The development of a finite elements based springback compensation tool for sheet metal products.

R. Lingbeek[1,2,3],J. Huétink[1], S. Ohnimus[2], M. Petzoldt[2], J. Weiher[2]

[1] *University of Twente, Faculty of Engineering Technology, P.O. Box 217, 7500 AE Enschede, The Netherlands*
[2] *INPRO Innovationsgesellschaft für fortgeschrittene Produktionssysteme in der Fahrzeugindustrie mbH Hallerstraße 1, D-10587 Berlin ,phone (+49) 30 3997 278, fax (+49) 30 3997 117*
[3] *Netherlands Institute for Metals Research, Rotterdamseweg137, 2628 AL Delft, The Netherlands*

Abstract

Springback is a major problem in the deep drawing process. When the tools are released after the forming stage, the product springs back due to the action of internal stresses. In many cases the shape deviation is too large and springback compensation is needed: the tools of the deep drawing process are changed so, that the product becomes geometrically accurate after springback. In this paper, two different ways of geometric optimization are presented, the Smooth Displacement Adjustment (SDA) method and the Surface Controlled Overbending (SCO) method. Both methods use results from a finite elements deep drawing simulation for the optimization of the tool shape. The methods are demonstrated on an industrial product. The results are satisfactory, but it is shown that both methods still need to be improved and that the FE simulation needs to become more reliable to allow industrial application.

Key words: springback compensation deep drawing finite elements

1. Introduction

Deep drawing is one of the most common manufacturing processes in the automotive industry. Most deep drawn products are parts of the car body, such as door panels, engine hoods and side impact protection bars. For these products, the geometrical tolerances are tight, and the tools are expensive. Therefore, accurate process planning is essential.

After the relatively limited analytical models of forming processes, for example for stretch-bending of metal sheets[1], the focus in forming simulation has now moved towards the Finite Elements (FE) method[2,3]. It is now possible to predict the shape of the final product of a realistic industrial forming operation, its internal stresses and process forces. Upon unloading after the forming stage, the product springs back due to internal stresses. For large parts such as car body panels, these springback deformations can be large, up to several millimeters. High strength steels and aluminum, used for lightweight products, show particularly large springback[4]. The calculation of springback has been implemented in most commercial forming simulation software packages. However, for industrial deep drawn products, the accuracy of the results has not yet reached an acceptable level [5].

When the product does not meet the geometrical requirements, the deep drawing tools are manually redesigned so, that the shape deviations due to springback are compensated. This is a complex and costly operation, because the springback can be quite large, and because it is also different for every product. At present time, it is a trial-and-error process of manufacturing tools, making a prototype product, measuring it, modifying CAD data and reworking the stamping tools. When the FE springback simulation has proven to be reliable, the results of this simulation can be used in this shape optimization loop. This speeds up the development of the toolset significantly, as was demonstrated in [6]. However, in this

optimization the shape optimization is still carried out manually, and still requires engineering experience.

The goal of this research was to develop an industrially applicable software tool that automatically alters the deep drawing tools so, that springback is compensated. The optimized geometrical data are transferred into a new CAD file, which are needed as a basis for the NC code for tool production right away. This way, expensive prototype tests can be avoided and the design optimization phase will be more effective, faster and more cost-efficient.

2. Evaluation of springback and process optimization for springback reduction

In general, sheet metal products are produced in several forming operations, for example deep drawing, trimming and hemming. Because each operation can be seen as a separate process with individual parameters, and because the results of subsequent operations depend on the previous ones, each forming operation needs to be optimized separately.

First, the deep drawing tools are directly derived from the CAD data. With these tools, a FE simulation is carried out, resulting in two meshes. The *reference mesh* represents the blank directly after the forming operation, with the tools still closed. When the tools are removed, the blank will spring back, resulting in the *springback mesh.* The reference mesh is considered the best obtainable geometry. It includes not only the product geometry, but also the die addendum, die-entry radii and possibly blank-cuts. Because of the fact that these parts of the blank have a major impact on its springback behavior, the springback compensation algorithm optimizes the complete blank, not only the product area.

Evaluating the shape of the meshes of the deformed blank and the deformed blank after springback is not a trivial task. In practice the product is not mechanically constrained anymore after the tools are released. Therefore, only a minimal amount of mechanical constraints is applied during the springback calculation. After the forming process has been completed, the product is fastened in a larger assembly, such as a car body. Using those ‘assembly constraints’, a second, mainly elastic, calculation can be performed to evaluate the product’s springback. Based on this calculation a decision can be made whether the forming process needs to be optimized or not.

The product shape can be checked for large shape distortions. But, the forces that act on the constrained nodes, modeling the assembly forces, are an equally important factor. When the force to push the product in the right shape exceeds 30N this is already unacceptable for car body panels, and reduction or compensation of springback are required. If the fastening forces are relatively small it is in many cases preferable not to compensate, and to check whether the product already meets its geometrical requirements after assembly. Most structural products are generally too rigid and cannot be bent back into shape during assembly because high internal stresses would be introduced in the assembly.

Before springback compensation is carried out, the deep drawing process should be optimized to reduce springback first. There are numerous methods to decrease the amount of springback. [7, 8] present many references on controlling the springback process.

Springback reduction already starts in the design phase. A relatively flat structure will be more prone to springback problems than a cup-shaped product. Adding reinforcement ribs can reduce springback problems drastically. Also, altering sheet thickness, or radii in the structure can improve the dimensional stability. Computer optimization of structural design features has been successfully implemented in [9]. Here, a simple structure (hat-profile) was optimized, with a limited set of shape-parameters. However, a realistic product may contain thousands of geometrical parameters, so this method is considered as highly impractical. Also, adding or changing structural design features is a task for the designer, because a computer cannot completely oversee the functional requirements for the structure. Therefore, we consider *redesign* outside the scope of the

project. It is, however, the most effective way of reducing springback.

3. Springback compensation algorithms

Even after the forming process has been optimized, springback may remain problematic. In this case springback compensation is needed. How an automatic springback compensation algorithm works is shown in figure 1.

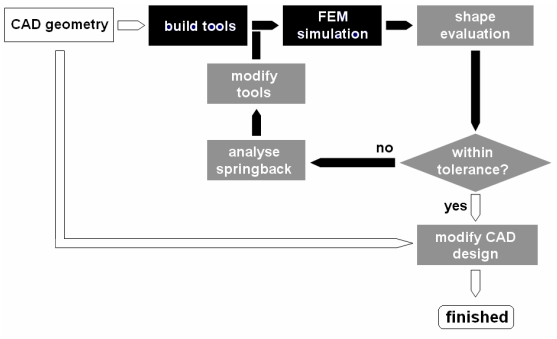

Figure 1. Automatic springback compensation

From the product's CAD geometry, deep-drawing tools are derived. With these tools a FE simulation is carried out. The algorithm evaluates the springback deformations, and changes the tools if the geometric deviation is outside the tolerances. The tools are modified, and a new FE simulation is carried out. If the product does not meet the tolerances yet, the tools are again modified. This loop is carried out until the product is geometrically accurate.

In literature many methods have been developed for compensating springback by changing the tool geometries. Generally, the target of the optimization is to reduce the *shape difference* between the reference mesh and the springback mesh. During the optimization the springback itself is *not* reduced. Actually, in most cases springback increases when the tools are optimized. The two methods that are discussed here are the Displacement Adjustment (DA) method and the Surface Controlled Overbending (SCO) method. Both methods are strictly geometrical and work in principally the same way as an engineer manually compensates springback.

It is also possible to use the blank's internal stresses directly after forming for springback compensation, as presented in [10,11]. Unfortunately, this method has shown to be unreliable [8]. Attempts have also been made to use the change in curvature of the blank geometry for compensation [12].

3.1 The Displacement Adjustment Method

So far, the Displacement Adjustment (DA) Method [13, 14] has proven to be the most effective. The principle behind the method is well known and has already been applied intuitively by process engineers; The idea is to (optically) measure the blank, and calculate the distance between the produced blank and the desired shape. The surface of the tools is then displaced with the same distance, but in the direction opposite to the springback deformation.

In the DA method this principle is applied to optimize the product shape, defined as a discrete FE mesh. $\vec{R}$ is the reference product geometry, given as a collection of *n* points in $\Re^3$, $\vec{S}$ is the product geometry after springback.

$$\vec{R} = \left\{ \vec{r}_i \middle| \vec{r}_i \in \Re^3, 1 \le i \le n \right\} \qquad (1)$$

$$\vec{S} = \left\{ \vec{s}_i \middle| \vec{s}_i \in \Re^3, 1 \le i \le n \right\} \qquad (2)$$

The compensated product geometry $\vec{C}$ can now be calculated, provided that the reference and springback nodes with identical number *i* are coupled: $\vec{r}_i$ becomes $\vec{s}_i$ after springback

$$\begin{aligned} &\vec{C} = \vec{R} + a\left(\vec{S} - \vec{R}\right) \\ &\Leftrightarrow \vec{c}_i = \vec{r}_i + a\left(\vec{s}_i - \vec{r}_i\right) \end{aligned} \quad \forall i \qquad (3)$$

The factor *a* is called the compensation factor. It is generally negative and varies between the values -2.5 and -1.0. When the method is applied iteratively, the results will become significantly better. The first compensated geometry $\vec{C}$ is now referred to as $\vec{C}^1$, and with this geometry a new FE simulation is

carried out. The resulting springback mesh $\vec{S}^1$ is now used to modify $\vec{C}^1$, delivering the second compensated geometry $\vec{C}^2$. Note that $\vec{S}^1$ and $\vec{R}$ are the results from different FE simulations.

$$\vec{C}^2 = \vec{C}^1 + a\left(\vec{S}^1 - \vec{R}\right) \quad (4)$$

So, equation (3) can be formulated recursively as follows.

$$\vec{C}^{t+1} = \vec{C}^t + a\left(\vec{S}^t - \vec{R}\right) \quad 0 < t < t_{\max} \quad (4)$$

The optimization process is stopped when the geometrical tolerance is reached:

$$\left\|\vec{S}^t - \vec{R}\right\|_{\max} < \varepsilon_1 \quad (5)$$

or when convergence is reached

$$\left\|\vec{C}^{t+1} - \vec{C}^t\right\|_{\max} < \varepsilon_2 \quad (6)$$

Note that the shape modification field $\vec{\Phi}$

$$\vec{\Phi}^t\left(\vec{R}, \vec{S}^t\right) = \left\{\vec{\varphi}_i^t \middle| \vec{\varphi}_i^t \in \Re^3, 1 \le i \le n\right\}$$
$$\vec{\varphi}_i^t = a\left(\vec{s}_i^t - \vec{r}_i\right) \quad (7)$$

is defined on the nodes of the reference mesh only, and that from the second iteration, it is applied to the geometry that has already been modified before. This is made clear in figure 2.

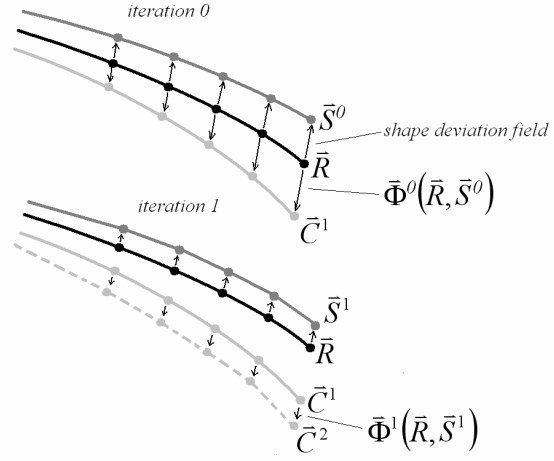

Figure 2. Sequential application of the DA principle

The DA method has been demonstrated to be effective and fast, but only on simple forming processes for two-dimensional products [13]. However, the most important problem of the DA method is that the shape modification field is defined on the nodes of the blank-mesh only. To be able to apply $\vec{\Phi}$ to any mesh, including the generally larger and topologically different tool meshes, or even the analytically defined CAD geometries, the discrete field needs to be approximated by an analytical function $\vec{\Psi}(x, y, z)$. With this addition, the method is now called the *Smooth Displacement Adjustment* (SDA) method [15].

The goal is to find the right function $\vec{\Psi}(x, y, z)$, that minimizes:

$$\left\|\vec{\Phi} - \vec{\Psi}(x, y, z)\right\|_{L2} \quad (8)$$

The exact definition of this norm is given in equation (13). The function $\vec{\Psi}(x, y, z)$, a summation of k_{max} polynomials is defined as follows:

$$\vec{\Psi}(x, y, z) = \sum_k \vec{a}_k \theta_k(x, y, z) \quad (9)$$

with $1 \le k \le k_{\max}$

The SDA method has been implemented using polynomial basis-functions:

$$\theta_k(x, y, z) = x^{f_k} y^{g_k} z^{h_k} \quad (10)$$

where the exponents f_k, g_k, h_k can be defined by the user.

The goal is now to find the right vector $\vec{a}_k$. For convenience, the problem of finding the optimal function $\vec{\Psi}$ can be split into its components:

$$\vec{\Psi}(x, y, z) = \begin{pmatrix} \Psi_x(x, y, z) \\ \Psi_y(x, y, z) \\ \Psi_z(x, y, z) \end{pmatrix} \quad (11)$$

Each component has its vector of (now scalar) *a*-values. The component Ψ_x can be calculated as follows, and the others are calculated in the same way.

$$\Psi_x(x,y,z) = \sum_k a_{x,k}\theta_k(x,y,z) \qquad (12)$$

The L2 norm is calculated as follows

$$\Pi = \frac{1}{2}\int_R (\Psi_x - \Phi_x)^2 dx \qquad (13)$$

For the optimal parameter-set $a_{x,n}, 0 < n < k_{\max}$, the potential Π should be minimal. Therefore:

$$\delta_{a_n}\Pi = 0 \quad \forall n \qquad (14)$$

$$\delta_{a_n}\Pi = \int_R (\Psi_x - \Phi_x) dx \cdot \delta_{a_n}\Psi_x \qquad (15)$$

Because of

$$\delta_{a_n}\Psi_x = \theta_n \qquad (16)$$

equation (14) can be rewritten as

$$\int_R \left(\sum_k a_{x,k}\theta_k - \Phi_x\right)\theta_n dx = 0 \Leftrightarrow \qquad (17)$$
$$\sum_k a_{x,k}\int_R \theta_k\theta_n dx = \int_R \Phi_x\theta_n dx \quad \forall n$$

finally, this set of equations can be written as a matrix-equation:

$$\{a_x\}^T[M] = \{c_x\} \qquad (18)$$

with

$$\{a_x\} = \begin{Bmatrix} a_0 \\ a_1 \\ \dots \\ a_k \end{Bmatrix}$$

$$\{c_x(x,y,z)\} = \begin{Bmatrix} \int \Phi_x\theta_0(x,y,z) \\ \int \Phi_x\theta_1 \\ \dots \\ \int \Phi_x\theta_k \end{Bmatrix}$$

$$[M(x,y,z)] = \begin{bmatrix} \int \theta_0(x,y,z)\theta_0(x,y,z) & \int \theta_0\theta_1 & \dots & \int \theta_0\theta_k \\ \int \theta_1\theta_0 & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \int \theta_k\theta_0 & \dots & \dots & \int \theta_k\theta_k \end{bmatrix}$$

Equation (18) can now be solved, and the vector a_x can be calculated.

To solve the integrals, the following summation for *n* nodes has been defined:

$$M_{kl} = \int_{\bar{R}} \theta_k(x,y,z)\theta_l(x,y,z)dx$$
$$\approx \sum_{i=1}^{n} \theta_k(\vec{r}_i)\theta_l(\vec{r}_i)\Delta x_i \qquad (20)$$
$$\{c_{x,k}\} \approx \sum_{i=1}^{n} \Phi_x(\vec{r}_i)\theta_k(\vec{r}_i)\Delta x_i \qquad (21)$$

Δx_n is calculated as follows: the elements in which node *n* is a member are found. If the elements are triangles, 1/3 of their area is summed to calculate Δx_n, if they are quadrangles, 1/4 of the areas is summed. Of course the mesh can be mixed as well. This procedure is visualized in figure 3.

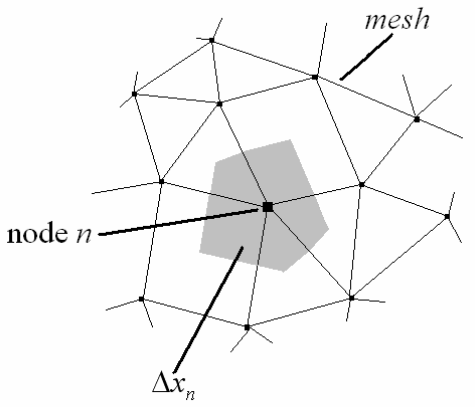

Figure 3. Calculation of Δx_n

In the same way as vector $\{a_x\}$, the two other coefficient-vectors $\{a_y\}$ and $\{a_z\}$ can be calculated and function $\vec{\Psi}(x, y, z)$ is complete:

$$\vec{\Psi}(x, y, z) = \sum_{k=1}^{k} \begin{pmatrix} a_{x,k}\theta_k(x, y, z) \\ a_{y,k}\theta_k(x, y, z) \\ a_{z,k}\theta_k(x, y, z) \end{pmatrix} \qquad (22)$$

The compensated geometry is now, calculated in the same way as equation (3):

$$\begin{aligned} &\vec{C} = \vec{R} + \vec{\Psi} \\ &\Leftrightarrow \vec{c}_i = \vec{r}_i + \vec{\Psi}(\vec{r}_i) \end{aligned} \qquad (23)$$

3.2 The Surface Controlled Overbending (SCO) method

As an alternative method the SCO strategy has been developed. This method is also based on the DA principle but overcomes the problems with modifying the topologically different tool-meshes. With the SCO method, the shape modifications on the tools are defined in such a way, that they can be applied to CAD geometries as well. The modified CAD geometries of the tools can be used to produce the toolset right away.

For convenience, equations (1) and (2) are repeated here. Both reference and springback geometries are represented by a set of 3 dimensional coordinates.

$$\vec{R} = \left\{ \vec{r}_i \middle| \vec{r}_i \in \Re^3, 1 \le i \le n \right\} \qquad (24)$$

$$\vec{S} = \left\{ \vec{s}_i \middle| \vec{s}_i \in \Re^3, 1 \le i \le n \right\} \qquad (25)$$

Both geometries are approximated with analytical surfaces that have an identical definition (but different parameter values). The reference mesh is approximated by the *reference surface* Ω, the springback mesh by the *springback surface* Ξ. The (limited number of) coefficients of these surfaces can be used as an input for a DA process, producing a *transformation surface* Θ. This surface describes the global shape of the compensated geometry, and is used to transform the tool meshes or CAD files directly.

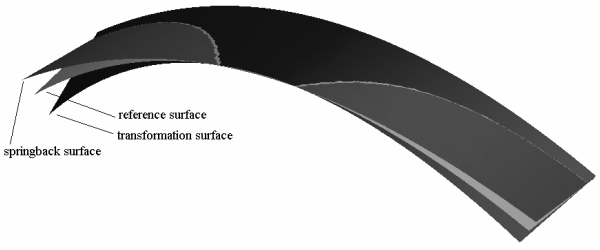

Figure 4. The control surfaces

Bezier and B-spline surfaces have been chosen as approximation surfaces, because they are capable of modeling complex shapes while remaining stable. Both surface descriptions are parametric. For parametric surfaces, each coordinate value is represented by a separate function of one or more parameters.

$$\vec{\Omega}(u, v) = \left(\omega_x(u, v), \omega_y(u, v), \omega_z(u, v)\right) \qquad (26)$$

The Bezier description of a surface with the degrees *n* and *m* is defined as follows:

$$\vec{\Omega}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i(u) B_j(v) \vec{P}_{ij} \qquad (27)$$

Note that equation (27) is a 3-dimensional vector function. As a function basis, the Bernstein polynomials are used. Equation (28) describes the *i*-th Bernstein polynomial of degree *n*:

$$B_{i,n}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i} \qquad (28)$$

Equation (27) can be written more conveniently in the following form:

$$\vec{\Omega}(u, v) = \{B(u)\}^T \left[\vec{P}_{ij}\right] \{B(v)\} \qquad (29)$$

The *n* by *m* matrix $\left[\vec{P}_{ij}\right]$ contains the control points that define the shape of the surface, vector $\{B(u)\}$ contains *n* and $\{B(v)\}$ *m* Bernstein polynomials. Together the control points form the control polygon, as visualized for a single parameter Bezier-curve in figure 5.

A B-spline surface can be regarded as a piecewise polynomial Bezier-surface. The mathematics of these surfaces is not very different, but a bit more involved. The reader is referred to [16] for an in-depth discussion.

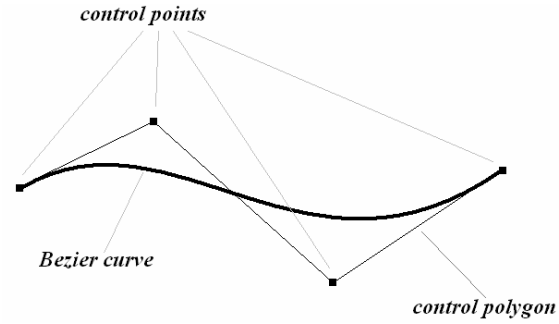

Figure 5. A Bezier curve

In order to fit a Bezier surface through the reference mesh $\vec{R}$, the matrix $\left[\vec{P}_{ij}\right]$ needs to be found that minimizes the norm

$$\left\|\vec{R}-\vec{\Omega}(u,v)\right\| \tag{30}$$

$$\Rightarrow \sum_i \left|\vec{r}_i-\vec{\Omega}(u_i,v_i)\right|^2 \tag{31}$$

in which $\vec{\Omega}(u_i,v_i)$ is the point on the surface that is closest to the node $\vec{r}_i$ in the reference mesh.

The degrees of the Bezier control-surface can become rather large, which implies that the matrix $\left[\vec{P}_{ij}\right]$ will become large too. An analytical solution for finding the right parameters in this matrix will become very complicated. Therefore Powell's multivariate minimization algorithm [17] was applied to minimize the norm in equation (31).

Finding u_i and v_i (defining the point closest to $\vec{r}_i$) is another problem. The basis of the solution is a *point inversion* algorithm, which calculates the *u* and *v* parameters for a given point *on* the surface.

Calculating the two parameters from a point given in three Cartesian coordinates is an overdefined problem:

$$\vec{p}_i=\vec{\Omega}(u_i,v_i)\Leftrightarrow\begin{pmatrix}p_{xi}\\p_{yi}\\p_{zi}\end{pmatrix}=\begin{pmatrix}\omega_x(u_i,v_i)\\\omega_y(u_i,v_i)\\\omega_z(u_i,v_i)\end{pmatrix} \tag{32}$$

This means that an exact solution cannot be found because the point is never exactly on the surface. Therefore, a numerical tolerance has to be specified. In theory, this calculation is solvable in closed form for surfaces with lower degrees. Because the degrees of the surface are generally higher, a numerical algorithm, derived from [16] has been chosen for this calculation.

This algorithm searches the point with parameters u_i and v_i on the surface that is *closest* to the input point. The input-point does not need to be on the surface, and this process is therefore called point-on-surface *projection*. With the closest point, the distance between the input point and the surface:

$$\left|\vec{r}_i-\vec{\Omega}(u_i,v_i)\right| \tag{33}$$

can be calculated. For a *curve*, with parameter *u* only, the distance between point $\vec{p}$ and the curve *C(u)* is minimal when the function

$$f(u)=\vec{C}'(u)\cdot(\vec{C}(u)-\vec{p}) \tag{34}$$

equals zero. This point is found using the Newton iteration scheme, with a good initial guess for *u*. Convergence is reached when the distance of the points is within a certain tolerance:

$$\left|\vec{C}(u_i)-\vec{p}\right|\le\varepsilon \tag{35}$$

or when the vector pointing at point $\vec{p}$ is normal to the surface:

$$\vec{C}'(u_i)\perp(\vec{C}(u_i)-\vec{p})$$
$$\Rightarrow\frac{\left|\vec{C}'(u_i)\cdot(\vec{C}(u_i)-\vec{p})\right|}{\left|\vec{C}'(u_i)\right|\cdot\left|(\vec{C}(u_i)-\vec{p})\right|}\le\varepsilon \tag{36}$$

with $\varepsilon<<1$. For a surface, the solution is identical, but the mathematics are a bit more

involved because of the two parameters; see [16].

The surface fitting process is numerically expensive, because many point projection processes are needed. The easiest way to speed up the process is to use only a part of the nodes in the mesh. The algorithm has been tested using only 10% of the nodes, and the fitting process was still very accurate. As the node-surface distance calculations are the dominant factor in the algorithm, the calculation time was reduced by more than 80%. Care has to be taken, when nodes are ignored in the fitting process. When a certain part of the structure contains exactly those nodes that are ignored, the shape of the mesh will get distorted and the approximation will be wrong. It is advised to check the results of the fitting visually. A better way is to start with only a small selection of nodes, do some approximation steps, and derive a global surface shape. This solution is used as a starting point for a more accurate, but slower approximation with more nodes. The solution that is found now can, again, be used a starting point for a very accurate approximation with all available nodes.

Finally, the result of the complete surface fitting procedure is a matrix $\left[\vec{\omega}_{ij}\right]$ of control points, describing the shape of the reference surface Ω. The exact same procedure is followed to find the control point matrix $\left[\vec{\xi}_{ij}\right]$ for the springback surface Ξ. The two parameter sets are now used to create a new parameter set $\left[\vec{\vartheta}_{ij}\right]$, describing the transformation surface Θ. The DA-principle, as defined in equation (3) is applied to the control point matrices:

$$\left[\vec{\vartheta}_{ij}\right] = \left[\vec{\omega}_{ij}\right] + a\left(\left[\vec{\xi}_{ij}\right] - \left[\vec{\omega}_{ij}\right]\right) \quad \forall i, \forall j \qquad (37)$$

The transformation surface describes the global shape of the compensated part. The reference surface and the transformation surface are now used to transform the geometry of the tools

A tool mesh is described in the same way as the reference and springback meshes:

$$\vec{T} = \left\{ \vec{t}_i \middle| \vec{t}_i \in \Re^3, 1 \le i \le n \right\} \qquad (38)$$

The following procedure describes the transformation of a single toolnode t_i. Obviously, the entire mesh is transformed in the same way.

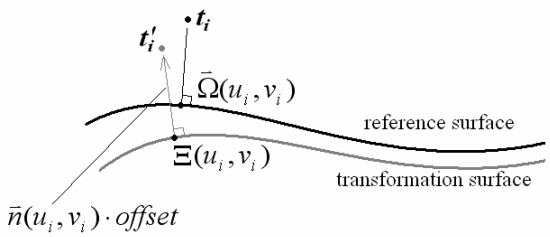

Figure 6. Transformation of a tool-node t_i

First, the node t_i is projected onto the reference surface with the point projection algorithm (eqs.32-36). The vector $\vec{O}_i$:

$$\vec{O}_i = \vec{t}_i - \vec{\Omega}(u_i, v_i) \qquad (39)$$

is normal to the reference surface. The length of this vector is called offset, and it can be made positive or negative, depending on whether the node t_i is above or underneath the surface.

Now, the point on the transformation surface with the parameters u_i and v_i is calculated, and the unity vector $\vec{n}$, normal to the transformation surface is calculated at this point. Finally, $\vec{n}$ is multiplied with the offset-value, and the resulting vector points at the transformed node t'_i:

$$t'_i = \vec{n}(u_i, v_i) \cdot offset \qquad (40)$$

Note that this process can principally be applied to any mesh. Problems may occur when the mesh is larger than the area defined for the Bezier surface. In this case, the surface must be extended beyond its regular parameter range.

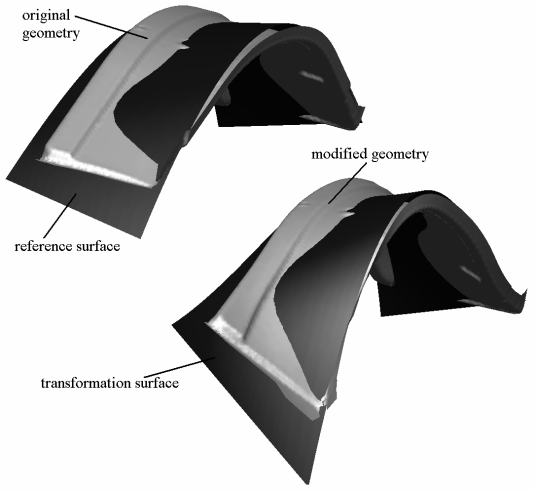

Figure 7. Modifying geometry with the control surfaces

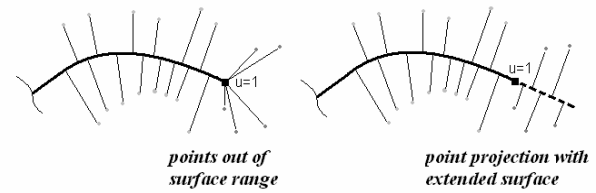

Figure. 8 Extending a parametric surface

Another difficulty is the number of variables in the matrix of control points. For a linear by quadratic Bezier surface, already 6 control points, each with 3 coordinate values, are needed. To reduce this number of free variables some limitations have been set. In general, a control point is allowed to move along a previously defined line only, reducing the amount of free variables to one per control point.

For each control point $\vec{P}_{ij}$, this line is defined with a base-point $\vec{b}_{ij}$ and a direction-vector $\vec{d}_{ij}$. These parameters are set as follows: For the fitting of the reference mesh a bounding box is constructed around the mesh. It is required that the deep drawing direction is set along the z-axis. The base points are now positioned equispaced in the x-y plane, and for each control point, the direction is set in the z-direction.

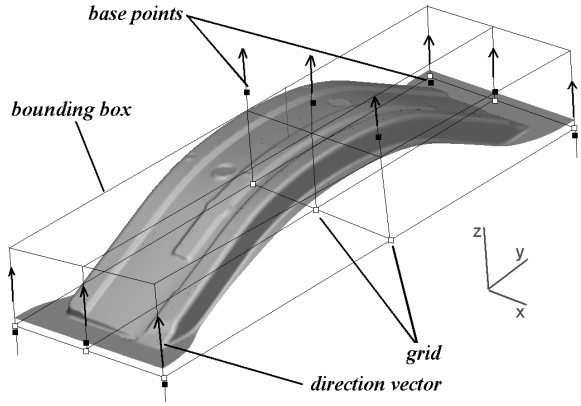

Figure 9. Constrained moving of control points

Once the surface has been fitted, the base points for the springback surface are now set as the control points of the reference surface. The direction vectors are set normal to the reference surface.

4. Compensation of an industrial part

In this section, an industrial product is analyzed and optimized with the algorithms developed in the previous sections. The product is a structural part provided by DaimlerChrysler, and was compensated in one iteration only. The product is a part of the body structure of the Mercedes-Benz E-class, and is located under the luggage compartment. It is made in a complex multistage deep drawing process. In this example, only one step of the forming process is analyzed. The previous forming stages have already been simulated and optimized separately, so the intermediate product is considered geometrically accurate. Only the flange of the product is deformed, the rest of the product is held in place by a large blankholder. It is shown how the algorithm parameters need to be set to acquire accurate results. The modification of a tool mesh is visualized. The resulting reduction in shape deviation is evaluated. The results of both methods are compared to investigate their advantages and disadvantages.

Figure 10. Flanging of the DC-part

FEM simulations performed with the commercial code INDEED are used as the basis for the compensation. In industrial applications, care has to be taken when compensating springback using simulation data, because the springback calculations are not yet completely reliable. When the springback calculation is flawed in some way, the compensation will be wrong as well, and the algorithm may even worsen the products geometrical accuracy. It is recommended to gain experience with springback calculations and check the results or calibrate the calculation by doing extensive measurements on real products first. One should realize that the compensation can only be as accurate as the FEM results and that inaccuracy builds up fast. For example: when an 80% accurate FEM analysis is used, and the springback compensation reaches 70% accuracy, the final effectivity of the procedure can only reach 56%. In this case, the calculated springback was in good agreement with practice.

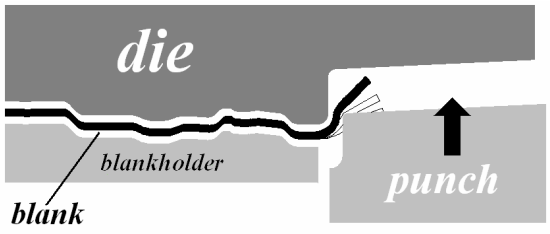

Figure 11. Schematic cross-section of the DC tools

In figure 12 (left) the norm of the shape deviation field between the reference and springback meshes is shown. Because the product is nearly symmetrical, only one half of the product is shown. The maximum shape deviation amounts 4.2mm The springback can be seen as a global mode, because the distance field is smooth.

Using the SCO method

After inspecting the springback deformation, suitable surface degrees are selected. In this example, a linear by cubic surface with 8 parameters is used. With this surface, it is possible to fit the product shape quite accurately: in one direction the product is relatively flat, and in the other direction the product has a relatively constant curvature. In this product, the springback deformation is a combination of torsion and bending in the length of the product. This can be modeled accurately with a linear by cubic surface as well.

The springback modeling capabilities of the surface can be checked by applying the algorithm to the reference mesh with a compensation factor of -1.0. Obviously, the shape of the compensated geometry should then closely resemble the springback mesh. If the shapes are significantly different, a surface with more control points should be used. The amount of movable control points is always a compromise between the algorithm's stability and the accuracy of the springback modeling. After the surface definition, the parameters for the surface-fitting process have to be set. Convergence is reached when the objective function from equation (31) does not change more than a specified parameter ε. Also, the maximum number of iterations needs to be specified. If no convergence is reached, the surface fitting is halted. It is highly recommended to review the input parameters then, and to restart the algorithm.

Now the SCO algorithm is applied and the die, blankholder and punch are modified. With the modified tools, a new FE simulation is carried out. In figure 12 the normal distance from the reference mesh to the springback mesh has been visualized. The left part is simulated with the original tools, the right part with the optimized tools. The overbending factor has been set at 250%. Note that this overbending factor is significantly larger than the 130% rule of thumb, applied in the industry.

After optimization the global mode of the springback has been removed, a smaller springback mode remains. The small flanges on the left and right of the product still show large springback. This is because they are not held in position by the (optimized) blankholder, and will be formed into the right shape in a subsequent forming phase. For this, the maximum shape deviation after optimization is 1.5mm, a decrease in shape deviation of 64%.

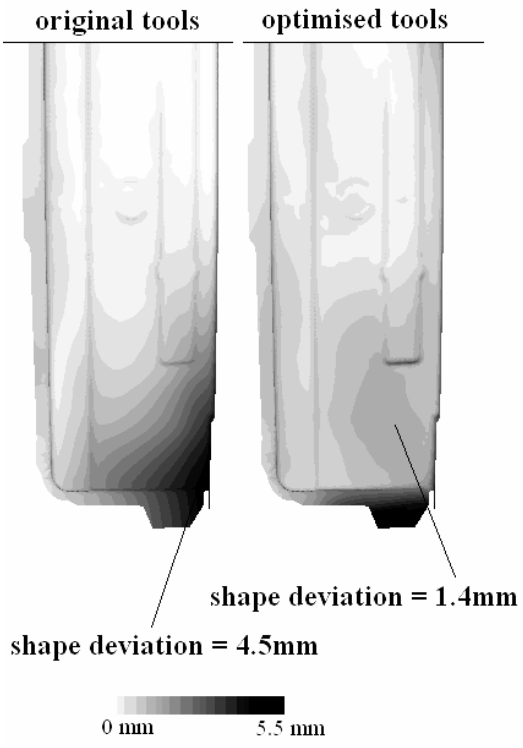

Figure 12. Distance plot for the original (left) and optimized toolset (right).

Using the SDA method

For the SDA method, the overbending factor has been set at 250% as well. This is the only parameter that needs to be set. For this method, it turned out that the gap width between the tools was changed after optimization. This changed the blank draw-in in the second FE simulation. The solution is to modify the die only, and manually offset the other tools, using the modified die-geometry.

The result of the analysis is shown in figure 13. With the SCO method, a springback reduction of 64% is obtained. The SDA method performs slightly better, the effectivity of the compensation amounts 71%. The advantage over the SCO method is not as large as expected, due to the coarseness of the interpolation function. Experiments with higher order polynomials have not been successful either, because the resulting functions suffer from instabilities. Most likely the advantage of the SDA method will become clearer when the algorithms are compared in an optimization with more iterations. Implementation of an iterative algorithm will be the first focus in future work.

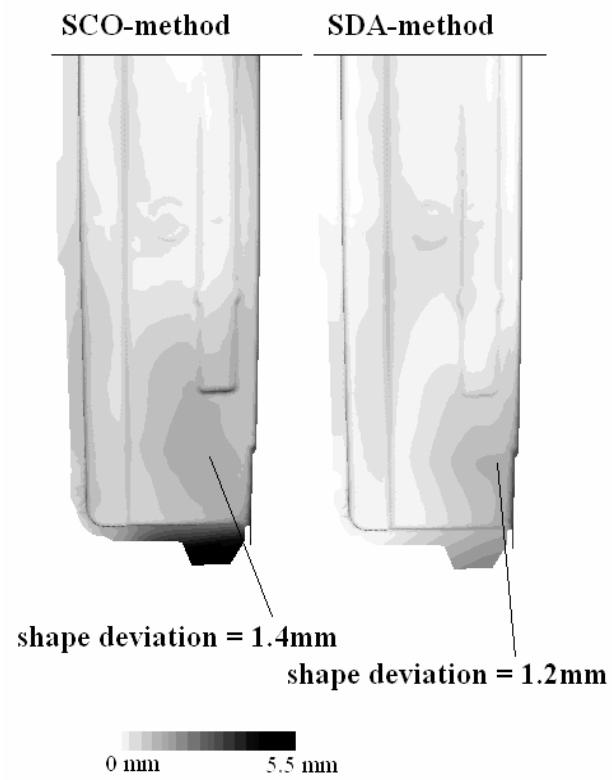

Fig 13. Distance plots for the products, optimized with the SCO and SDA methods.

5. Modification of CAD data

To make a springback compensation algorithm useable, the same geometry modification that is applied to the tool *meshes* needs to be applied to the CAD data of the tools. The smooth and continuous description of geometry in CAD files is required for the generation of NC code for the milling robot. The process engineer can still make manual changes to the tool design, after the algorithm has been applied.

In CAD programs, the geometry of complex deep drawn sheet metal products is represented mainly by parametric freeform surfaces. NURBS-surfaces (non-uniform rational b-spline surfaces) are the most flexible freeform description of geometry. Simpler non-freeform geometric entities, such as flat and cylindrical surfaces, are also used. To allow for free modification of geometry, these surfaces need to be converted into a freeform description. When all geometric entities are in freeform description, the geometry can be modified by applying the shape modification function of equation (22) to the control points of the surfaces. This has already been demonstrated in [18].

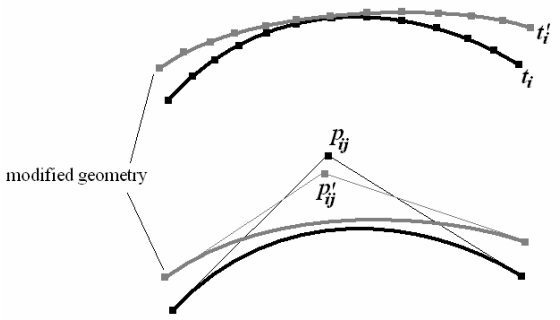

Fig 14. Modificatition of meshes (top) and CAD geometry (bottom)

The same strategy can be followed for the SCO method. Instead of transforming the set of tool-nodes from equation (38), the control points are now modified with the control-surfaces. Modification of geometry with control-surfaces has already been implemented in sophisticated freeform-surface modeling software such as ICEM-surf. As becomes clear from figure 14, the modified CAD geometry will not be exactly identical to the modified mesh, because the control points for the parametric surfaces are outside the surfaces themselves. However, for slightly curved geometries, the control points are very close to the surfaces, making the difference in shape negligible. When small radii are present in certain surfaces, these surfaces can be refined. Refining means that more control points are added to the surface description without altering its shape. If the amount of control points is increased, they get closer to the surface.

The number of control points of the surfaces also define how accurate the shape transformation will be. When, for example, the CAD-geometry contains a linear by linear surface, and the product needs to be bent due to springback compensation, the linear by linear surface needs to get higher surface degrees because otherwise it will remain flat.

Another problem is that a CAD geometry consists of more than one parametric surface. The surfaces are connected with a set of boundary conditions to maintain the smoothness requirements that are a very important feature for ‘Class A’ car body panels. These boundary conditions need to be present in the CAD data structure, so they can be taken into consideration. Unfortunately this is currently not the case in most CAD systems.

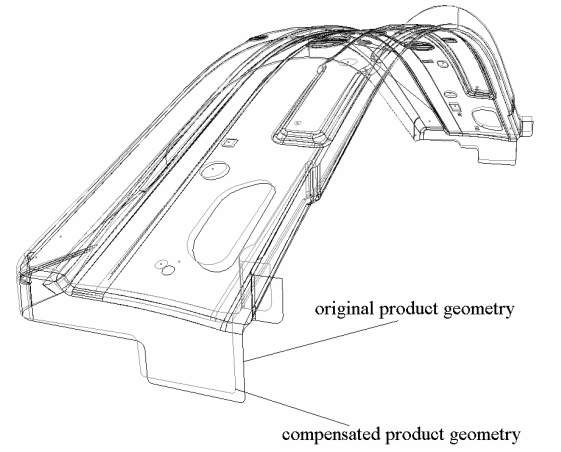

Fig 15. Modification of CAD geometry

To demonstrate the process, the reference and transformation surfaces and the CAD data of the DaimlerChrysler part have been imported into ICEM-surf. The surface degrees in the product’s CAD geometry had to be increased to 8, to make the shape modification sufficiently accurate. The results are shown in figure 15. To evaluate the difference between the modified CAD geometry and the identically modified mesh, the (normal) distance between both geometries was measured, and amounts less than 0.2 mm, an excellent result.

The smoothness of the surfaces was generally retained, but two small gaps occurred. This is related to the structure of the CAD geometry,

and the use of *trimmed surfaces*. A trimmed surface is a part of a parametric surface, that is bounded by a (parametric) trimming curve, as shown in figure 16 (top). Adjacent regular surfaces share their boundary curve: the curve where one of the parameters is either 0 or 1. These boundary curves need to be identical to make the transition at least C-0 continuous (figure 16, bottom left). For two adjacent trimmed surfaces the shared boundary curve is also identical, but it should lie in both surface bases as well (figure 16, bottom right). This boundary condition is hard to retain when the surface bases are both changed.
The preservation of higher order continuity is generally very problematic because the parameterization of both trimmed surfaces is entirely different.

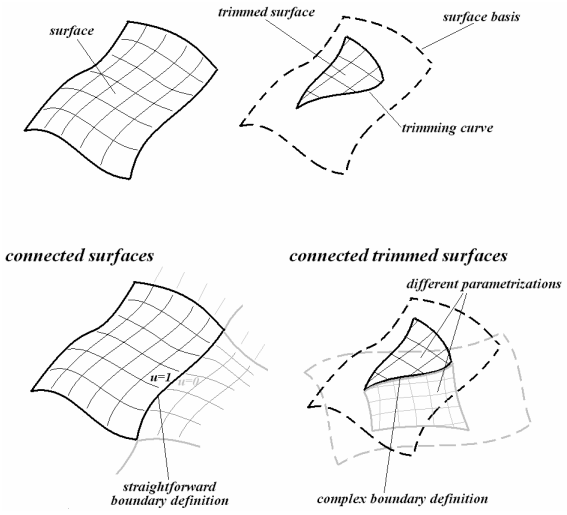

Fig 16. Trimmed surfaces

6. Conclusion

Both the SDA and SCO methods have proven to be effective in compensating springback in industrial products. With the SCO method and ICEM-surf, modification of CAD geometries is principally possible as well. This makes the method already useable in industrial practice. The overbending factor depends strongly on the geometry and material of the product, and on the forming process. The effectivity of the springback compensation also varies greatly; Geometrically stable products that show slight springback are harder to compensate than instable products that show large springback.

When both algorithms are applied to the same problem and the same compensation factor is used, the results are comparable. The SDA and SCO methods both have their limitations. With the SDA method, the approximation/extrapolation function filters out detailed information in the shape modification field. Therefore, the shape modification possibilities are limited, and the effectivity will be lower than the standard DA method. Right now, polynomials are used as an approximation function. When the degree of the polynomials is raised, more details are captured, but instabilities occur. The problem is even clearer for the SCO method: the control surfaces only allow 'quasi 2D' shape modifications, such as bending, or the application of torsion and camber. To allow detailed compensation, the control surface must contain many control points. However, having too many control-points will make the surface fitting process , and consequently the springback compensation unstable.

Most likely, the algorithms need to be applied iteratively to obtain maximum effect. In literature, it is shown that the DA method is robust, and it converges rapidly. However, this was demonstrated for an academic example-product, a two-dimensional pure bending problem [13].

Most importantly, the deep drawing simulations that form the basis of the compensation algorithm still need to become much more reliable to achieve industrially applicable springback compensation. However, the experiences and results that were obtained with this project provide a promising outlook for the future. They show that FE deep drawing simulations can be used not only for feasibility checks, but also for the (re)design of deep drawing tools.

Acknowledgements

The authors wish to express their thanks to DaimlerChrysler for supplying the demonstration part and for continuous support during the project.

ICEM-technologies is thanked for their support during the research on CAD modifications.

References

[1] A. El-Domiaty, M. Shabara and M. Al-Ansary, Determination of stretch-bendability of sheet-metals, Int. Journ. of Mach. Tools. Manuf., 36(5), p. 635-650, 1996

[2] Y. Huang, T. Chen, An elasto-plastic finite-element analysis of sheet metal camber process, J. of Materials Processing Techn., 140(1-3), p. 432-440, 2003

[3] L. Lei, S. Hwang and B. Kang, Finite element analysis and design in stainless steel sheet forming and its experimental comparison, J. of Materials Processing Techn., 110 (1), p. 70-77, 2001

[4] Z. Tekiner, An experimental study on the examination of springback of sheet metals with several thicknesses and properties in bending dies, J.of Materials Processing Techn. 145 (1), p. 109-117, 2004

[5] D. Yang et al (eds.)., NUMISHEET 2002 vol.2 benchmark problems and results, Jeju Island, 2002

[6] E. Chu , L. Zhang , S. Wang, X. Zhu,B. Maker, Validation of springback predictability with experimental measurements and die compensation for automotive panels, proceedings NUMISHEET 2002, D. Yang et al. (eds.), p.313-318, 2002

[7] R. Wagoner, Fundamental aspects of springback in sheet metal forming, proceedings NUMISHEET2002, , D. Yang et al. (eds.), p.13-19, 2002

[8] R. Wagoner, Design of sheet forming dies for springback compensation, proceedings ESAFORM2003, V. Brucato (ed.), p.7-14, Salerno, 2003

[9] O. Ghouati, D. Joannic, J. Gelin, Optimisation of process parameters for the control of springback in deep drawing, proceedings Numiform98, J. Huetink, F. Baaijens (eds.), Balkema, Rotterdam 1998

[10] A.P. Karafillis, M.C. Boyce, Tooling design in sheet metal forming using springback calculations, Int. J. Mach. Tools. Manuf., 34, p.113, 1992

[11] A.P. Karafillis, M.C. Boyce, Tooling and Binder for sheet metal forming processes compensating Springback Error, Int. J. Mach. Tools. Manuf., 36, p.503, 1996

[12] F. Valente, Compensation of springback of sheet metal forming by die design, proceedings ICAFT 2003, R. Neugebauer (ed.), p.133-147 Verlag Wissenschaftliche Scripten, Zwickau 2003

[13] W. Gan, R. H. Wagoner, Die Design Method for Sheet Springback, Submitted to the International Journal of Mechanical Sciences, Feb. 2003.

[14] W. Gan, R. Wagoner, K. Mao, S. Price, F. Rasouli, Practical Design Methods Using FE Modeling Applied to Sheet Parts and Dies, Submitted to the Journal of Engineering Materials and Technology – Transactions of the ASME, May 22, 2003.

[15] J. Weiher, B. Rietman, K. Kose, S. Ohnimus, M. Petzoldt, Controlling springback with compensation strategies, NUMIFORM 2004 proceedings, p.1011-1015 Columbus, 2004

[16] L. Piegl, W. Tiller, The NURBS book, Springer, Berlin, 1997

[17] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C, Cambridge University Press, 1992

[18] T. Sederberg, S. Perry, Freeform deformation of solid geometric models, Computer Graphics, 20(4):151-160, SIGGRAPH proceedings, 1986