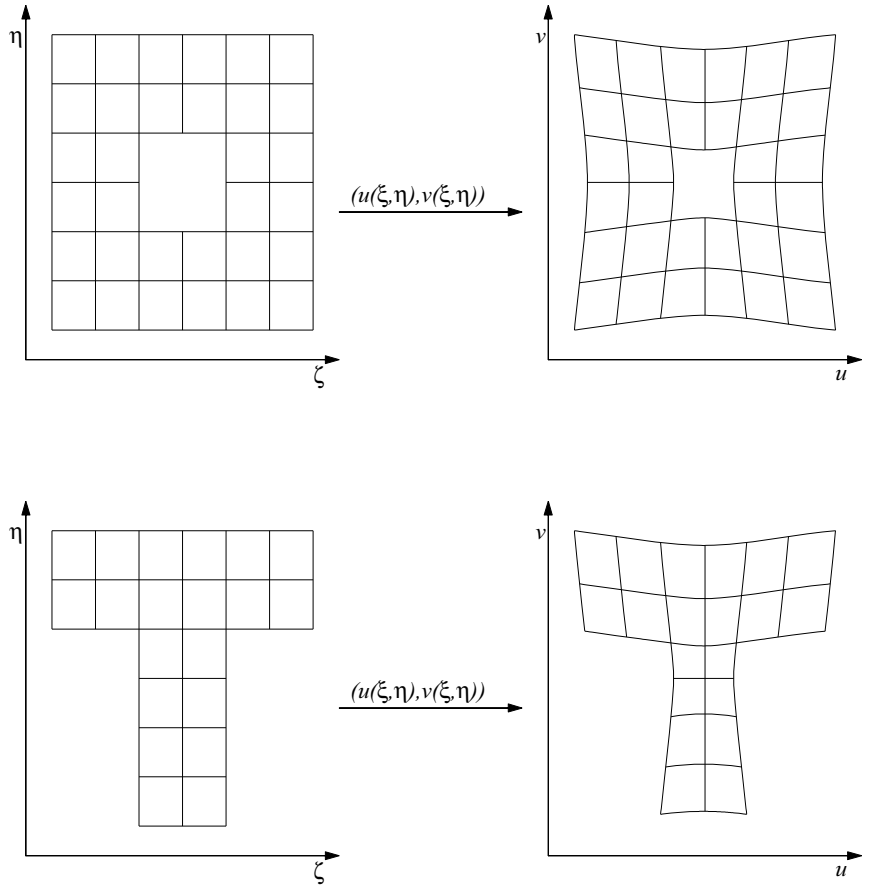
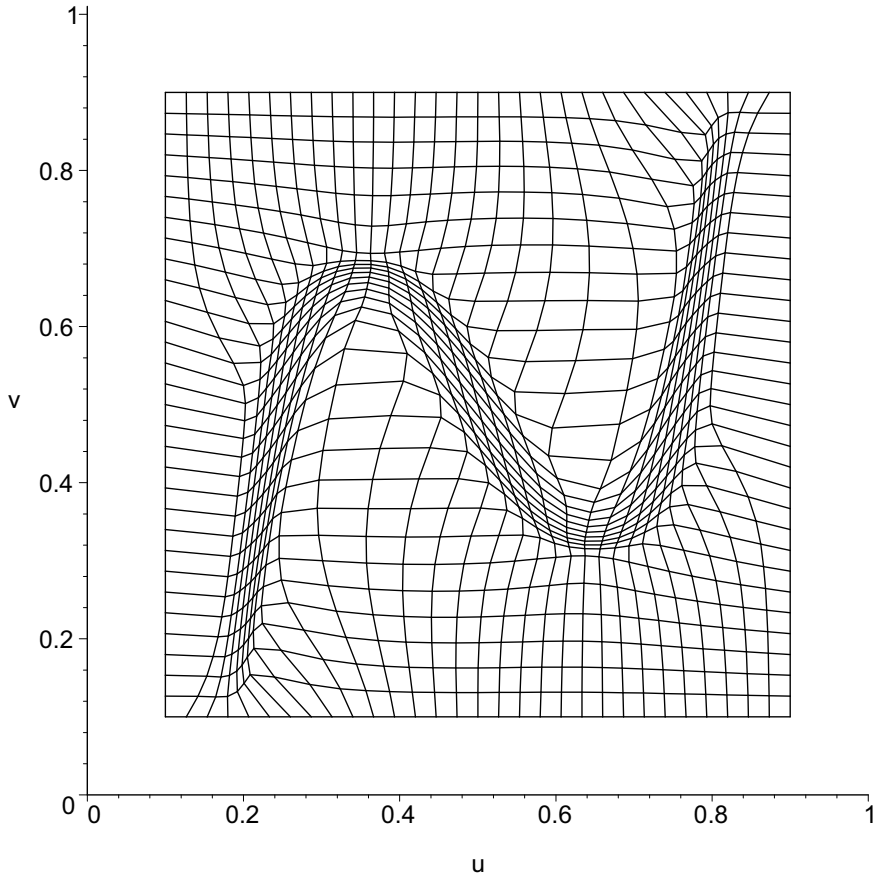


Fig. 5.13. Examples of block-structured grids

This section presents the grid generation technique based on variational grid generation method of Ivanenko [39]. The grid is considered as a discrete version of a mapping $\{u(\xi,\eta), v(\xi,\eta)\}$, where the pair $\{u, v\}$ minimizes a Dirichlet functional [39] (5.12).

The discrete version of the solution is the so called adaptive-harmonic grid characterized by small cells in the regions of large gradients of the control function $f(u, v)$ (see, for example, Fig. 5.14). The grid can be also interpreted as such a mapping from the unit square onto the parametric domain that the corresponding mapping of the surface $f(u, v)$ onto the parametric domain is harmonic. Further theoretical details can be found in [39].

Let us introduce the following data structure suitable for either regular or block-structured grids. $C_O(N, k)$ defines correspondence between the local and the global node numbers.



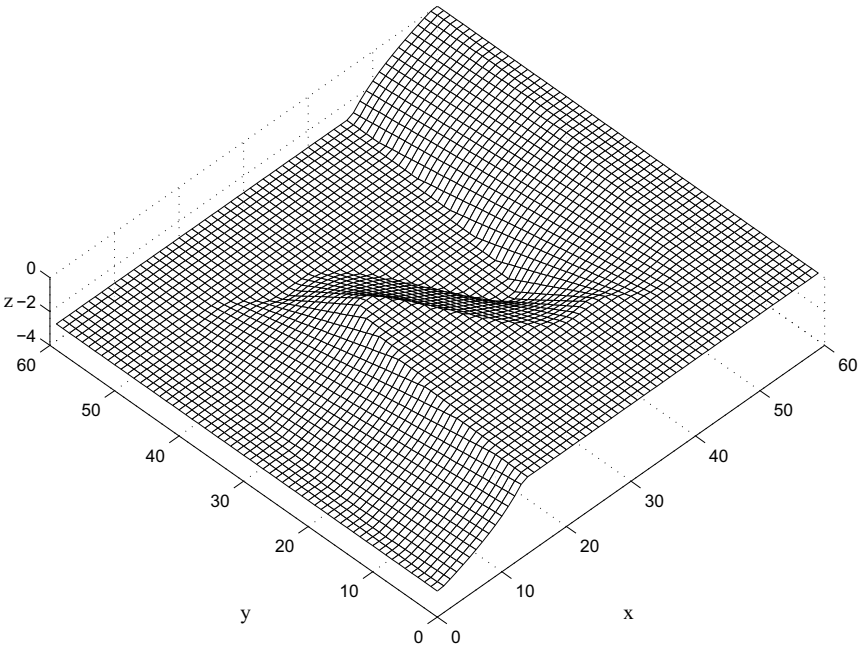
(a)

Fig. 5.14a. Example of adaptive-harmonic grid

$$C_O(N,k) = n \quad n = 1,\ldots,N_n \quad N = 1,\ldots,N_e \quad k = 1,2,3,4, \tag{5.14}$$

where n is a global node number, N_n is a total number of grid nodes, N is an element number, N_e is a number of elements, k is a local node number in the element. The cell vertices are numbered from 1 to 4 in the clockwise direction. Each vertex is associated with a triangle: vertex 1 with $\triangle_{412}$, vertex 2 with $\triangle_{123}$, vertex 3 with $\triangle_{234}$, and vertex 4 with $\triangle_{341}$. It can be easily demonstrated that $2A_{k-1,k,k+1} = J_k$ [39], where A denotes the area of the corresponding triangle and $J_k, k = 1,2,3,4$ is the Jacobian of the mapping given by

$$J_k = (u_{k-1} - u_k)(v_{k+1} - v_k) - (u_{k+1} - u_k)(v_{k-1} - v_k), \tag{5.15}$$



(b)

Fig. 5.14b. Control function

where $k-1=4$ if $k=1$ and $k+1=1$ if $k=4$. An example of such data structure is shown in Fig. 5.15.

The function $N'(n,d)$ defines the correspondence between the grid node n, and its neighbors in the d direction, $d \in \{\text{left,right,up,down}\}$. Functional (5.12) is approximated by a discrete functional as follows

$$I \approx I^h = \sum_{N=1}^{N_e} \sum_{k=1}^{4} \frac{1}{4} [F_k]_N. \tag{5.16}$$

where

$$\begin{aligned}
F_k &= \frac{D_1 \left[1+(f_u)_k^2\right] + D_2 \left[1+(f_v)_k^2\right] + 2D_3 (f_u)_k (f_v)_k}{J_k \left[1+(f_u)_k^2+(f_v)_k^2\right]^{\frac{1}{2}}}, \\
D_1 &= (u_{k-1}-u_k)^2 + (u_{k+1}-u_k)^2, \\
D_2 &= (v_{k-1}-v_k)^2 + (v_{k+1}-v_k)^2, \\
D_3 &= (u_{k-1}-u_k)(v_{k-1}-v_k) + (u_{k+1}-u_k)(v_{k+1}-v_k), \\
J_k &= (u_{k-1}-u_k)(v_{k+1}-v_k) - (u_{k+1}-u_k)(v_{k-1}-v_k).
\end{aligned} \tag{5.17}$$

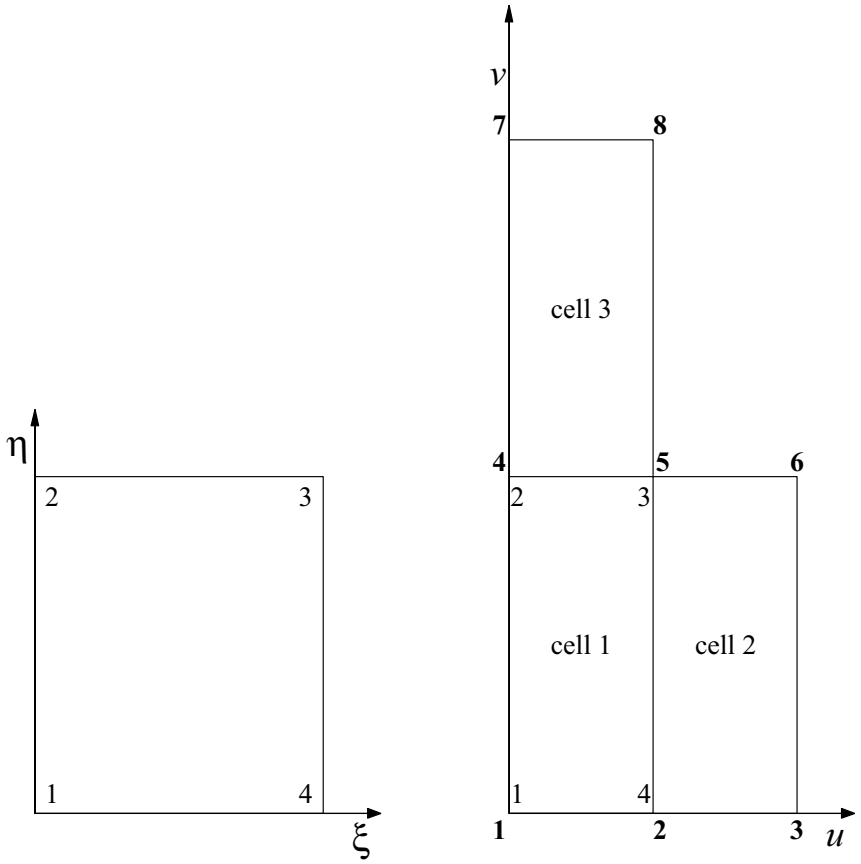


Fig. 5.15. Correspondence of node numbers for a mapping of the unit square in the (ξ, η) plane on to the quadrilateral cell 1 of the grid in the (u, v) plane

Here $(f_u)_k$ and $(f_v)_k$ are the derivatives of the control function in the u and v direction, respectively, computed in the node number k of the cell number N. The convexity of the grid cells requires that the Jacobian be positive [37], i.e.,

$$[J_k]_N > 0 \quad k = 1, 2, 3, 4 \quad N = 1, \ldots, N_e, \tag{5.18}$$

The discrete functional (5.16) is minimized numerically using an initial grid as the first iteration. The minimization is arranged in such a way that the minimum is attained on a grid composed of convex quadrilaterals. This property improves the stability of the grid generation procedure. Moreover, it can be shown that if the set of convex grids corresponding to the mapping is not empty, the system of algebraic equations

$$\begin{aligned} R_u &= \frac{\partial I^h}{\partial u_n} = 0, \\ R_v &= \frac{\partial I^h}{\partial v_n} = 0, \end{aligned} \tag{5.19}$$

has at least one solution which is a convex grid.

Suppose the grid at the l^{th} step of the iteration is determined, the grid at the $(l+1)^{\text{th}}$ step is computed using the quasi-Newtonian procedure as follows:

$$\begin{aligned} u_n^{l+1} &= u_n^l - \tau \left(R_u \frac{\partial R_v}{\partial v_n} - R_v \frac{\partial R_u}{\partial v_n} \right) \left(\frac{\partial R_u}{\partial u_n}\frac{\partial R_v}{\partial v_n} - \frac{\partial R_v}{\partial u_n}\frac{\partial R_u}{\partial v_n} \right)^{-1}, \\ v_n^{l+1} &= v_n^l - \tau \left(R_v \frac{\partial R_u}{\partial u_n} - R_u \frac{\partial R_v}{\partial u_n} \right) \left(\frac{\partial R_u}{\partial u_n}\frac{\partial R_v}{\partial v_n} - \frac{\partial R_v}{\partial u_n}\frac{\partial R_u}{\partial v_n} \right)^{-1}, \end{aligned} \tag{5.20}$$

where τ is the iteration parameter, which is chosen so that the grid remains convex. At each step, condition (5.18) is verified. If (5.18) is not satisfied, then τ is reduced by half and the grid at the $(l+1)^{\text{th}}$ step is re-computed. The derivation of the computational formulas for (5.20) suitable for computer programming is given in the Appendix.

In the case of the tool path generation, the gradient of the control function f corresponds to the machining error (scallop) between the tool tracks. Since the tool path is a discrete set of points the derivatives of the control function $f_u(u,v)$ and $f_v(u,v)$ for a given surface can not be explicitly evaluated. Therefore, these derivatives are generated "artificially" as follows. Initially, $(f_u)_n$ and $(f_v)_n$ are set to 0. Next,

$$\begin{aligned} (f_u)_n^{l+1} &= \begin{cases} (f_u)_n^l + \lambda_+ \text{ if } s_u(n) > 0, \\ (f_u)_n^l - \lambda_- \text{ otherwise}, \end{cases} \\ (f_v)_n^{l+1} &= \begin{cases} (f_v)_n^l + \lambda_+ \text{ if } s_v(n) > 0, \\ (f_v)_n^l - \lambda_- \text{ otherwise}, \end{cases} \end{aligned} \tag{5.21}$$

where λ_+ and λ_- are the prescribed increment and decrement, respectively, and $s_u(n)$ and $s_v(n)$ are the scallop estimates evaluated at the node n as follows:

$$\begin{aligned} s_u(n) &= \max_{d \in \{\text{left,right}\}} \left[D(n, N'(n,d)) - T(n, N'(n,d)) \right], \\ s_v(n) &= \max_{d \in \{\text{up,down}\}} \left[D(n, N'(n,d)) - T(n, N'(n,d)) \right]. \end{aligned} \tag{5.22}$$

where $D(n,m)$ is the distance between nodes n and m given by

$$D(n,m) = |S(u_n, v_n) - S(u_m, v_m)|. \tag{5.23}$$

$T(n,m)$ is an estimate of the machining strip width calculated at the midpoint $S(\frac{u_n+u_m}{2}, \frac{v_n+v_m}{2})$.

The grid generation algorithm consists of the following steps:

1. Generate the initial convex grid. The grid is generated manually or by interpolating. Note that interpolation may generate a grid with the nodes outside the boundary of the region. In that case, several iterations can be performed by the classic Brackbill's and Saltzman's method [15] which will move the nodes back inside the region. The initial grid should not satisfy the scallop constraints. If the grid satisfies the constraints for every node then the adaptation is not necessary. On the other hand, it means that the number of nodes has been overestimated and should be reduced. The initial grid size (l_r rows and l_c columns) can be estimated as follows

$$\begin{aligned} l_r &= \left\lceil \frac{W_v}{2r} \right\rceil, \\ l_c &= \left\lceil \frac{W_u}{2r} \right\rceil, \end{aligned} \tag{5.24}$$

where W_u and W_v are the surface dimensions in the u and v directions, respectively and r is the radius of the cutting tool.

2. Adapt the grid by minimizing the Dirichlet functional until a prescribed number of iterations has been performed.

3. Refine the grid by insering additional nodes. If the grid does not satisfy the scallop height constraint, more grid points are needed. For a structured grid having l_r rows and l_c columns, the numbers of rows and columns at the next step are estimated as follows

$$\begin{aligned} l_r^{\text{new}} &= l_r + \left\lceil \frac{\max_{1\le i\le l_r} \left(\sum_{j=1}^{l_c-1} \left[D\left(n_{i,j}, n_{i,j+1}\right) - T\left(n_{i,j}, n_{i,j+1}\right) \right] \right)}{2r} \right\rceil \\ l_c^{\text{new}} &= l_c + \left\lceil \frac{\max_{1\le j\le l_c} \left(\sum_{i=1}^{l_r-1} \left[D\left(n_{i,j}, n_{i+1,j}\right) - T\left(n_{i,j}, n_{i+1,j}\right) \right] \right)}{2r} \right\rceil, \end{aligned} \tag{5.25}$$

where $n_{i,j}$ is the grid node on row i and column j. For block-structured grid, the grid is first partitioned into smaller blocks of structured grids and the refinement is performed for each block. For column insertion, the grid is partitioned vertically as shown in Fig. 5.16(b) whereas for row insertion, the grid is partitioned horizontally as shown in Fig. 5.16(c) (see also Fig. 5.23 in Example 5.6).

4. Adapt the grid by minimizing the Dirichlet functional until all the grid points satisfy the scallop constraint or until a prescribed number of iterations has been performed.

5. If the scallop constraint has not been satisfied for some points, goto the refinement step 3.

The algorithm converges faster and exhibits more stability if the third term

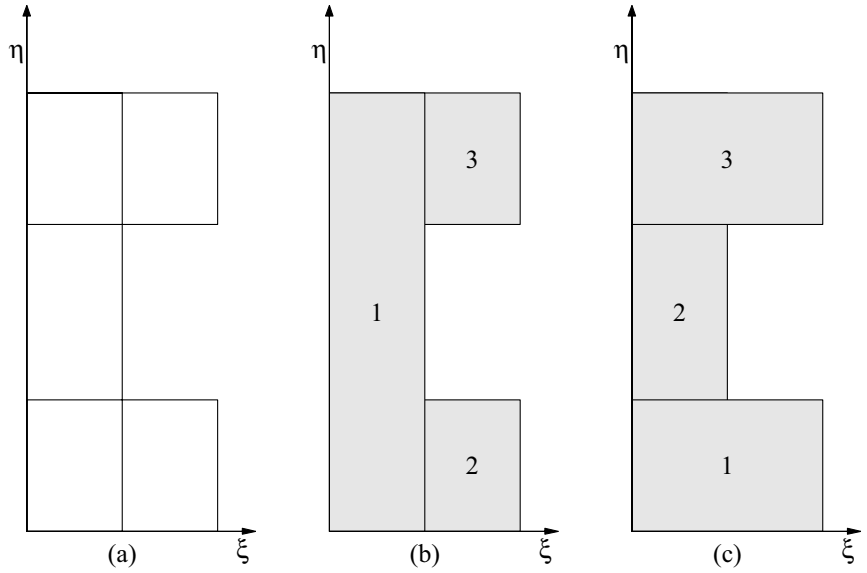


Fig. 5.16. Partitioning of block-structured grid for refinement

in the nominator of functional (5.12) is neglected. In order to interpret this term geometrically for each F_k in (5.16), consider two vectors from local node k to local nodes $k-1$ and $k+1$ given by

$$\begin{aligned} \mathbf{v}_{k,k-1} &= [u_k - u_{k-1}, v_k - v_{k-1}], \\ \mathbf{v}_{k,k+1} &= [u_k - u_{k+1}, v_k - v_{k+1}]. \end{aligned} \tag{5.26}$$

Clearly, $\mathbf{v}_{k,k-1} \cdot \mathbf{v}_{k,k+1} = (u_k - u_{k-1})(u_k - u_{k+1}) + (v_k - v_{k-1})(v_k - v_{k+1})$ $= D_3 = |\mathbf{v}_{k,k-1}||\mathbf{v}_{k,k+1}| \cos\vartheta$, where ϑ is the angle between the two vectors. Therefore

$$F_k = \frac{\frac{D_1}{c}\left[1 + (f_u)_k^2\right] + \frac{D_2}{c}\left[1 + (f_v)_k^2\right] + 2(f_u)_k(f_v)_k \cos\vartheta}{\frac{J_k}{c}\left[1 + (f_u)_k^2 + (f_v)_k^2\right]^{\frac{1}{2}}}, \tag{5.27}$$

where $c = |\mathbf{v}_{k,k-1}||\mathbf{v}_{k,k+1}|$. Therefore, omitting the last term leads to an adaptation with less impact on the angles between coordinate lines. Finally, recall that convexity of the cells is provided by an appropriate τ in (6).

5.5 Space-Filling Curve Generation on Block Structured Grid

Space-filling curve tool paths on block-structured grids can be generated by imposing one extra requirement on the grid generation algorithm presented

in the previous section. At each refinement stage, the size of any sub-block of the grid partitioned vertically or horizontally must be even. This is a sufficient condition for the existing of Hamiltonian path, i.e., the space-filling curve on the block structured grid. The proof of this claim is quite straightforward since all nodes in the blocks of the even size can all be covered by small rectangular circuits. The merging stage of the space-filling curve generation algorithm then merges all small circuits into one bigger circuit. Figure 5.17 illustrates the space-filling curve generation on block structured grid. First, a vertex is placed at the center of each grid cell (Fig. 5.17(b)). Each vertex is then connected to the vertices of the neighbor cells (Fig. 5.17(c)). Finally, small circuits are created by constructing small rectangular cyclic paths over every 4 adjacent vertices, i.e, by connecting the vertices on even rows and columns with the vertices on odd rows and columns, respectively, as shown in Fig. 5.18. Merging of small circuits to create the space-filling curve is then performed (see Sect. 4.3.2).

The combination of space-filling curve and the grid generation techniques allows construction of tool paths on surfaces with complex irregular boundaries, cuts off, pockets, islands, etc. Examples are given in the next section.

5.6 Examples and Discussion

This section presents a variety of examples that demonstrate the efficiency and advantages of the use of curvilinear grid in tool path generation.

Example 5.5. This example demonstrates the use of variational grid generation technique to overcome the inefficiency of tool path generation by the conventional isoparametric method. Consider a unimodal surface with an exponential peak along a line in the parametric domain (u, v). The surface is given by (see Fig. 5.19)

$$\begin{aligned} x &= 100u - 50, \\ y &= 100v - 50, \\ z &= 10\exp(-40(2u - 0.5 - v)^2) - 15. \end{aligned} \tag{5.28}$$

A curvilinear grid constructed for a flat-end tool with radius 3 mm and the surface tolerance of 0.1 mm is shown in Fig. 5.20. The comparison of the zigzag and SFC tool paths generated by traditional isoparametric method and curvilinear grid method is presented in Table 5.4. The zigzag tool path based on the adaptive grid is shown in Fig. 5.21(c). The length of the tool path is 2558.28 mm. The SFC tool path requires a basic (isoparametric) grid with a small spatial step. Consequently it leads to a SFC of about 3066.87 mm (Fig. 5.21(b)). Finally, the curvilinear grid adapted to the zones where the small distance between the tracks is required leads to a shorter path of about 2519.88 mm (Fig. 5.21(d)). The length of the zigzag and SFC tool paths

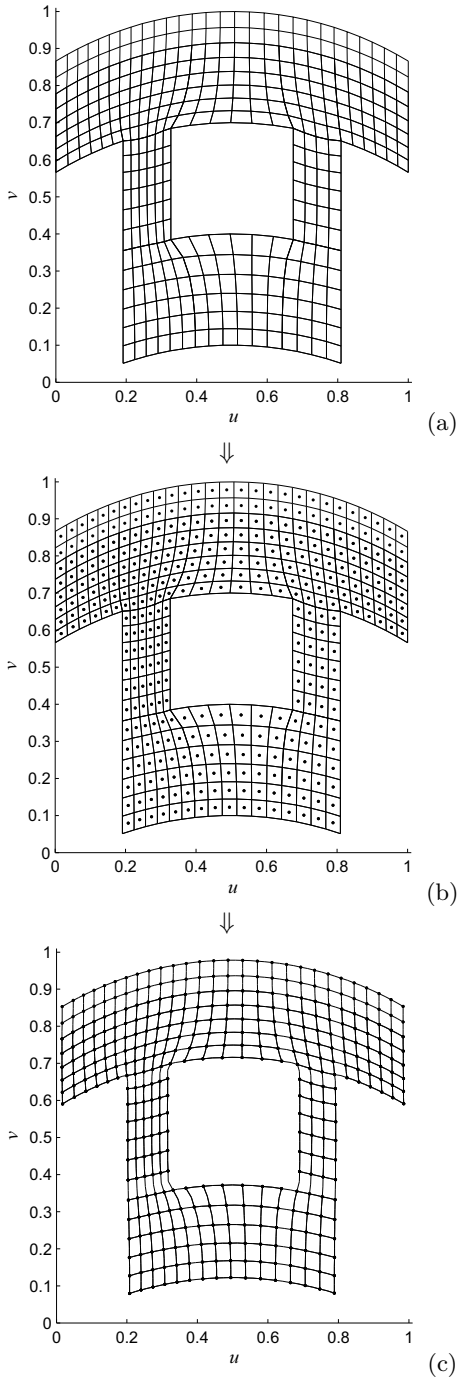


Fig. 5.17. Undirected graph construction for SFC tool path generation

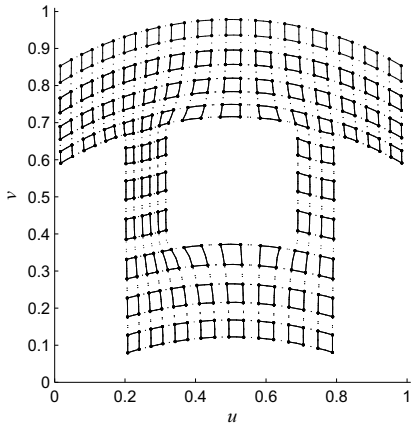


Fig. 5.18. Covering of grid in Fig. 5.17 by small circuits

based on the adaptive grid are shorter by 45.76% and 17.84%, respectively, when compared with the zigzag and SFC based on isoparametric tool path method (Fig. 5.21).

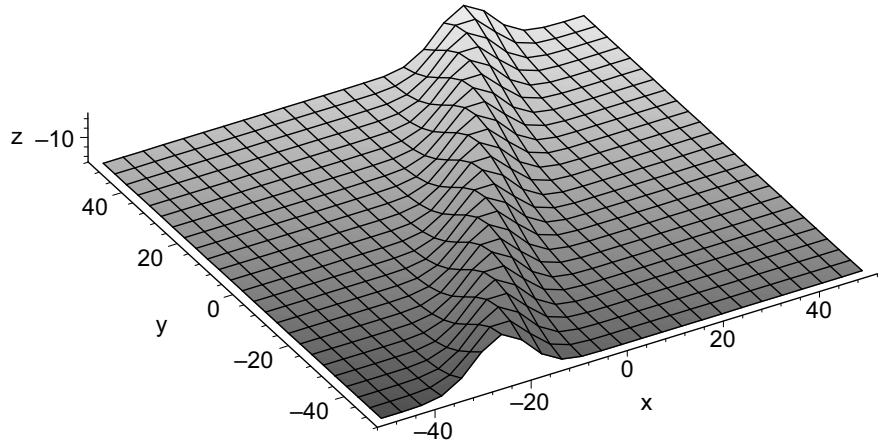
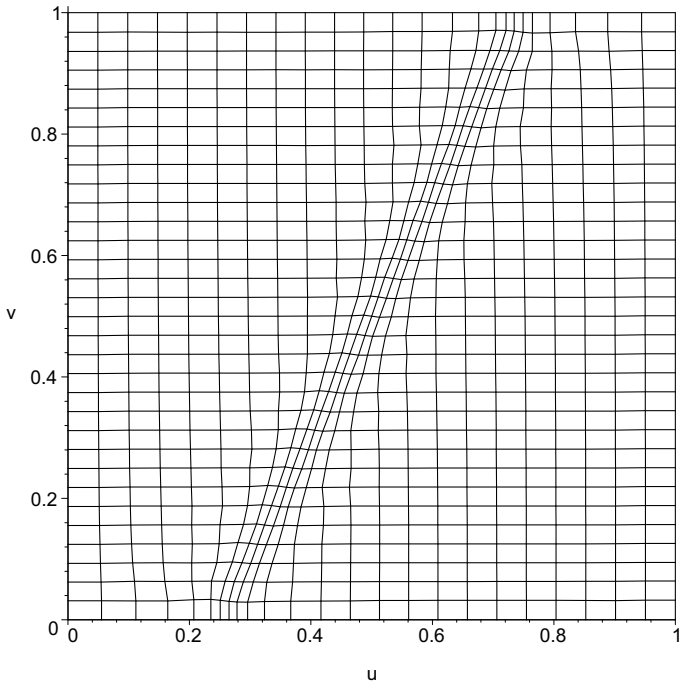
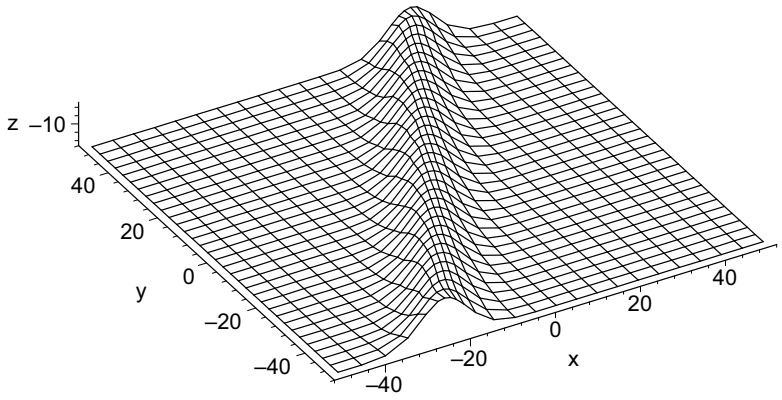


Fig. 5.19. Unimodal surface with exponential peak along a line

The SFC tool path is better than the standard zigzag path since it turns in the optimal direction, that is, in the direction which maximizes the machining strip. Furthermore, a step between two adjusted tracks on the rectangular grid is set to the minimal step ensuring the required scallop along the two tracks. The curvilinear grid makes it possible to vary the distance between the tracks making the step small only where necessary. Therefore, the SFC constructed within a new basic grid leads to better results.

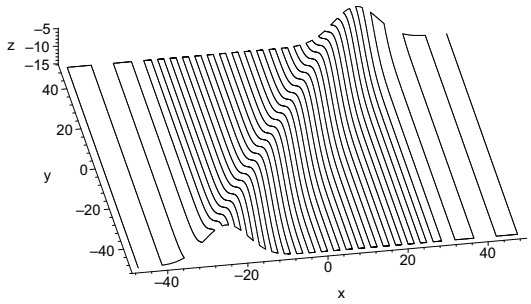


(a)

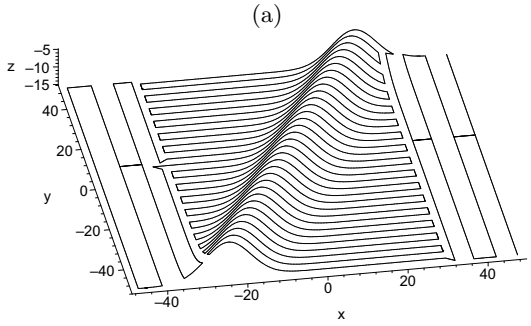


(b)

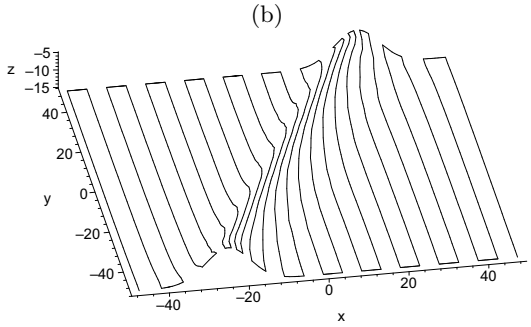
Fig. 5.20. Curvilinear grid adapted to the unimodal surface with exponential peak along a line in (u, v) domain (**a**) and workpiece coordinate system (**b**)



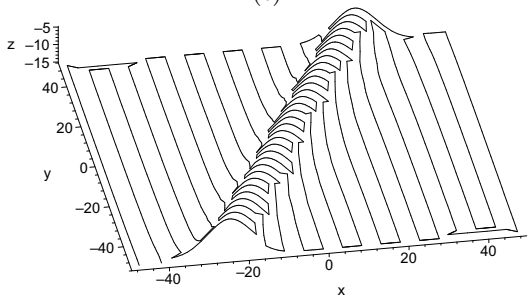
(a)



(b)



(c)



(d)

Fig. 5.21. (**a**) isoparametric tool path, (**b**) SFC tool path from two isoparametric tool paths, (**c**) zigzag tool path from adaptive grid, (**d**) SFC tool path from adaptive grid

Table 5.4. Comparison of tool paths based on variational grid generation techniques versus the isoparametric tool path scheme

	Total path length (mm)	
Method used for constructing grid	Vertical zigzag	SFC
Overlaying of two isoparametric tool paths	4716.91	3066.87
Curvilinear grid generation	2557.74	2508.25

The plots of convergence rates are shown in Fig. 5.22. Note that the convergence rate of the algorithm increases a little when the third term of the numerator in Direchlet functional (5.12) is dropped. Although, the improvement in convergence rate is not evidently large, the complexity of the algorithm is reduced by eliminating this third term. Consequently, the grid requires less amount of time to compute.

Example 5.6. This example demonstrates the use of curvilinear grid and SFC to construct tool paths to machine surfaces with complex irregular boundaries, cuts off, and islands. Consider a two-bell surface given by (4.8) in Example 4.3. The surface is to be machined only in the area within the boundaries shown in Fig. 5.24(a) in (u, v) domain. The initial grid and the first refined grid are depicted in Fig. 5.23. The final grid and SFC tool path are shown in Figs. 5.24 and 5.25, respectively. The simulaton of the cutting in Unigraphics 18 for a flat-end tool with radius 3 mm and the surface tolerance of 0.05 mm is shown in Fig. 5.26. The plots of convergence rates are shown in Fig. 5.27. The convergence rate of the algorithm also increases a little when the third term of the numerator in Direchlet functional (5.12) is dropped.

Example 5.7. Consider a single blade of an impeller depicted in Fig. 5.28(a). This single blade is described by parametric equations. When the blade is broken (see Fig. 5.28(b)), it is sometimes possible to fill the broken blade with a new material and mill only the filled part. Since this part of the blade contains irregular boundaries in (u, v) domain, the traditional isoparametric method cannot be used to construct the tool path. Applying the grid generation method on this parametric surface with complex irregular boundaries, the final curvilinear grid adapted to the shape of the broken blade is shown in Fig. 5.29 (parametric domain) and Fig. 5.30 (workpiece coordinates). The grid is adapted to the surface that is to be milled using ball-end tool of radius 5 mm and machined surface tolerance of 0.05 mm. The calculation of machining strip width for ball-end mill is given in Sect. 3.2. The SFC tool path is shown in Fig. 5.31 (parametric domain) and Fig. 5.32 (workpiece coordinates). The virtual cutting with the SFC tool path is shown in Fig. 5.33.

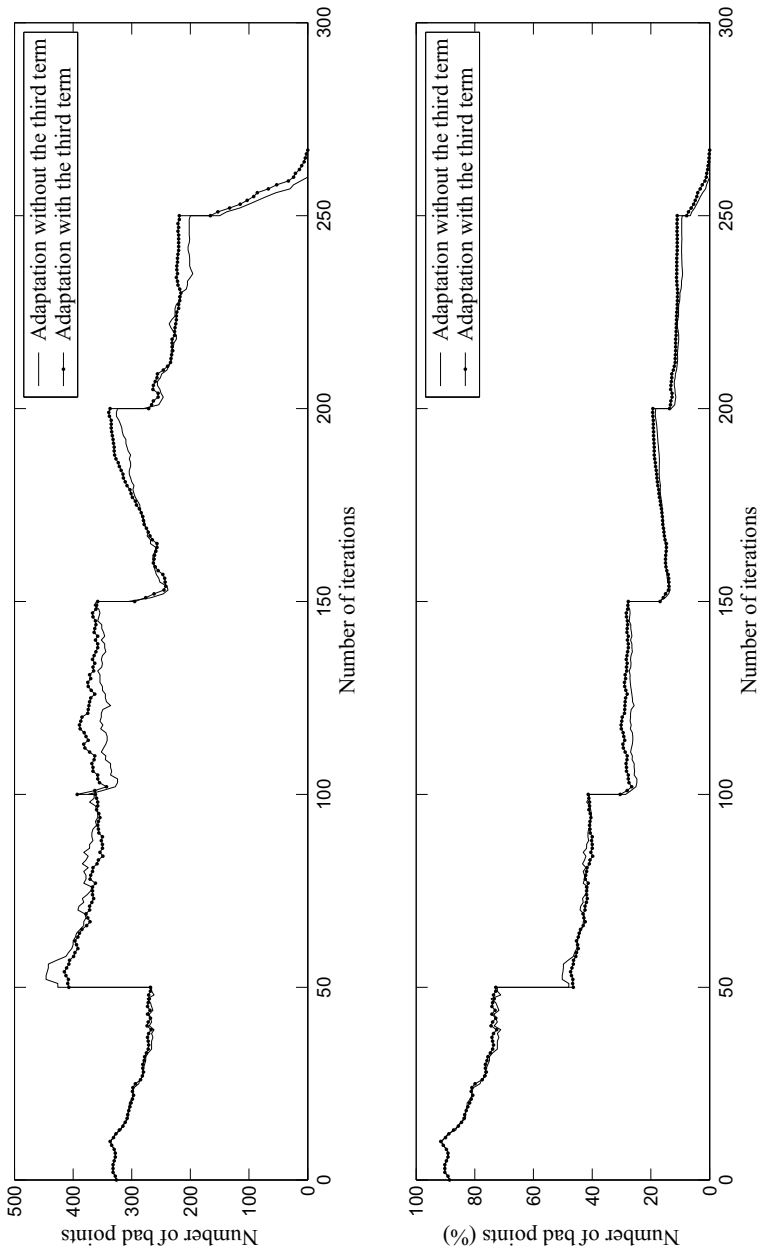
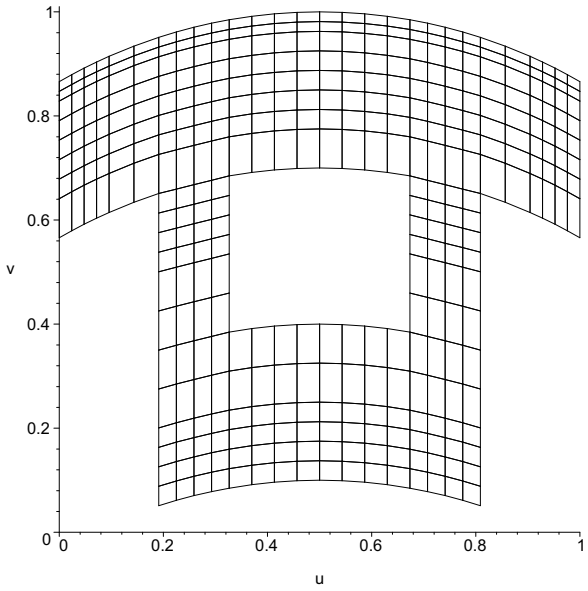
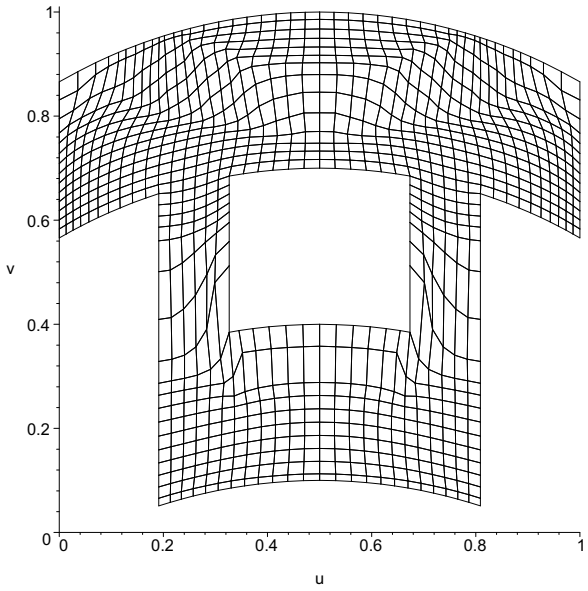


Fig. 5.22. Convergence rate of the grid generation technique in term of the number of iterations for Example 5.4



(a)



(b)

Fig. 5.23. Grid refinement: (**a**) initial grid, (**b**) grid after first refinement

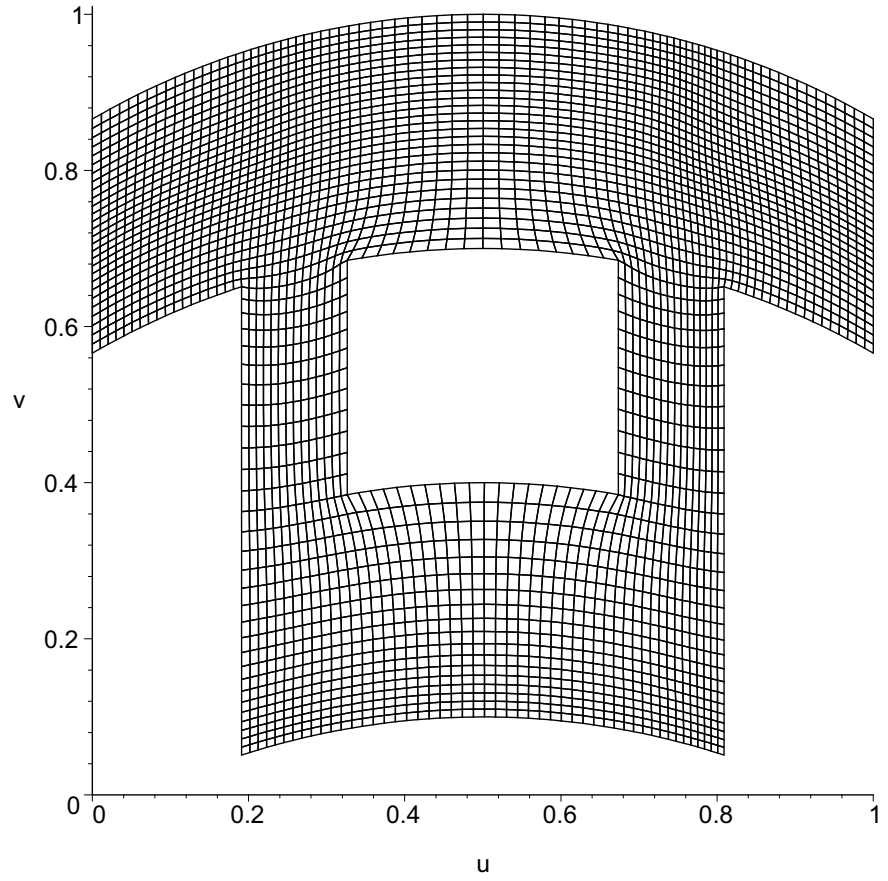


Fig. 5.24a. Curvilinear grid adapted to the surface in Example 5.6 in (u, v) domain

Appendix: Derivation of Computational Formulas for Adaptive-Harmonic Grid Generation

The minimization of the discrete functional (5.16) requires computation of F_k and its derivatives. Each computation of F_k involves three grid nodes on the triangle $\triangle_{k-1,k,k+1}$. For conciseness, node $k-1$, k and $k+1$ are denoted by a, b and c, respectively. F_k is denoted by F and is now expressed as

$$F = \frac{X}{Y}, \tag{5.29}$$

where

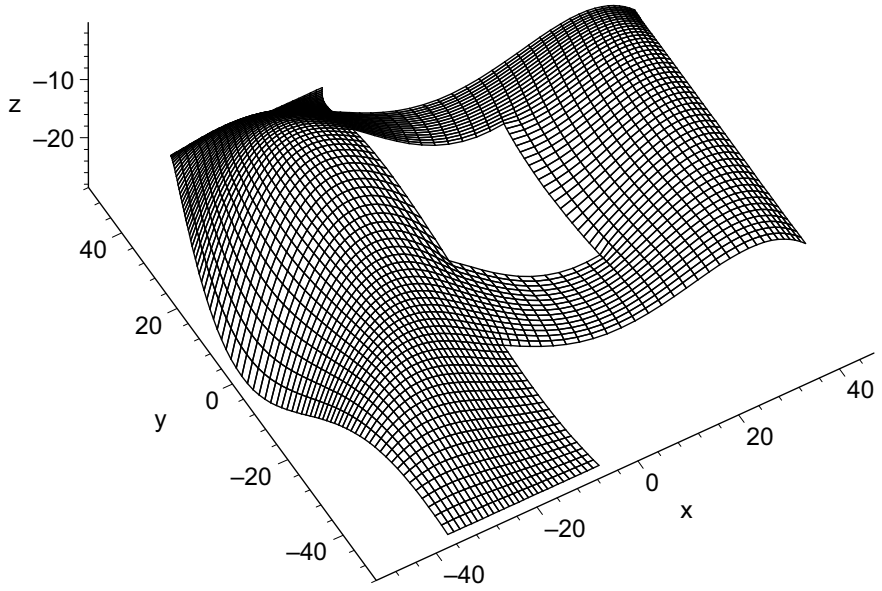


Fig. 5.24b. Curvilinear grid adapted to the surface in Example 5.6 in workpiece coordinate system

$$
\begin{aligned}
X &= \frac{D_1\left[1+(f_u)_k^2\right]+D_2\left[1+(f_v)_k^2\right]+2D_3(f_u)_k(f_v)_k}{\left[1+(f_u)_k^2+(f_v)_k^2\right]^{\frac{1}{2}}},\\
Y &= (u_a-u_b)(v_c-v_b)-(u_c-u_b)(v_a-v_b),\\
D_1 &= (u_a-u_b)^2+(u_c-u_b)^2,\\
D_2 &= (v_a-v_b)^2+(v_c-v_b)^2,\\
D_3 &= (u_a-u_b)(v_a-v_b)+(u_c-u_b)(v_c-v_b).
\end{aligned}
\tag{5.30}
$$

The formulas for the derivatives of F can then be expressed as the derivatives of two functions X and Y and are given below

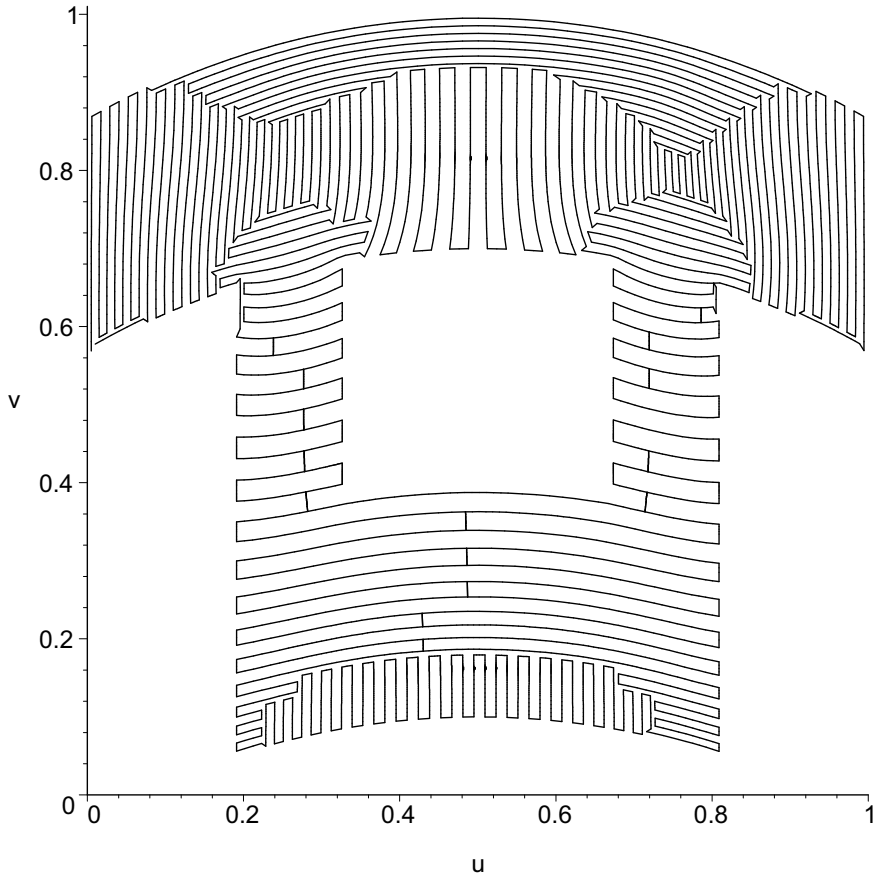


Fig. 5.25a. SFC tool path for the surface in Example 5.6 in (u, v) domain

$$
\begin{aligned}
F_u &= \frac{X_u Y - X Y_u}{Y^2} = \frac{X_u - F Y_u}{Y}, \\
F_v &= \frac{X_v Y - X Y_v}{Y^2} = \frac{X_v - F Y_v}{Y}, \\
F_{uu} &= \frac{(X_{uu} - F_u Y_u - F Y_{uu}) Y - (X_u - F Y_u) Y_u}{Y^2}, \\
&= \frac{X_{uu} - 2 F_u Y_u - F Y_{uu}}{Y}, \\
F_{vv} &= \frac{(X_{vv} - F_v Y_v - F Y_{vv}) Y - (X_v - F Y_v) Y_v}{Y^2}, \\
&= \frac{X_{vv} - 2 F_v Y_v - F Y_{vv}}{Y}, \\
F_{uv} &= \frac{(X_{uv} - F_v Y_u - F Y_{uv}) Y - (X_u - F Y_u) Y_v}{Y^2}, \\
&= \frac{X_{uv} - F_v Y_u - F_u Y_v - F Y_{uv}}{Y}.
\end{aligned}
\tag{5.31}
$$

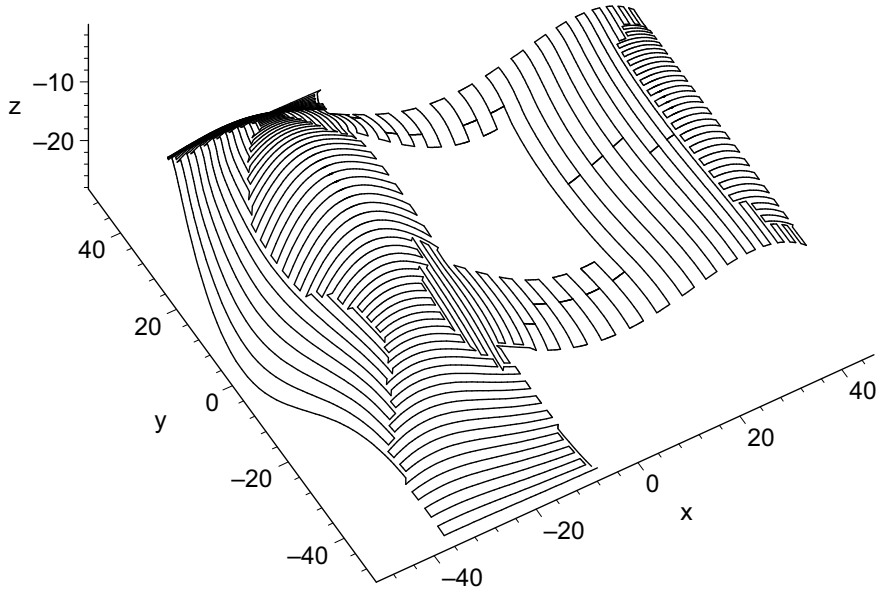


Fig. 5.25b. SFC tool path for the surface in Example 5.6 in workpiece coordinate system

The derivatives at node a are obtained by substituting u by u_a and v by v_a and are given by

$$
\begin{aligned}
Y_u = v_c - v_b, \quad Y_v = u_b - u_c, \quad Y_{uu} = 0, \quad Y_{vv} = 0, \quad Y_{uv} = 0,\\
X_u = 2\frac{\left[1+(f_u)_k^2\right](u_a-u_b)+(f_u)_k(f_v)_k(v_a-v_b)}{[1+(f_u)_k^2+(f_v)_k^2]^{\frac{1}{2}}},\\
X_v = 2\frac{\left[1+(f_v)_k^2\right](v_a-v_b)+(f_u)_k(f_v)_k(u_a-u_b)}{[1+(f_u)_k^2+(f_v)_k^2]^{\frac{1}{2}}},\\
X_{uu} = 2\frac{1+(f_u)_k^2}{[1+(f_u)_k^2+(f_v)_k^2]^{\frac{1}{2}}}, \quad X_{vv} = 2\frac{1+(f_v)_k^2}{[1+(f_u)_k^2+(f_v)_k^2]^{\frac{1}{2}}},\\
X_{uv} = 2\frac{(f_u)_k(f_v)_k}{[1+(f_u)_k^2+(f_v)_k^2]^{\frac{1}{2}}}.
\end{aligned}
\tag{5.32}
$$

The derivatives at node b are obtained by substituting u by u_b and v by v_b and are given by

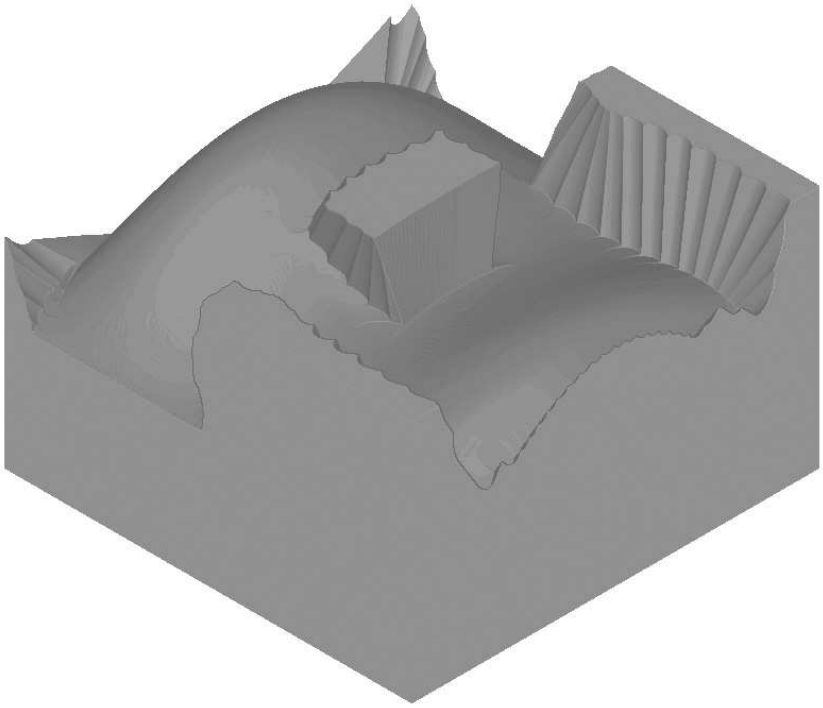


Fig. 5.26. Simulation result of five-axis machining with SFC tool path in Unigraphics 18 for the surface in Example 5.6

$$
\begin{aligned}
&Y_u = v_a - v_c, \quad Y_v = u_c - u_a, \quad Y_{uu} = 0, \quad Y_{vv} = 0, \quad Y_{uv} = 0,\\
&X_u = 2\frac{\left[1 + (f_u)_k^2\right](2u_b - u_a - u_c) + (f_u)_k(f_v)_k(2v_b - v_a - v_c)}{[1 + (f_u)_k^2 + (f_v)_k^2]^{\frac{1}{2}}},\\
&X_v = 2\frac{\left[1 + (f_v)_k^2\right](2v_b - v_a - v_c) + (f_u)_k(f_v)_k(2u_b - u_a - u_c)}{[1 + (f_u)_k^2 + (f_v)_k^2]^{\frac{1}{2}}},\\
&X_{uu} = 4\frac{1 + (f_u)_k^2}{[1 + (f_u)_k^2 + (f_v)_k^2]^{\frac{1}{2}}}, \quad X_{vv} = 4\frac{1 + (f_v)_k^2}{[1 + (f_u)_k^2 + (f_v)_k^2]^{\frac{1}{2}}},\\
&X_{uv} = 4\frac{(f_u)_k(f_v)_k}{[1 + (f_u)_k^2 + (f_v)_k^2]^{\frac{1}{2}}}.
\end{aligned}
\tag{5.33}
$$

Finally, the derivatives at node c are obtained by substituting u by u_c and v by v_c and are given by

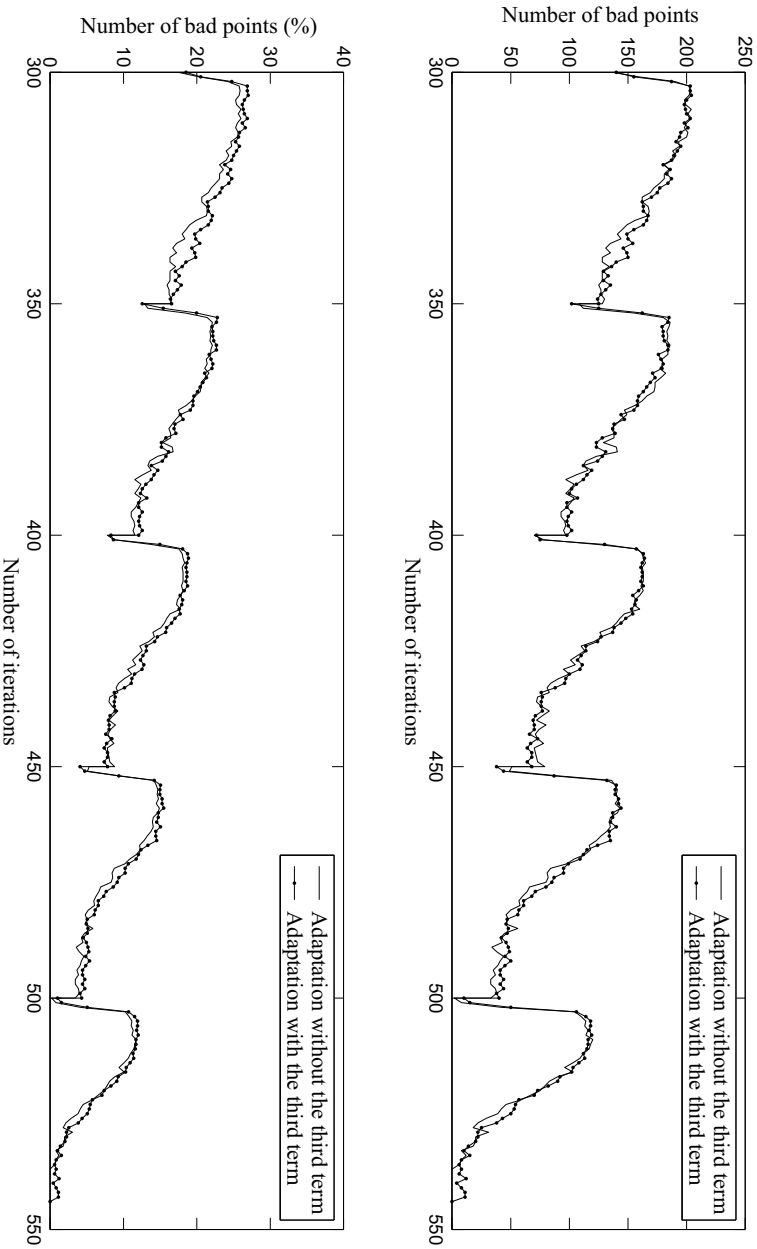
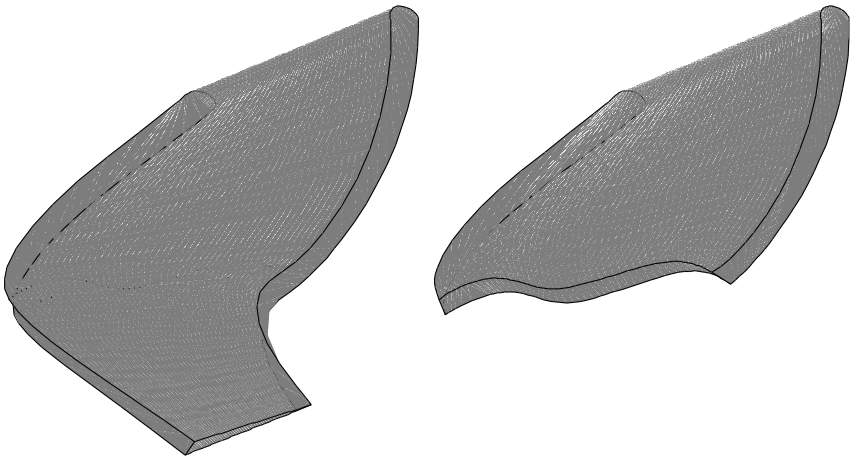


Fig. 5.27. Convergence rate of the grid generation technique in term of the number of iterations for Example 5.6



(a) (b)

Fig. 5.28. (**a**) a full single blade of an impeller, (**b**) part of a single blade

$$
\begin{gathered}
Y_u = v_b - v_a, \quad Y_v = u_a - u_b, \quad Y_{uu} = 0, \quad Y_{vv} = 0, \quad Y_{uv} = 0, \\
X_u = 2\frac{\left[1 + (f_u)_k^2\right](u_c - u_b) + (f_u)_k(f_v)_k(v_c - v_b)}{[1 + (f_u)_k^2 + (f_v)_k^2]^{\frac{1}{2}}}, \\
X_v = 2\frac{\left[1 + (f_v)_k^2\right](v_c - v_b) + (f_u)_k(f_v)_k(u_c - u_b)}{[1 + (f_u)_k^2 + (f_v)_k^2]^{\frac{1}{2}}}, \\
X_{uu} = 2\frac{1 + (f_u)_k^2}{[1 + (f_u)_k^2 + (f_v)_k^2]^{\frac{1}{2}}}, \quad X_{vv} = 2\frac{1 + (f_v)_k^2}{[1 + (f_u)_k^2 + (f_v)_k^2]^{\frac{1}{2}}}, \\
X_{uv} = 2\frac{(f_u)_k(f_v)_k}{[1 + (f_u)_k^2 + (f_v)_k^2]^{\frac{1}{2}}}.
\end{gathered}
\tag{5.34}
$$

For the cell number N, $N \in \{1, \ldots, N_e\}$ and triangle number k, $k \in \{1, 2, 3, 4\}$, the values of F and its derivatives on u_a and v_a are computed with the use of (5.31) and (5.32). The computed values are then added to the appropriate array elements

$$
\begin{gathered}
I^h_{\text{new}} = I^h_{\text{old}} + F, \\
[R_u]_{n,\text{new}} = [R_u]_{n,\text{old}} + F_u, \\
[R_v]_{n,\text{new}} = [R_v]_{n,\text{old}} + F_v, \\
[R_{uu}]_{n,\text{new}} = [R_{uu}]_{n,\text{old}} + F_{uu}, \\
[R_{vv}]_{n,\text{new}} = [R_{vv}]_{n,\text{old}} + F_{vv}, \\
[R_{uv}]_{n,\text{new}} = [R_{uv}]_{n,\text{old}} + F_{uv},
\end{gathered}
\tag{5.35}
$$

where $n = C_O(N, k-1)$.

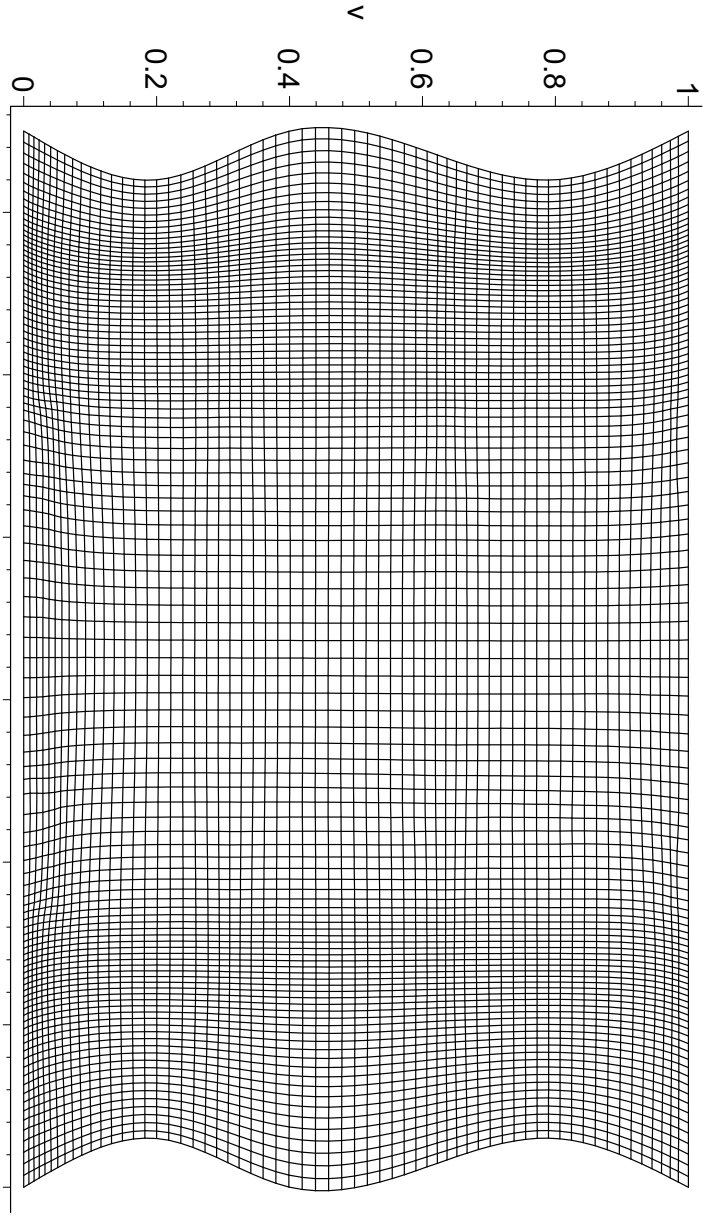


Fig. 5.29. Curvilinear grid adapted to part of a surface of the blade in (u, v) domain

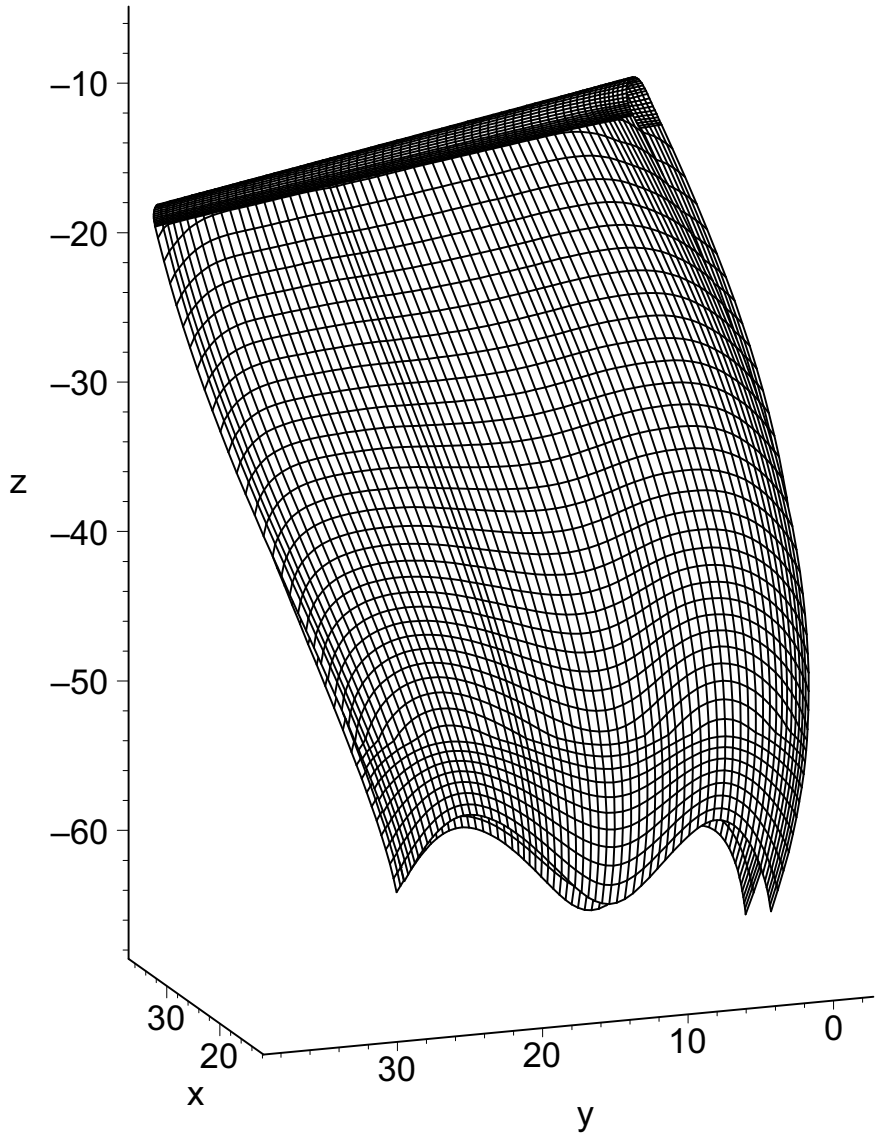


Fig. 5.30. Curvilinear grid adapted to part of a surface of the blade in workpiece coordinate system

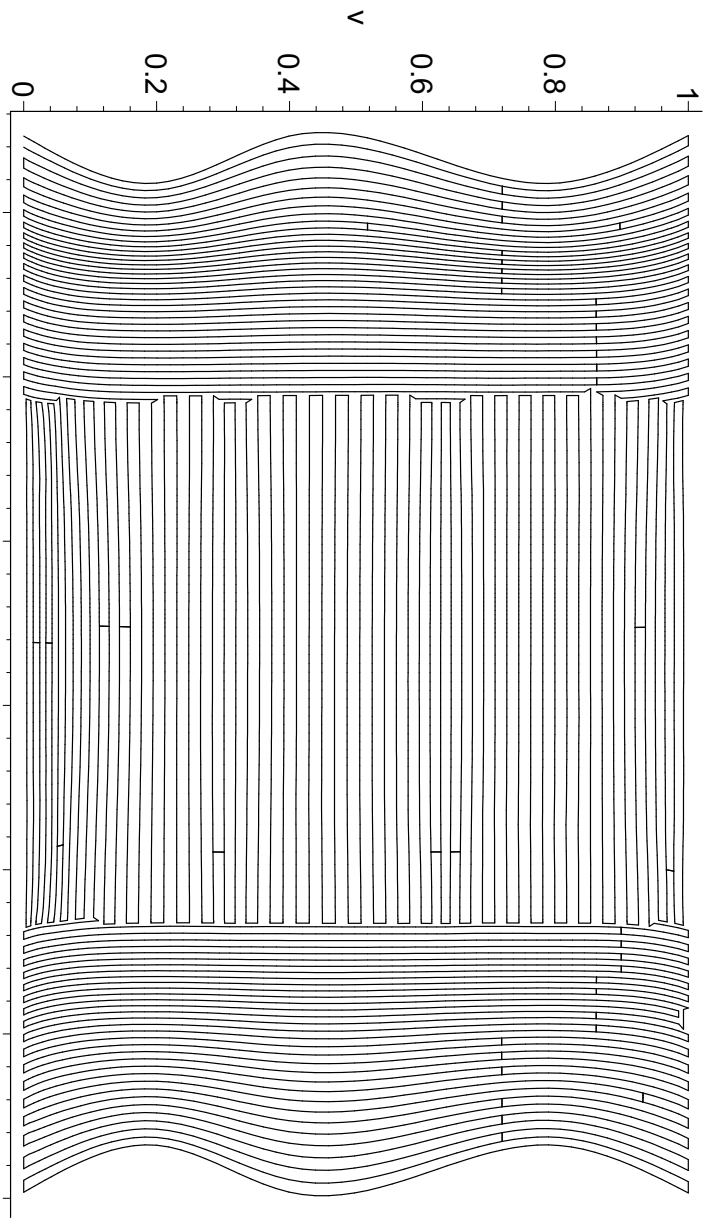


Fig. 5.31. SFC tool path for milling of the broken blade in (u, v) domain

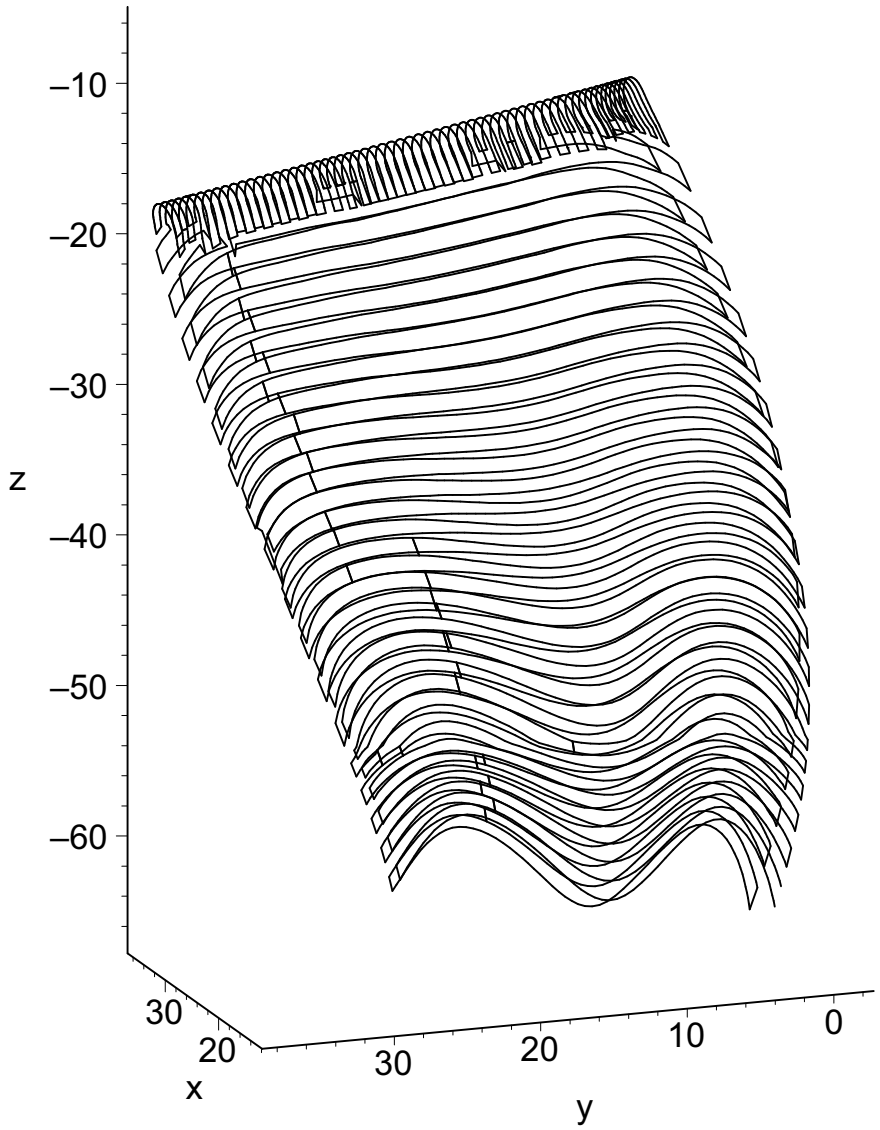


Fig. 5.32. SFC tool path for milling of the broken blade in workpiece coordinate system

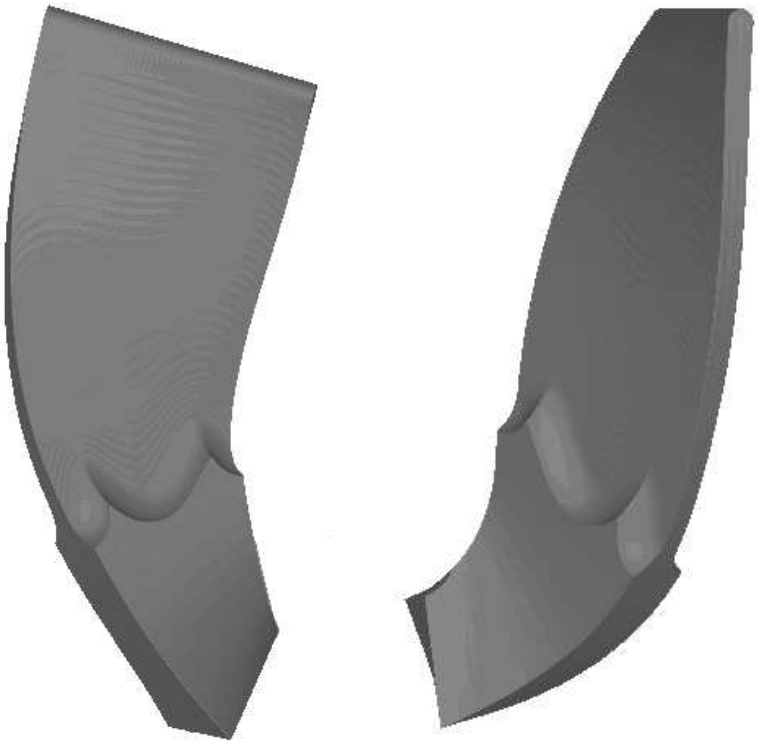


Fig. 5.33. Virtual cutting of the blade in Unigraphics 18

Similarly for node b, the correspondence between local and global node number is given by $n = C_O(N, k)$ and the values of F and its derivatives on u_b and v_b are computed with the use of (5.31) and (5.33). Finally, for node c, the correspondence between local and global node number is given by $n = C_O(N, k + 1)$ and the values of F and its derivatives on u_c and v_c are computed with the use of (5.31) and (5.34).

References

[1] Amazigo, J. C. and Rubenfeld, L. A. 1980. *Advanced Calculus and Its Applications to the Engineering and Physical Sciences.* John Wiley & Sons, Inc., New York, USA.

[2] Anderson, D. and Ray, M. 1982. The use of solution adaptive grids in solving partial differential equations. *Applied Mathematics and Computation*, 10-11:317–338.

[3] Anderson, D. A. 1987. Equidistribution schemes, poisson generators, and adaptive grids. *Applied Mathematics and Computation*, 24(3):211–227.

[4] Arina, R. and Tarditi, S. 1994. Orthogonal block structured surface grids. In Baines, M. J. and Morton, K. W., editors, *Numerical Methods for Fluid Dynamics IV*, pages 291–298. Oxford University Press.

[5] Babuska, I. and Rheinboldt, W. 1979. A-posteriori error estimates for the finite element method. *International Journal for Numerical Methods in Engineering*, 12:1597–1615.

[6] Bahvalov, S. 1969. On optimization of methods for solving boundary problems with boundary layers. *Computational Mathematics and Mathematical Physics*, 9(4):841–859.

[7] Baker, T. 1989. Developments and trends in three-dimensional mesh generation. *Applied Numerical Mathematics*, 5(4):275–304.

[8] Berger, M. and Oliger, J. 1984. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53(3):484–512.

[9] Bieterman, M. B. and Sandstrom, D. R. 2003. A curvilinear tool-path method for pocket machining. *Journal of Materials Processing Technology*, 125(4):709–715.

[10] Blacker, T. and R.J., M. 1993. Seams and wedges in plastering: A 3D hexahedral mesh generation algorithm. *Engineering with Computers*, 2(9):83–93.

[11] Blacker, T. and Stephenson, M. B. 1991. Paving : a new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 32(4):811–847.

[12] Boender, E. 1994. Reliable Delaunay-based mesh generation and mesh improvement. *Communications in Numerical Methods in Engineering*, 10(10):773–783.

[13] Brackbill, J. U. 1993. An adaptive grid with directional control. *Journal of Computational Physics*, 108(1):38–50.

[14] Brackbill, J. U., B., K. D., and M., R. H. 1988. FLIP: A low-dissipation, particle-in-cell method for fluid flow. *Computational Physics Communications*, 48(1):25–38.

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

[16] Carey, G. 1997. *Computational Grids: Generations, Adaptation & Solution Strategies*. Taylor & Francis, London.

[17] Carey, G. and Dinh, H. 1985. Grading function and mesh redistribution. *SIAM Journal of Numerical Analysis*, 22(3):1028–1040.

[18] Charakhch'yan, A. 1994. Compound difference schemes for time-dependent equations on non-uniform meshes. *Communications in Numerical Methods in Engineering*, 10(2):93–110.

[19] Charakhch'yan, A. A. and Ivanenko, S. A. 1997. A variational form of the Winslow grid generator. *Journal of Computational Physics*, 136(2):385–398.

[20] Dannenhoffer, J. 1988. A comparison of two adaptive grid techniques. In *Numerical Grid Generation in Computational. Fluid Mechanics '88*, pages 319–328.

[21] Dar'in, N. and Majukin, V. . 1988. On one approach to adaptive grid generation in time-dependent problems. *Computational Mathematics and Mathematical Physics*, 28(3):454–460.

[22] De Boor, C. 1974. Good approximation by splines with variable knots II. In Watson, G. A., editor, *Lecture Notes in Mathematics*, volume 363, pages 12–20. Springer-Verlag, Berlin.

[23] Degtyarev, L. and Ivanova, T. 1993. Adaptive grid method in one-dimensional nonstationary convection - diffusion problems. *Differential Equations*, 29(7):1179–1192.

[24] Diaz, A., Kikuchi, N., and Taylor, J. 1983. A method of grid optimization for finite element method. *Computational Methods in Applied Mechanics and Engineering*, 41(1):29–45.

[25] Dvinsky, A. 1991. Adaptive grid generation from harmonic maps on riemanian manifolds. *Journal of Computational Physics*, 95(3):450–476.

[26] Eells, J. and Sampson, J. 1964. Harmonic mappings of riemannian manifolds. *American Journal of Mathematics*, 86(1):109–160.

[27] Eiseman, P. 1987. Adaptive grid generation. *Computational Methods in Applied Mechanics and Engineering*, 64(1-3):321–37664.

[28] Emelyanov, K. 1994. Optimal grid generation and its application in singular problems. *Computational Mathematics and Mathematical Physics*, 34(6):936–943.

[29] Farrashkhalvat, M. and Miles, J. 2003. *Basic Structured Grid Generation: With an Introduction to Unstructured Grid Generation*. Elsevier.

[30] Frey, W. and Field, D. 1991. Mesh relaxation: A new technique for improving triangulation. *International Journal for Numerical Methods in Engineering*, 31(6):1121–1133.

[31] George, P. 1991. *Automatic Mesh Generation: Application to Finite Element Methods*. John Wiley and Sons, New York.

[32] Godunov, S. and Prokopov, G. 1967. On computing conformal mappings and finite-difference grid generation. *Computational Mathematics and Mathematical Physics*, 7(5):1031–1059.

[33] Godunov, S. and Prokopov, G. 1972. The use of moving meshes in gas-dynamics calculations. *Computational Mathematics and Mathematical Physics*, 12(2):182–194.

[34] Golias, N. and Tsiboukis, T. 1994. An approach to refining three-dimensional tetrahedral meshes based on Delauney transformations. *International Journal for Numerical Methods in Engineering*, 37:793–812.

[35] Grebennikov, A. 1976. On the choice of nodes in spline approximation. *Computational Mathematics and Mathematical Physics*, 16(1):219–223.

[36] Ivanenko, S. 1995. Adaptive-harmonic grid generation and its application for numerical solution of the problems with boundary and interior layers. *Computational Mathematics and Mathematical Physics*, 35(10):1203–1220.

[37] Ivanenko, S. A. 1988. Generation of non-degenerate meshes. *Computational Mathematics and Mathematical Physics*, 28(5):141–146.

[38] Ivanenko, S. A. 1993. Adaptive grids and grids on surfaces. *Computational Mathematics and Mathematical Physics*, 33(9):1179–1193.

[39] Ivanenko, S. A. 1999. Harmonic mappings. In *Handbook of Grid Generation*, chapter 8, pages 8(1–43). CRC Press, Inc., Boca Raton, FL, USA.

[40] Jaquotte, O.-P. 1988. A mechanical model for a new grid generation method in computational fluid dynamics. *Computational Mathematics and Applied Mechanics in Engineering*, 66:323–338.

[41] Jeong, J. and Kim, K. 1999. Generation of tool paths for machining free-form pockets with islands using distance maps. *International Journal of Advanced Manufacturing Technology*, 15(5):311–316.

[42] J.F., T., Soni, B., and Weatherill, N. 1999. Handbook of grid generation. *CRC Press*.

[43] Jiang, B. and Carey, G. 1987. Adaptive refinement for least squares finite elements with element-by-element conjugate gradient solution. *International Journal for Numerical Methods in Engineering*, 24(3):569–580.

[44] Kennon, S. and Anderson, D. 1988. Unstructured grid adaption for non-convex domains. In *Proceedings of the Conference on Numerical Grid Generation in Computational Fluid Mechanics*, pages 599–609.

[45] Kennon, S. and Dulikravich, G. 1986. Generation of computational grids using optimization. *AIAA Journal*, 24(7):1069–1073.

[46] Liseikin, V. 2006. *A Computational Differential Geometry Approach to Grid Generation*. Springer,, Berlin, New York.

[47] Liseikin, V. D. 1999. *Grid Generation Methods*. Springer-Verlag, Berlin, Germany.

[48] Lo, C. C. 1999. Efficient cutter-path planning for five-axis surface machining with a flat-end cutter. *Computer-Aided Design*, 31(9):557–566.

[49] Makhanov, S. S. 1999. An application of variational grid generation techniques to the tool-path optimization of industrial milling robots. *Journal of Materials Processing Technology*, 39(9):1524–1535.

[50] Makhanov, S. S., Batanov, D., Bohez, E., Sonthipaumpoon, K., Anotaipaiboon, W., and Tabucanon, M. 2002. On the tool-path optimization of a milling robot. *Computers & Industrial Engineering*, 43(3):455–472.

[51] Makhanov, S. S. and Ivanenko, S. A. 2003. Grid generation as applied to optimize cutting operations of the five-axis milling machine. *Applied Numerical Mathematics*, 46(3-4):331–351.

[52] Muller, K. 1981. Moving finite elements II. *SIAM Journal of Numerical Analysis*, 18(6):1033–1057.

[53] Muller, K. and Muller, R. 1981. Moving finite elements I. *SIAM Journal of Numerical Analysis*, 18(6):1019–1032.

[54] Nakahashi, K. and Deiwert, G. S. 1986. Three-dimensional adaptive grid method. *AIAA Journal*, 24(6):948–045.

[55] Oden, J., Demkowicz, L., Strouboulis, T., and Devlow, P. 1986. Adaptive methods for problems in solid and fluid mechanics. In Babuska, I.,

editor, *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, pages 249 – 280. John Willey.

[56] Pardhanani, A. and Carey, G. 1988. Optimization of computational grids. *Numerical Methods for Partial Differential Equations*, 4(2):95–117.

[57] P.D., T. 1982. Composite three dimensional grids generated by elliptic systems. *AIAA Journal*, 20(9):1195–1202.

[58] Pereira, V. and Sewell, E. 1975. Mesh selection for discrete solutions of boundary value problems in ordinary differential equations. *Numerical Mathematics*, 23(3):261–268.

[59] Prokopov, G. 1970. Construction of orthogonal difference grids from conformal mappings. In *Proceedings N45, Keldysh Institute of Applied Mathematics*, Moscow (in Russian).

[60] Rank, E. and Babushka, I. 1987. An expert system for the optimal mesh design in hp-version of the finite element method. *International Journal for Numerical Methods in Engineering*, 24(11):2087–2106.

[61] Roache, P. and Steinberg, S. 1985. A new approach to grid generation using a variational formulation. In *Proceeding of AIAA 7-th Computational Fluid Dynamics Conference*, pages 360–370.

[62] Russel, R.D.and Christiansen, J. 1978. Adaptive mesh selection strategies for solving boundary value problems. *SIAM Journal of Numerical Analysis*, 15(1):59–80.

[63] Ryskin, G. and L.G., L. 1983. Orthogonal mapping. *Journal of Computational Physics*, 50(1):71–100.

[64] Saltzman, J. 1986. Variational methods for generating meshes on surfaces. *Journal of Computational Physics*, 63(1):1–19.

[65] Samareh-Abolhassani, J. and Stewart, J. 1994. Surface grid generation in a parameter space. *Journal of Computational Physics*, 113(1):112–121.

[66] Serezhnikova, T., Sidorov, A., and Ushakova, O. 1989. On one method of construction of optimal curvilinear grids and its applications. *Soviet Journal Numerical Analysis and Mathematical Modelling*, 4(2):137–155.

[67] Sidorov, A. 1966. On one algorithm for computing optimal difference grids. In *Proceedings of the Steklov Institute of Mathematics*, pages 147–151.

[68] Soni, B. 1993. Elliptic grid generation system: Control functions revisited - I. *Applied Mathematics and Computation*, 59:151–163.

[69] Spitaleri, R. and Micacchi, V. 1998. A multiblock multigrid generation method for complex simulations. *Mathematics and Computers in Simulation*, 46(1):1–12.

[70] Steger, J. and Chaussee, D. 1980. Generation of body-fitted coordinates using hyperbolic partial differential equations. *SIAM Journal on Scientific and Statistical Computing*, 1(4):431–437.

[71] Steinberg, S. 1994. *Fundamentals of Grid Generation*. CRC Press.

[72] Thompson, J. 1984. Grid generation techniques in computational fluid dynamics. *AIAA Journal*, 22(11):1505–1523.

[73] Thompson, J., Warsi, Z., and .Mastin, C. 1985. *Numerical grid generation: Foundations and Applications*. North-Holland.

[74] Thompson, J. F., editor 1982. *Numerical Grid Generation*. North-Holland.

[75] Tikhonov, A. and Gorbunov, A. 1969. Error estimates for the Runge-Kutta methods and choice of optimal grids. *Computational Mathematics and Mathematical Physics*, 4(2):232–242.

[76] Tu, Y. and Thompson, J. 1991. Three-dimensional solution - adaptive grid generation on composite configurations. *AIAA Journal*, 29(12):2025–2026.

[77] Tucker, A. B., editor 1997. *The Computer Science and Engineering Handbook*. CRC Press.

[78] Vabistchevitch, P. 1989. Adaptive block grids in problems of mathematical. *Computational Mathematics and Mathematical Physics*, 29(6):902–914.

[79] White, A. J. 1979. On selection of equidistributing meshes for two-point boundary-value problems. *SIAM Journal of Numerical Analysis*, 16(3):472–502.

[80] White, A. J. 1982. On the numerical solution of initial/boundary value problems in one space dimension. *SIAM Journal of Numerical Analysis*, 19(4):683–697.

[81] Winslow, A. M. 1966. Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh. *Journal of Computational Physics*, 1(2):149–172.

6

Optimization of Rotations

6.1 Introduction

Given the general optimization context given in the preceding chapters, we will now analyze a particular but important problem of sequencing the machine rotations. There exists a variety of research focused on the *orientation of the cutting tool*. The survey of which is given in Chap. 1. However, the accuracy is also affected by *the way the orientations are being achieved*. In other words, the kinematics error depends not only on the characteristics of the surface versus the tool orientation but on the previous rotations as well. The "history" of rotations becomes in particular important in the vicinity of the stationary points of the desired surface where the tool could make abrupt changes in the orientation. However, such analysis is not provided by commercial CAD/CAM systems such as Unigraphics, EdgeCam, Vericut, etc. Besides, only a few recent research papers deal with the subject.

Recall that a stationary point on a surface is either a maximum or a minimum point or a saddle point. Mathematically, it means that if (u_s, v_s) is the stationary point then

$$\left.\frac{\partial S}{\partial u}\right|_{(u_s,v_s)} = \left.\frac{\partial S}{\partial v}\right|_{(u_s,v_s)} = 0. \tag{6.1}$$

Jung et al. [4] analyze the sequence of rotations to minimize the number of the phase reverse steps at discontinuities of the first derivative of the surface (corners, etc.). A method of avoiding singularities has been presented by Affouard et al. [1]. An algorithm based on global optimization with regard to feasible rotations of the machine was proposed in [11] and developed in [9].

The optimization is based on the idea that there exist several combinations of rotation angles to establish the required orientation. Therefore, one can formulate a problem of finding a set of the rotation angles sequenced in such a way that a certain cost function (for instance the kinematics error) is reduced. The problem can be solved by the shortest path optimization with

regard to the feasible rotations found from the kinematics equations. The cost function can be represented in terms of either the kinematics error or the angle variation. Such optimization increases the accuracy of machining and is the most appropriate in the case of a rough cut.

The shortest path procedure applies to either the entire set of trajectories or to only the most inappropriate overcuts inside the workpiece. In the latest case the algorithm generates an interesting family of solutions characterized by smaller overcuts obtained at the expense of increased undercuts.

Note that a fine cut of a smooth surface employing small spatial and angular steps may not demonstrate the detrimental effects near the stationary points. However, a rough cut characterized by large gradients could produce considerable errors.

The large gradients lead to sharp angular jumps and it is because of the sharp angular jumps that the machine produces the loop-like trajectories of the tool. Moving along such trajectories could destroy the workpiece and even lead to collisions with the machine parts.

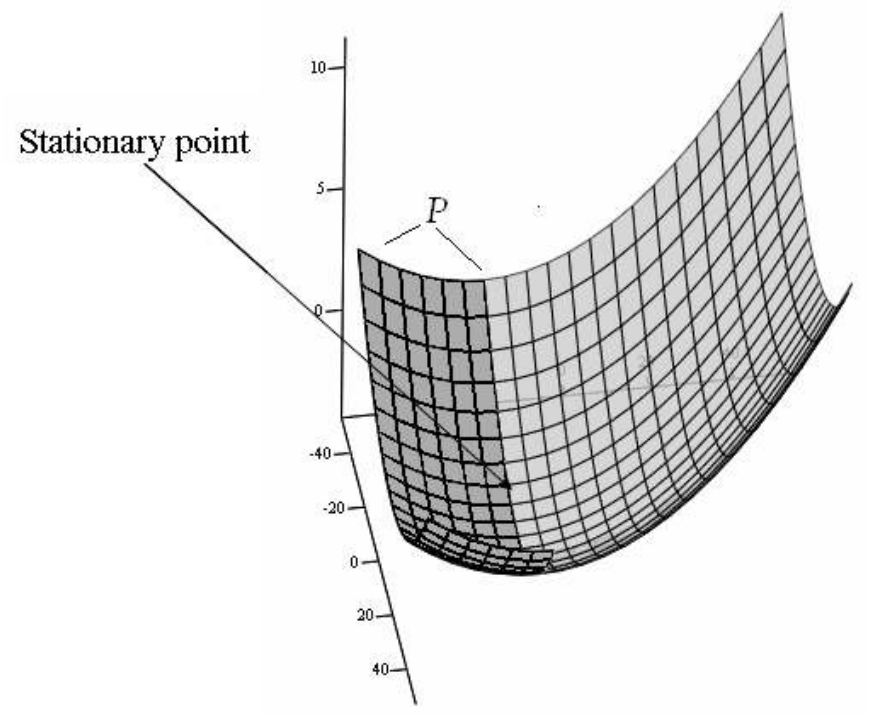
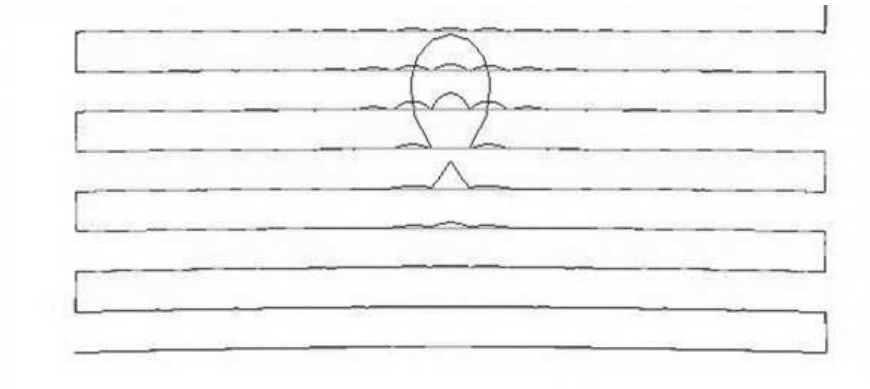


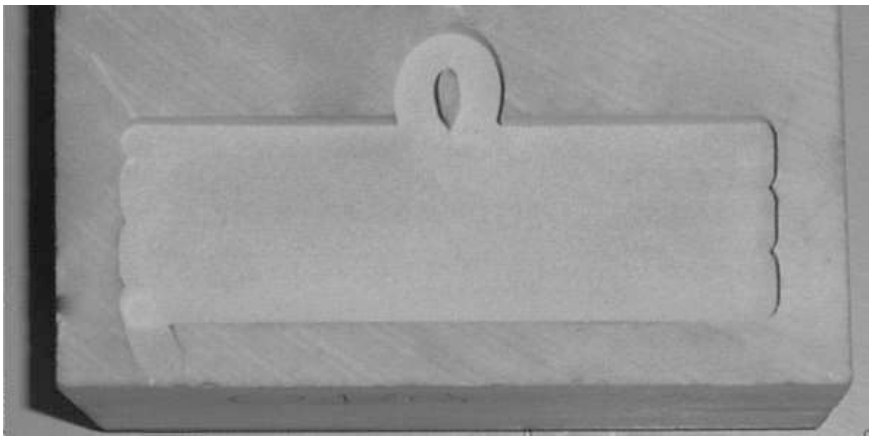
Fig. 6.1. An experimental part surface S_1 (*P is the machined area*)

Our introductory example presents a cut performed for a part surface S_1 (Fig. 6.1) on HERMLE UWF902H. Figure 6.2 shows a large loop in the case of machining a curve belonging to the surface located close to a stationary point. Clearly, linearization of the tool path is not always applicable. It could be even unsafe to assume the linear trajectories, since moving the tool along the actual loop-like trajectories could destroy the workpiece and lead to collisions with the machine parts.

Figure 6.3 shows that changing the sequence of the rotations leads to a much lesser loop and consequently to a significant reduction of the kinematics error. Our second example demonstrates a similar trajectory when machining a single curve belonging to surface S_2 (Fig. 6.4) on MAHO 600E.

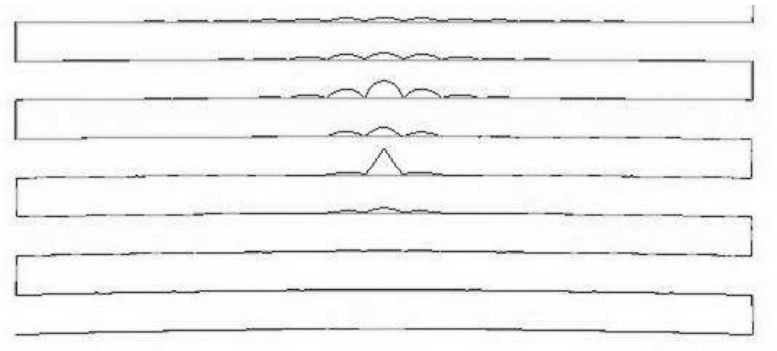


(a)

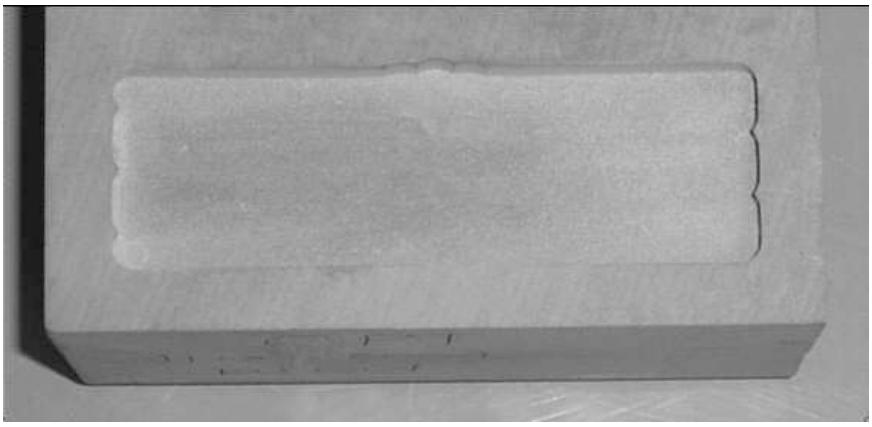


(b)

Fig. 6.2. (**a**) conventional tool path simulated by the virtual machine, (**b**) surface machined by HERMLE UWF902H



(a)



(b)

Fig. 6.3. (**a**) optimized tool path simulated by the virtual milling machine, (**b**) optimized surface machined by HERMLE UWF902H

Figure 6.5 shows that as opposed to a linearized version of the tool path, the real machining produces a loop-like trajectory induced by the large angular steps. Such trajectories could be repaired (see Fig. 6.6) by adjusting the rotation angles in such a way that the kinematics error is minimized.

In this chapter, we will also introduce a point inserting algorithm which can be interpreted as an angular interpolation scheme in the areas of large kinematics errors. The additional positions are created by finding numerically a grid of points uniformly distributed in the angular space.

The procedures are in particular beneficial for high speed milling characterized by small spatial steps. In this case a further decrease of the step size leads to a substantial increase in the machining time due to delays in the servo-update rate.

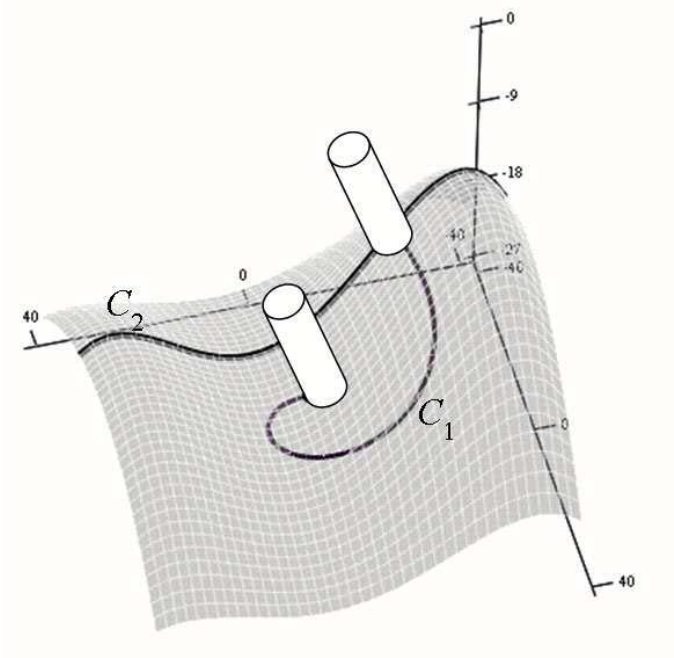
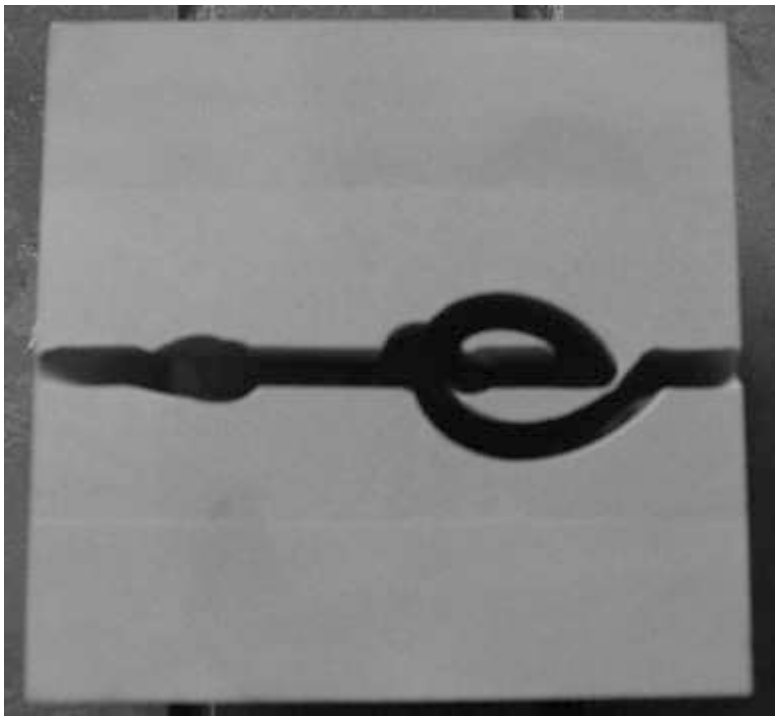


Fig. 6.4. Nonlinearity of the tool path due to rotations in the workpiece coordinates (C_1), the experimental cut (C_2)

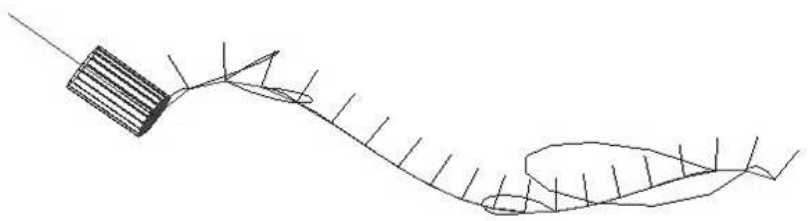
Finally, there is always a limit of the angular speed of specific machine parts. As a result, a shorter tool path with many turns may require more time than a longer tool path with fewer turns [2, 3]. If the maximum angular speed is exceeded, the controller detects this event and reduces the angular speed increasing the machining time. The angle correction algorithms minimize the total angle variation, thus, reducing the probability of such an event.

6.2 Kinematics Error and Angle Variation

Let $W^D_{p,p+1}(t) \in S(u,v)$ be a curve between W_p and W_{p+1} extracted from the surface in such a way that it represents a desired tool trajectory between Π_p and Π_{p+1} (see also Sect. 3.4). We define the *total error* as a sum of deviations of $W^D_{p,p+1}(t)$ from the actual trajectories $W_{p,p+1}(t)$. The total error is given by

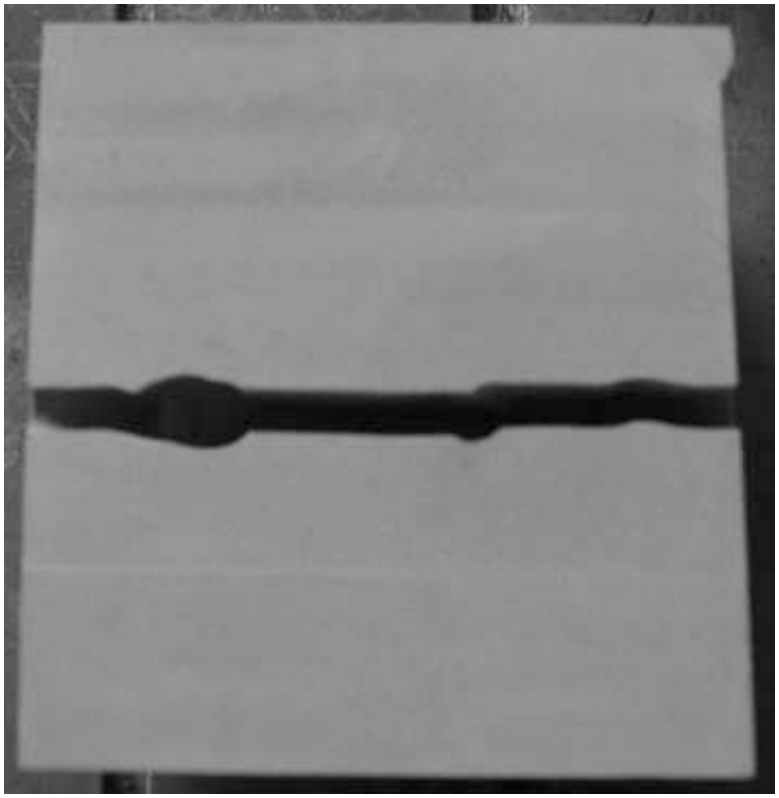


(a)

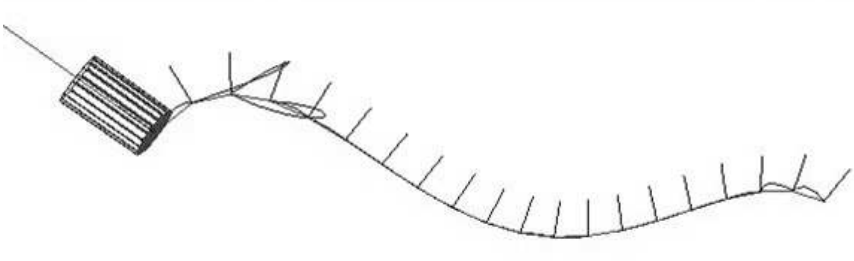


(b)

Fig. 6.5. (**a**) a loop-like trajectory induced by large gradients of the rotation angles damages the workpiece, (**b**) the trajectory simulated by the virtual milling machine



(a)



(b)

Fig. 6.6. (**a**) the "repaired" trajectory, (**b**) the repaired trajectory simulated by the virtual milling machine

$$\begin{aligned}\epsilon &= \sum_p \left[\int_0^1 \left|W^D_{p,p+1}(t) - W_{p,p+1}(t)\right|^2 dt\right]^{\frac{1}{2}}, \\ &= \sum_p \Bigg[\int_0^1 \Big[\big(x^D_{p,p+1}(t) - x_{p,p+1}(t)\big)^2 + \big(y^D_{p,p+1}(t) - y_{p,p+1}(t)\big)^2 \\ &\qquad + \big(z^D_{p,p+1}(t) - z_{p,p+1}(t)\big)^2\Big] dt\Bigg]^{\frac{1}{2}}.\end{aligned} \tag{6.2}$$

The total error above is approximated as follows:

$$\begin{aligned}\epsilon = \sum_p \frac{1}{L_p} \Bigg[\sum_{l_p=1}^{L_p} \Big[&\Big(x^D_{p,p+1,l_p} - x_{p,p+1,l_p}\Big)^2 + \Big(y^D_{p,p+1,l_p} - y_{p,p+1,l_p}\Big)^2 \\ &+ \Big(z^D_{p,p+1,l_p} - z_{p,p+1,l_p}\Big)^2\Big]\Bigg]^{\frac{1}{2}},\end{aligned} \tag{6.3}$$

where L_p is a number of the sampling points between W_p and W_{p+1}.

Remark 6.1. The definitions above can be simplified by replacing the desired trajectories $W^D_{p,p+1}$ by linear trajectories given by $W^L_{p,p+1}(t) = tW_{p+1} + (1-t)W_p$, although care should be taken when using this option. As opposed to the machine coordinates M, the trajectories in the workpiece coordinates W are not linear. However, we may use the linear trajectories as a reference, noting that $\epsilon \equiv \|W^D - W\| \leq \|W^D - W^L\| + \|W^L - W\|$, where L is a piecewise linear approximation of S. Hence, when the points are close enough, the error is approximately $\epsilon \approx \epsilon_L = \|W^L - W\|$. Note that if the orientation of the tool is fixed through the entire cut, then $\epsilon_L = 0$. In other words, the three-axis mode leads to $\epsilon_L = 0$ since all the trajectories become linear. Therefore, minimization with regard to ϵ_L must be subjected to constraints specifying the orientations of the tool. Finally, the linearization is the simplest option which can be used even when the actual surface is not known, for instance, for a given G-code. Actually, the desired trajectory W^D can be extracted from the surface by a variety of ways, for example, using interpolation in the parametric space, the geodesic curves, etc. However, the results presented in this chapter are valid irrespectively of the method to obtain W^D. Finally, the error induced by linear approximation depends on the maximum distance h between the CC points as $k\|\dfrac{d^2W^D}{dt^2}\|_C h^2$, where k is a constant (see, for instance, [10]).

Let us now introduce the total angle variation. Consider two positions W_p and W_{p+1}. The corresponding kinematics error between the two positions is then defined by

$$\epsilon_{p,p+1} \equiv \left[\int_0^1 \left[W^D_{p,p+1}(t) - W_{p,p+1}(t)\right]^2 dt\right]^{\frac{1}{2}}. \tag{6.4}$$

Clearly, $\epsilon_{p,p+1} \equiv \epsilon(\Delta a_{p,p+1}, \Delta b_{p,p+1}, \Delta l_{p,p+1})$ where $\Delta l_{p,p+1} = \|W^D_{p,p+1} - W_{p,p+1}\|_E$ is the spatial step in the workpiece coordinate system. Furthermore, $\Delta a_{p,p+1} \equiv a_{p,p+1}(\Delta l_{p,p+1}) = a_{p+1} - a_p$ and $\Delta b_{p,p+1} \equiv b_{p,p+1}(\Delta l_{p,p+1}) = b_{p+1} - b_p$ are the angular steps. Machining experiments show that the angular steps are often more important than the spatial step. In particular, during a rough cut, decreasing the angular steps leads to a larger decrease in the error than a decrease in the spatial step. The angle variation between positions W_p and W_{p+1} is given by:

$$c_{p,p+1} = \int_0^1 \sqrt{\left(\frac{\partial a_{p,p+1}}{\partial t}\right)^2 + \left(\frac{\partial b_{p,p+1}}{\partial t}\right)^2}\, dt. \tag{6.5}$$

Assuming that $a_{p,p+1}$ and $b_{p,p+1}$ are changing linearly, yields $c_{p,p+1} = \sqrt{(\Delta a_{p,p+1})^2 + (\Delta b_{p,p+1})^2}$. The total angle variation across the entire tool path is then simply given by

$$c = \sum_p c_{p,p+1} = \sum_p \sqrt{(\Delta a_{p,p+1})^2 + (\Delta b_{p,p+1})^2}. \tag{6.6}$$

Note that there often exists a neighborhood of the CL point where one of the angles changes faster than another one. In this case $\epsilon_{p,p+1} \approx O\left((\Delta a_{p,p+1})^{m_1}\right)$ or $\epsilon_{p,p+1} \approx O\left((\Delta b_{p,p+1})^{m_2}\right)$, where m_1 and m_2 depend on the surface. In this case, minimization of the total angle variation produces excellent results.

Finally, the case of a rough cut often requires to differentiate between the overcut and undercut error. When too much material is removed from the part, it is said to be gouged or overcut. The undercut is a situation when not enough material is removed. For simplicity, we define the undercut and overcut error as follows

$$\epsilon_U = \sum_p \int_0^1 \begin{cases} (W^D_{p,p+1} - W_{p,p+1})^2 & \text{if } z^D_{p,p+1}(t) \le z_{p,p+1}(t) \\ 0 & \text{otherwise} \end{cases} dt\ , \tag{6.7}$$

$$\epsilon_O = \sum_p \int_0^1 \begin{cases} (W^D_{p,p+1} - W_{p,p+1})^2 & \text{if } z^D_{p,p+1}(t) > z_{p,p+1}(t) \\ 0 & \text{otherwise} \end{cases} dt\ . \tag{6.8}$$

The corresponding undercut and overcut angle variation are then defined by

$$c_U = \sum_p c_{U,p,p+1}, \tag{6.9}$$

$$c_O = \sum_p c_{O,p,p+1}, \tag{6.10}$$

where

$$c_{U,p,p+1} = \sqrt{(\Delta a_{p,p+1})^2 + \Delta b_{p,p+1})^2}\Delta^U_{p,p+1},$$

$$c_{O,p,p+1} = \sqrt{(\Delta a_{p,p+1})^2 + \Delta b_{p,p+1})^2}\Delta^O_{p,p+1},$$

$$\Delta^U_{p,p+1} = \int_0^1 \begin{cases} 1 \text{ if } z^D_{p,p+1}(t) \le z_{p,p+1}(t) \\ 0 \text{ otherwise} \end{cases} dt\ ,$$

$$\Delta^O_{p,p+1} = \int_0^1 \begin{cases} 1 \text{ if } z^D_{p,p+1}(t) > z_{p,p+1}(t) \\ 0 \text{ otherwise} \end{cases} dt\ ,$$

Clearly, $\epsilon = \epsilon_O + \epsilon_U$ and $c = c_O + c_U$. Since overcut is not repairable, reducing the overcut error is often more important than reducing the undercut or even the total error.

6.3 Optimization Problem

Let us formulate the following minimization problem:

$$\min_{\Lambda}(w_U\epsilon_U + w_O\epsilon_O), \tag{6.11}$$

where Λ is a set of possible rotations and w_U, w_O the weighting coefficients representing the importance of the undercut and overcut, respectively. For instance, if $w_U = w_O = 1$, then the total kinematics error will be minimized. If $w_U = 0$, $w_O = 1$, only kinematics errors along the overcut will be minimized. If $w_U = 0.1$, $w_O = 1.0$, then the kinematics error is minimized with regard to both the undercut and overcut, but the importance of the overcut is 10 times larger than that of the undercut. The weights may also depend on the kinematics error itself. For example, one may establish the following rule "if $\epsilon < \epsilon'$ then $w_O = 1$, $w_U = 0$ else $w_O = w_U = 1$", where ϵ' is a prescribed threshold. In other words, if the total kinematics error is not too large, we minimize only the overcut, however if the kinematics error is more than a prescribed ϵ' then the minimization applies to the total kinematics error.

How to evaluate set Λ? Without loss of generality, consider the configuration depicted in Fig. 2.4. Recall that the machine movements prescribed by a sequence of the machine commands are executed simultaneously allowing for five degrees of freedom of the tool. The goal of the simultaneous movements is transporting the tool tip to location $W = (x_w, y_w, z_w)$ in the workpiece coordinate system or $M = (x_m, y_m, z_m)$ in the machine coordinate systems and establishing orientation $\mathfrak{R} = (a, b)$. In accordance with the reference coordinate system in Fig. 2.4(b), the angle a is between the X_1-axis and the projection of the orientation vector onto the X_1-Y_1 plane and the positive direction of the X_1-axis. The angle b is between the projection of the orientation vector onto the X_1-Z_1 plane and the positive direction of the X_1-axis.

The above makes it possible to represent $\mathfrak{R}$ in terms of the components of the orientation vector $I = (I_x, I_y, I_k)$. First of all, note that the tool orientation vector in O_4 is $(0, 0, 1)$. In other words, the rotations must be performed

in such a way that the tool orientation vector becomes collinear with the Z_4-axis. Observe that the resulting coordinate transformation depends on the zero position $\mathfrak{R} = (0, 0)$ which can assigned by a special machine command. It is not hard to see clearly that at this configuration only $b = 0$ is important. Therefore, without loss of generality, we assign $\mathfrak{R} = (0, 0)$ to the rightmost position of the tilt table shown in Fig. 2.4(a).

From (2.4), the equation relating the two rotation angles $\mathfrak{R} = (a, b)$ and the tool orientation vector I is given by

$$\begin{aligned} I_x &= \cos(a)\cos(b), \\ I_y &= \sin(a)\cos(b), \\ I_z &= -\sin(b). \end{aligned} \tag{6.12}$$

Consider a solution of the above system given by

$$\begin{aligned} a_{\text{base}} &= \begin{cases} \tan^{-1}\left(\dfrac{I_y}{I_x}\right) & \text{if } I_x > 0 \text{ and } I_y \geq 0, \\ \tan^{-1}\left(\dfrac{I_y}{I_x}\right) + \pi & \text{if } I_x < 0, \\ \tan^{-1}\left(\dfrac{I_y}{I_x}\right) + 2\pi & \text{otherwise,} \end{cases} \\ b_{\text{base}} &= -\sin I_z, \end{aligned} \tag{6.13}$$

Given a_{base} and b_{base}, it is not hard to construct a full set of four solutions given by (see the four solutions in Fig. 6.7)

$$\begin{aligned} \Lambda = \big\{ &(a_{\text{base}}, b_{\text{base}}), (a_{\text{base}} - 2\pi, b_{\text{base}}), \\ &(a_{\text{base}} - \pi, -b_{\text{base}} - \pi), (a_{\text{base}} + \pi, -b_{\text{base}} - \pi) \big\}. \end{aligned} \tag{6.14}$$

Similar solutions can be established for other types of the machine kinematics such as, the 1-1 machine and the 0-2 machine (see Chap. 2).

Note that the number of solutions depends on the number of the rotational degrees of freedom d as $O(2^d)$. For example, for a six-axis machine we will have eight solutions, however, some solutions might not be achievable due to the machine limits.

Furthermore, it is clear that the problem can be solved by shortest path optimization. The corresponding graph is constructed in such a way that each position $\mathit{\Pi}_p$ is characterized by 4 graph nodes Λ_p, where the edge between the nodes represents the kinematics error or the angle variation (see Fig. 6.8). Therefore, such minimization could be performed by a conventional so-called greedy discrete algorithm.

The above shortest path scheme is visualized in Fig. 6.9 and Fig. 6.10, where l denotes a coordinate along the tool path. The 4 options are represented by 4 trajectories in the angular space (a, b). It is plain that the trajectories are close when b is near $-\pi/2$. In this case it is possible to change the trajectory

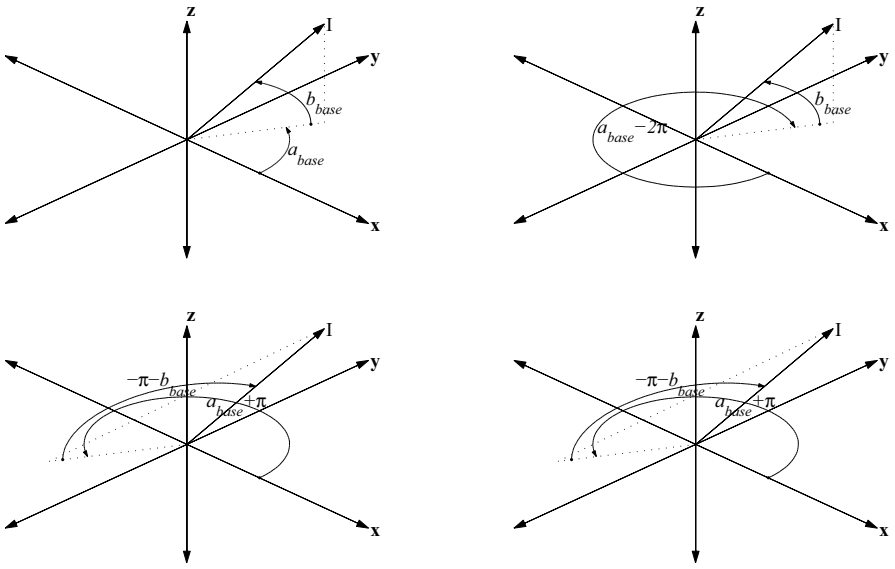


Fig. 6.7. The set of feasible rotations

by considering the shortest path producing minimal error. Figure 6.10 shows that the optimization may only be required near the stationary points, namely, when b is close to $-\pi/2$ and a is "jumping" from 0 to π or from $\pi/2$ to $3\pi/2$, etc. Consequently, the computational load could be substantially reduced by introducing an "optimization windows" in the neighborhood of the stationary points. This idea discussed in details in [8].

6.4 Optimization of Rotations: Examples and Practical Machining

Example 6.1. **A trajectory passing through a stationary point - across or around the hill**

Consider two successive points $\mathit{\Pi}_1 = (W_1, \mathfrak{R}_1)$ and $\mathit{\Pi}_2 = (W_2, \mathfrak{R}_2)$. Let the average space step $s = |W_1 - W_2|$ be 1 mm. Consider a fixed $\mathfrak{R}_1 \equiv (a_1, b_1) = (0, -85^\circ)$ and a varying $\mathfrak{R}_2 \in [120, 180] \times [-60, -80]$. $\mathfrak{R}_1$ and $\mathfrak{R}_2$ represent a possible combination of the rotation angles in the neighborhood of a stationary point (see Fig. 6.11).

Consider a linear trajectory T and the actual trajectories T_{across}, T_{around} corresponding to $\mathfrak{R}_2 \equiv (a_2, b_2) = (-10^\circ, -95^\circ)$ and $\mathfrak{R}_{2,\text{base}} \equiv (a_{2,\text{base}}, b_{2,\text{base}}) = (170, -85)$, respectively. Note that $a_2 = a_{2,\text{base}} - 180^\circ$, $b_2 = -b_{2,\text{base}} - 180^\circ$. Let the distance between the midpoint of the linear trajectory and the center of the a-rotation (A-axis) r_a be 10 mm. Let us compute the corresponding trajectory and the kinematics error. We have $\epsilon_{\text{across}} = 1.49$

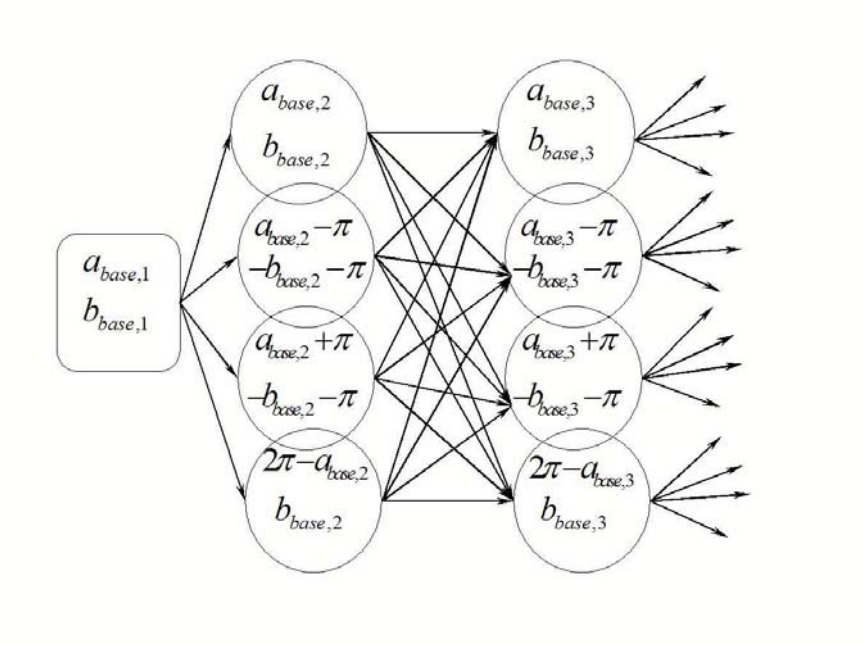


Fig. 6.8. A graph corresponding to the set of the feasible rotations

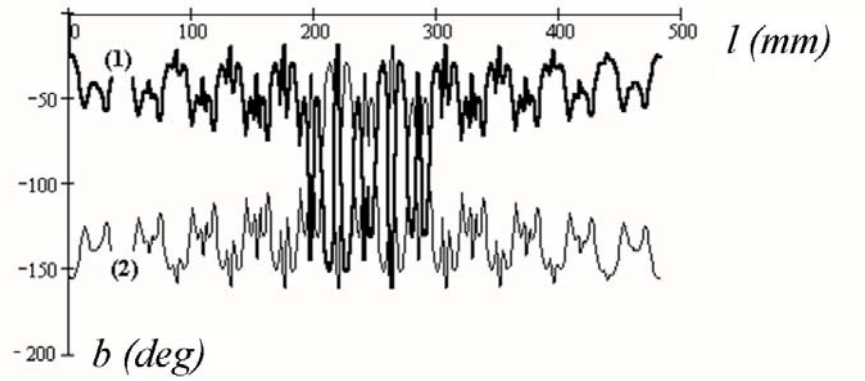


Fig. 6.9. Trajectories corresponding to (1) b_{base} and (2) $\pi - b_{\text{base}}$ and the optimized trajectory composed from (1) and (2) (*bold line*), l is the coordinate along the tool path

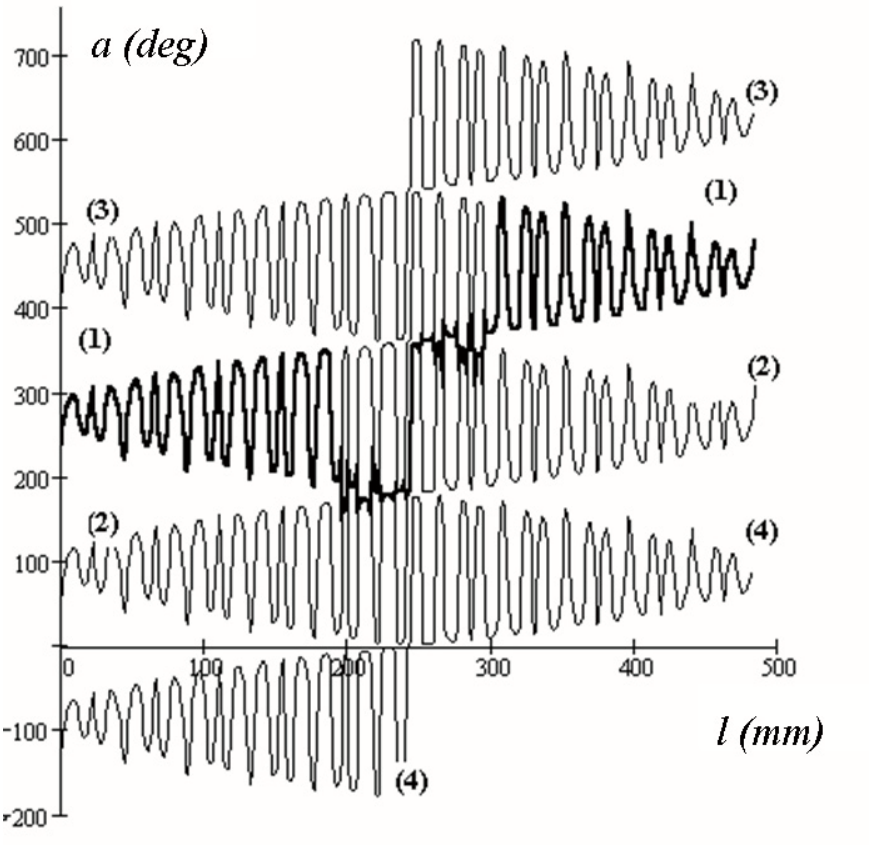


Fig. 6.10. Trajectories corresponding to (1) a_{base} (2) $a_{\text{base}} - \pi$ (3) $a_{\text{base}} + \pi$ (4) $a_{\text{base}} - 2\pi$ and the optimized trajectory (*bold line*) composed from the trajectories (1) (2) and (4)

mm, $\epsilon_{\text{around}} = 8.89$ mm. In words, moving "across the hill" by T_{across} is better than T_{around} - "around the hill".

However, it is not always the case. A small r_a may entail another choice. For instance, r_a= 0.5 mm produces $\epsilon_{\text{around}} = 0.71$ mm, $\epsilon_{\text{across}} = 1.59$ mm. Therefore, when the cutter location point is close to the center of the a-rotations, the tool should move "around the hill".

Given an average space step, one could evaluate the kinematics error for varying r_a. Such evaluation makes it possible to pre-compute the movements of the tool and to reduce the computational load spent on evaluating the error. Figure 6.12 shows the kinematics error corresponding to T_{across} and T_{around} as functions of a_2 and b_2 for $r_a = 0.1$ mm, 5 mm and 20 mm, respectively. Clearly, $r_a = 0.1$ mm requires T_{around} for any angle, $r_a = 20$ mm entails T_{across}

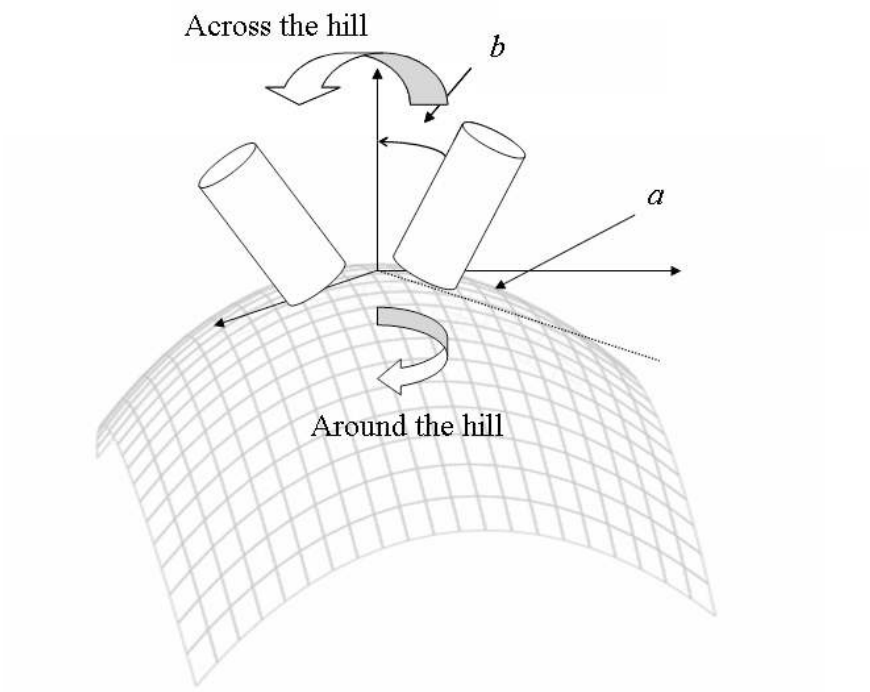


Fig. 6.11. Around or across the hill?

whereas $r_a = 5$ mm requires a further evaluation of the kinematics error for each particular pair (a, b).

Such pre-computing performed for a set of prescribed spatial steps could constitute an optimization strategy for a practical five-axis machining. For instance, minimization of the kinematics error of the milling machine MAHO 600E for $s = 1$ mm requires T_{around} if $r_a \in [0, 5]$ mm and T_{across} if $r_a > 10$ mm. However, when $r_a \in (5, 10)$ mm, the decision can not be pre-computed and shall be made by evaluating $\epsilon^{\text{kinematic}}$, explicitly.

Example 6.2. **Optimization with regard to the kinematics error**

We have shown that loops due to large variations in the rotation angles can be almost entirely eliminated for a trajectory having a single stationary point. Table 6.1 shows the performance of the method for an entire surface S_1 (Fig. 6.1) machined on MAHO 600E.

The optimization was performed without discriminating between ϵ_U and ϵ_O. Consider Table 6.1. Clearly, minimizing the total angle variation leads to minimizing the total kinematics error. On the contrary, an increase of the number of the CL points does not necessarily lead to a decrease of the error (see the lines 40×20 and 100×20) since the grids are not nested (the points of the coarse grid do not necessarily belong to the fine grid).

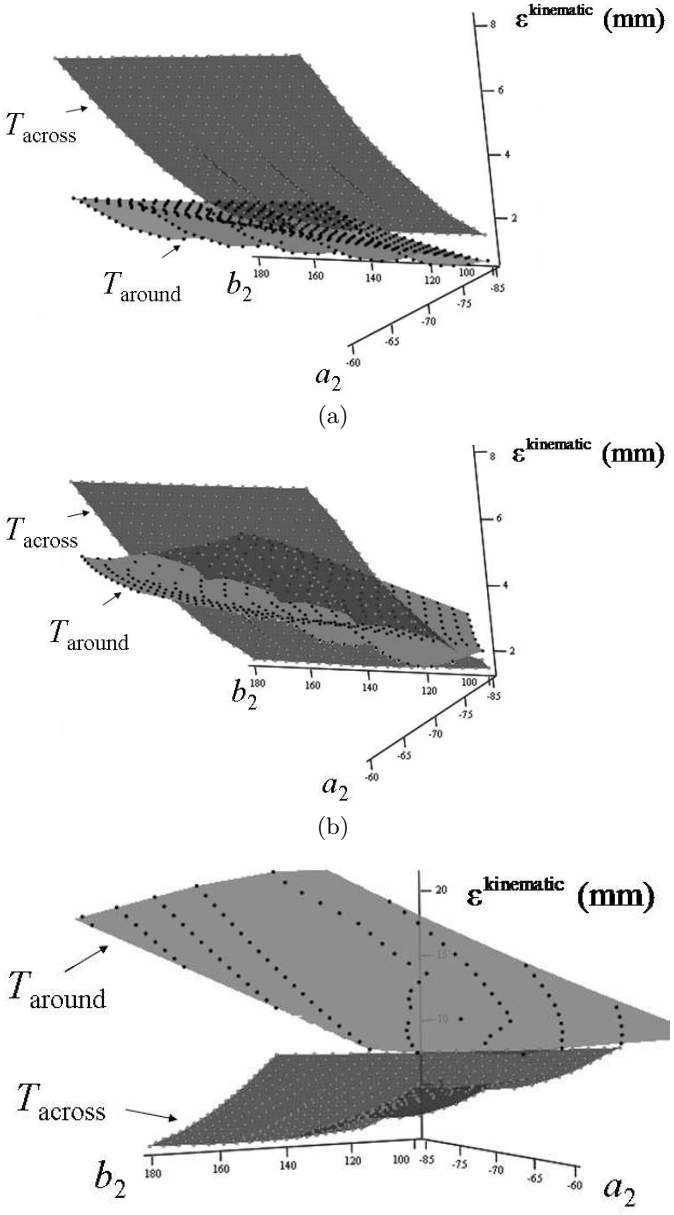


Fig. 6.12. The kinematic error as a function of a_2 and b_2 for rotation radius r_a = 0.1 mm (**a**), 5 mm (**b**) and 20 mm (**c**)

Table 6.1. Kinematics error for the optimized and non-optimized tool path, surface S_1 on MAHO 600E

	Max error (mm) /Max angle var (deg)			Path length (mm)
Grid size	No optimization	Optimization	Decrease (%)	Non-opt/Opt
10×20	23.862 / 168.779	12.426 / 78.072	47.925 / 53.743	2825.67 / 2255.54
15×20	19.300 / 162.186	8.517 / 101.524	55.870 / 34.403	2500.30 / 2123.02
20×20	20.228 / 160.080	7.558 / 84.927	62.636 / 46.947	2367.57 / 2101.15
30×20	16.253 / 141.890	7.162 / 88.230	55.934 / 37.818	2183.63 / 2038.27
40×20	8.711 / 103.885	6.878 / 88.399	21.042 / 14.907	2069.32 / 2020.11
100×20	7.395 / 90.898	7.103 / 89.102	3.949 / 1.976	1916.11 / 1911.17
130×20	3.999 / 68.416	3.999 / 68.416	0.000 / 0.000	1876.49 / 1876.49

Example 6.3. **Optimization of the tool path with regard to overcut and undercut**

A very useful optimization can be performed when the undercuts and overcuts are differentiated. In order to detect them numerically every trajectory $W_{p,p+1}(t)$ is subdivided by a certain number of points t_k. At every point $W_{p,p+1,k} = (x_{p,p+1,k}, y_{p,p+1,k}, z_{p,p+1,k})$ we find the corresponding parametric coordinates (u_k, v_k) by solving numerically the system

$$\begin{aligned} x_{p,p+1,k} &= x_S(u,v) \\ y_{p,p+1,k} &= y_S(u,v) \end{aligned} \tag{6.15}$$

Note that for explicit surfaces the above equations are linear and, therefore, a numerical solution is not required. Now, given the solution (u_k, v_k), we calculate $z_S(u_k, v_k)$ for every k. Next, we calculate the angle variations $c_{U,p,p+1}$ and $c_{O,p,p+1}$ for every p and for every graph node. Finally, we use the weights w_O and w_U and apply the Dijkstra's shortest path algorithm (see, for instance, [12]).

Consider machining surface S_2 on HERMLE UWF902H. Let $w_O = 1$, $w_U = 0$ that is, the minimization is performed only with regard to the overcuts. Figure 6.13a displays the conventional tool path with overcuts (the thick curves). Figure 6.13b shows how minimization of the total error eliminates the loops located at the right side of the workpiece. However, minimization of only the overcut error produces an entirely different path. The overcut loops at the boundary of the workpiece have been replaced by large but harmless undercuts (see Fig. 6.13c). Table 6.2 shows the behavior of the undercut and overcut error along with the angle variation as the number of points in the cutting direction increases. The decrease in the overcut error has been achieved at the expense of the increase of the undercuts and the length of the path.

Let us now demonstrate the techniques by optimizing the overcuts on MAHO 600E. The configuration of this machine will create an entirely different set of trajectories. The conventional tool path and the tool path optimized by means $w_O = 1$, $w_U = 1$ are displayed in Figs. 6.14(a-b), whereas a tool path optimized with $w_O = 1$, $w_U = 0$ in Fig. 6.14c. The corresponding machined surfaces are displayed in Figs. 6.15(a-c). Clearly, optimization with regard to only the overcuts makes it possible to cut a rough but reasonable surface which could be further refined whereas cutting without the overcut optimization may destroy the workpiece. The behavior of the kinematics error versus the number of points and the maximum angle variation is displayed in Table 6.3. Comparing HERMLE UWF902H and MAHO 600E (Tables 6.2 and 6.3) shows that the accuracy of the two cuts without optimization is comparable, whereas the optimized cut on MAHO 600E outperforms HERMLE UWF902H.

Table 6.2. Overcut error optimization, surface S_2 on HERMLE UWF902H

	Max error (mm) / Max angle var (deg)				Path length (mm)	Angular variation (deg)
	No optimization		Optimization			
Grid size	Overcut	Undercut	Overcut	Undercut	Non-opt/Opt	Non-opt/Opt
10×20	19.11 / 139.78	23.40 / 169.95	13.33 / 109.90	56.16 / 356.22	2929.02 / 3641.59	7763.87 / 9317.35
15×20	9.54 / 32.96	19.28 / 162.13	9.52 / 32.76	60.93 / 364.84	2676.55 / 3985.66	7921.93 / 10131.55
20×20	8.07 / 88.17	20.21 / 160.01	4.41 / 23.83	60.93 / 364.84	2486.07 / 5021.87	7972.65 / 11477.04
30×20	6.28 / 149.23	16.24 / 141.87	2.40 / 45.76	51.55 / 372.37	2253.76 / 3845.65	8018.01 / 10119.48
40×20	5.41 / 139.58	8.81/ 104.05	1.60 / 14.187	13.24 / 130.02	2117.19 / 2166.59	8035.95 / 7993.36
100×20	0.00 / 4.00	7.40 / 90.83	0.00 / 4.00	7.13 / 89.27	1924.58 / 1919.35	8055.03 / 8032.72
130×20	0.00 / 3.09	4.00 / 68.34	0.00 / 3.09	4.00 / 68.34	1880.72 / 1880.72	8056.50 / 8056.50

Table 6.3. Overcut error optimization, surface S_2 on MAHO 600E

	Max error (mm) / Max angle var (deg)					
	No optimization		Optimization		Path length (mm)	Angular variation (deg)
Grid size	Overcut	Undercut	Overcut	Undercut	Non-opt/Opt	Non-opt/Opt
10×20	8.41 / 169.66	23.86 / 170.03	1.84 / 28.26	38.46 / 157.94	2825.67 / 3575.11	7768.03 / 7867.60
15×20	7.23 / 164.57	19.30 / 162.20	1.98 / 40.47	32.92 / 159.15	2500.30 / 2417.30	7926.18 / 7339.96
20×20	6.78 / 159.42	20.23 / 160.83	0.96 / 26.98	7.56 / 87.37	2367.57 / 2101.14	7976.88 / 7148.41
30×20	6.25 / 149.31	16.25 / 141.94	0.00 / 12.57	7.16 / 89.19	2183.63 / 2032.20	8022.24 / 7848.15
40×20	5.39 / 139.65	8.71 / 104.12	0.00 / 9.64	6.88 / 88.56	2069.32 / 2020.11	8040.19 / 7788.45
100×20	0.00 / 4.00	7.40 / 90.90	0.00 / 4.00	7.10 / 89.20	1916.11 / 1911.17	8059.30 / 8036.42
130×20	0.00 / 3.09	4.00 / 68.42	0.00 / 3.09	4.00 / 68.41	1876.49 / 1876.49	8060.76 / 8080.76

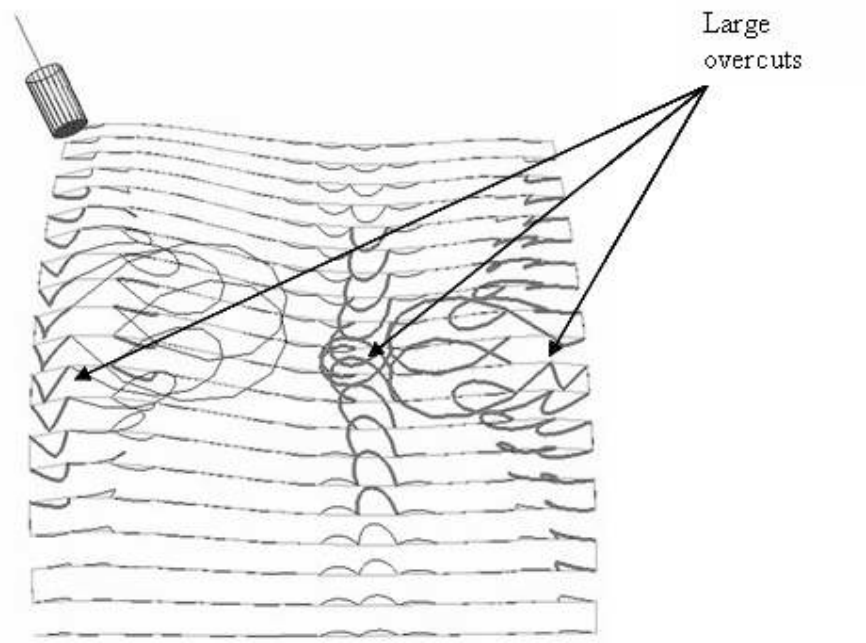


Fig. 6.13a. Conventional tool path for S_2 on HERMLE UWF902H showing large overcuts (*thick curves*)

6.5 Uniform Angular Grids

This section introduces uniform angular grids inserted locally in such a way that the resulting total angle variation is minimized or at least reduced. Many algorithms to calculate the so-called feedrate (the step between the consecutive CL points) have been proposed. The simplest approach is to linearize the surface along the tool path, calculate the error and insert additional points until the error is within the required tolerance (see [5]). Another approach is to interpolate the desired trajectory by a spline or a NURBS and establish the required feedrate using properties of the curve. The feedrate may depend on the space coordinates of the curve as well as on the angular changes (see, for example, [13]). A combination of the actual kinematics of the machine with an interpolating method is made by Lo [6].

A general framework for applying grid generation to tool path optimization proposed in [7] and in [8] is given in Chap. 5. In particular, grid generation based on the Dirichlet functional, invoked in an iterative loop with a suitable preprocessing, may substantially decrease the kinematics error. The grid generation is applicable if $\epsilon \to 0$ as the area of the grid cell A tends to zero. If it is the case, a functional representing the so-called equi-distribution principle [8] and subjected to the machine constraints can be introduced.

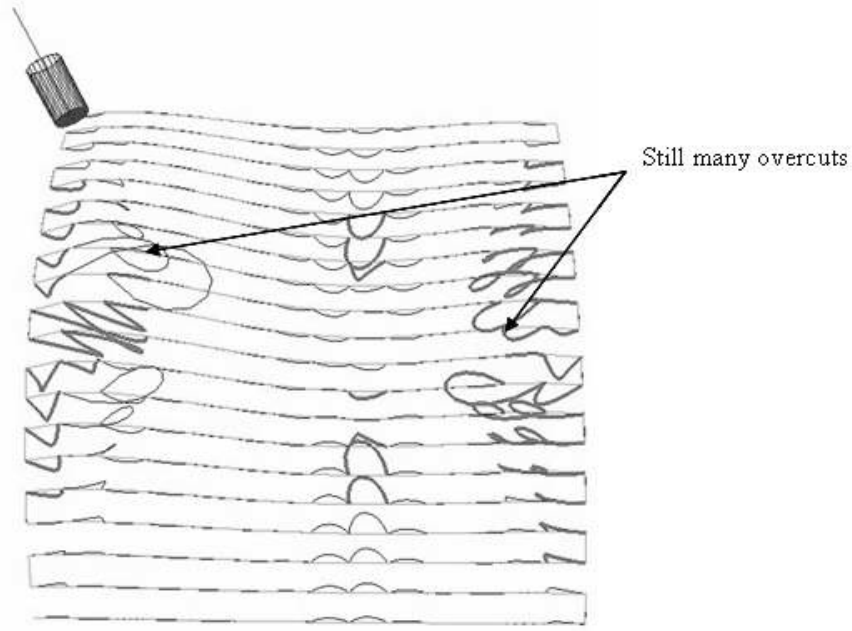


Fig. 6.13b. The tool path optimized with regard to the total error. There are still many trajectories with large overcuts

Now we will consider a different situation characterized by a rough cut, when the tool rotations angles jump as described above. In this case the adaptive grid may require very small spatial cells. As a result the algorithm may create twisted and degenerated cells which may not converge or display a very slow convergence.

Suppose we have constructed a "basic" tool path represented by a rectangular or a curvilinear grid described in Chap. 5. Suppose we are allowed to "inject" a certain number of additional points to decrease the error. Where and how should these points be inserted? Let us employ local grids with equal angular increments around points characterized by large angle variations. We shall call such grids the uniform angular grids.

Recall that near the stationary points there often exists a neighborhood such that one of the rotations is not performed or one of the angles changes faster than another angle. In this case it is sufficient to construct a grid uniform only with regard to the "fast" angle.

The algorithm consists of the following steps. First of all, it detects points characterized by sharp angular variation. Next, for each point the algorithm determines the positions between which the uniform angular grid should be constructed. Usually the interval includes a few tool positions right before and after the singularity which creates the kinematics loop. The space between every pair of the selected points W_p and W_{p+1} must be subdivided in such a

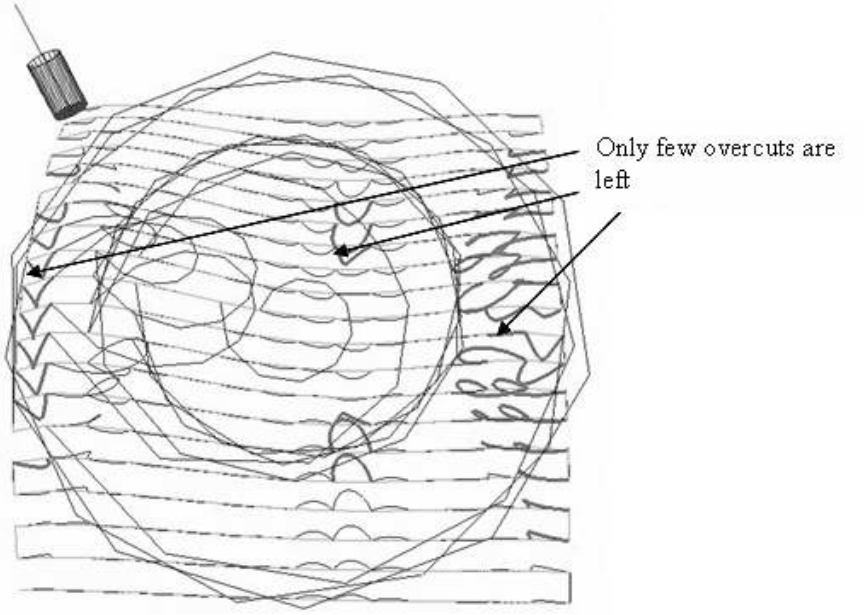


Fig. 6.13c. The tool path optimized with regard to the overcut error. The large overcuts have been replaced by large but harmless undercuts

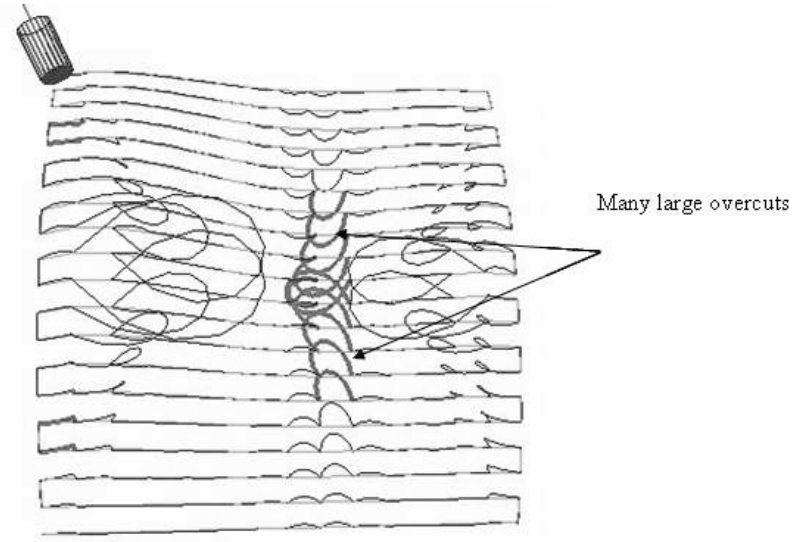


Fig. 6.14a. Conventional tool path for S_2 on MAHO 600E

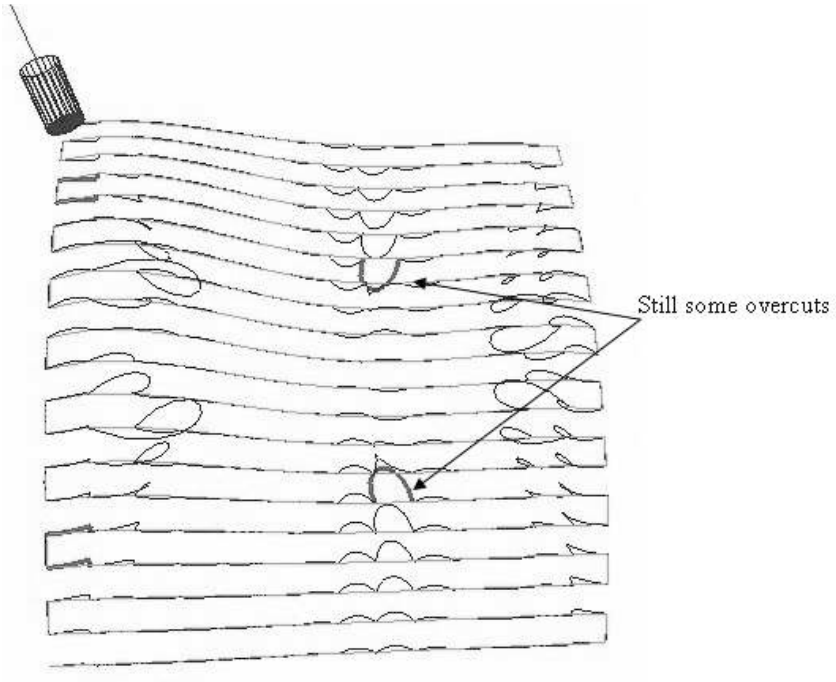


Fig. 6.14b. Optimization with regard to the total error

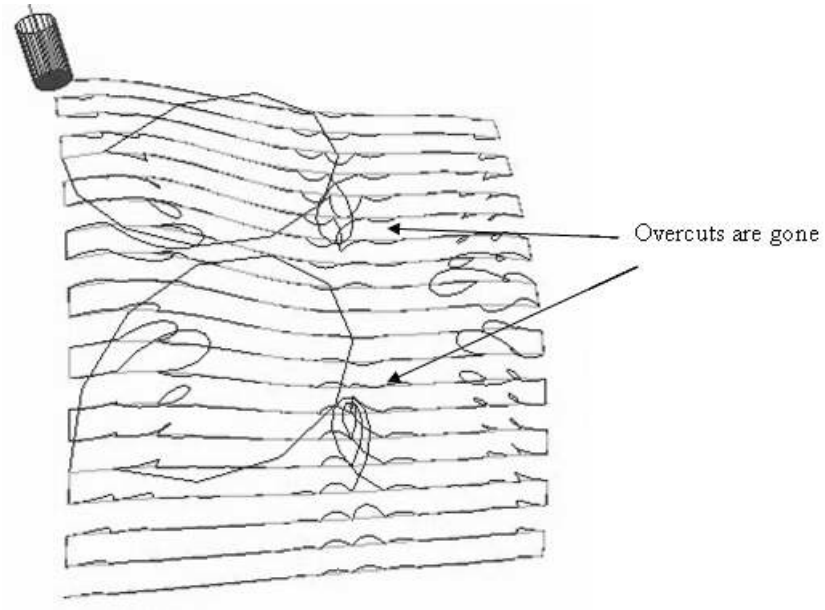


Fig. 6.14c. Optimization with regard to the overcut error, S_2 on MAHO 600E

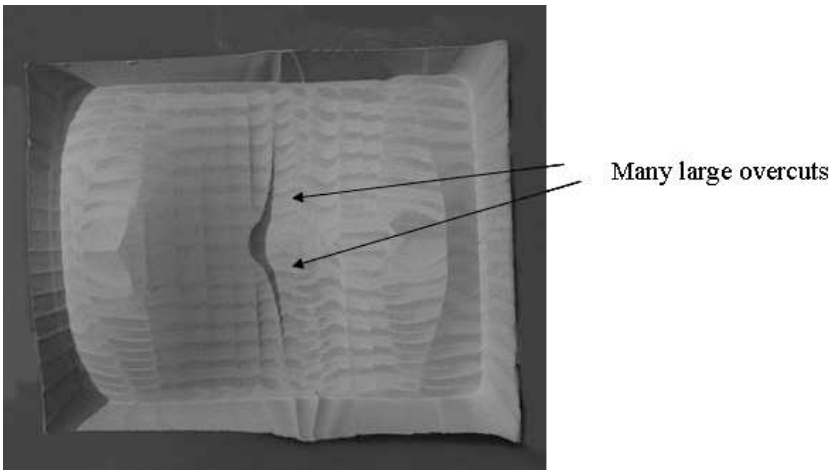


Fig. 6.15a. Without optimization, S_2 on MAHO 600E (corresponds to Fig. 6.14a)

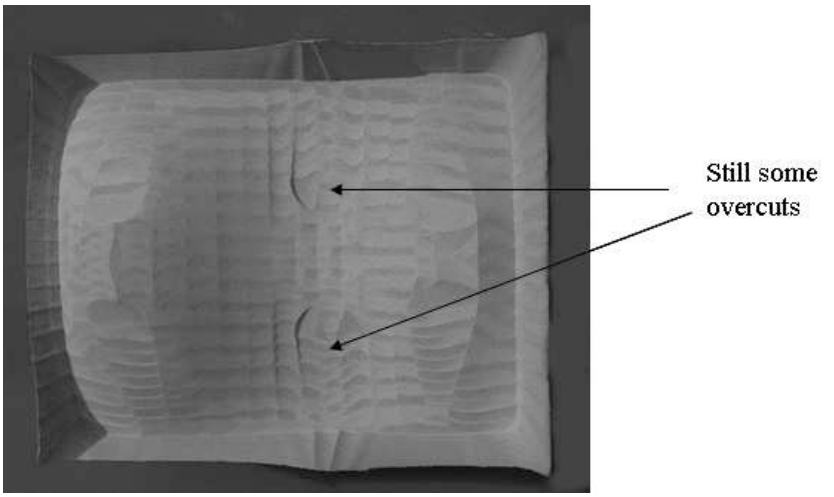


Fig. 6.15b. Optimization with regard to the total error, S_2 on MAHO 600E (corresponds to Fig. 6.14b)

way that $\Delta a_{p,p+1,i,i+1} \approx \text{const}$ or $\Delta b_{p,p+1,i,i+1} \approx \text{const}$ for every subinterval $[i, i+1]$.

We have already noted that the stationary point is cut by either changing sharply the first or the second rotation angle. Irrespectively whether the singular point is a CC point or not, the tool will follow a path "across or around the hill" depending on the choice of the rotation angles. Consequently, the algorithm selects the appropriate angle (say, angle a) and subdivides the angular interval $\Delta a_{p,p+1}$ into equal subintervals $\Delta a_{p,p+1,i,i+1}$ such that

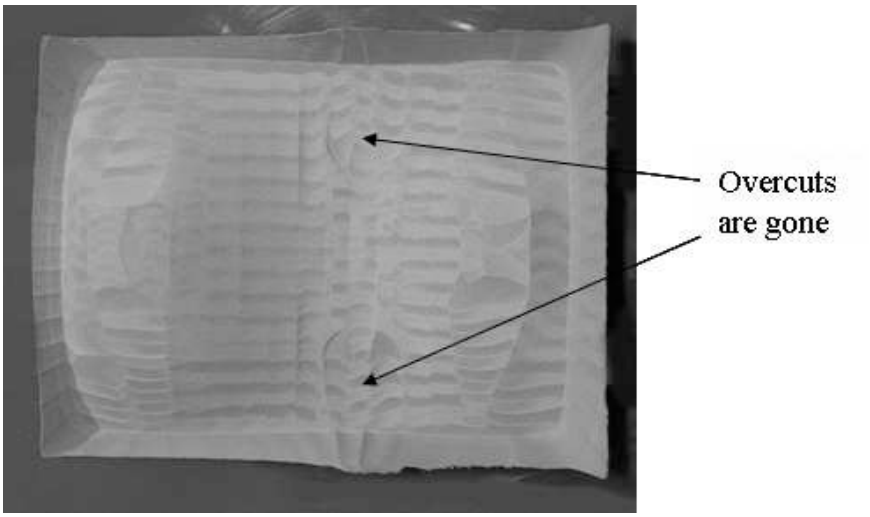


Fig. 6.15c. Optimization with regard to the overcut error, S_2 on MAHO 600E (corresponds to Fig. 6.14c)

$a_{p,p+1,i} = a_p + \Delta_{p,p+1} i$, where $\Delta_{p,p+1}$ is the corresponding step. The corresponding spatial positions s_i are found from the surface equation as follows. First of all, we specify a parametric spatial trajectory $T = T(s)$ on the surface through the stationary point. Next, for each i we solve numerically the equation $a(T(s)) = a_{p,p+1}$, where $a(T(s))$ denotes the rotation angle on $T(s)$ calculated from the surface normal.

6.6 Uniform Angular Grids: Numerical and Machining Experiments

Consider machining surface S_2 by MAHO 600E. We are allowed to "inject" additional points in order to decrease the error. First of all, consider the surface machined using the isoparametric tool path represented by a local uniform grid (see Fig. 6.14a). The large error loops appearing due to sharp angle variations can be eliminated by inserting the additional points in these areas using a uniform angular distribution of the points shown in Fig. 6.16b. The substantial reduction of the error (50 times!) has been achieved by inserting only 8 points inside each loop characterized by large error.

It may seem that inserting an equi-spaced set of points may lead to the same or similar result. However, it is not the case. The uniform spatial distribution of the points does not have a significant impact on the error. The error has been reduced only by 4 times by inserting the 8 additional points. Table 6.4 displays the kinematics errors and the rotation angles before and after ap-

plying the proposed point insertion. Clearly, the uniform angular grid allows to substantially decrease the error, whereas the conventional is not efficient.

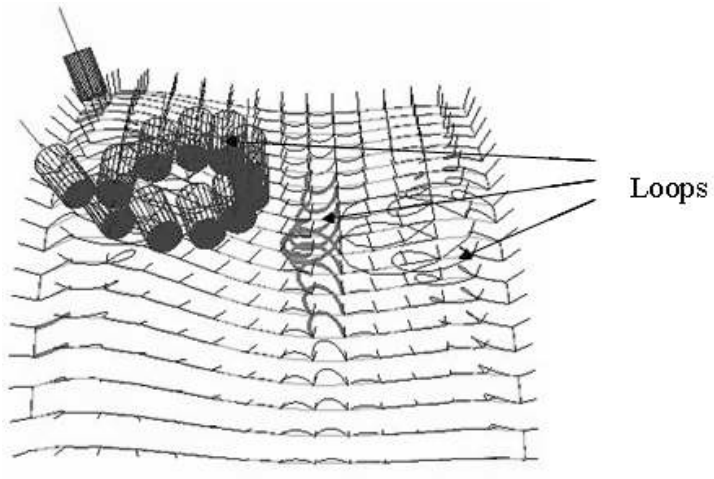


Fig. 6.16a. Tool path and tool orientations, S_2 on MAHO 600E, before insertion

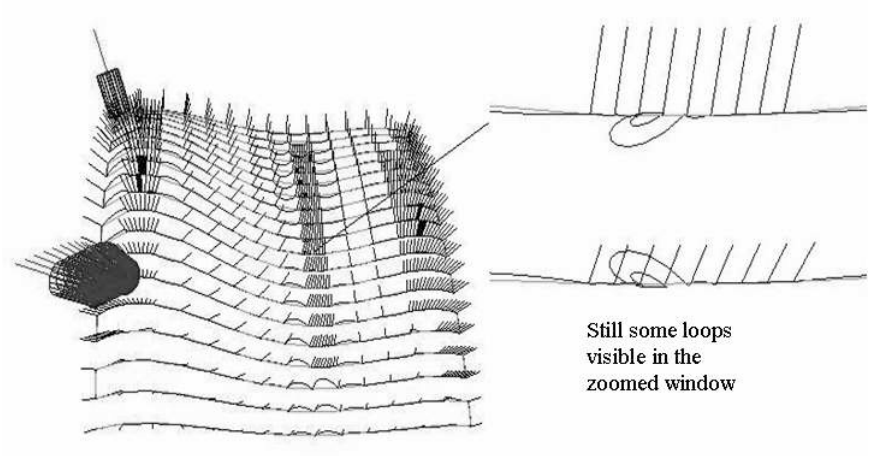


Fig. 6.16b. Tool path and tool orientations, S_2 on MAHO 600E, conventional point insertion

However, when a large number of the additional points have been inserted, the error decrease is approximately the same for the space or angular inserting. Therefore, the method applies to rough cuts characterized by sharp variations

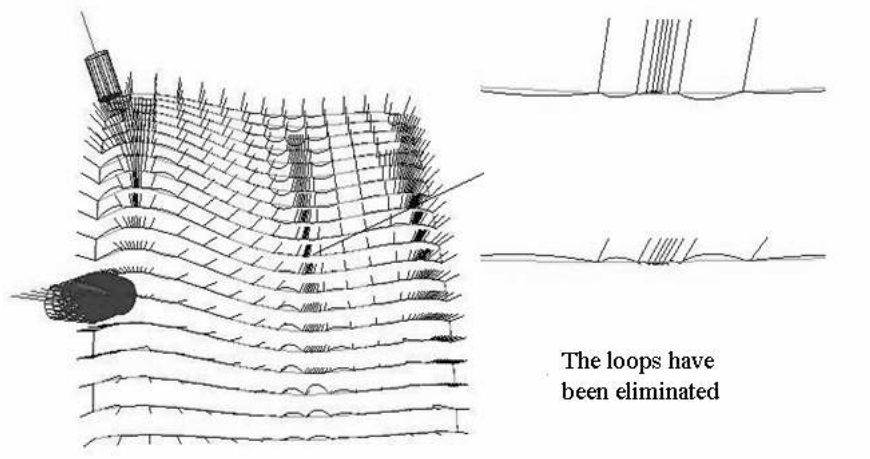


Fig. 6.16c. Tool path and tool orientations, S_2 on MAHO 600E, angular grid insertion

Table 6.4. Error versus number of inserted points, the basic grid size 15×20

Number of inserted points	Max error (mm) Conventional / Angular grid	Angular variation (deg) Conventional / Uniform angular grid
0	19.300 / 19.300	162.202 / 162.202
8	4.241 / 0.416	68.660 / 20.631
16	1.707 / 0.138	43.016 / 10.629
32	0.490 / 0.092	20.647 / 5.352
64	0.158 / 0.089	10.955 / 2.680
128	0.099 / 0.089	5.378 / 1.340

of the rotation angles. Figures 6.16(b-c) explain advantages of the angular grids. Clearly, the conventional uniform grid does not remove the kinematics loops although the amplitude has been decreased. As opposed to that, the angular grid changes the behavior of the trajectories by entirely eliminating the loops. Figures 6.17(a-b) show the difference between the workpieces machined using the conventional and the angular grid technologies.

The cutting experiments have been performed with the flat-end cutter selected due to its popularity for fast, rough milling when the sharp angular variations are the most expected. However, the angular grids are applicable to the ball nose cutter being often used for finish machining. As a matter of fact, the algorithms apply to a general APT cutter (Automatic Programmed Tool) without major modifications. The equations of the APT cutter include the most popular shapes such as the flat-end shaped cutter, toroidal cutter, the ball nose cutter (see the Appendix).

Combination of the two proposed methods might lead to even better optimizations. However usually, the angular grid should be constructed before the

shortest path procedure or in an iterative loop presented below. Even when the angular grid is applicable after the shortest path routine, the implementation is not as straightforward as it may seem since constructing the angular grid after the shortest path optimization requires to keep track of changes associated with every point. Furthermore, applying the angular grid after the optimal sequencing is often not possible because inserting even one point may create an entirely different graph for the shortest path optimization.

Let us illustrate this case by an example. Consider surface S_2 with the tool path obtained after the optimal sequencing (see Fig. 6.18). The largest loop between point 141 and 142, which can not be treated by the optimal sequencing, is on the right side of the surface. The angles before applying the algorithm are $a_{141} = 319°$, $b_{141} = -79°$, $a_{142} = 224°$, $b_{142} = -80°$. The shortest path optimization produces $a_{141} = 319°$, $b_{141} = -79°$, $a_{142,\text{new}} = a_{142} + 180° = 404°$, $b_{142,\text{new}} = -b_{142} - 180° = -100°$. Inserting a point in the middle of the loop yields $a_{\text{mid}} = 267°$, $b_{\text{mid}} = -82°$.

Taking into account the "history", we modify the pair of angles as follows $a_{\text{mid}} = a_{\text{mid}} + 180° = 447°$, $b_{\text{mid}} = -b_{\text{mid}} - 180° = -98°$ which leads to a larger loop depicted in Fig. 6.19. Note that $a_{\text{mid}} \notin [a_{141}, a_{142}]$ anymore. In other words, the additional point has destroyed the particular shortest path. Since the remaining part of the shortest path depends on these angles, the entire path has been destroyed and the optimization should be performed again.

The following procedure works very well when combining the two methods. First, we apply the optimal sequencing. Second, we select the largest loops appearing after applying the optimal sequencing. Next, we go back to the original angles and insert angular grids into the selected loops. Finally, we apply the optimal sequencing again to the modified tool path.

If the required tolerance has not been achieved, or even some new loops have appeared, we select the largest loops again and go back to inserting points into the original tool path.

Finally, apart from the machine kinematics, the accuracy of machining is often affected by the tool accelerations and decelerations due to frequent changes of the tool path directions. Since the phenomenon often occurs in the areas of sharp variations of the rotation angles, let us discuss the proposed method with the reference to the acceleration errors.

In order to reduce the acceleration error the entire tool path is usually treated as an interpolating curve parameterized with regard to the chord length between two consecutive reference points. Generating the tool positions by incrementing the chord length leads to the feedrate instabilities due to the difference between the chord and the arc length. The instabilities induce undesirable accelerations and jerk fluctuations (see our literature survey on interpolators in Chap. 1).

Our procedures apply to the regions where the angular variations are the most source of errors. However, the procedures do not treat the accelerations explicitly. Therefore, when such accelerations lead to large inaccuracies, the

proposed algorithms should be combined with the above mentioned spline interpolating methods. However, observe that reducing the *angular* variation leads to equal increments in the angular space. Therefore, it reduces *angular accelerations* appearing in the five-axis mode due to sharp variations of the tool orientations.

Ideally, in five-axis machining the tool path must be regarded as a curve in the five dimensional space and parameterized with regard to its arc length in 5D. Such parameterizations should be associated with several sources of error such as the kinematics error, errors due to accelerations, etc. Unfortunately, such parameterizations still constitute an open problem.

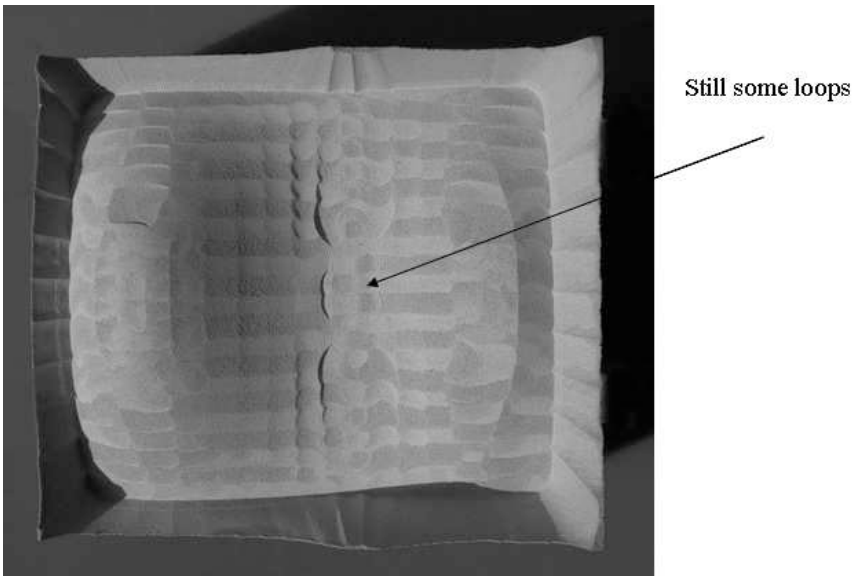


Fig. 6.17a. Spatial grid for S_2 on MAHO 600E (corresponds to Fig. 6.16b)

Appendix: The APT cutter

The cutter surface is composed of the truncated cone, torus and lower cone:

$$\Pi_C = \Pi_{TC} \cup \Pi_T \cup \Pi_{LC},$$

where

$$\Pi_{TC} = \begin{bmatrix} (R_1 + t\tan\beta_2)\sin\gamma \\ (R_1 + t\tan\beta_2)\cos\gamma \\ t + h - R_2\sin\beta_2 \end{bmatrix},$$

$\gamma \in [0, 2\pi]$, $t \in [h - R_2\sin\beta_2, L]$,

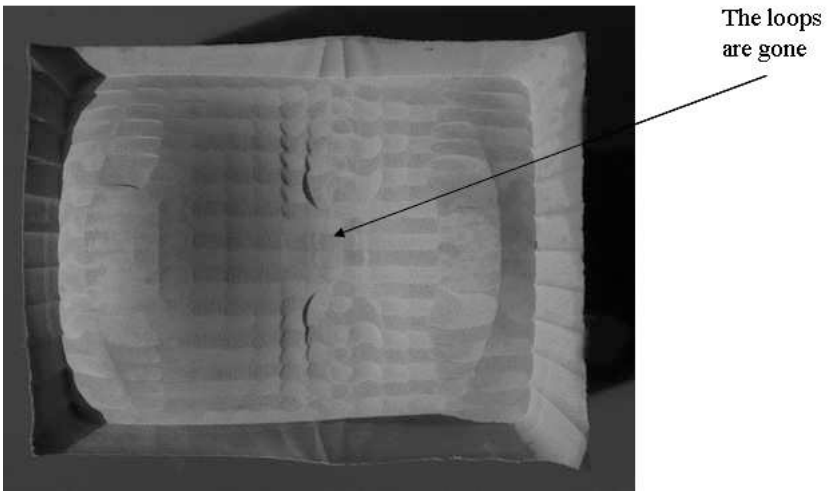


Fig. 6.17b. Angular grid insertion for S_2 on MAHO 600E (corresponds to Fig. 6.16c)

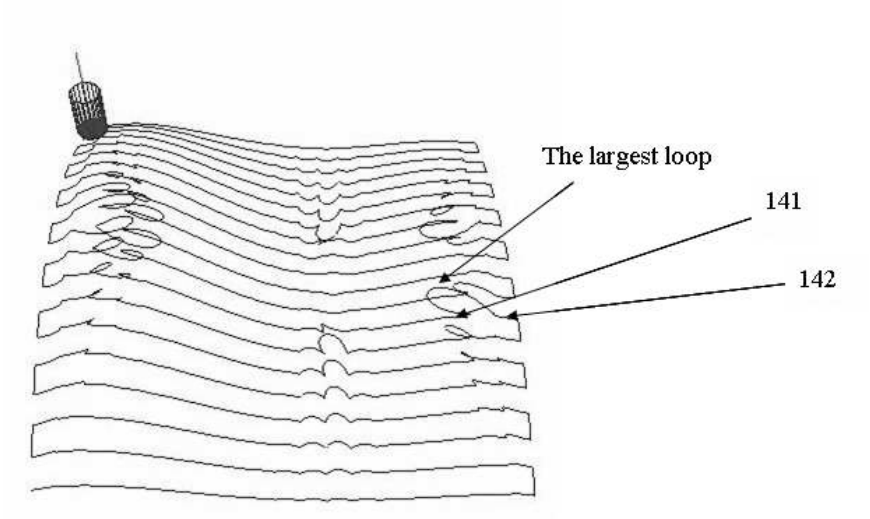


Fig. 6.18. Tool path for surface S_2 on MAHO 600E after the optimal sequencing

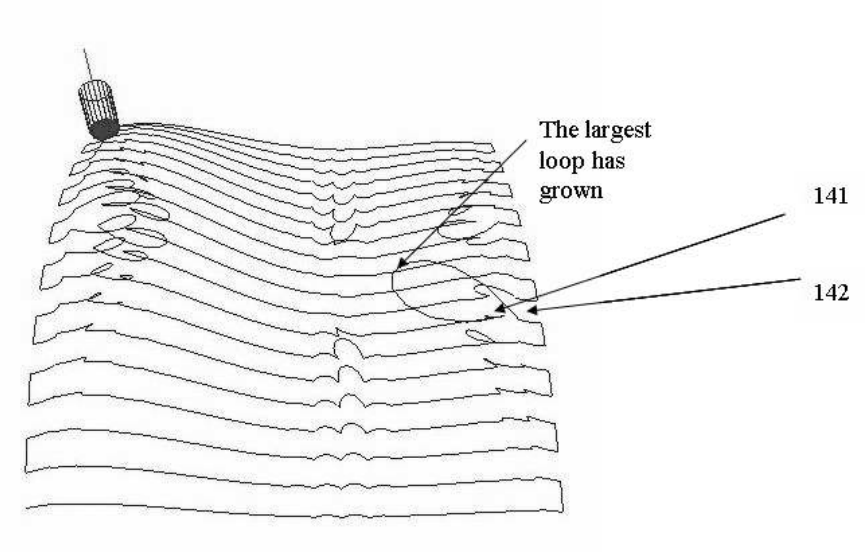


Fig. 6.19. Tool path for surface S_2 on MAHO 600E after the optimal sequencing and inserting one point into the largest loop

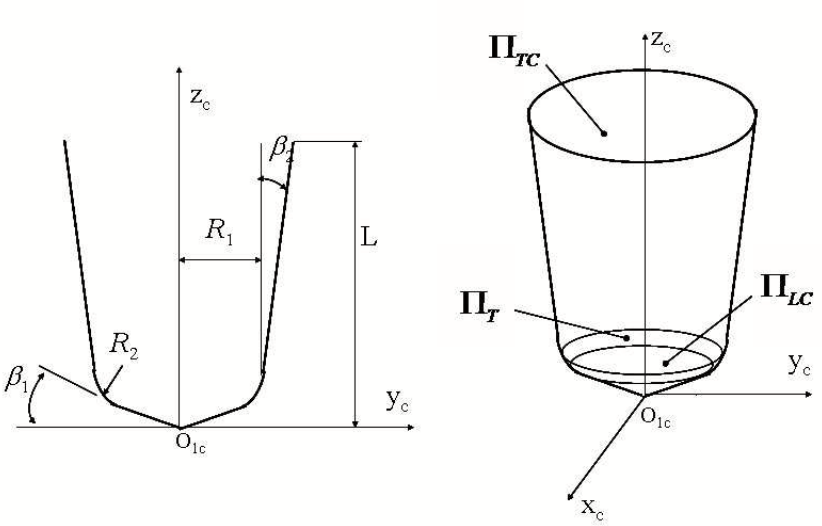


Fig. 6.20. APT cutter geometry

$h = R_2 \cos\beta_1 + R_2 \cos\beta_1 \tan\beta_1 + R_1 \tan\beta_1 - R_2 \tan\beta_1 \cos\beta_2$,

$$\Pi_T = \left\{ \begin{array}{c} x_c \\ y_c \\ z_c \end{array} \right\} = \left[\begin{array}{c} (e + R_2 \sin\chi)\sin\gamma \\ (e + R_2 \sin\chi)\cos\gamma \\ h - R_2 \cos\chi \end{array} \right],$$

$\chi \in [\beta_1, \frac{\pi}{2} - \beta_2]$, $e = R_1 - R_2 \cos\beta_2$,

$$\Pi_{LC} = \left\{ \begin{array}{c} x_c \\ y_c \\ z_c \end{array} \right\} = \left[\begin{array}{c} t\cot\beta_1 \sin\gamma \\ t\cot\beta_1 \cos\gamma \\ t \end{array} \right],$$

$t \in [0, h - R_2 \cos\beta_1]$.

In the case of specific fillet-end cutter, $\beta_1 = \beta_2 = 0$. The flat-end cutter is a particular case of the fillet-end cutter when $R_2 = 0$.

References

[1] Affouard, A., Duc, E., Lartigue, C., Langeron, J.-M., and Bourdet, P. 2004. Avoiding 5-axis singularities using tool path deformation. *International Journal of Machine Tools and Manufacture*, 44(4):415–425.

[2] Arkin, E. M., Bender, M. A., Demaine, E. D., Fekete, S. P., Mitchell, J. S. B., and Sethia, S. 2001. Optimal covering tours with turn costs. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001)*, pages 138–147.

[3] Arkin, E. M., Bender, M. A., Demaine, E. D., Fekete, S. P., Mitchell, J. S. B., and Sethia, S. 2005. Optimal covering tours with turn costs. *SIAM Journal on Computing*, 35(3):531–566.

[4] Jung, Y. H., Lee, D. W., Kim, J. S., and Mok, H. S. 2002. NC post-processor for 5-axis milling machine of table-rotating/tilting type. *Journal of Materials Processing Technology*, 130-131:641–646.

[5] Li, F., Wang, X. C., Ghosh, S. K., Kong, D. Z., Lai, T. Q., and Wu, X. T. 1995. Tool-path generation for machining sculptured surface. *Journal of Materials Processing Technology*, 48(1):811–816.

[6] Lo, C. C. 1999. Efficient cutter-path planning for five-axis surface machining with a flat-end cutter. *Computer-Aided Design*, 31(9):557–566.

[7] Makhanov, S. S., Batanov, D., Bohez, E., Sonthipaumpoon, K., Anotaipaiboon, W., and Tabucanon, M. 2002. On the tool-path optimization of a milling robot. *Computers & Industrial Engineering*, 43(3):455–472.

[8] Makhanov, S. S. and Ivanenko, S. A. 2003. Grid generation as applied to optimize cutting operations of the five-axis milling machine. *Applied Numerical Mathematics*, 46(3-4):331–351.

[9] Makhanov, S. S. and Munlin, M. Optimal sequencing of rotation angles for five-axis machining. *International Journal of Advanced Manufacturing Technology*. in press.

[10] Marchuk, G. I. 1982. *Methods of Numerical Mathematics*. Springer, New York.

[11] Munlin, M.-A., Makhanov, S. S., and Bohez, E. L. J. 2004. Optimization of rotations of a five-axis milling machine near stationary points. *Computer-Aided Design*, 36(12):1117–1128.

[12] Weiss, M. A. 1993. *Data structures and algorithm analysis in C.* Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA.

[13] Xu, H.-Y. 2003. Linear and angular feedrate interpolation for planar implicit curves. *Computer-Aided Design*, 35(3):301–317.

7

Theory of Optimal Setup for Five-Axis NC Machining

7.1 Introduction

In five-axis machining, the forward step (kinematics) error due to nonlinearity of the machine kinematics can significantly affect the quality of the machined surfaces. In a simple approach, kinematics error between two CC points can be reduced to within a prescribed tolerance by inserting enough points. However, this method is often inefficient, especially for high speed milling where a tool path containing a large number of points can slow down the milling process and is, therefore, undesirable. A technique of angle variation presented in Chap. 6 can also be used to minimize the kinematics error without inserting additional points. The error after minimization using this angle adjustment method can still, however, exceed the prescribed tolerance and more points or another minimization techniques are required.

In can be shown that an initial workpiece setup as well as parameters related to the machine configuration such as the position of each center of rotation and the tool length can affect the machining error due to the kinematics of the machine. Analysis of an initial setup is not currently provided by commercial CAD/CAM software tools such as Unigraphics, EdgeCam, and Vericut. However, in the academic literature, a few researchers have addressed the subject. Jung et al. [3] propose a general five-axis postprocessor, but the possibility of optimization has not been raised. Sijie et al. [6] optimize the position and orientation of the stock relative to the CAD modeling surface but machine related parameters were not optimized. Surveys and analysis of five-axis configuration are given in [2, 7, 8]. However, optimizing machining setups for specific surfaces has apparently been overlooked.

This chapter presents a new optimization model designed to minimize kinematics error introduced by the initial setup of a five-axis milling machine. An initial setup consists of the position and orientation of the workpiece with respect to the mounting table and, optionally, the machine's initial configuration. Depending on the type of the machine (see machine classification in Sect. 2.3), some parameters have no effect on the kinematics of the machine.

These parameters are termed invariant parameters (see Sect. 7.2.1). Furthermore, for each type of machine, there exists variables that are dependent and only one of these need to be optimized (see Sect. 7.3.2). After identifying the sets of invariant and dependent variables, a corresponding system of nonlinear equations is constructed and solved numerically.

For demonstration propose, the accuracy achieved through use of the proposed optimization of the initial setup is compared with the accuracy achieved through use of a "standard" positioning of the workpiece where a blank part is placed so that the bottom of the machined part is at a certain distance above the mounting table to ensure against collision avoidance. The center of the base is positioned at the center of the mounting table, and the **z**-axis in workpiece coordinates is perpendicular to the mounting table.

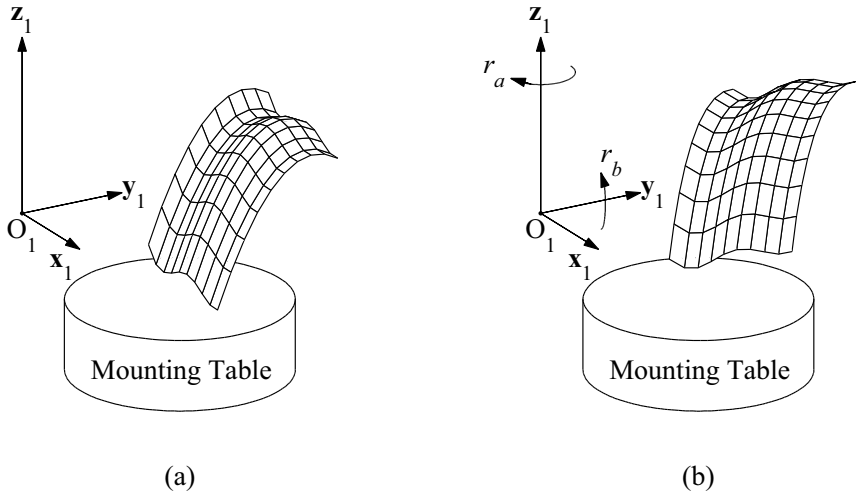


Fig. 7.1. Changing an initial workpiece setup by rotating around the $\mathbf{z}_1$-axis and the $\mathbf{y}_1$-axis

The following introductory example shows that optimization can be surprisingly effective. Let W_1 and W_2 be two successive spatial positions of the tool tip (CC points) in workpiece coordinates and let I_1 and I_2 be the corresponding tool orientations. The actual tool trajectory between W_1 and W_2 is generally nonlinear due to nonlinearity of the machine's kinematics; it also depends on the initial workpiece position and orientation with respect to the mounting table. Figure 7.1 shows two different initial workpiece setups. The workpiece orientation in Fig. 7.1(b) is obtained by first rotating the workpiece in Fig. 7.1(a) around the $\mathbf{z}_1$-axis by r_a and then rotating it around the $\mathbf{y}_1$-axis by r_b. Finally, it is shifted by T_{12}.

The new workpiece setup changes the cutter contact points and the orientations of the tool as follows:

$$\begin{aligned} W'_p &= R_b[r_b]R_a[r_a]W_p + T_{12}, \\ I'_p &= R_b[r_b]R_a[r_a]I_p, \end{aligned} \tag{7.1}$$

where R_a and R_b are the rotation matrices corresponding to r_a and r_b. Figure 7.2 illustrates the effect of workpiece setup on the tool's actual trajectory. Astonishingly, with the setup shown in Fig. 7.2(c), the error between the desired tool trajectory $W_{12}^D(t)$ and the actual tool trajectory $W_{12}(t)$ has been reduced by a factor of one thousand! (the error in the optimal setup is 0.03 mm). This error reduction cannot be achieved by using the method of angle adjustment alone (see Sect. 3.4 and Fig. 3.8 on Pages 63-67).

In general, the optimization parameters $\mathfrak{M}$ are (see Figs. 2.4-2.6):

1. The workpiece setup $\mathfrak{M}_S = \{T_{12}, r_a, r_b\}$, where $T_{12} = (T_{12,x}, T_{12,y}, T_{12,z})$ is the position of the origin of the workpiece coordinates with respect to O_2 (the coordinate system of the first rotary part) and r_a and r_b are the angles of rotation corresponding to the initial orientation of the workpiece.
2. The machine design $\mathfrak{M}_T = \{T_{23}, T_{34}, T_4\}$, where $T_{23} = (T_{23,x}, T_{23,y}, T_{23,z})$ denotes the position of the origin of the first rotary part in the coordinate system of the second rotary part, $T_{34} = (T_{34,x}, T_{34,y}, T_{34,z})$ denotes the position of the second rotary part in the spindle coordinate system, and T_4 is a vector describing the tool length. Note that the tool length L is being treated as a translation T_4 which is either $(0, 0, L)$ or $(0, 0, -L)$ depending on the position of the tool tip in the spindle coordinate system.

Consider the following optimization problem:

$$\min_{\mathfrak{M}} \ \epsilon, \tag{7.2}$$

where ϵ is the kinematics error defined as the difference between the desired and the actual trajectory of the tool tip. $\mathfrak{M} = \{\mathfrak{M}_S, \mathfrak{M}_T\}$ consists of the twelve parameters introduced above. It will be proven that only six parameters from the twelve need to be considered, since the remaining six parameters do not have any effect on the kinematics error. The actual identities of the six parameters to be ignored depend on the type of five-axis milling machine being used (see Sect. 2.3). The six parameters are selected as follows. First, for each type of machine, some of the setup parameters have no effect on the tool trajectory at all. These parameters can be immediately excluded from the optimization. Next, for each type of machine, an additional set of dependent parameters is found. Although each of these parameters does affect the tool trajectory when manipulated independently, each can be replaced by a linear combination of the other parameters, so these linearly dependent parameters can also be eliminated. Finally, the elimination of invariant and linearly dependent variables leads to a nonlinear system of equations that can be solved numerically by a Newton-Raphson procedure.

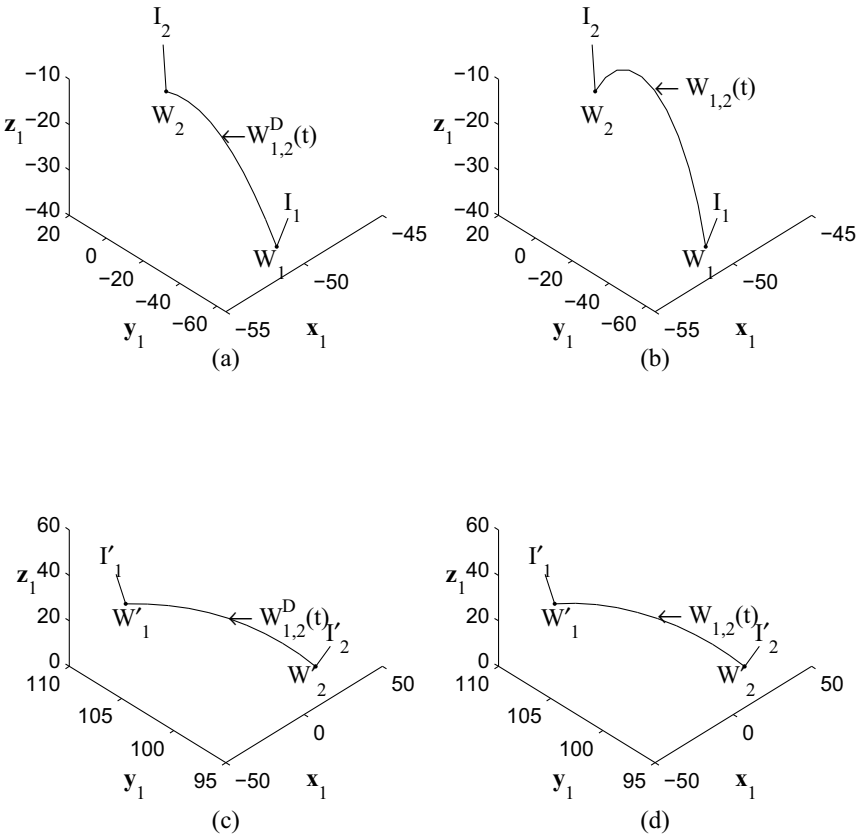


Fig. 7.2. The desired tool trajectory (**a**) and the actual tool trajectory before optimization (**b**). The desired tool trajectory (**c**) and the actual tool trajectory for the new (optimal) setup (**d**) (the workpiece is rotated by 99.23° around $\mathbf{z}_1$-axis and $-41.28°$ around the $\mathbf{y}_1$-axis and shifted by [21.47 49.33 33.67] mm). The desired and actual tool trajectories almost coincide in the optimal setup. The error has been reduced by 99.9%

The solution of this system establishes, for a particular machine and desired surface, not only the optimal position and orientation of the workpiece with respect to the mounting table, but also the optimal initial configuration of the five-axis machine. As previously mentioned, the method has proven extremely effective in many cases. Moreover, it requires nothing of the operator except changing the initial setup, which is a very simple, nearly zero-cost, operation.

7.2 Tool Trajectory Analysis

This section presents a theoretical analysis of the tool tip trajectory for each of the three types of five-axis machines. First, parameters that do not affect the tool trajectory are identified. These parameters will be excluded from the optimization procedure. Next, the effect of initial workpiece setup on tool trajectory is described.

7.2.1 Invariant Parameters

Five-axis machines are characterized by nonlinear kinematics due to the two additional degrees of freedom that control tool orientation. In general, the trajectory of the tool tip is nonlinear. The closed-form representation of the tool tip trajectory for general machine kinematics is given in (3.48). In each of the three types of five-axis machines (see Sect. 2.3), there are some parameters that have no effect on the tool trajectory. These parameters are identified by the following theorems.

Theorem 7.1. *The kinematics of the 2-0 machine implies that the tool trajectory between two arbitrary points is invariant with respect to T^*_{34} and $T_{23,y}$.*

Proof. Denote $A[a(t)]$ by A, $B[b(t)]$ by B, $A[a_p]$ by A_p, $B[b_p]$ by B_p. Represent $\mathfrak{K}^{-1}(W)$ by $\mathfrak{K}^{-1}(W) = GB\hat{\mathfrak{K}}^{-1}(W) + T^*_{34}$, where $\hat{\mathfrak{K}}^{-1}(W) = A(W + T_{12}) + T_{23}$ (see (2.5)). The tool tip trajectory (3.47) is then given by

$$W_{p,p+1}(t) = \mathfrak{K}(tGB_{p+1}\hat{\mathfrak{K}}^{-1}_{p+1} + tT^*_{34} + (1-t)GB_p\hat{\mathfrak{K}}^{-1}_p + (1-t)T^*_{34}), \quad (7.3)$$

where $\hat{\mathfrak{K}}^{-1}_p$ denotes $\hat{\mathfrak{K}}^{-1}(\mathfrak{M}, \mathfrak{R}_p, W_p)$. Clearly, from (2.5), $W = \mathfrak{K}(M) = A^{-1}(B^{-1}G^{-1}(M - T^*_{34}) - T_{23}) - T_{12}$. Consider $W = \mathfrak{K}(M) = \hat{\mathfrak{K}}(S) = A^{-1}(S - T_{23}) - T_{12}$ where $S = B^{-1}G^{-1}(M - T^*_{34})$. Rewriting (7.3) in terms of $\hat{\mathfrak{K}}(S)$ yields

$$\begin{aligned} W_{p,p+1}(t) &= \hat{\mathfrak{K}}(B^{-1}G^{-1}(tGB_{p+1}\hat{\mathfrak{K}}^{-1}_{p+1} + tT^*_{34} \\ &\qquad + (1-t)GB_p\hat{\mathfrak{K}}^{-1}_p + (1-t)T^*_{34} - T^*_{34})), \\ &= \hat{\mathfrak{K}}(tB^{-1}B_{p+1}\hat{\mathfrak{K}}^{-1}_{p+1} + (1-t)B^{-1}B_p\hat{\mathfrak{K}}^{-1}_p). \end{aligned} \quad (7.4)$$

Since neither $\hat{\mathfrak{K}}$ nor $\hat{\mathfrak{K}}^{-1}$ depends on T^*_{34}, $W_{p,p+1}(t)$ does not depend on T^*_{34} either. Therefore, the tool trajectory is invariant with respect to T^*_{34}.

Let us now prove the invariance with regard to $T_{23,y}$. Substituting $\hat{\mathfrak{K}}^{-1}(W) = A(W + T_{12}) + T_{23}$ and $\hat{\mathfrak{K}}(S) = A^{-1}(S - T_{23}) - T_{12}$ into (7.4) yields

$$
\begin{aligned}
W_{p,p+1}(t) &= \hat{\mathfrak{K}}(tB^{-1}B_{p+1}\hat{\mathfrak{K}}_{p+1}^{-1} + (1-t)B^{-1}B_p\hat{\mathfrak{K}}_p^{-1}),\\
&= \hat{\mathfrak{K}}(tB^{-1}B_{p+1}(A_{p+1}(W_{p+1}+T_{12})+T_{23})\\
&\quad + (1-t)B^{-1}B_p(A_p(W_p+T_{12})+T_{23})),\\
&= A^{-1}(tB^{-1}B_{p+1}(A_{p+1}(W_{p+1}+T_{12})+T_{23})\\
&\quad + (1-t)B^{-1}B_p(A_p(W_p+T_{12})+T_{23})-T_{23})-T_{12},\\
&= A^{-1}(tB^{-1}B_{p+1}A_{p+1}(W_{p+1}+T_{12})\\
&\quad + (1-t)B^{-1}B_pA_p(W_p+T_{12})\\
&\quad + tB^{-1}B_{p+1}T_{23} + (1-t)B^{-1}B_pT_{23} - T_{23}) - T_{12}.
\end{aligned}
\tag{7.5}
$$

The terms involving T_{23} in (7.5) can be written $(tB^{-1}B_{p+1}T_{23} + (1-t)B^{-1}B_pT_{23} - T_{23})$. Since B^{-1}, B_{p+1} and B_p are rotations around the **y**-axis, they do not change the y-component of T_{23}. Consequently, $(tB^{-1}B_{p+1}T_{23} + (1-t)B^{-1}B_pT_{23} - T_{23})_y = 0$. Therefore, the tool trajectory does not depend on $T_{23,y}$. This completes the proof of the Theorem.

Theorem 7.2. *The kinematics of the 1-1 machine implies that the tool trajectory is invariant with respect to $T_{12,z}$, T_{23} and $T_{34,y}$.*

The proof of Theorem 7.2 is analogous to that of Theorem 7.1.

Theorem 7.3. *The kinematics of the 0-2 machine implies that the tool trajectory is invariant with respect to T_{12} and $T_{23,y}$.*

The proof of Theorem 7.3 is analogous to that of Theorem 7.1.

7.2.2 Workpiece Setup and the Tool Trajectory

Changing the workpiece setup $\mathfrak{M}_S = \{T_{12}, r_a, r_b\}$ changes the coordinates of the cutter contact points, the orientations of the tool, and, consequently, the tool trajectory. Suppose the workpiece has been rotated clockwise by r_a around the $\mathbf{z_1}$-axis and by r_b around the $\mathbf{y_1}$-axis (Figure 7.1). The following theorems develop a closed-form representation of the tool trajectory for arbitrary r_a, r_b for the three basic types of the five-axis machines.

Theorem 7.4. *Let r_a and r_b be the rotation angles defining the orientation of the workpiece, and let R_a and R_b be the corresponding rotation matrices. The kinematics of the 2-0 machine is given by*

$$
M = GB[b']\,(A[a']\,(W' + T_{12}) + T_{23}) + T_{34}^{*}, \tag{7.6}
$$

where

$$
\begin{aligned}
W' &= R_b[r_b]R_a[r_a]W\\
&= \begin{bmatrix} \cos(r_b)\cos(r_a)x + \cos(r_b)\sin(r_a)y - \sin(r_b)z \\ -\sin(r_a)x + \cos(r_a)y \\ \sin(r_b)\cos(r_a)x + \sin(r_b)\sin(r_a)y + \cos(r_b)z \end{bmatrix},
\end{aligned}
$$

$$I' = \begin{bmatrix} \cos(a - r_a)\cos(b)\cos(r_b) + \sin(b)\sin(r_b) \\ \sin(a - r_a)\cos(b) \\ \cos(a - r_a)\cos(b)\sin(r_b) - \sin(b)\cos(r_b) \end{bmatrix}.$$

Proof. The relationships between the tool orientation $I = (I_x, I_y, I_z)$ and the two rotary angles a and b are given by (see Sect. 2.4)

$$\begin{aligned} I_x &= \cos(a)\cos(b), \\ I_y &= \sin(a)\cos(b), \\ I_z &= -\sin(b). \end{aligned} \tag{7.7}$$

After the rotations, the tool orientation becomes

$$I' = R_b[r_a]R_a[r_b]I. \tag{7.8}$$

After some algebraic manipulation, (7.8) becomes

$$I' = \begin{bmatrix} \cos(a - r_a)\cos(b)\cos(r_b) + \sin(b)\sin(r_b) \\ \sin(a - r_a)\cos(b) \\ \cos(a - r_a)\cos(b)\sin(r_b) - \sin(b)\cos(r_b) \end{bmatrix} \tag{7.9}$$

Theorem 7.5. *Let r_a and r_b be the rotation angles defining the orientation of the workpiece, and let R_a and R_b be the corresponding rotation matrices. The kinematics of the 1-1 machine is given by*

$$M = GA[a']\,(W' + T_{12}) + T_{23} + B^{-1}[b']T^*_{34}, \tag{7.10}$$

where

$$I' = \begin{bmatrix} \cos(a + r_a)\sin(b)\cos(r_b) - \cos(b)\sin(r_b) \\ -\sin(b)sin(a + r_a) \\ \cos(a + r_a)sin(b)\sin(r_b) + \cos(b)\cos(r_b) \end{bmatrix}.$$

The proof is analogous to that of Theorem 7.4.

Theorem 7.6. *Let r_a and r_b be the rotation angles defining the orientation of the workpiece, and let R_a and R_b be the corresponding rotation matrices. The kinematics of the 0-2 machine is given by*

$$M = GW' + T_{12} + A^{-1}[a'](T_{23} + B^{-1}[b']T^*_{34}) \tag{7.11}$$

where

$$I' = \begin{bmatrix} (\cos(b)\cos(r_a) - \sin(a)\sin(b)\sin(r_a))\cos(r_b) - \cos(a)\sin(b)\sin(r_b) \\ -\cos(b)\sin(r_a) - \sin(a)\sin(b)\cos(r_a) \\ (\cos(b)\cos(r_a) - \sin(a)\sin(b)\sin(r_a))\sin(r_b) + \cos(a)\sin(b)\cos(r_b) \end{bmatrix}.$$

The proof is analogous to that of Theorem 7.4.

Finally, substituting the new rotation angles $\mathfrak{R}'_p = (a'_p, b'_p)$ obtained from I' and the new coordinates W' into (3.48) yields

$$\begin{aligned} W'_{p,p+1}(t) = \mathfrak{K}(\mathfrak{M}, t\mathfrak{R}'_{p+1} + (1-t)\mathfrak{R}'_p, \\ t\mathfrak{K}^{-1}(\mathfrak{M}, \mathfrak{R}'_{p+1}, W'_{p+1}) + (1-t)\mathfrak{K}^{-1}(\mathfrak{M}, \mathfrak{R}'_p, W'_p)), \end{aligned} \tag{7.12}$$

where $\mathfrak{K}$ corresponds to either (2.2) or (2.7) or (2.10).

7.3 Least-Squares Optimization and Dependent Variables

This section introduces a definition of the kinematics error in the least-squares sense. The optimal setup is defined as a set of optimization parameters minimizing the kinematics error. A system of non-linear equations to minimize the error is deduced and analyzed. The dependent optimization parameters are excluded from the minimization procedure. Finally, the minimal set of optimization parameters is presented for each machine type.

7.3.1 Least-Squares Optimization

Recall that $W^D_{p,p+1}(t) \equiv (x^D_{p,p+1}(t), y^D_{p,p+1}(t), z^D_{p,p+1}(t)) \in S(u,v)$ is defined as a curve between W_p and W_{p+1} extracted from the surface in such a way that it represents the desired tool trajectory between Π_p and Π_{p+1}. The error is represented as a deviation between $W^D_{p,p+1}(t)$ and $W_{p,p+1}(t) \equiv (x_{p,p+1}(t), y_{p,p+1}(t), z_{p,p+1}(t))$, namely,

$$\begin{aligned}\epsilon &= \sum_p \int_0^1 \left|W^D_{p,p+1}(t) - W_{p,p+1}(t)\right|^2 dt,\\ &= \sum_p \int_0^1 \Big[\big(x^D_{p,p+1}(t) - x_{p,p+1}(t)\big)^2 + \big(y^D_{p,p+1}(t) - y_{p,p+1}(t)\big)^2 \\ &\qquad + \big(z^D_{p,p+1}(t) - z_{p,p+1}(t)\big)^2\Big]\, dt,\end{aligned} \tag{7.13}$$

where

$$\begin{aligned}W_{p,p+1}(t) = R_a^{-1}[r_a]R_b^{-1}[r_b]\mathfrak{K}(\mathfrak{M}, t\mathfrak{R}'_{p+1} + (1-t)\mathfrak{R}'_p,\\ t\mathfrak{K}^{-1}(\mathfrak{M}, \mathfrak{R}'_{p+1}, R_b[r_b]R_a[r_a]W_{p+1})\\ + (1-t)\mathfrak{K}^{-1}(\mathfrak{M}, \mathfrak{R}'_p, R_b[r_b]R_a[r_a]W_p)).\end{aligned}$$

Consider now the least-squares minimization problem described by (7.2). Differentiating with regard to the optimization parameters yields

$$\frac{\partial \epsilon}{\partial \nu_p} = \sum_p \int_0^1 \left|W^D_{p,p+1}(t) - W_{p,p+1}(t)\right| \frac{\partial W_{p,p+1}(t)}{\partial \nu_p} dt = 0, \tag{7.14}$$

where $\nu = (\nu_1, \ldots, \nu_n)$ is a vector comprising the optimization parameters. The number of parameters n is at most six, but if the optimization is only performed with respect to the machine setup, the number of the parameters will vary with the type of machine.

7.3.2 Dependent Variables

Let us now analyze dependent variables. As an example, consider the 2-0 machine. Observe that $T_{34} - T_4$ has been replaced by one variable T^*_{34} (see Sect. 2.4). Consequently, if $T^*_{34} = T^*_{34,o}$ is a component of the solution of the transformed optimization problem then any T_{34} and T_4 that satisfy $T_{34} - T_4 = T^*_{34,o}$ are components of the solutions of the original optimization problem. In terms of the cost function, the dependence is expressed as follows: ν_1 and ν_2 are dependent if $\epsilon(\nu_1, \nu_2, \nu_3, \ldots, \nu_n) \equiv \hat{\epsilon}(c_1\nu_1 + c_2\nu_2, \nu_3, \ldots, \nu_n)$, where c_1 and c_2 are constants. In this case $\dfrac{\partial\hat{\epsilon}}{\partial v_1} = 0$ and $\dfrac{\partial\hat{\epsilon}}{\partial v_2} = 0$ produce the same equations. Therefore, $c_1\nu_1 + c_2\nu_2$ can be replaced by one variable. When the optimal solution to the transformed problem is found, either ν_1 or ν_2 is considered a free variable.

The definition of dependence can be easily generalized to the case of n dependent variables. It will now be established that (7.13) for the kinematics in (2.2) has an additional, less obvious, pair of dependent variables.

Theorem 7.7. *$T_{12,z}$ and $T_{23,z}$ are linearly dependent for the 2-0 machine.*

Proof. Consider a tool trajectory given by $W_{p,p+1}(t) = \hat{\mathfrak{K}}(tB^{-1}B_{p+1}\hat{\mathfrak{K}}^{-1}_{p+1} + (1-t)B^{-1}B_p\hat{\mathfrak{K}}^{-1}_p)$, where $\hat{\mathfrak{K}}^{-1}_p = A_p(W_p + T_{12}) + T_{23}$ (see (7.4)). Since A_p and A_{p+1} are rotations around the **z**-axis, they do not change the z-component of T_{12}. Therefore, $(A_pT_{12} + T_{23})_z = (A_{p+1}T_{12} + T_{23})_z = T_{12,z} + T_{23,z}$. Hence, $(A_pT_{12} + T_{23})_z$ and $(A_{p+1}T_{12} + T_{23})_z$ can be replaced by one variable. Furthermore, since $\hat{\mathfrak{K}}(S) = A^{-1}(S - T_{23}) - T_{12}$, $(A^{-1}T_{23} + T_{12})_z = T_{23,z} + T_{12,z}$ can be replaced by the same variable. Consequently, $T_{12,z}$ and $T_{23,z}$ are dependent.

Remark 7.1. The dependency of $T_{12,z}$ and $T_{23,z}$ is illustrated by Fig. 7.3 (see also Fig. 2.4 for comparison). In words, since $T_{12,z} + T_{23,z} = c$ (c is the distance between the center of the workpiece coordinate system O_1 and the center of the coordinate system of the second rotary part O_3), the optimal workpiece position relative to the second rotary part is fixed. However, the height of the mounting table can be reduced or increased along the $\mathbf{z}_2$-axis (the O_2 coordinate system).

Consider now the 1-1 and the 0-2 machines.

Theorem 7.8. *T_{34} and T_4 are linearly dependent for the 1-1 machine.*

The proof is analogous to that of Theorem 7.7.

Theorem 7.9. *$T_{23,x}$ and $T_{34,x}$ are linearly dependent for the 0-2 machine.*

The proof is analogous to that of Theorem 7.7.

The results presented in this section are summarized in Table 7.1.

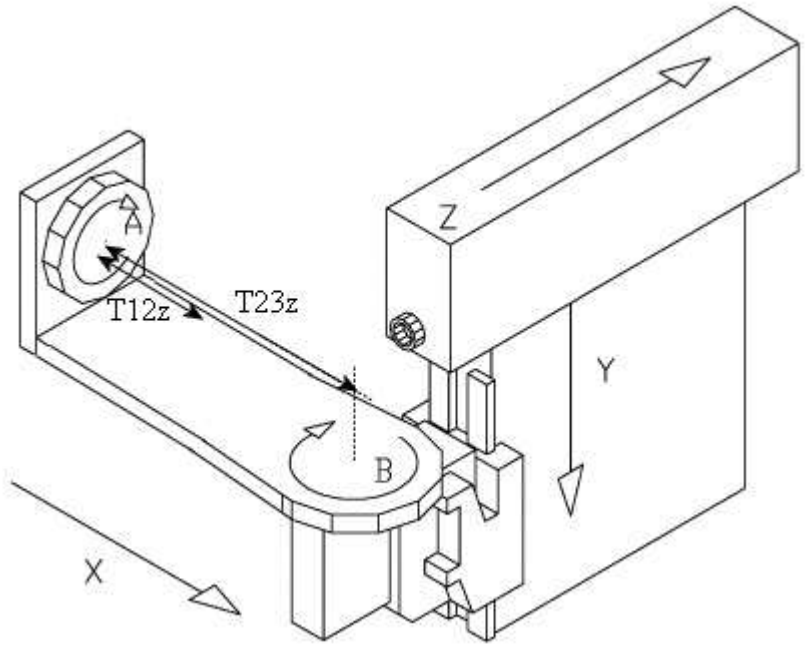


Fig. 7.3. Dependent parameters $T_{12,z} + T_{23,z} = c$ of the 2-0 machine

Table 7.1. Minimal sets of optimization parameters

	Optimization parameters		
Machine type	Workpiece setup	Tool length	Machine settings
The 2-0 machine	$r_a, r_b, T_{12,x}, T_{12,y}, T_{12,z}$	none	$T_{23,x}$
The 1-1 machine	$r_a, r_b, T_{12,x}, T_{12,y}$	$T_{4,z}$	$T_{34,x}$
The 0-2 machine	r_a, r_b	$T_{4,z}$	$T_{23,x}, T_{23,y}, T_{34,y}$

7.4 Examples and Discussion

7.4.1 Numerical Method

This section describes a numerical solution of system (7.14) by a standard Newton-Raphson procedure. Consider a system $F(v) = 0$, where F is a vector function deduced from (7.14). A number of numerical methods can be used to solve this system of nonlinear equations [1]. Among the simplest is the standard Newton method given by

$$\nu^{\text{new}} = \nu^{\text{old}} - J^{-1} F(\nu^{\text{old}}), \tag{7.15}$$

where J is the Jacobi matrix. It is well known that the performance of Newton's method depends on the initial approximation. The process may not

converge, and the global optimum may not be obtained in the case of multiple optima. To counter these limitations, Newton's method is executed several times with a different initial approximation in each run. Starting points are arranged in a regular grid pattern or sampled from a uniform distribution over feasible parameter space. The required derivatives of the objective function with respect to the rotation angles are obtained with the chain rule, as follows:

$$\frac{\partial \epsilon(r_a, r_b)}{\partial r_L} = \frac{\partial \epsilon}{\partial r_L} + \frac{\partial \epsilon}{\partial a'_p}\frac{\partial a'_p}{\partial r_L} + \frac{\partial \epsilon}{\partial b'_p}\frac{\partial b'_p}{\partial r_L} + \frac{\partial \epsilon}{\partial a'_{p+1}}\frac{\partial a'_{p+1}}{\partial r_L} + \frac{\partial \epsilon}{\partial b'_{p+1}}\frac{\partial b'_{p+1}}{\partial r_L}, \quad (7.16)$$

where $L = a$ or $L = b$. a' and b' are machine rotation angles which are function of r_a and r_b (see Theorems 7.4-7.6). The next section demonstrates the performance of this simple approach with several experiments.

7.4.2 Examples

To demonstrate the performance of the proposed procedure, the optimization is applied to two test surfaces, a sweep surface [4, 5] (Figure 7.4(a)) and a two-bell surface (Figure 7.4(b)). The sweep surface is given by

$$\begin{aligned} x &= 100u - 50, \\ y &= 100v - 50, \\ z &= -40(v - 0.8)^2 - 50\sin(3u - 0.3)\sin(u - 0.5)\sin(u - 0.8) - 5. \end{aligned} \quad (7.17)$$

The two-bell surface is given by

$$\begin{aligned} x &= 100u - 50, \\ y &= 100v - 50, \\ z &= 400v(1 - v)(3.55u - 14.8u^2 + 21.15u^3 - 9.9u^4) - 28. \end{aligned} \quad (7.18)$$

where $0 \leq u, v \leq 1$.

Let $W_p = S(u_p, v_p)$ and $W_{p+1} = S(u_{p+1}, v_{p+1})$ be two successive CC points. The desired curve between the two CC points is extracted from the surface as follows:

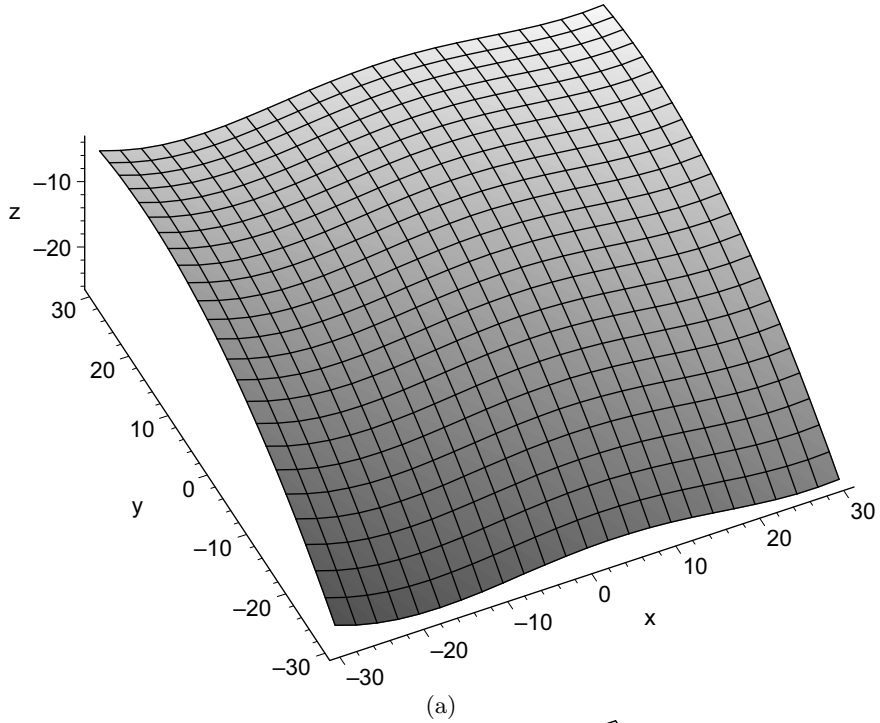
$$W^D_{p,p+1}(t) = S((1 - t)u_p + tu_{p+1}, (1 - t)v_p + tv_{p+1}), \quad (7.19)$$

where t is a fictitious time coordinate along the curve ($0 \leq t \leq 1$). The mean error is an approximation of (7.13), given by

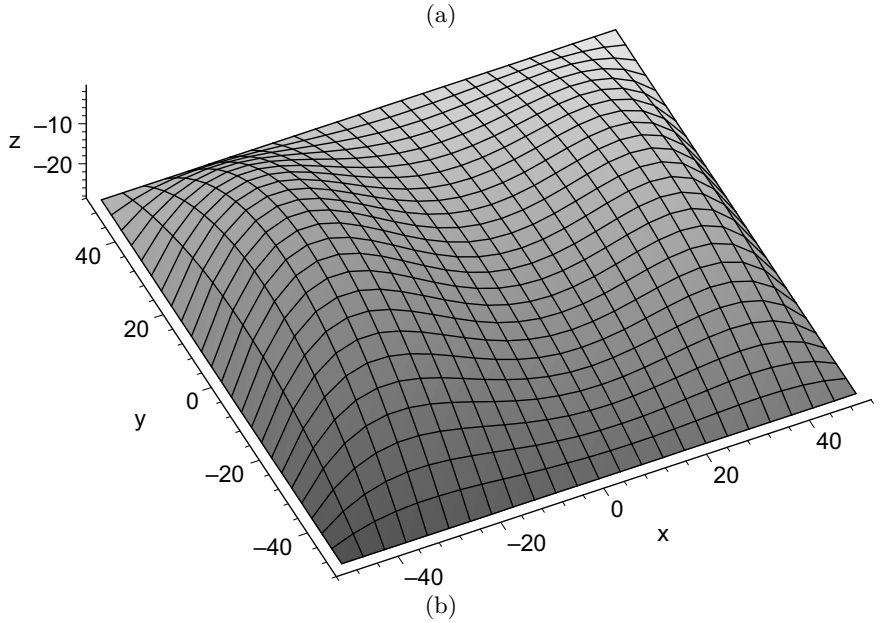
$$\bar{\epsilon} = \frac{1}{N_{pt}} \sum_p \sum_t \left| W^D_{p,p+1}(t) - W_{p,p+1}(t) \right|^2, \quad (7.20)$$

where N_{pt} is the total number of sampled points.

The performance of the optimization is presented by Table 7.2 (the sweep surface) and Table 7.3 (the two-bell surface).



(a)



(b)

Fig. 7.4. Test surfaces: (**a**) sweep surface, (**b**) two-bell surface

Table 7.2. Performance of the optimization method, the sweep surface

Machine type	The mean error (mm) / reduction			
	Before optimization	Optimization wrt. the workpiece setup	Optimization wrt. the workpiece setup and the tool length	Optimization wrt. the entire set
The 2-0 machine	0.5730	0.0179 / 96.88%	-	0.0176 / 96.92%
The 1-1 machine	0.5785	0.0167 / 97.12%	0.0146 / 97.47%	0.0145 / 97.49%
The 0-2 machine	0.0522	0.0396 / 24.14%	0.0273 / 47.78%	0.0160 / 69.34%

Table 7.3. Performance of the optimization method, the two-bell surface

Machine type	The mean error (mm) / reduction			
	Before optimization	Optimization wrt. the workpiece setup	Optimization wrt. the workpiece setup and the tool length	Optimization wrt. the entire set
The 2-0 machine	0.5408	0.1118 / 79.33%	-	0.1102 / 79.62%
The 1-1 machine	0.6208	0.5064 / 18.43%	0.2136 / 65.59%	0.2133 / 65.64%
The 0-2 machine	0.5564	0.5403 / 2.90%	0.1572 / 71.75%	0.0454 / 91.83%

The desired and actual tool trajectory obtained for the sweep surface is depicted in Figs. 7.5-7.7 along with the corresponding error. The tool trajectories are depicted in the original workpiece coordinates, i.e., each tool trajectory, beginning from the optimal setup, is transformed back to workpiece coordinates for comparison. The proposed optimization is compared with the results obtained by traditional positioning of the workpiece, in which the origin of the workpiece coordinates is aligned with the center of the mounting table, and the **z**-axis is perpendicular to the mounting table. The results reveal that the error can be reduced by as much as 97%. Table 7.3 shows a similar accuracy increase achieved for the geometrically complex two-bell surface.

The sweep surface produced by MAHO600E (a 2-0 machine, Fig. 2.1) with and without optimization is shown in Fig. 7.8. The surface in Fig. 7.8(d) was produced with a setup optimized with regard to r_a and r_b. The optimization leads to an error reduction of about 81.45%. Note that in order to better visualize the errors, a very small number of CC points was used (102 points).

Another measure to evaluate the performance of optimization is the number of CC points required to maintain the kinematics error within a prescribed tolerance. This estimate is in particular important for high speed milling when an increase in the number of the points leads to a substantial increase in the machining time.

Table 4 displays the number of CC points required to cut the workpiece to within a prescribed maximum machining error. The optimal setup applied to both surfaces, reduces the number of CC points by 68% and 30%, respectively. Note that although the reduction achieved for the two-bell surface is not particularly large, the optimization is still appropriate since it requires nothing of the operator but changing the initial setup. This is, a simple, nearly zero cost, operation.

Table 7.4. Performance of the optimization measured by the required number of CC points

	Number of CC points / reduction	
Surface	Before optimization	Optimization wrt. r_a and r_b
Sweep surface	3900	1233 / 68.4%
Two-bell surface	7925	5544 / 30.0%

2-0 machine. The maximum allowed machining error = 0.01 mm.

References

[1] Bertsekas, D. P. 1995. *Nonlinear Programming.* Athena Scientific, Belmont, MA, USA.

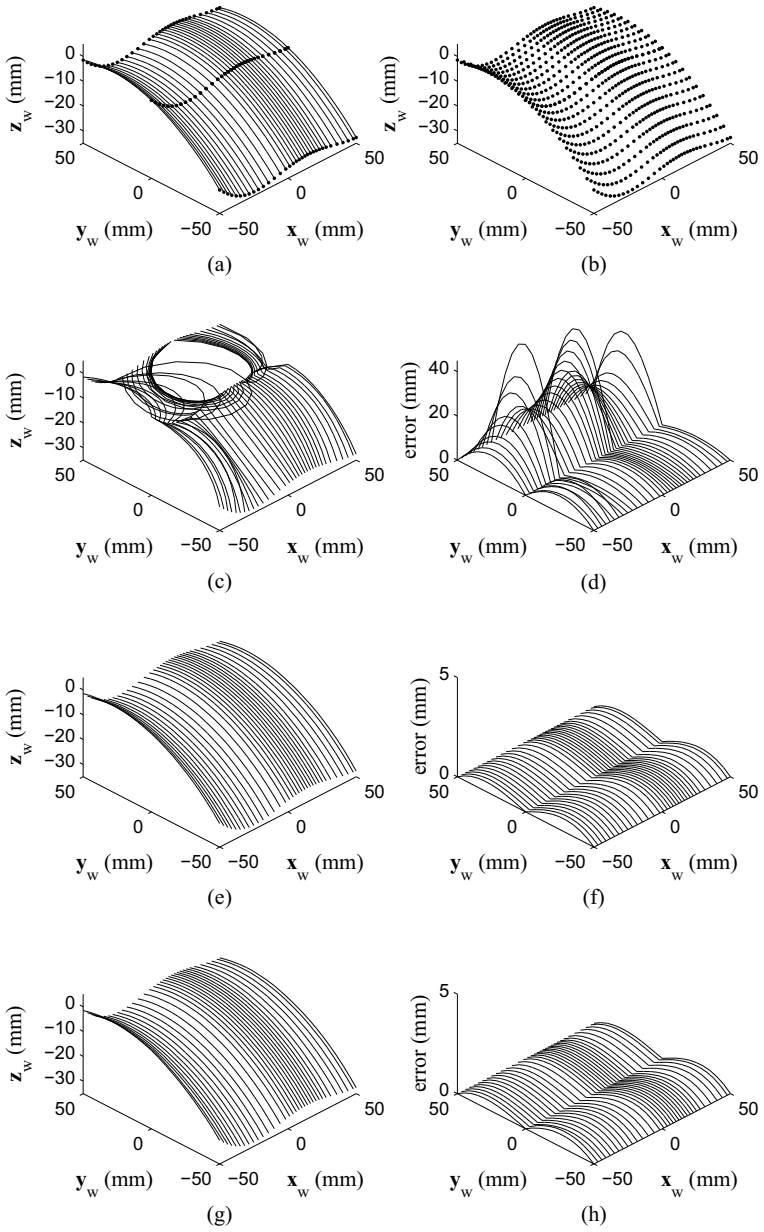


Fig. 7.5. (**a**) CC points on the desired tool trajectory, (**b**) the extracted tool trajectory (sampling points), (**c**) the actual tool trajectory on the 2-0 machine before optimization and (**d**) the error, (**e**) the tool trajectory with an optimal setup r_a, r_b, T_{12} and (**f**) the error, (**g**) the tool trajectory with an optimal setup r_a, r_b, T_{12}, T_{23} and (**h**) the error

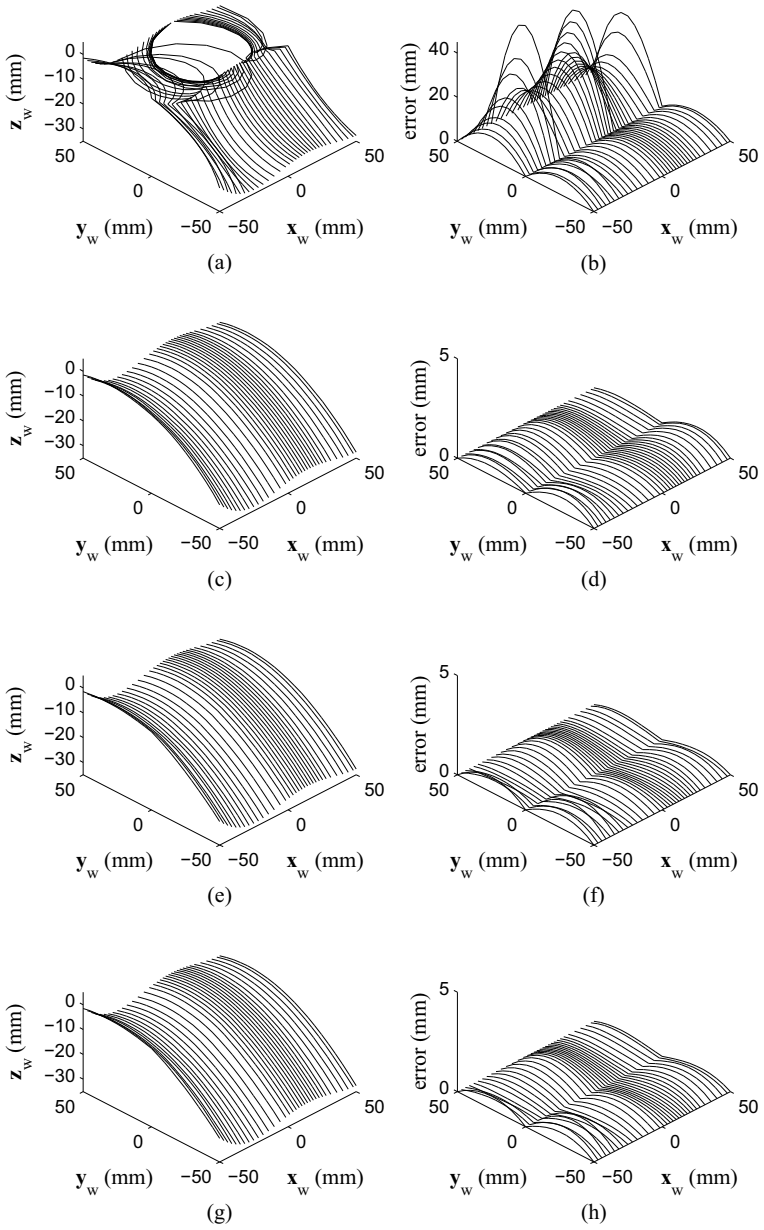


Fig. 7.6. (**a**) the actual tool trajectory on the 1-1 machine before optimization and (**b**) the error, (**c**) the tool trajectory with an optimal setup r_a, r_b, T_{12} and (**d**) the error, (**e**) the tool trajectory with an optimal setup r_a, r_b, T_{12}, T_4 and (**f**) the error, (**g**) the tool trajectory with an optimal setup r_a, r_b, T_{12}, T_{34}, T_4 and (**h**) the error

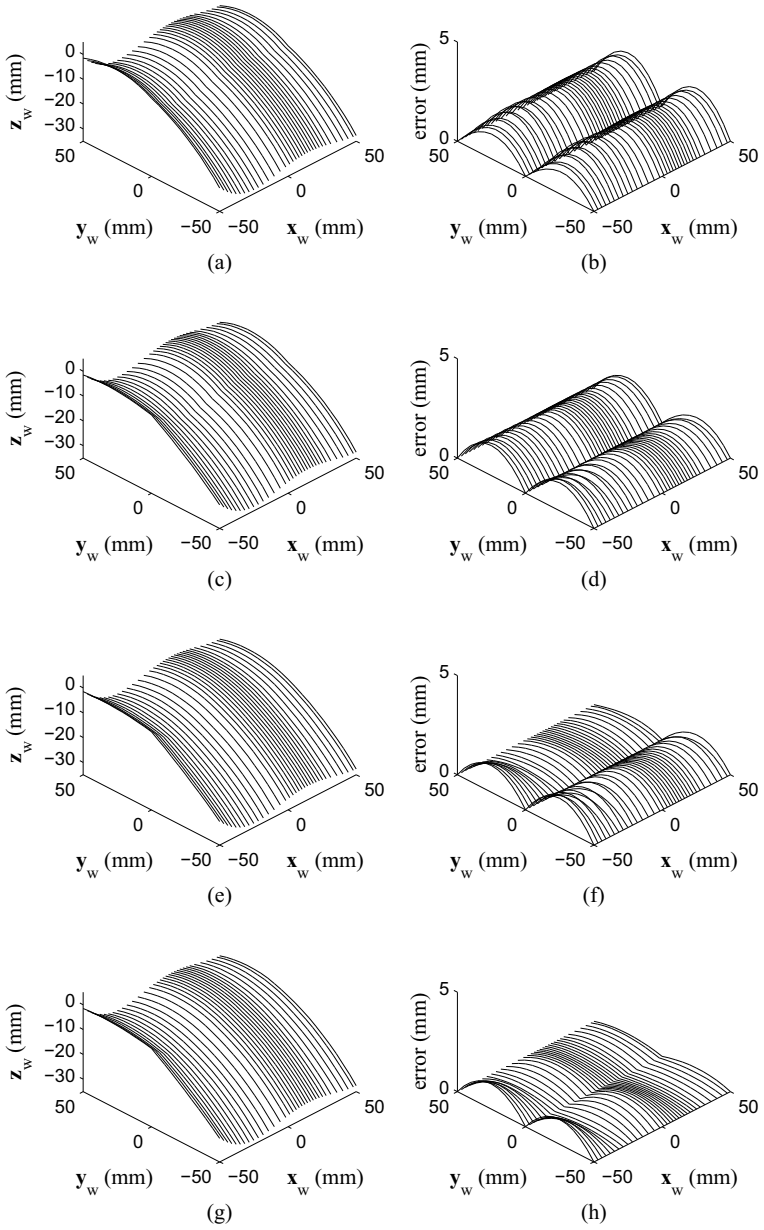


Fig. 7.7. (**a**) the actual tool trajectory on the 0-2 machine before optimization and (**b**) the error, (**c**) the tool trajectory with an optimal setup r_a, r_b and (**d**) the error, (**e**) the tool trajectory with an optimal setup r_a, r_b, T_4 and (**f**) the error, (**g**) the tool trajectory with an optimal setup r_a, r_b, T_{23}, T_{34}, T_4 and (**h**) the error

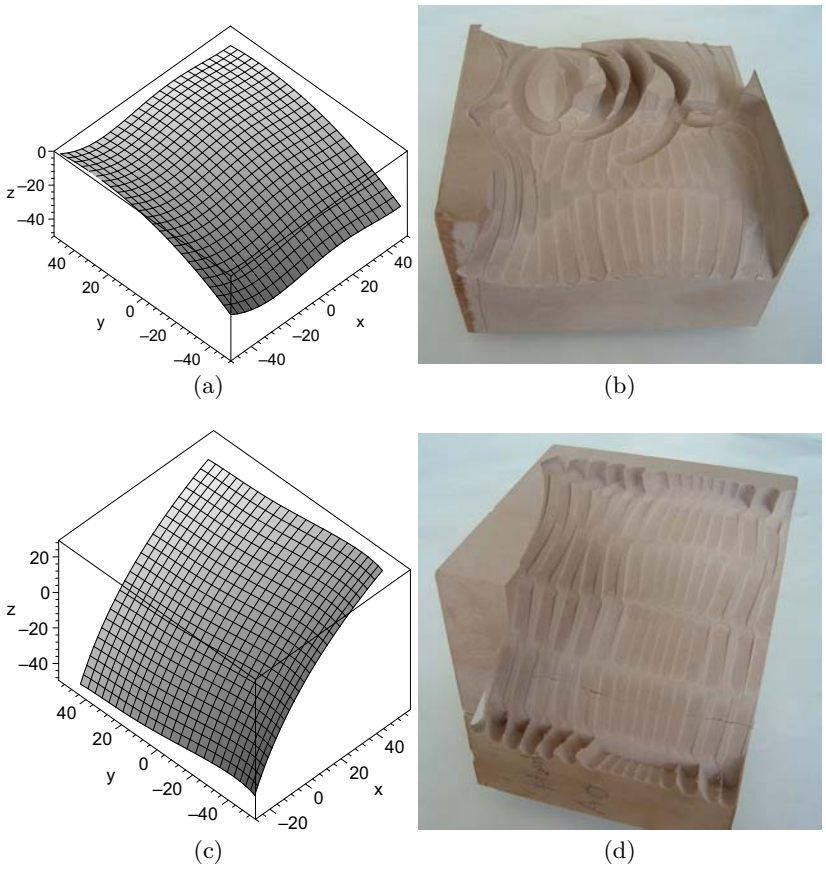


Fig. 7.8. The sweep surface machined with different setups. The workpiece on the top (**a**) and (**b**) has been cut with a conventional setup ($r_a = 0$, $r_b = 0$). The workpiece at the bottom (**c**) and (**d**) has been cut with $r_a = 1.6008$ radian and $r_b = 0.5236$ radian (the optimal setup with regard to r_a and r_b)

[2] Bohez, E. L. J. 2002. Five-axis milling machine tool kinematic chain design and analysis. *International Journal of Machine Tools and Manufacture*, 42(4):505–520.

[3] Jung, Y. H., Lee, D. W., Kim, J. S., and Mok, H. S. 2002. NC post-processor for 5-axis milling machine of table-rotating/tilting type. *Journal of Materials Processing Technology*, 130-131:641–646.

[4] Piegl, L. and Tiller, W. 1995. *The NURBS book*. Springer-Verlag, London, UK.

[5] Rogers, D. F. 2001. *An introduction to NURBS: with historical perspective*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[6] Sijie, Y., Yunfei, Z., Fangyu, P., and Xide, L. 2004. Research on the localisation of the workpieces with large sculptured surfaces in NC machining.

The International Journal of Advanced Manufacturing Technology, 23(5-6):429–435.

[7] Tutunea-Fatan, O. R. and Feng, H.-Y. 2004. Configuration analysis of five-axis machine tools using a generic kinematic model. *International Journal of Machine Tools and Manufacture*, 44(11):1235–1243.

[8] Warkentin, A., Hoskins, P., Ismail, F., and Bedi, S. 2001. Computer-aided 5-axis machining. In *Systems Techniques and Computational Methods*, volume 1 of *Computer-Aided Design, Engineering, and Manufacturing: Systems Techniques and Applications*, chapter 3, pages 3001–3034. CRC Press, Inc., Boca Raton, FL, USA.

Index