Engineering with Computers 8, 163–176 (1992)

Engineering with Computers

New and Effective Remeshing Scheme for the Simulation of Metal Forming Processes

R. Schneiders,[1] W. Oberschelp
Lehrstuhl für Angewandte Mathematik insb. Informatik, Aachen, F.R. Germany

R. Kopp, M. Becker[1]
Institut für Bildsame Formgebung, Aachen, F.R. Germany

Abstract. Remeshing is an important problem encountered often in the FEM-simulation of metal forming processes. This paper describes the remeshing scheme developed at LAMI and IBF for the FINEL FEM-program. A novel automatic, two-dimensional, quadrilateral mesh generator is introduced. Mesh generation is performed by means of image processing and computational geometry. Other features of the remeshing scheme are boundary simplification, smoothing, and rezoning. Examples of applications are given.

1 Introduction

The finite element method (FEM) can now be successfully used for the simulation of metal forming processes with complicated die geometries. At Institut für Bildsame Formgebung (IBF) the FEM program FINEL was developed for the coupled thermal–mechanical 2-D analysis of the forging process [1]. A problem arising especially in forging simulation is the deformation of the body and the distortion of the quadrilateral mesh (e.g. Fig. 1). To get useful results, the mesh must be regenerated. Manual remeshing is a very time consuming task, so that simulations of large deformations where new meshes must periodically be created are practically impossible. Therefore it is important that remeshing runs without user intervention.

The aim of this paper is to describe a fully automated remeshing scheme added to FINEL (see Fig. 2). Owing to the complexity of the task, the present version is restricted to 2-D and purely geometrical algorithms. Aspects such as, for example, local fine adjustment, have deliberately been excluded, but are planned as later extensions to the system.

The remeshing scheme should fulfill the following requirements:

- The mesh consists of convex quadrilateral elements having internal angles close to 90 degrees (this is important to prevent fast mesh distortion).
- The region to be meshed is a polygonal area which is defined by the boundary of the distorted mesh.
- Since volume conservation is preserved only numerically, it is sufficient to mesh the region approximately. The user can, however, guarantee exact remeshing by appropriate choice of simplification parameters.
- Meshing should be robust in the sense that regions of arbitrary shape can be remeshed.

Most FEM programs for metal forming simulations use the bilinear displacements–constant pressure element pair since it is more accurate in predicting plastic deformations. The reason is that bilinear quadrilateral elements are linear strain elements, while triangular elements are constant strain elements [11]. Forging simulations at IBF with the ABAQUS FEM program using triangular elements led to significantly worse results than FINEL or ABAQUS simulations with quadrilateral elements.

The disadvantage of quadrilateral elements is that mesh generation is a very difficult task. While there exists a variety of programs for generating triangular meshes, very few mesh generators for quadrilateral meshes were developed in the past.

An automated remeshing scheme was developed by Baehmann et al. [4]. Mesh generation is performed by the quadtree method [5]. Habraken and Radu [6] propose a scheme which seems to be fully automated, although they don't describe the mesh generator. The forging simulation program

[1] R. Schneiders and M. Becker were supported by the Deutsche Forschungsgemeinschaft under grant no. KO579/31-1.

Offprint requests: Dipl. Math. Dipl. Inf. Robert Schneiders, Lehrstuhl für angewandte Mathematik insb. Informatik, RWTH Aachen, Ahornstr. 55, 51 Aachen, Germany.

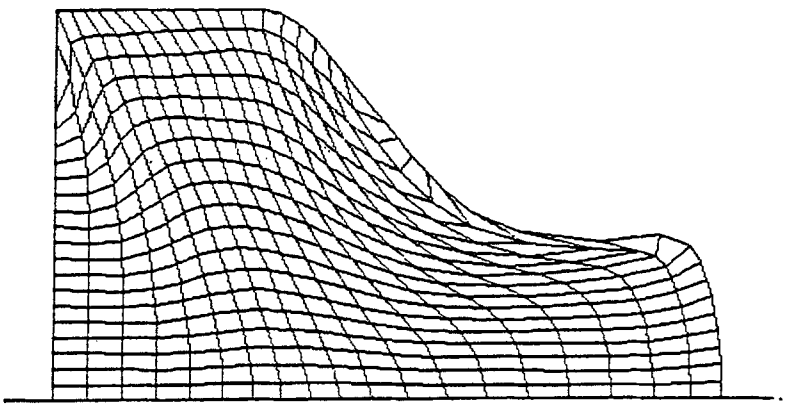

Fig. 1. Distorted mesh [2].

EALPID has remeshing capabilities [7], mesh generation is based on [8] and [9]. Good remarks on remeshing can be found in Cheng [10].

One of the earliest mesh generators for quadrilateral elements is the TRIQUAMESH program [8]. Other mesh generators were developed by Baehmann and Shephard [5], Talbert and Parkinson [12], and Heighway [13]. An overall picture of the finite element mesh generation field is presented by Ho-Le [11].

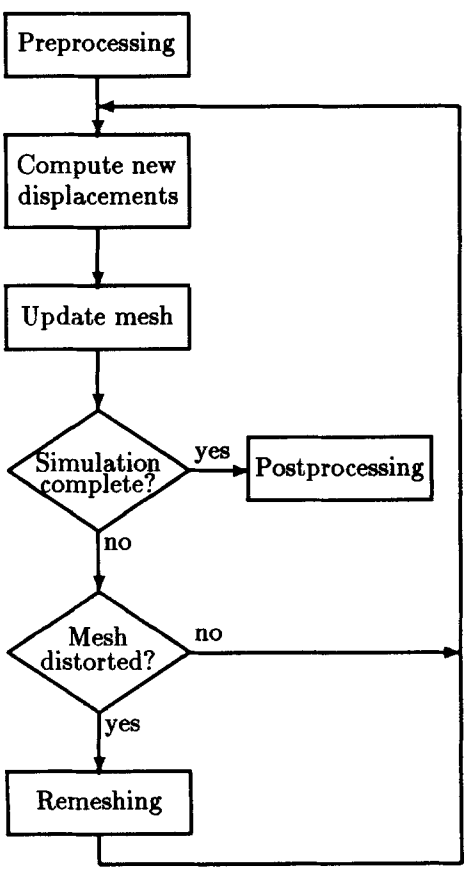

Fig. 2. FINEL flowchart [2].

The rest of the paper is organized as follows: section 2 describes a new mesh generator for polygonal areas using quadrilateral elements. Mesh generation is performed by means of computational geometry and image processing. This ensures a minimum quality of element distribution. Section 3 gives an outline of the complete remeshing scheme: measuring mesh quality, boundary simplification, smoothing, and rezoning. The application of the program is illustrated in section 4. Conclusions and further prospects can be found in section 5.

2 Mesh Generation

The region to be meshed is defined by the distorted mesh (hence there is no need to deal with curved geometries). This is a polygonal area and must be specified by the surrounding polygon from the calling module. The other input value is the mean side length h of the mesh.

The algorithm works on the following principle: the entire region is covered with a square mesh (see Fig. 3).

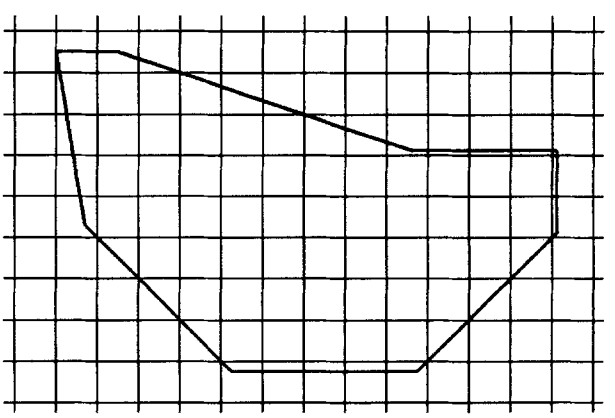

Fig. 3. Square mesh.

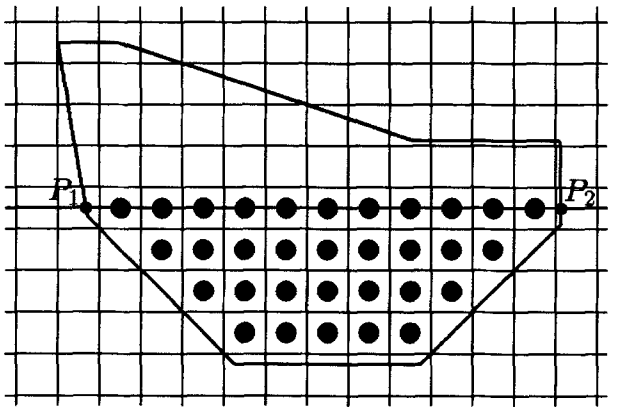

Fig. 4. Determination of the interior pixels.

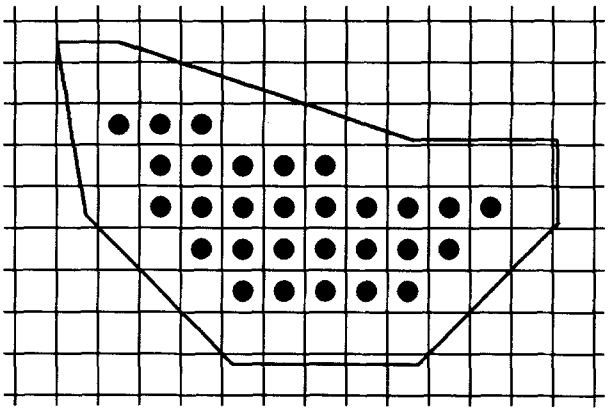

Fig. 6. Modified inner region.

Then mesh generation is done in two major steps:

1. Generation of an initial mesh from the squares inside the polygon
2. Meshing of the boundary region

2.1 Generation of the Initial Mesh

The region is covered with a mesh of squares having side length h. It can be regarded as a picture, the squares being called "pixels." A scanline algorithm well known in computer graphics is used to determine the pixels inside the polygon (see e.g. [15]). The algorithm makes a top–bottom scan of the picture. It uses the intersections between the polygon and an individual scanline situated in the middle of the current row. The pixels between two intersection points P_{2i-1} and P_{2i} are classified as "interior" pixels. Figure 4 gives an example of how the algorithm works.

The interior pixels define the "inner" region R of the picture (Fig. 5).

In the next step, the pixels touched by or having a distance of less than $h/2$ to the polygon are removed from R. The result is shown in Fig. 6. The choice of $h/2$ leads to a mean distance of h between R and the polygon. Then the pixels of R are used to construct the initial mesh (Fig. 7). Notice that there is a one-to-one correspondence between R and the initial mesh.

Errors can occur at this stage if the inner region is empty or is disconnected. This can be avoided by the choice of a smaller h value.

Phase 2 of the algorithm meshes the "boundary region" which means the region between the initial mesh and the polygon. It is required that the boundary of the initial mesh is given as a double linked list of nodes in ccw order. The list is created using a modification of the "tracer" algorithm, an efficient algorithm quite common in computer graphics [14,16]. The algorithm can be described in terms of an observer who walks along the boundary of R and selects the leftmost pixel available. He follows the

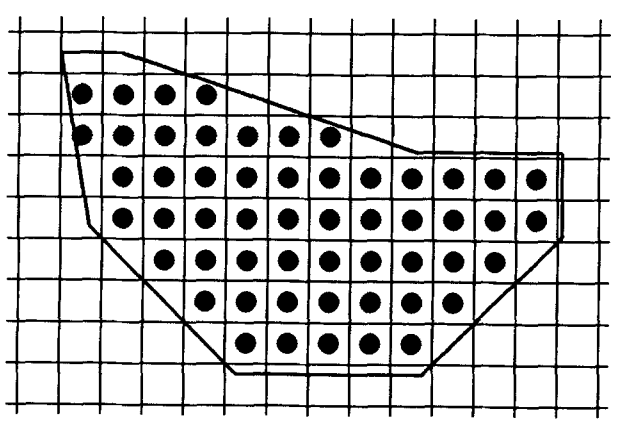

Fig. 5. Inner region: pixels marked by the scanline algorithm.

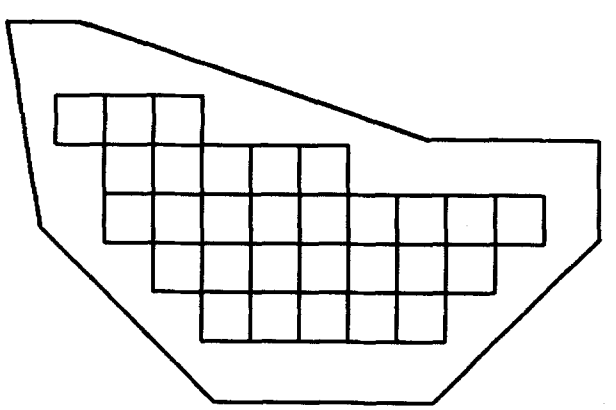

Fig. 7. Initial mesh.

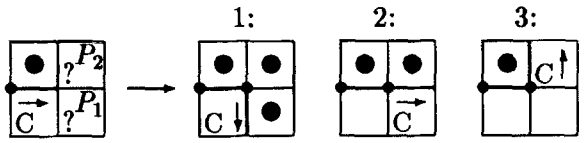

Fig. 8. Choosing a new current pixel.

boundary in a way that there is always a pixel of R (i.e., an element of the initial mesh) at the left (Fig. 9).

An outline of the algorithm is given in Fig. 10. An initial noninterior pixel A lying next to R is found in a top–bottom, left–right scan of the picture. A is chosen to be the observers's first "current pixel" C. Then he can move into four search directions S (right, up, left, down). In principle, the situations given in Fig. 8 may occur. First he considers the pixel P_1 (straight ahead). If it is in R, he corrects his search direction and turns to the right to guarantee that the initial mesh lies at the left hand. If it does not, he tests the pixel P_2. If it belongs to R, the observer selects P_1 and moves forward, otherwise he selects P_2 as the new current pixel.

The procedure is continued until the initial pixel is found. During the walk the boundary points of the initial mesh are determined and appended to the list.

2.2 *Meshing to the Boundary Region*

Having executed the steps of the previous section, the task is to mesh the boundary region. This is done in three steps:

1. Connect the corners of the initial mesh to the polygon.
2. Define the meshing at the corners.
3. Mesh the boundary between two corners.

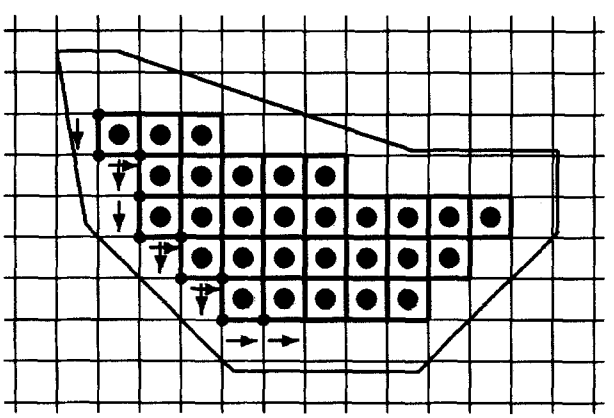

Fig. 9. Tracing the mesh boundary.

```
procedure tracer;

type pixel = record
               i, j: integer;
               end;

var A,  {* starting pixel}
    B,  {* test pixel}
    C:  {* current pixel}
       pixel;

        {*                        2   *}
var S:  {* search direction: 4    0 *}
        {*                        6   *}
       integer;

{* R is the set of interior pixels *}
{* S: addition/subtraction mod 8   *}

begin

find initial pixel A;
C := A;
S := 6;

repeat
    if B, the S-neighbour of C,
      is in R, then
        {* case 1 *}
        S := S - 2;
    else if B, the (S+1)-neighbour of C,
      is in R, then
        {* case 2 *}
        C := B;
    else
        {* B is in R *}
        {* case 4 *}
        C := B;
        S := S + 2;
      end;
  until C=A and S=6;

end; {* tracer *}
```

Fig. 10. Tracer algorithm.

The corners e_i of the initial mesh are classified as "inner" or "outer" corners (see Fig. 11 for explanation).

First the outer corners are examined. A search is made over the entire polygon for a point P which satisfies the following conditions (Fig. 12):

1. The angle between the lines $\overline{EP}$ and g does not

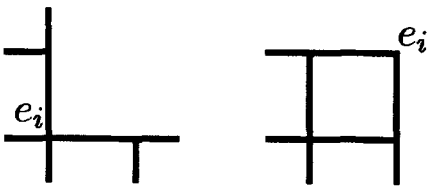

Fig. 11. Inner and outer corners.

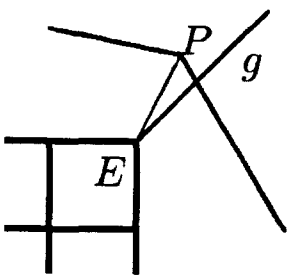

Fig. 12. Finding polygon points for outer corners.

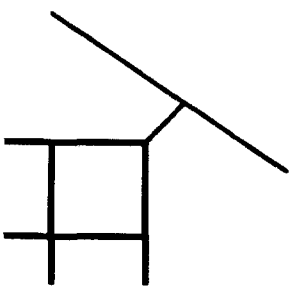

Fig. 13. Inserting a new point into the polygon.

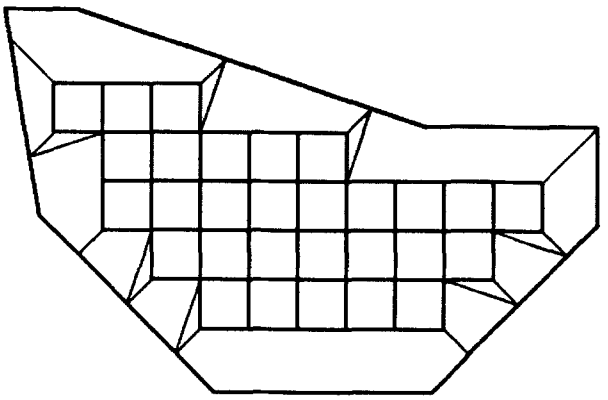

Fig. 14. Connecting the corners to the polygon.

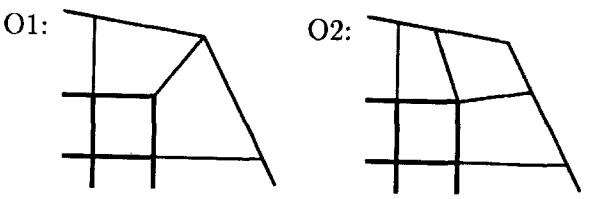

Fig. 15. Possible meshing of an outer corner.

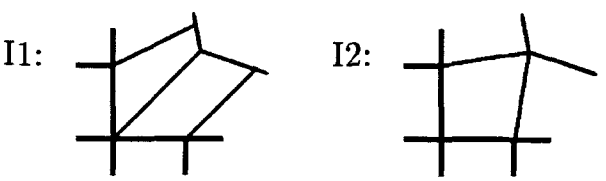

Fig. 16. Possible meshings of an inner corner.

exceed 35 degrees (g is the diagonal through the outer region around E).

2. The distance between E and P is limited to a value of $2h$.
3. The function

$$f(P) = (2h - \|P - E\|) \cdot \cos(P - E, g)$$

takes its maximum value over all points of the polygon which satisfy conditions 1 and 2 at P (P should be as close to E as possible while having a small angle to g).

If there does not exist such a point, a new one is inserted into the polygon. The point is assigned to the corner (Fig. 13).

After completing the task for the outer corners, the same procedure is executed for the inner corners. The intermediate result is shown in Fig. 14 (observe that a point of the polygon may be assigned to more than one corner).

The next step is to decide how to mesh the boundary region at the corners. Meshing at an outer corner can be done in two ways, O1 and O2, demonstrated in Fig. 15. Meshing with method O2 yields good results if the corresponding angle at the polygon does not exceed 120 degrees.

Figure 16 shows two ways to mesh inner corners (with corner meshing parameters I1 and I2). I2 is used if the corresponding angle is bigger than 175 degrees.

Meshing with method O2 or I2 gives better results than "simple" meshing (see Fig. 22), but if two corners are contiguous, one must take care that meshing is done correctly. Modification of the corner meshing parameters may be necessary. In the example given in Fig. 17 two corrections must be done:

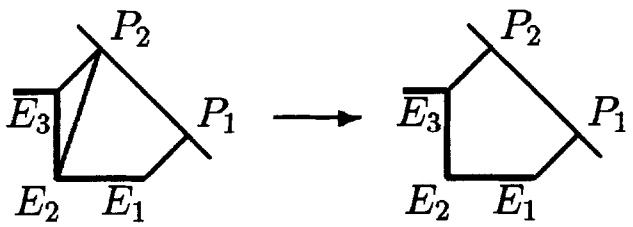

Fig. 17. Meshing at contiguous corners.

- set the meshing of E_2 to I1
- remove the edge E_2P_2

In general, for two adjacent corners the algorithm must take into account four kinds of corner meshing for E_1, four for E_2, and the possibility that the same polygon point is assigned to E_1 and E_2 or not. This gives 32 possible cases (not listed here, for a complete description see [17]).

Redefining of the corner meshing is done as follows: the list of boundary nodes is traversed. In each step three subsequent corners are examined:

```
var E1, E2, E3,     {* subs. corners *}
    first_corner:
  integer;

begin

determine first_corner;
E1 := first_corner;
E2 := successor (E1);
E3 := successor (E2);

repeat
  examine E1, E2 and E3;
  redefine corner meshing parameter at
    E2 and E3 if necessary;
  remove connections between E1/E2 and
    the polygon if necessary;
  E1 := E2;
  E2 := E3;
  E3 := next_corner;
  until E1 = first_corner;

end;
```

This part of the algorithm guarantees that no triangles are generated in the next step. Lots of special case logic is needed, see [17].

Now the noncorner points of the boundary are connected to the polygon. The points between two corners E_1E_2 are linked to the corresponding part P_1P_2 of the polygon. If there exist points between P_1 and P_2, this task must be done in two steps: connect the points between P_1 and P_2 to the boundary region, then apply transformation of similitude (take care that a point of the mesh boundary is assigned to only one point of the polygon). Meshing at E_1 and E_2 must be done according to the parameters defined previously.

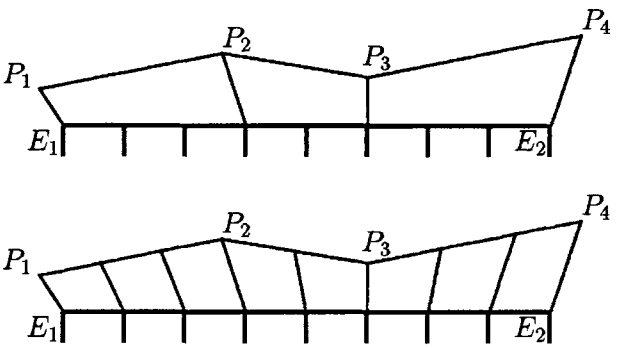

Fig. 18. Meshing at noncorner points.

The principle is demonstrated by the example in Fig. 18; the intermediate result is shown in Fig. 19.

The generated mesh consists of pre-elements having more than three nodes. This is important as the tesselation of a three-node element would make it necessary to split the neighbor elements which belong to the initial mesh. In the last step quadrilaterals are generated from pre-elements. Again lots of special case logic is needed [17]. Examples for the tesselations of five-node elements are shown in Fig. 20 (the tesselation must be done with respect to the tesselation of the neighbor elements). Theoretically there exists an infinite number of possible elements, but in practice there is no need to take care for elements having more than six nodes (this is an effect of boundary simplification; see section 3). If elements with more than six nodes occur, a restart of the algorithm with a slightly modified side length h might lead to better results. Experience shows that the cases occurring in practice are treated well

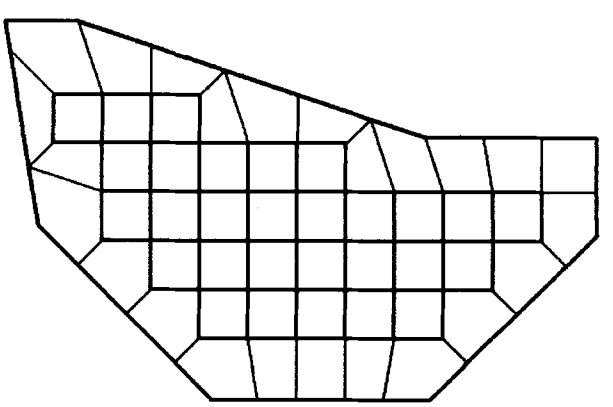

Fig. 19. Mesh consisting of pre-elements.

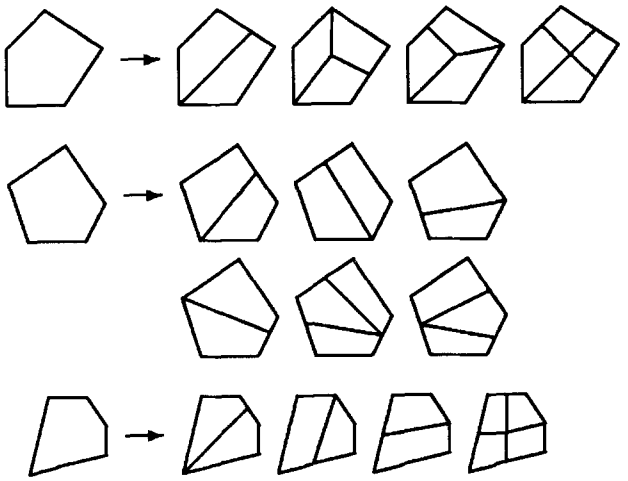

Fig. 20. Partitioning five-node elements.

by the algorithm. The generated mesh for the standard example is shown in Fig. 21.

The example in Fig. 22 shows the meshing of a part of the boundary region. The corners are connected to the polygon and the corresponding corner meshing parameters are determined. Modification of the parameters is not necessary. The second picture shows the generated mesh.

2.3 *Improvement of the Algorithm*

Consider the situation in Fig. 23: meshing cannot be done using the given mean side length h. Figure 24 shows that all interior pixels marked by the scanline algorithm have a distance of less than $h/2$ to the polygon. These pixels are removed subsequently, and no initial mesh can be generated.

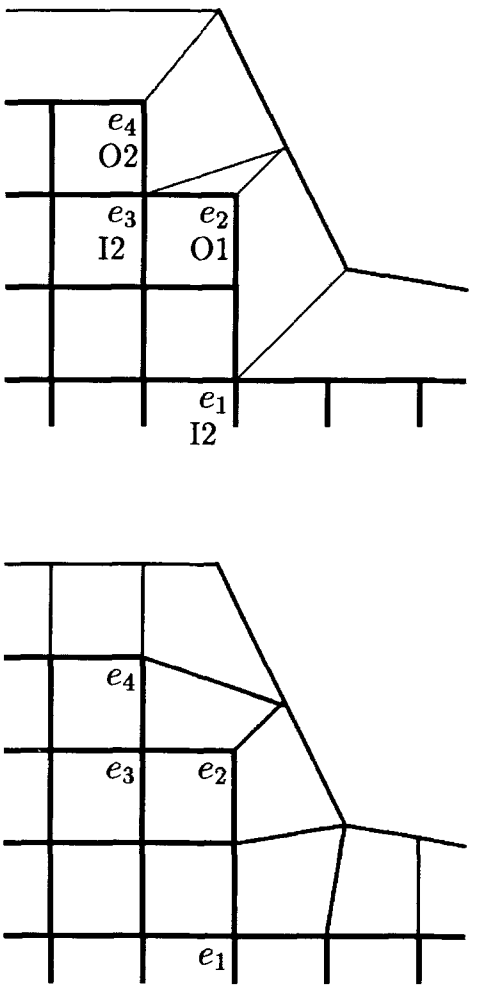

Fig. 22. Example of boundary meshing.

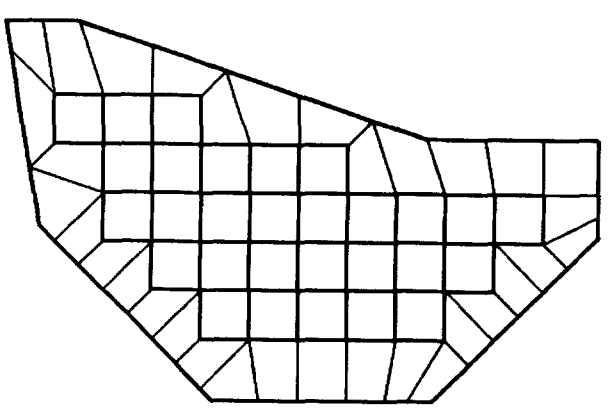

Fig. 21. Final mesh.

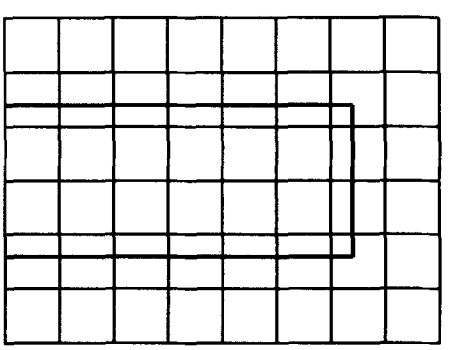

Fig. 23. Narrow region.

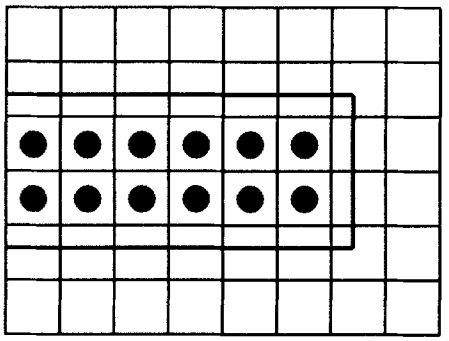

Fig. 24. Corresponding inner region.

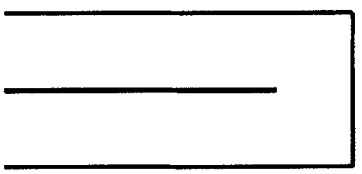

Fig. 25. Part of the initial mesh.

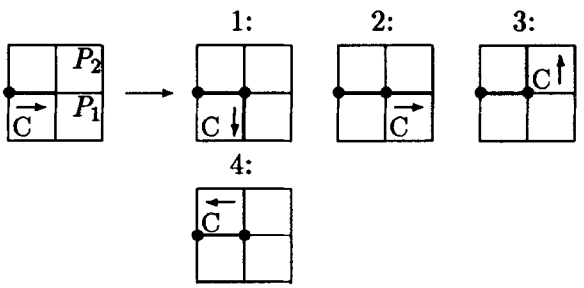

Fig. 26. Special cases.

We can overcome this difficulty if we proceed as follows:

- find the interior pixels (scanline algorithm)
- construct the initial mesh
- delete the nodes lying too close to the polygon
- for any deleted node, remove the corresponding edges and elements

The initial "mesh" consists of quadrilaterals and edges (Fig. 25).

The list of boundary nodes can be constructed using a slight modification of the tracer algorithm of section 2.1. Remember that the "observer" follows the boundary in such a way that there is always a pixel of R (an element of the initial mesh) at the left. This condition is weakened: there must always be an edge of the initial mesh at the left. This leads to four possible cases shown in Fig. 26. An example of boundary following is shown in Fig. 27.

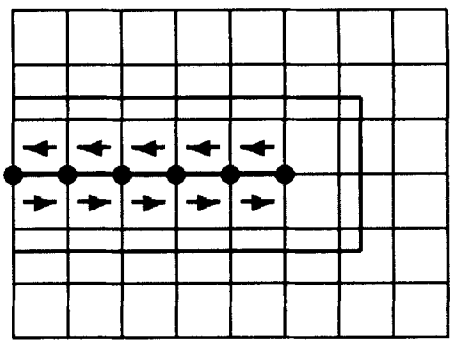

Fig. 27. Tracing the mesh boundary.

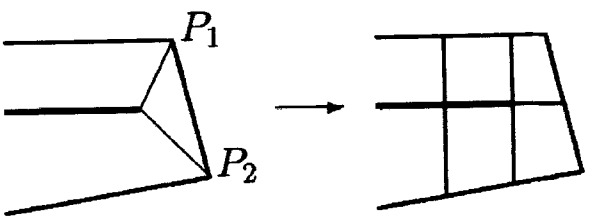

Fig. 28. Meshing at a sharp corner.

While tracing the boundary of the initial mesh, inner, outer and "sharp" corners (see Fig. 25) can occur. A sharp corner is connected to two points of the polygon (Fig. 28). Then the corresponding points on the polygon are searched according to the rules for inner and outer corners (see section 2.2). The meshing at a sharp corner is determined by the angles at P_1 and P_2. Figure 28 gives an example [meshing is (O2, O2)].

Modification of the corner parameters of contiguous corners does not cause further difficulties; it is done according to the rules for outer corners (see [17]). Mesh generation is then performed as described in section 2.2.

3 Remeshing

The complete remeshing scheme consists of the following components:

1. Mesh quality testing
2. Boundary simplification
3. Mesh generation
4. Smoothing and boundary projection
5. Rezoning

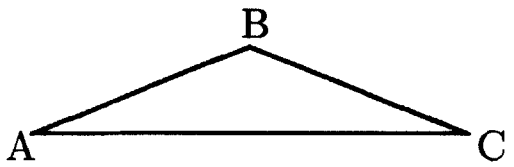

Fig. 29. Boundary simplification: principle.

3.1 *Mesh Quality Testing*

Several criteria for measuring geometrical degeneration of finite elements (element error) were tested [18]. The following criterion [19] was chosen: each four-node element is partitioned into four triangles by stretching the diagonals. The element error is represented by the maximum ratio of the circumcircle to the incircle radii of the triangles.

After extensive tests, the following heuristic criteria were selected for quality appraisal of the complete mesh:

- the maximum element error
- the ratio of maximum error to mean error
- the Gaussian normal distribution of element errors

Upper limits for these criteria can be entered individually.

3.2 *Boundary Simplification*

If the boundary of the distorted mesh defines the input for the mesh generator, the number of elements grows. This follows from the fact that the new boundary contains the nodes of the old boundary and several inserted nodes. This may cause troubles, since the number of elements may exceed the maximum number of elements after several remeshings. On the other side, FINEL conserves volume only numerically by the penalty method. Losses up to 1% may occur. Therefore it is sufficient to mesh a simplified boundary and reproject the new mesh to the old boundary.

Of the three succeeding points A, B, and C (Fig. 29), point B is omitted if

$$\frac{|AB| + |BC|}{|AC|} < 1 + \varepsilon$$

is true for the lines AB, BC, and AC. This approach eliminates not only quasilinear points but also extremely small boundary sections.

Any desired degree of boundary simplification can be selected by the choice of ε (default: $\varepsilon = 0.1$). Figure 30 shows an example of a simplified boundary which was calculated using a value of $\varepsilon = 0.1$. Note that the old boundary and the simplified boundary have some points in common.

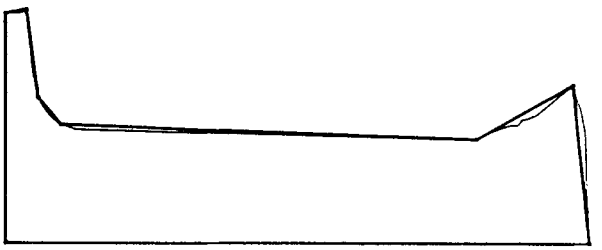

Fig. 30. Boundary simplification: example.

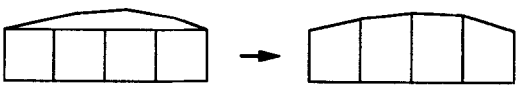

Fig. 31. Projection onto the old boundary.

3.3 *Mesh Generation*

Mesh generation can be controlled by the choice of the mean side length h and the boundary simplification parameter ε. If no valid mesh is generated, return codes of the mesh generator determine the reaction:

- *No connected initial mesh generated:* restart the mesh generation with a smaller h value.
- *Pre-elements with more than six nodes:* decrease h or increase ε.
- *Too many quadrilaterals generated:* h is too small.

The remeshing scheme uses this information to generate a new FEM mesh.

The node numbering is computed using the familiar algorithm of Gibbs, Poole and Stockmeyer [20].

3.4 *Boundary Projection and Smoothing*

The boundary nodes of the mesh generated are projected onto the old boundary (Fig. 31).

In order to improve the gradation and shapes of the mesh elements, a smoothing procedure is performed. The following iteration rule [6] has been adopted without modification:

$$x_i^{new} = \frac{\sum_{j=1}^{N_e} x_{js} A_j}{\sum_{j=1}^{N_e} A_j} \tag{1}$$

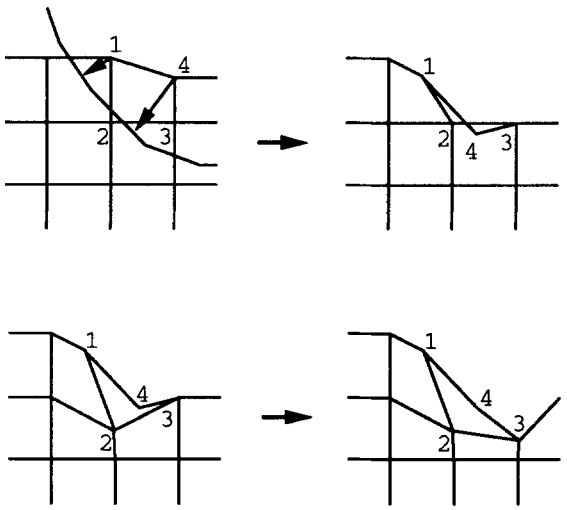

Fig. 32. Mesh repair.

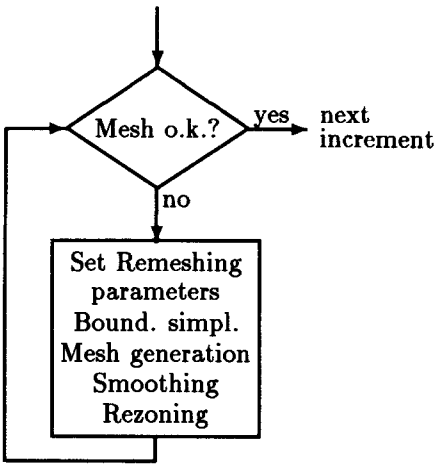

Fig. 33. Automatic remeshing.

where

x_i = mesh node
N_e = number of elements contiguous to x_i
x_{js} = center of the element j
A_j = area of element j

Iteration is done until

$$\frac{|\Delta_i - \Delta_{i-1}|}{|\Delta_0|} < \delta$$

where Δ_i is the biggest nodal movement of the ith mesh. δ can be specified by the user (default: $\delta = 0.001$).

With the exception of the corner nodes (which belong only to a single element), the boundary nodes are also adjusted, but then immediately reprojected onto the accurate boundary. This smoothing procedure tends to form equal sided elements.

Note that in forming technology a mesh repair made only by means of adjustment [10] is rarely sufficient, since mesh generation stems not only from local deformations but also from changes in the overall shape of the workpiece (e.g. rectangle in forging).

While projecting the boundary nodes of the mesh generated onto the old boundary, both inappropriately large and completely invalid elements (with negative area) may result. These mesh defects can, however, likewise readily be eliminated using slightly modified nodal adjustment [Eq. (1)] (see Fig. 32):

- initially perform the iteration for internal nodes (nonboundary nodes of the new mesh) only
- set the area of the degenerated elements at zero [set $A_j = 0$ in (1) if element j is degenerated].

After a few iterations, the mesh lies within the desired contour and can be completed by the usual adjustment process [Eq. (1)] (see also Fig. 38).

3.5 *Rezoning*

Local data (nodal velocity, temperature, equivalent strain) are transferred from the old to the new mesh using the formula [Eq. (2)]

$$D_j = \frac{\sum_{k=1}^{N_n} \frac{D_k}{R_{kj}^2}}{\sum_{k=1}^{N_n} \frac{1}{R_{kj}^2}} \tag{2}$$

where

D_j = data of the new point j
D_k = data of the old point k
R_{kj} = distance between the point j and the point k
N_n = number of points in the old mesh in the vicinity of the new point j

Before the FINEL computation run, the following parameters must be specified by the user:

- quality testing: upper limits for maximum error, error rate and standard error
- smoothing: convergence parameter δ
- boundary simplification parameter ε
- mesh generation: mean side length h

Automatic remeshing is performed as shown in Fig. 33.

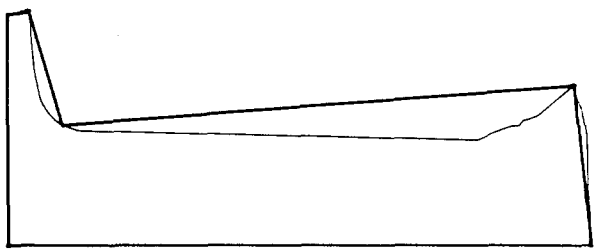

Fig. 34. Original and simplified boundary [2].

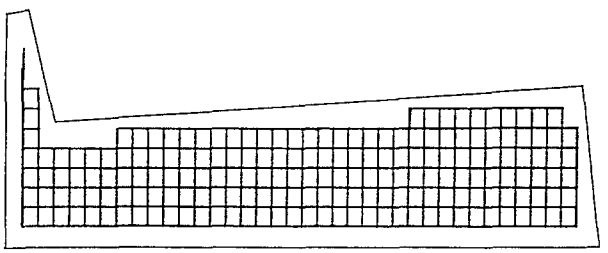

Fig. 35. Initial mesh [2].

If no valid mesh is generated, the parameters are modified by the program according to the rules mentioned above. If the geometry to be meshed is too complex, the number of trials will exceed a maximum value and the program will terminate with an error code (e.g. the required number of nodes exceeds the maximum value).

4 Examples

The following example shows a complete remeshing process. The process runs without user intervention. The original boundary is simplified using an ε value of 0.5 Fig. 34).

In the next step the initial mesh is generated (Fig. 35; observe that the left upper region could only be filled using a line). After connecting the corners to the polygon, the boundary region is meshed (Fig. 36). Note that meshing is done very well at the right upper corner.

The generated mesh is projected to the original boundary, smoothed, and adjusted (Figs. 37 and 38). The final mesh is shown in Fig. 39.

The next example shows the simulation of the die forging of a shaft with flange (direct extrusion). The 3-D problem was reduced to 2-D by using axial symmetry. The pictures (Figs. 40–45) show the mesh on the left half and the equivalent strain on the right half. The whole process took about 32.000 CPU-seconds on a FPS 264-minisupercomuter.

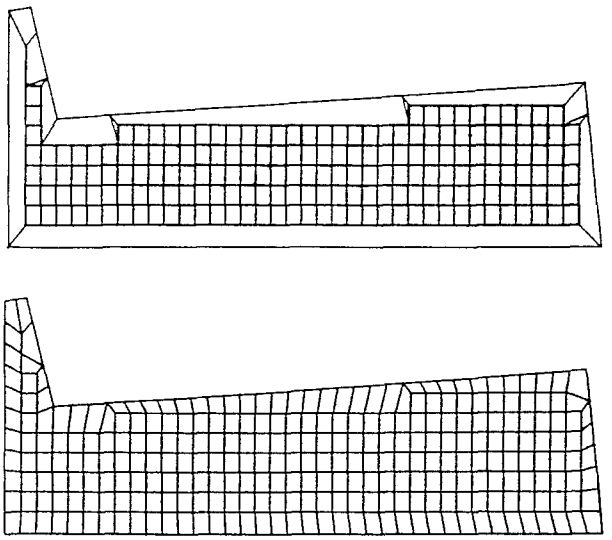

Fig. 36. Meshing of the boundary region [2].

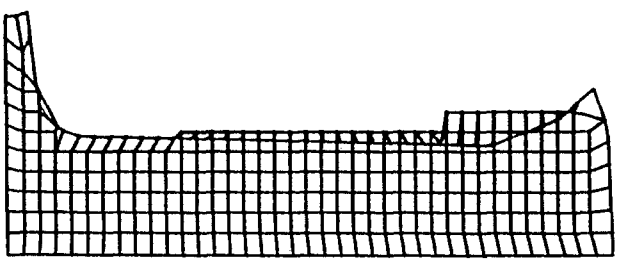

Fig. 37. Reprojection [2].

Forging is done in three steps: first a cylindrical preform is created. Then the block is deformed into an intermediate form using the die shown in Fig. 40. Figures 40–42 show examples of remeshed geometries and the predicted equivalent strains. In the last step the final shape is achieved using the die shown in Fig. 43. In this phase remeshing is enforced very often by the distorted quadrilaterals occurring in the left upper region of the geometry. Figures 44 and 45 demonstrate the quality of the remeshing scheme.

5 Summary

The FEM system FINEL in connection with the remeshing scheme presented here permits simulation of large deformations without user inverventions. The advantages of the remeshing scheme are:

- The mesh generator leads to good results if the mean side length h of the mesh is lower than the

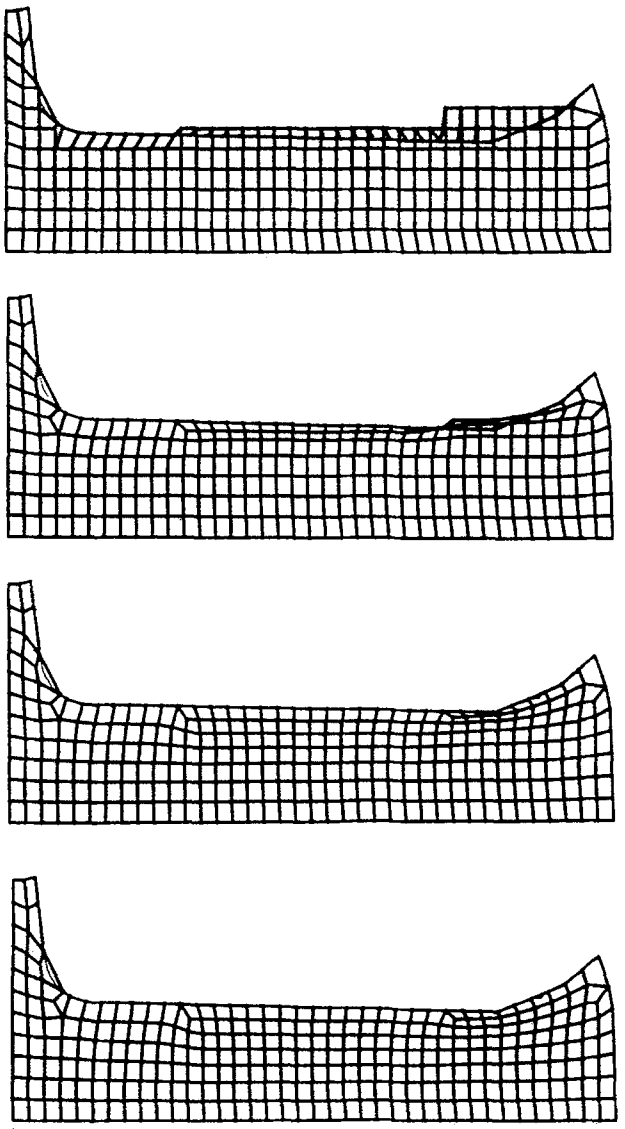

Fig. 38. Modified nodal adjustment [2].

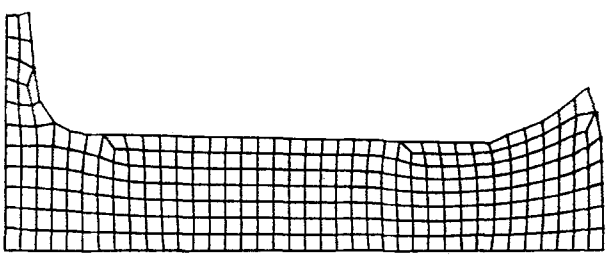

Fig. 39. Final mesh [2].

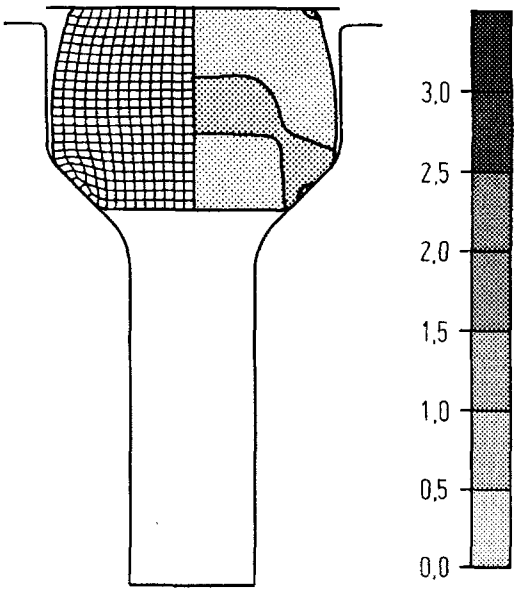

Fig. 40. Forging into intermediate form, I.

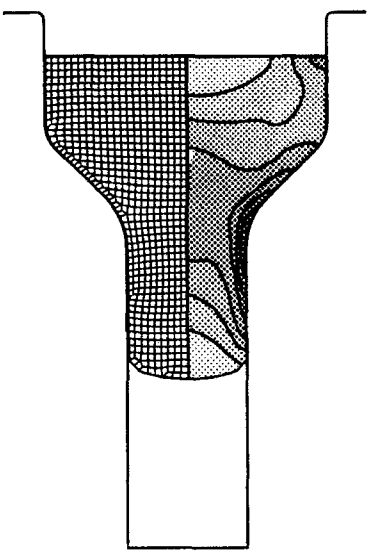

Fig. 41. Forging into intermediate form, II.

length of the smallest polygon edge. This can be achieved by boundary simplification.

- The meshes have interior elements of excellent quality.
- Experience shows that remeshing yields good results in forging simulations with even complex geometries.
- Remeshing is very fast (the time consumed is less than 1% of the total computing time).

The remeshing scheme does not allow local fine adjustment. This is a topic of current research.

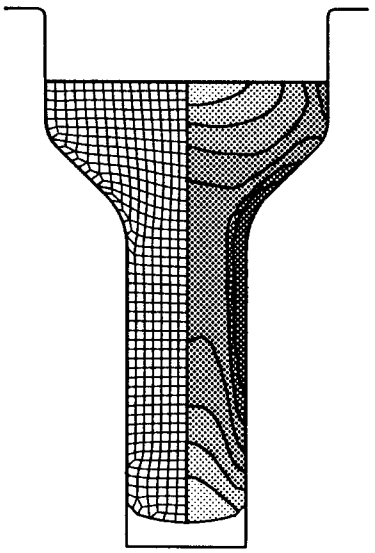

Fig. 42. Forging into intermediate form, III.

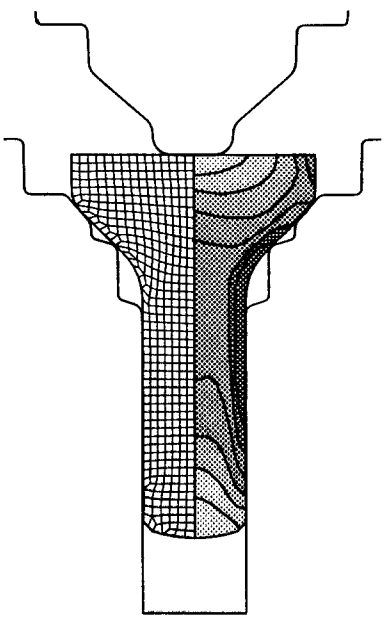

Fig. 43. Changing the die.

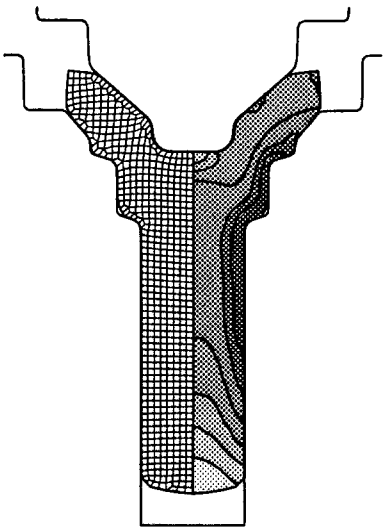

Fig. 44. Achieving the final state, I.

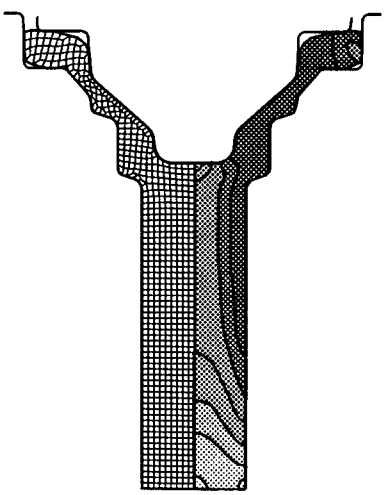

Fig. 45. Achieving the final state, II.

Acknowledgment

The authors wish to acknowledge the Deutsche Forschungsgemeinschaft.

References

1. M.L. Cho (1987) Bewertung der Anwendbarkeit der FEM fuer die Umformtechnik, Verlag Stahleisen
2. R. Kopp, M. Becker (1990) A concept for dynamic Remeshing at FEM-simulation shaved by the example of the forging process, ICTP 90, Kyoto, Japan
3. P. Brooks (1987) Verifying finite element results; a graphical approach. J. Eng. Comp. Appli., 19–27
4. P.L. Baehmann, M.S. Shephard, R.A. Ashley, A. Jay (1988) Automated metal forming analysis utilizing adaptive remeshing and evolving geometry, Comp. Struc. 30, 319–326
5. P.L. Baehmann, S.L. Wittchen, M.S. Shephard, K.R. Grice, M.A. Yerry (1987) Robust, geometrically based, automatic two dimensional mesh generation, Int. J. Num. Meth. Eng. 24, 1043–1078
6. A.M. Habraken, J.P. Radu (1989) Simulation of forging applications with the finite element method. NUMIFORM 89, Thompson et al. (eds.), 543 ff.
7. L.J. Hageman (1987) Automatic adaptive remeshing in EALPID, an advanced forging simulation program, Comp. Eng.

8. A.J.G. Schoofs, L.H.Th.M. von Beukering, M.C. de Sluiter (1979) TRIQUAMESH: A general purpose two dimensional mesh generator. Adv. Eng. Software 1 (3)
9. W.T. Wu, S.I. Oh, T. Altan, R.A. Miller (1990) Automated mesh generation for forming simulation—I, Proc. ASME Int. Comp. Eng. Conf., Boston
10. J.H. Cheng (1988) Automatic adaptive remeshing for finite element simulation of forming processes, Int. Jou. Num. Meth. Eng. 26, 1–18
11. K. Ho-Le (1988) Finite element mesh generation methods: A review and classification. Comp. Aid. Des. 20, No. 1
12. J.A. Talbert, A.R. Parkinson (1990) Development of an automatic two dimensional finite element mesh generator using quadrilateral elements and Bézier curve boundary definition, Int. Jou. Num. Meth. Eng. 29, 1551–1567
13. E.A. Heighway (1983) A mesh generator for automatically subdividing irregular polygons into quadrilaterals, IEEE Trans. Magn. MAG-19, No. 6, 2535 ff.
14. W. Oberschelp, M. Bierwagen (1991) Mathematische Methoden der Bildcodierung und Computergrafik, Internal report RWTH
15. D. Hearn, M.P. Baker (1986) Computer Graphics. Prentice Hall, Englewood Cliffs, NJ
16. T. Pavlidis (1982) Algorithms for Graphics and Image Processing, Computer Science Press
17. R. Schneiders (1989) Generierung von zulässigen Viereck-snetzen für die FEM-Methode, Lehrstuhl für angewandte Mathematik insb. Informatik
18. U. Strieder (1989) Entwicklung von Kriterien zur Messung der geometrischen Entartung von 2-dimensionalen FEM-Netzen aus 4-Knoten-Elementen, Int. Report IBF, Aachen
19. K. Lange, W. Osen (1985) Cold extrusion processes combined with radial extrusion. Man. Eng. Transactions
20. Comm. ACM, Vol. 2.4, Algorithm 508
21. M. Becker (to appear) Anwendung von höheren Optimierungsmethoden in der Umformtechnik, Ph.D. thesis