11. Varilux Comfort, natural vision by large viewing zones and a soft periphery

A new design generation

From about 1980 on, after the years of the Varilux invention and market introduction, the different designs developed by the Essilor competitors made the new spectacle lens concept increasingly popular. These designs could be roughly divided in two type classes. The first type of surface was the "hard design", characterized by large viewing zones for far and near vision, a rather short power increase and moderate to strong lateral astigmatism. The second lens type was the "soft progressive design" distinguished by a longer progression channel with smaller FV and NV fields and low aberrations in the periphery. The "hard" type was well adapted for static viewing conditions, for example reading, whereas the "soft" type was very performant for dynamic viewing situations as head and eye movements.

To overcome these boundaries and to develop a lens combining the advantages of both types allowing the wearer to see "almost naturally" was the challenge for a new lens generation.

The development of a power profile, which minimizes the physiological effort of lowering the head and the eyes when reading, is described by Christian Miège and Claude Pedrono in a scientific paper [1] and in the patents US 5 270 745 and US 5 272 495. They propose a principal meridian consisting substantially of three straight line segments: the first segment is extending vertically from the top of the lens to the fitting center, the second segment is extending obliquely from the fitting center to the point where the power increase reaches 85% of the add power, and the third segment is starting from this point until the bottom of the lens passing through the near vision reference point [2].

In the US patent 5 488 442 the inventors C. Harsigny, C. Miege, J.P. Chauveau and F. Asbahs specify the characteristics of the surface design which combines the rather short meridian of US 5 270 745 with a soft lens periphery. They conclude that power and surface cylinder gradients have a primordial importance for the quality of extra-foveal and dynamic vision. The patent includes quantitative conditions for the location and the maximum value of the gradients.

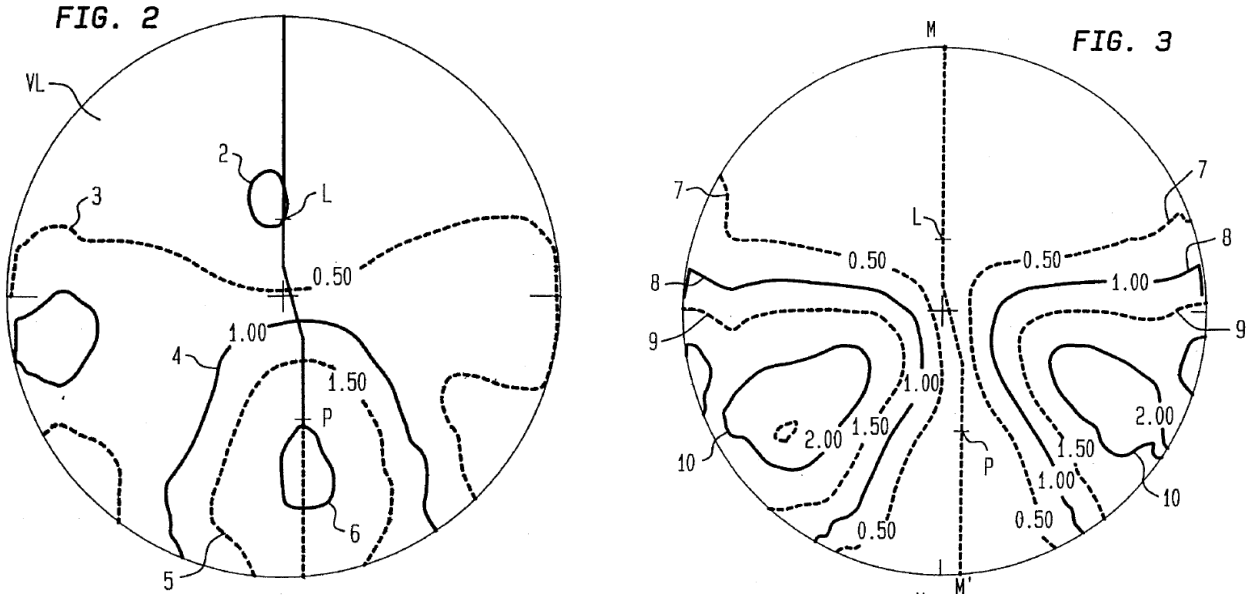

Fig 1: Patent US 5 488 442, Varilux Comfort add 2

Finally, at this time a powerful optimization tool was ready for a first test: the design loop [3]. This notion specifies an iterative process improving step by step the geometry of the lens taking into account the feedback of clinical trials.

For the calculation of such a new design, the mathematical tools used until now were not flexible enough, but in the 1980's a new method allowing to calculate freeform surfaces made its entry into the field of spectacle lens computation (for example [4]).

11.1 Freeform surfaces built with splines

The designs in the foregoing chapters were calculated following the concept of a central backbone, the main meridian, and orthogonal surface sections affixed to it. The orthogonal sections were given by rather simple functions with a limited number of parameters. For Varilux 2 as well as for Progressif R, after taking into account the condition that the meridian is an umbilic, only one parameter remains for the modeling of the orthogonal section. Consequently the surfaces generated by these sections were far from representing the optimum geometry for the minimization of aberrations. In order to bypass this severe restriction, the Varilux 2 design in the periphery departs from the exact conic section geometry in constructing secondary umbilics and isoprismatic lines.

What the designer is looking for, is a mathematical method which provides the freedom to compute a general aspheric surface directly according to the specifications, independent from a limited number of variables and the corresponding number of conditions to define them.

Freeform surfaces were and are customary in the shipbuilding, aircraft and automotive industry and the physical splines, for example wooden planks bent around pegs laid out on the floor, were used already for designing during World War Two. Isaac Jacob Schoenberg introduced the notion of the mathematical spline. You will find many articles about the history of splines on the internet. In the late 1950's and in the early 1960's Citroen, Renault and General Motors were modeling car bodies using splines or similar techniques. Today computer aided design is applied for construction work in all technical branches. And this is by far not all, …. if you watched the liquid metal cyborg in "Judgement Day" you saw splines in action.

A simple definition is the following one:

A spline of degree k is a function that is piecewise-defined by polynomial functions of degree k, and at the places where the polynomial pieces connect (which are known as knots) it is (k-1) times continuously differentiable

In the optimization process described here we will consider uniform cubic splines i.e. polynomials of degree 3 with knots, which are equally spaced. Any spline function of degree 3 defined over a uniform knot sequence can be represented as a linear combination of uniform cubic Basis splines (B-splines). Cubic B-splines have a non-zero value (its "support") over only 4 successive segments and at the knots the cubic spline, its first and second derivative are continuous (C2 continuity). And this characteristic is exactly the minimum requirement we need, if we want to describe and analyze the curvature and the astigmatism of an optical surface

The optimization method to create the surface will be an approximation technique: the surface shall pass as close as possible the given control vertices. These control points represent the surface or one of the surface characteristics which we want to realize. The advantage using B-splines is local control, which mean, that altering the position of a data point changes the surface only in the neighborhood of this point (if you will try to optimize a progressive surface you will find this is still enough work to do).

There are many text-books introducing to splines in geometric modeling, the book "Splines for use in Computer Graphics & Geometric Modeling", written by the "Killer B's" offers good explanations and a clear structure [5].

11.2 Establishing the Merit Function

As in the chapters before we restrict the optimization to the progressive front surface without taking into account the aberrations caused by the thick lens, as for example the oblique astigmatism.

In [6] and [7] the authors give a general description how to optimize a progressive surface minimizing a merit function taking into account several different optical and physiological parameters.

The design calculations of this publication follow the approach developed by J. Loos, G. Greiner and H.-P. Seidel [8]. We formulate the requirements for an appropriate front surface by the error functional

$$J(f) = \int_{\Omega} (\alpha(\kappa_1 - \kappa_2)^2 + \beta(\frac{\kappa_1+\kappa_2}{2} - H_{soll})^2) g * du * dv \qquad (1)$$

Where

* f(u, v) represents the front surface defined on a quadratic (u,v)-parameter range Ω

* $\kappa_1, \kappa_{2,} H_{soll}$ are the common symbols for the principal curvatures and the targeted mean curvature

* g the area element $g = \sqrt{1 + f_u^2 + f_v^2}$ with f_u and f_v the first derivatives of f(u, v)

*α, β are the weight functions enforcing low astigmatism and small deviations from required mean curvature

Multiplied by (n-1), $(\kappa_1 - \kappa_2)$ is the surface astigmatism and $(\kappa_1 + \kappa_2)/2$ the mean power.

Thus, the designer has first to specify the desired distribution of the mean curvature H_{soll} and of the weight functions α,β and then to determine f(u,v) so, that the functional J(f) becomes minimal.

The ideal power distribution would be to have horizontal isopower-lines, the power increasing consistently with the progression of the principal meridian from the FV to the NV part. Generally the optical engineer will choose higher values for α and β in the far and near vision regions as well as in the central progression than in the lens periphery.

11.3 Minimizing the error functional

The problem reminds physicists of the minimization of the action integral S, which can be solved using the Euler Lagrange equations. Here we will evaluate a numerical solution, as there is no way for an analytic expression of the solution.

The functional (1) is well defined for functional surfaces

$$F(u,v) = (u,v,f(u,v))$$

which can be differentiated continuously up to the order of two, i.e. $f(u,v) \in C^2(\Omega)$.

Accordingly to the approach of Ritz-Galerkin we choose the subspace of $C^2(\Omega)$ of bicubic tensor product B-splines as an approximate solution of the minimization problem:

$$f(u,v) = \sum_i \sum_j x_{ij} * N_i(u) * N_j(v) \qquad (2)$$

with the control vertices x_{ij} and the univariate cubic B-spline functions N_i.

Now we have to determine the x_{ij} so that $J(f)$ becomes minimal.

Introducing the mean curvature H and the Gaussian curvature G into (1) we obtain

$$\int\int (4\alpha * (H^2 - G) + \beta * (H - H_{soll})^2) * g du dv =$$

$$= \int\int ((4\alpha + \beta) * H^2 - 4\alpha * G - 2\beta * H * H_{soll} + \beta * {H_{soll}}^2) * g du dv \quad (3)$$

With

$$H = \frac{(1+f_v^2)*f_{uu} - 2f_u*f_v*f_{uv} + (1+f_u^2)*f_{vv}}{2g^3} \quad , \quad G = \frac{f_{uu}*f_{vv} - f_{uv}^2}{g^4}$$

the integrand can now be written as

$$F^t * A(f_u, f_v) * F + 2B(f_u, f_v)^t * F + C$$

where

$$F^t = (f_{uu}, f_{uv}, f_{vv})$$

Thus, the integrand is a quadratic expression in the second derivatives of the surface function $f(u,v)$, the coefficients a_{ij} and b_i depending exclusively on the first derivatives. If we could consider the first derivatives to be constant, the functional $J(f)$ would represent a quadratic functional. Minimizing a quadratic functional can be reduced to solving a linear system of equations. We will develop the linear equation system in paragraph 11.5.

The analysis of the expressions of the arrays A and B calculated below shows that their derivatives with respect to f_u and f_v vanish for $f_u = 0$ and $f_v = 0$. As the boundary conditions of the problem require that in the lens center $(0,0)$ the first derivatives f_u and f_v are zero and as spectacle surfaces are rather flat, a functional depending only on the second derivatives could be a reasonable approximation.

The initial surface is the sphere with a radius which is the mean of the far vision (6 D) and near vision (8 D) sphere.

Thus, we calculate $f(u,v)$ by iteration. For each new step the ${f_u}'s$ and ${f_v}'s$ are the results of the preceding iteration step and kept constant. Now the task is the minimization of a quadratic functional and the solution of the corresponding system of linear equations gives the improved values for the first derivatives for the next step until the process converges. This iterative solution method is known as the approximation with frozen first order derivatives.

11.4 The bicubic tensor product B-spline surface

A uniform bicubic B-spline surface over the quadratic (u, v)- parameter region

[-n $*$ k_d , n $*$ k_d] x [-n $*$ k_d , n $*$ k_d] is represented by

$$f(u,v) = \sum_{i=-(n+1)}^{(n+1)} \sum_{j=-(n+1)}^{(n+1)} x_{ij} * N_i(u) * N_j(v) \qquad (2)$$

k_d is the constant knot distance, the control vertices x_{ij} are defined over the knots of the parameter grid, $N_i(u) * N_j(v)$ is a tensor product B-spline, where $N_i(u)$ and $N_j(v)$ are univariate, uniform cubic B-splines.

Each cubic B-spline N_i is a piecewise cubic polynomial with non-zero values over only 4 successive intervals. At the joints of the segments the polynomial pieces are continuous up to the second derivative. Starting from one B-spline the totality of basis splines can be obtained by simple translation by multiples of the knot distance.

Thus, the representation of the functional surface is $(u, v, f(u, v))$.

If we define the first univariate cubic B-spline between -2 and +2 with knot distance 1 (which we will call $B_0(u)$), we get

$$|u| < 1 \qquad B_0(u) = \frac{2}{3} - u^2 + \frac{1}{2} * |u|^3$$

$$1 \leq |u| < 2 \qquad B_0(u) = 1/6 * (2 - |u|))^3$$

As the cubic tensor product B-spline is the product of two univariate cubic B-splines, any tensor product B-spline $N_i(u) * N_j(v)$ will be non-zero over 16 square regions of the (u,v)-mesh. With this system of basis-splines any C2 tensor product spline surface can be represented.

11.5 Establishing the linear equation system (LES)

In 11.3 we mentioned that minimizing a quadratic functional can be reduced to solving a system of linear equations.

Applying the method of the frozen first derivatives we saw that we can consider approximately the merit function $J(f)$ as a quadratic functional $J(F)$ with $F^t = (f_{uu}, f_{uv}, f_{vv})$.

$$J(F) = \iint (F^t * A(f_u, f_v) * F + 2B(f_u, f_v)^t * F + C) * g du dv \qquad (4)$$
$$= Q2(F, F) + Q1(F) + Q0$$

With the representation of F by tensor product B-splines (2) the system of linear equations to solve has the following form

$$\sum_k \sum_l \; Q2(N_i * N_j, N_k * N_l) * x_{kl} + 1/2 * Q1(N_i * N_j) = 0$$

Calculating A and B using the expressions in (3) we obtain for the detailed linear equation system

$$\iint du * dv * LS_{ij}(u,v) = 0$$

LS_{ij}(u,v)=

$$\left\{\begin{array}{c}\sum_k \sum_l x_{kl} * \left[\begin{array}{c} a_{11} * Ni''(u)Nk''(u)Nl(v)Nj(v) + a12 * \left(Ni''(u)Nk'(u)Nl'(v)Nj(v) + Ni'(u)Nk''(u)Nl(v)Nj'(v)\right) + \\ a22 * Ni'(u)Nk'(u)Nl'(v)Nj'(v) + a13 * \left(Ni''(u)Nk(u)Nl''(v)Nj(v) + Ni(u)Nk''(u)Nl(v)Nj''(v)\right) + \\ a23 * \left(Ni'(u)Nk(u)Nl''(v)Nj'(v) + Ni(u)Nk'(u)Nl'(v)Nj''(v)\right) + a33 * Ni(u)Nk(u)Nl''(v)Nj''(v) \end{array}\right] \\ \\ -(b1 * Ni''(u)Nj(v) + \ b2 * Ni'(u)Nj'(v) + b3 * Ni(u)Nj''(v)) \end{array}\right\}$$

$$g = \sqrt{(1 + {f_u}^2 + {f_v}^2)}$$

$$b1 = \beta * H_{soll} * (1 + fv^2)/2g^2 \qquad b2 = -\beta * H_{soll} * f_u * f_v/g^2 \qquad b3 = \beta * Hsoll * (1 + f_u^2)/2g^2$$

$$a11 = (\alpha + \beta/4) * (1 + fv^2)^2/g^5 \quad a12 = -2 * (\alpha + \beta/4) * (1 + fv^2) * fufv/g^5$$

$$a22 = (4\alpha * (1 + f_u^2)(1 + f_v^2) + \beta * f_u^2 * f_v^2)/g^5$$

$$a13 = [(\beta/4 - \alpha) * (1 + fu^2)(1 + fv^2) + 2\alpha * fu^2 fv^2]/g^5$$

$$a23 = -2(\alpha + \beta/4) * (1 + fu^2)fufv/g^5 \qquad a33 = (\alpha + \beta/4) * (1 + fu^2)^2/g^5$$

11.6 Optimization Method

The computer optimization program in the *Matlab* language solving the LES of paragraph 11.5 has been established by Matthias Innmann, M. Sc.(computer science), University of Erlangen.

The output of the program is a tensor product B-spline patch represented graphically either as a 3D surface plot or by isolines for the optical power and astigmatism of the surface. The necessary input variables are the design parameters H_{soll}, α and β.

In this special case the intention was to approximate the Varilux Comfort design given by the two patents US 5 270 745 and US 5 272 495 for the main meridian and US 5 488 442 for the surface characteristics. Fig 2 and Fig 3 in US 5 488 442 (see Fig 1 above) depict the isolines for the add power and the surface astigmatism of the Varilux Comfort geometry add 2. These 2 graphs representing a lens diameter of 60 mm have been evaluated on a 24 x 24 mm grid with steps of 2 mm. Concerning the power distribution, all elements of the add matrix have been increased by 6 D, which takes into account the power in the far vision region.

In order to start the optimization with a reasonable estimate for the grid values of α and β we used an ansatz of the type

$$\alpha_{ij} = \frac{1}{(Ast_{ij} + C1)^e} + C2$$

The same relation was established between β and the power. A rough first approximation of the constants gave C1 values around 3, C2 distinctly smaller than one and an exponent e of about 4. Now we started the calculations of the surface by adjusting the α values trying to reproduce the essential qualitative characteristics of Comfort, i.e. the isopower and isoastigmatism-lines. "Qualitative" means that no effort was invested to measure and to adjust the typical dimensions of the design, as for example the extension of the FV, NV and IV zones. The modification of the α–matrix was done by hand and even though the first surface has already been close to the typical Comfort design, the final optimization was a tedious, long lasting process. A specific problem is the fact, that J(f) is a global criteria.

The optimization tests showed, that the desired results were achieved by $\frac{\alpha max}{\beta max}$ ratios between 1/5 and 1/25, depending on the characteristical details of the design, for example, if the isoastigmatism lines are extending more or less into the far vision part (see below).

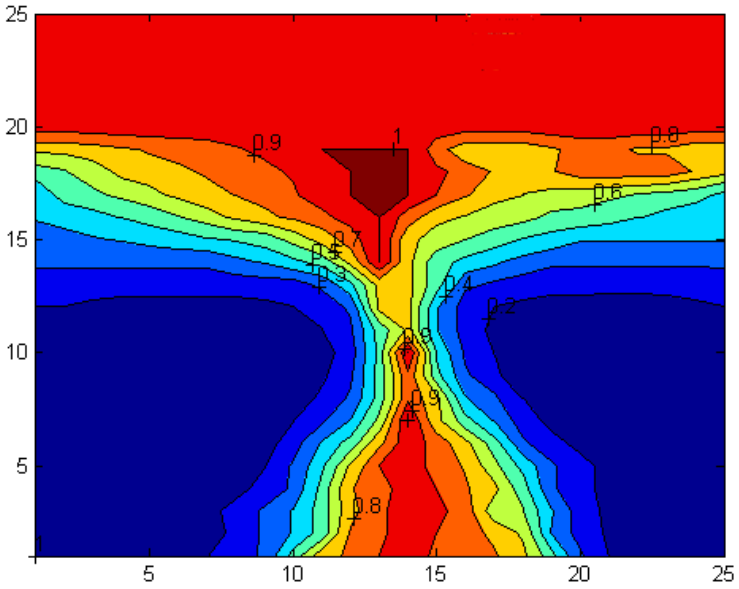

Fig 2: initial α- matrix

The initial approximate α-matrix (Fig 2) gives a design with poor far vision quality but low astigmatism in the lateral NV region, which has been corrected in the final matrix (Fig 3).

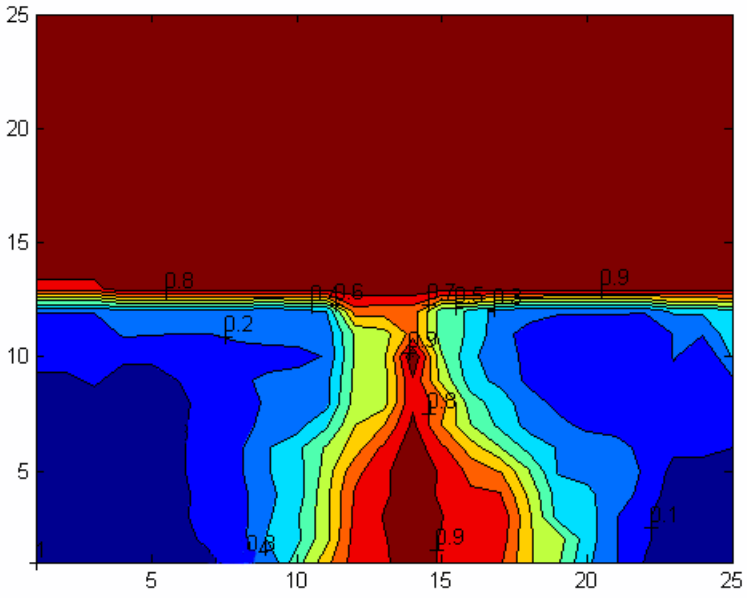

Fig 3: optimized α-matrix

We have to take into account that this way to work is more adapted to the improvement of an existing design but not appropriate to conceive a fundamentally new geometry.

For the creation of an entirely new design it seems recommendable to work with a graphical tool in order to have the possibility to model not only point by point, but also extended areas. So, the initial α and β matrices were converted into png files and these files were processed with the raster graphics editor GIMP. This way to work needs some practice, too, but provides valuable insight into the construction. For example it shows, that for the first steps it is sufficient to change the weights over extended areas.

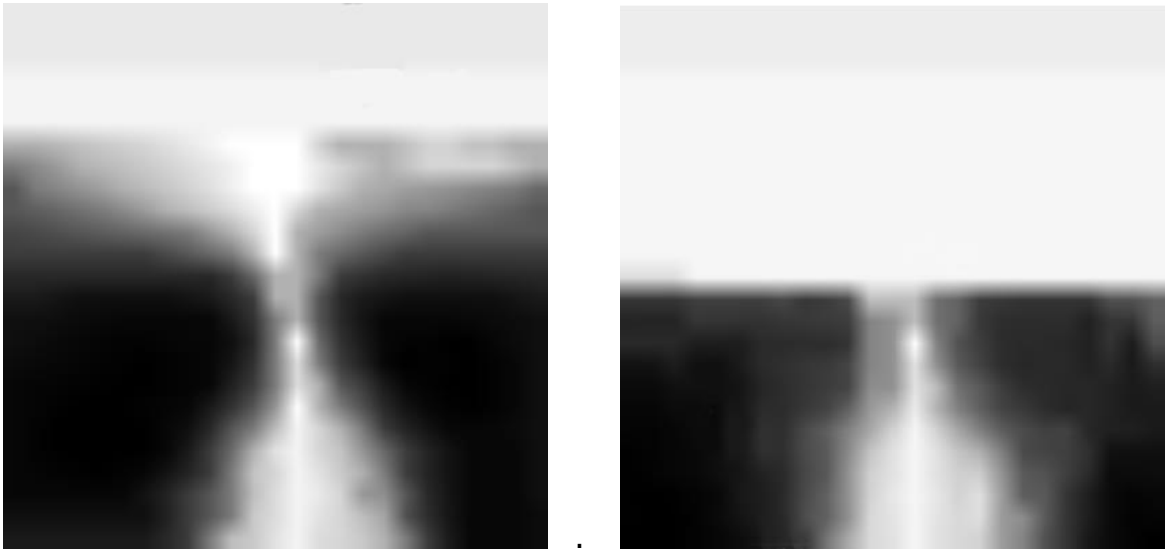

Fig 4: PNG format of the α-matrix before and after processing

Although the subjective visual information of the 2 images in Fig 4 is rather limited, it is easy to see that much effort had to be made to shift the aberrations from the far vision part into the lateral near vision region.

If the selected [x, z]-range is [-24, 24]mm x [-24, 24]mm, the solution is calculated as 12x12 tensor product B-spline patch for a uniform knot spacing k of 4 mm, and as 24x24 B-spline surface patch for a knot spacing of 2 mm, respectively. Including the boundary conditions for the surface in the zero point ($f(0,0) = 0, f_u(0,0) = 0, f_v(0,0) = 0$) there are 15x15+3 respectively 27x27+3 equations to be solved per iteration step.

The optimization was done using the 0.5 D isoastigmatism lines as criteria, for a fine tuning at least 0.25 D steps are necessary. The numbers of the α –matrix have been adjusted locally, depending on the need to correct the initial design to come close to the target, a final smoothing process of all the 625 array elements has not been realized. So the results cannot represent the optimum design , so for example the astigmatism island in the upper center of the FV part (see Fig 6 and 7) can certainly be reduced.

The computation was done on a common laptop with an *Intel®Core ™*i7-3610QM CPU 2.3GHz and a main memory of 7.9 GB. The *Matlab* version used was R 2014a completed with the parallel computing toolbox for the accelerated calculation of the matrices of the linear equation system.

For the knot distance k=4 convergence was achieved after 8 iteration steps with about 2.2 s/step. The corresponding values for k=2 are 7 iteration steps with about 10.7 s/step. The iteration was stopped, when the relative difference of the integral J(F) between 2 consecutive runs was $<10^{-9}$.

Fig 5 shows a typical graph for the evolution of the value J(F) per iteration step.

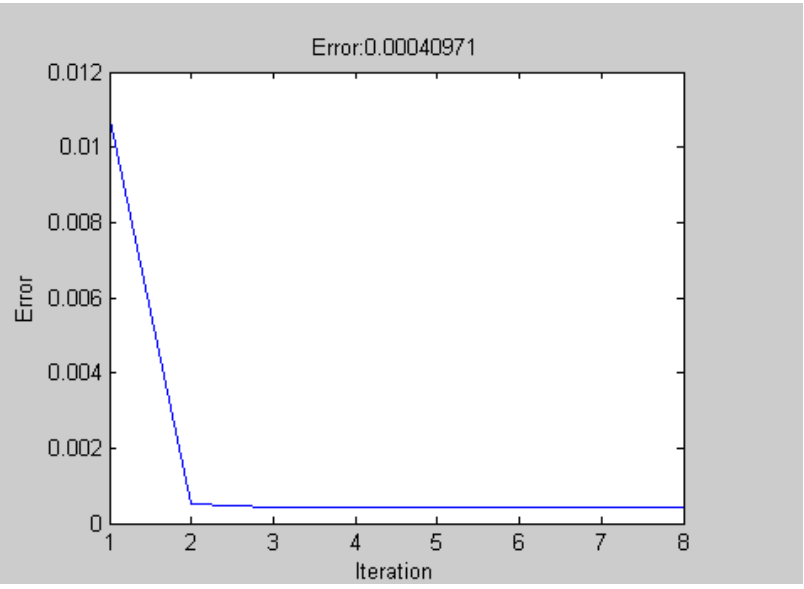

Fig 5: Iterative evolution of J(F)

11.7 Results and analysis

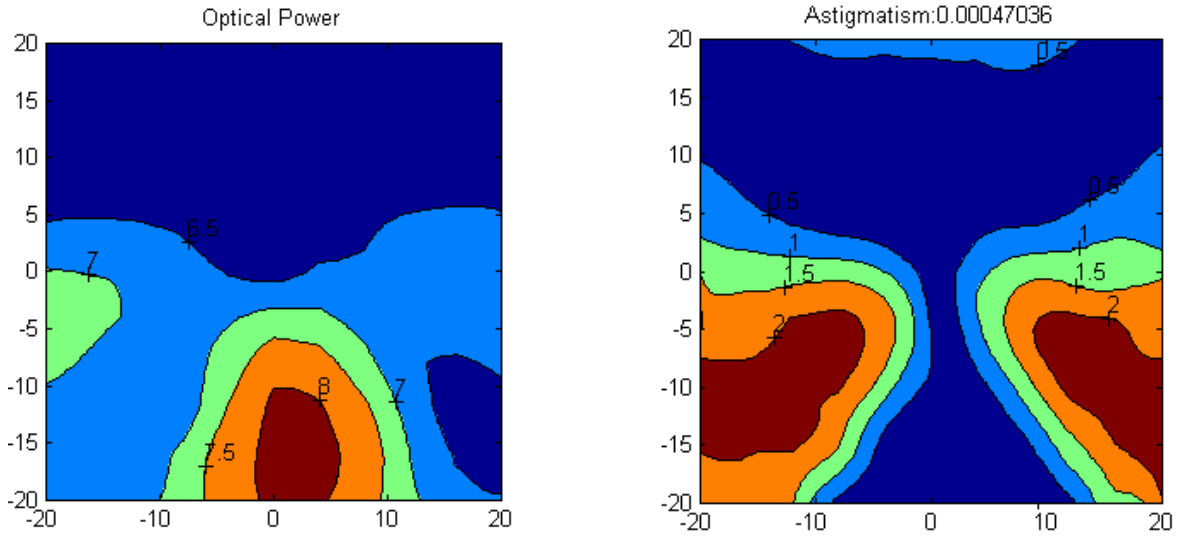

Fig 6: Power- and Astigmatism- Isolines for [-20, 20], k=4 and $\frac{\alpha max}{\beta max}$=1/25

Fig 6 and 7 show the results of two optimization approaches with far vision power 6 D and add 2 D. Fig 6 represents a design, where 0.5 D line extends higher into the lateral FV-part than for the geometry in Fig 7 (which is a big difference with regard to the α-matrices).

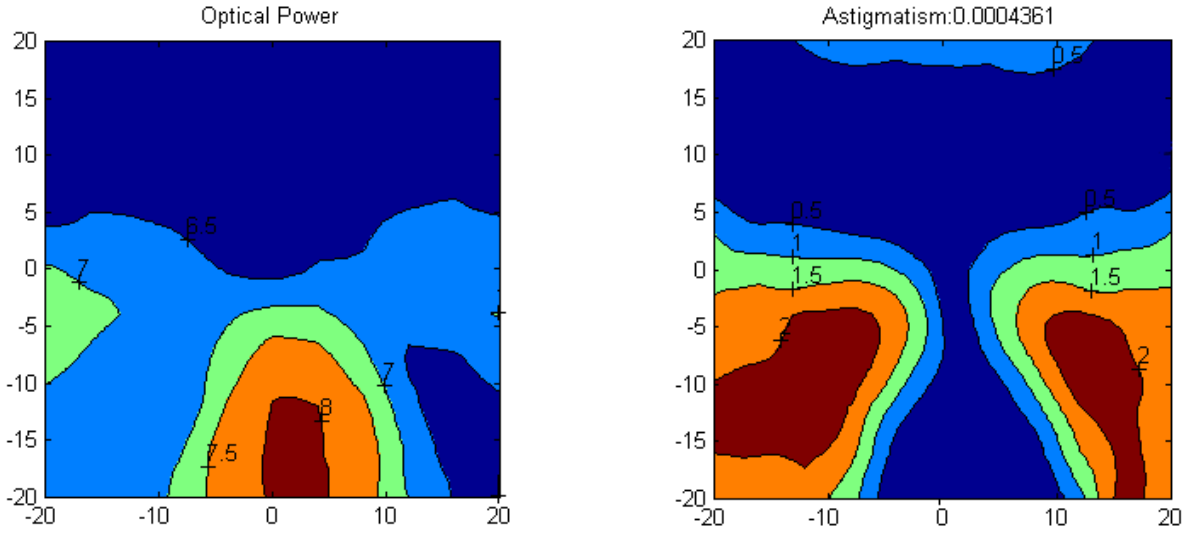

Fig 7: Power- and Astigmatism-Isolines for [-20, 20], k=4 and $\frac{\alpha max}{\beta max}$ =1/5

Both designs are distinguished by the geometry characteristics of the Comfort surface represented by patent US 5 488 442 (see Fig 1 of this chapter) and by patent US 5 270 745.

- Asymmetrical design
- The progression corridor and the near vision region follow the path of the converging eye
- In the upper lens portion the isocylinder- and the isopower-lines are substantially horizontal
- The power gradient of the surface has its maximum value on the principal meridian in the intermediate vision region
- The power increase between the points +4 mm (fitting cross) and -8 mm corresponds to 85% of the add power (corresponding to an effective progression length of 12 mm)
- The design provides extensive near and distance vision regions
- The surface astigmatism has a maximum value of about the add power
- The astigmatism gradient has a maximum value in the lateral portions of the intermediate region and is distinctly lower in the NV periphery

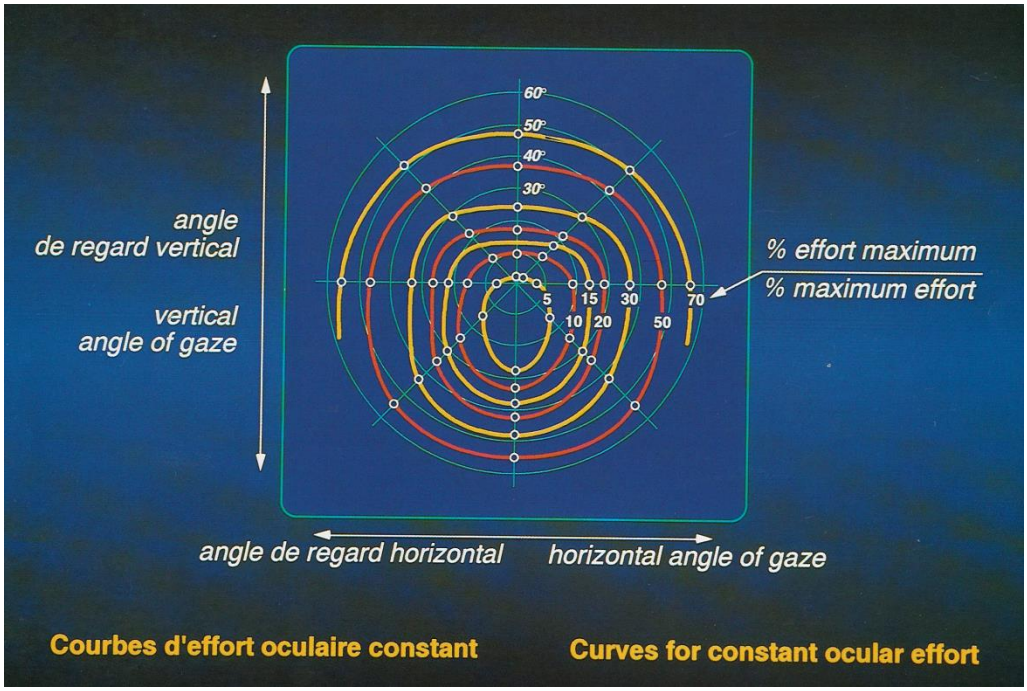

Fig 8: Curves of constant ocular effort for a point of fixation at 1m

Based on measurements of the ocular effort (Fig 8) the surface has been designed with an elevated near vision zone, which reaches 85% of the add power only 12 mm below the fitting cross. The head position and the eye inclinations of the wearer, when reading, were more natural than with other designs.

The position and extension of the zones for foveal vision, which had to be free from any perceptible imaging error, were determined with a measuring device registering the head and eye movements for different daily visual tasks (Fig 9). With the same method it could be confirmed that the lateral field of view for a progressive lens is determined less by the level of aberrations than by the gradient of aberrations. Correspondingly, the Comfort lens is distinguished by a low rate of change for power and astigmatism, which offers favorable conditions for the peripheral and dynamic vision.

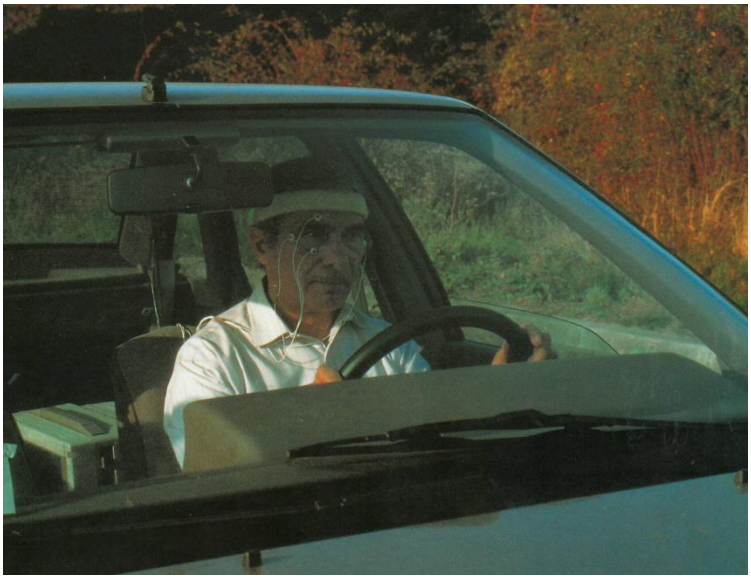

Fig 9: Apparatus for registering head- and eye movements

The whole dimension of these physiological studies and their impact on the design characteristics is described in detail in [1], [2].

The product was launched in Europe in 1993, in Asia and USA in 1994. It became the bestselling progressive brand ever and a Comfort lens adapted to the visual needs of the cyber era is still one of the cornerstones of the Essilor product range of today.

References

1. Christian Miège, Claude Pedrono: Varilux Comfort: the physiological concepts on which this new design is based. Optometriste, vol. 15, no. 5, 1993 and Optical Prism, oct 1993
2. Werner Köppen: Varilux Comfort. Deutsche Optikerzeitung, no.9, 1993
3. Jean-Louis Mercier, Christian Miège, Gilles Le Saux, Jean-Pierre Chauveau: The design loop for progressive lenses. Points de vue, no.34, spring 1996
4. Gerhard Fürter, Hans Lahres: Multifocal spectacle lens with a dioptric power varying progressively between different zones of vision. Patent US 4 606 622
5. Richard H. Bartels, John C. Beatty, Brian A. Barsky: An introduction to splines in computer graphics and geometric modeling. Morgan Kaufmann Publishers Inc., 1987
6. P. Allione, F. Ahsbahs, G.Le Saux: Application of optimization in computer-aided ophthalmic lens design. SPIE, vol. 3737,138-148
7. Eric F. Barkan, David H. Sklar: Method for improving progressive lens design and resulting article. Patent US 4 838 675
8. Joachim Loos, Günther Greiner, Hans-Peter Seidel: A variational approach to progressive lens design. Computer-Aided Design, vol. 30, no. 8, 1988

Annex : *Matlab* program for Minimizing the Error Functional

```
%% global variables
n = 525;                    % actually (n - 1) * 1000
Df = 6;
Dn = 8;

xRange = [-24 24];            % x range in mm
zRange = [-24 24];          % z range in mm

kd = 2;                     % knot distance for BSpline in mm
Ns = 2 * kd + 1;            % number of samples per knot interval

alphaWeight = 1;             % scaling factor for alpha
betaWeight = 5;              % scaling factor for beta

fileAlpha = 'alpha.txt';    % file with alpha(u, v)
fileBeta = 'beta.txt';      % file with beta(u, v)
fileHsoll = 'Hsoll.txt';    % file with hsoll(u, v)

isoLevelsAsti = 0:0.5:5;    % iso lines for astigmatism
isoLevelsPow = 1.5:0.5:10;  % iso lines for optical power

maxIter = 10;               % maximum number of iteration (since error difference
will never be = 0)
epsilon = 10e-9;           % epsilon for iteration

visible = 1;                % 1: on, 0: off
plotControlPoints = 1;
fileOutput = 0;
parallelComp = 1;


%% initialize program
p = gcp('nocreate');
if isempty(p) && parallelComp
    parpool('local');
end


%% place control points on initial surface
rho = 2 * n / (Df + Dn);

xSteps = (xRange(1, 1) - kd):kd:(xRange(1, 2) + kd);
zSteps = (zRange(1, 1) - kd):kd:(zRange(1, 2) + kd);

[Cx, Cz] = meshgrid(xSteps, zSteps);
Cy = rho - sqrt(rho^2 - Cx.^2 - Cz.^2);
```

```
%% load constraints
C = Constraints(fileAlpha, fileBeta, fileHsoll, xRange, zRange, n);


Error = zeros(1, maxIter + 1);


%% plots
if visible == 1
    h = figure;
else
    h = figure('visible', 'off');
end

prevError = 0;


%% iteration
for iter = 1:maxIter+1
    %% evaluate current surface
    [fx, fz, fy] = B0Spline.evalSurf(kd, Ns, Cx, Cz, Cy);

    hx = (max(fx) - min(fx)) / (size(fx, 2) - 1);
    hz = (max(fz) - min(fz)) / (size(fz, 1) - 1);

    %% create integrator only once (for performance issues)
    I = Integrator(size(fy, 1), size(fy, 2), hx, hz);


    %% find indices for boundary constraints
    % (f(0,0) = 0, \nabla f(0,0) = 0)

    idxI00 = floor(min(xRange) / (min(xRange) - max(xRange)) * size(fy, 2) + 1);
    idxJ00 = floor(min(zRange) / (min(zRange) - max(zRange)) * size(fy, 1) + 1);

    %% precomputed values
    A = imresize(C.A, size(fy), 'bilinear');
    B = imresize(C.B, size(fy), 'bilinear');
    Hsoll = imresize(C.H, size(fy), 'bicubic');

    A = alphaWeight * A;
    B = betaWeight * B;

    %% compute curvatures
    Fu = B0Spline.evalSurfDu(kd, Ns, Cx, Cz, Cy);
    Fv = B0Spline.evalSurfDv(kd, Ns, Cx, Cz, Cy);
    Fuu = B0Spline.evalSurfDuu(kd, Ns, Cx, Cz, Cy);
    Fuv = B0Spline.evalSurfDuv(kd, Ns, Cx, Cz, Cy);
    Fvv = B0Spline.evalSurfDvv(kd, Ns, Cx, Cz, Cy);

    G = evalG(Fu, Fv);
    H = evalH(Fu, Fv, Fuu, Fuv, Fvv, G);
    K = evalK(Fuu, Fvv, Fuv, G);
```

```
    %% current error
    E1 = 4 * A .* (H.^2 - K);
    E2 = B .* (H - Hsoll).^2;



    I1 = I.integrate(E1 .* G);
    I2 = I.integrate(E2 .* G);
    Error(iter:end) = I1 + I2;

    errorDiff = abs(Error(iter) / prevError);
    prevError = Error(iter);

    subplot(2, 3, 1)
    surf(fx, fz, fy, 'FaceColor', 'interp', 'EdgeColor', 'none')
    if plotControlPoints
        hold on
        surf(Cx(2:end-1, 2:end-1), Cz(2:end-1, 2:end-1), Cy(2:end-1, 2:end-1),
'FaceColor', 'None')
        hold off
    end
    title('Surface')
    xlabel('u')
    ylabel('v')
    axis square

    subplot(2, 3, 2)
    cont = contourf(fx, fz, H * n, isoLevelsPow);
    title('Optical Power')
    xlabel('u')
    ylabel('v')
    axis square
    clabel(cont)

    subplot(2, 3, 3)
    contourf(fx, fz, K)
    title('K: Gaussian Curvature')
    xlabel('u')
    ylabel('v')
    axis square

    subplot(2, 3, 4)
    plot(Error);
    xlabel('Iteration')
    ylabel('Error')
    title(strcat('Error: ', num2str(Error(iter))))

    subplot(2, 3, 5)
    contourf(fx, fz, abs(H - Hsoll) * n)
    title(strcat('|H - H_{soll}|: ', num2str(I2)))
    xlabel('u')
    ylabel('v')
    axis square
```

```
subplot(2, 3, 6)
cont = contourf(fx, fz, 2 * n * sqrt(H.^2 - K), isoLevelsAsti);
title(strcat('Astigmatism: ', num2str(I1)))
xlabel('u')
ylabel('v')
axis square
clabel(cont)

pause(0.01)

% save to file
if fileOutput
    saveas(h, strcat('plot', num2str(iter), '.fig'));
end

% break iteration if maxIter is reached or
% error does not decrease any more
if iter == maxIter+1 || abs(errorDiff - 1) < epsilon
    break;
end

tic
curTime = toc;      % for performance evaluation


%% set up Dx = b
% compute constant parts (for performance issues)
Fv2 = 1 + Fv.^2;
Fuv2 = 2 * Fu .* Fv;
Fu2 = 1 + Fu.^2;

ABG4 = (4 * A + B) ./ (4 * G.^5);
AG4 = 4 * A ./ G.^3;

BHG = B .* Hsoll ./ (2 * G.^2);

% set up matrix D and rhs b
length = size(Cx);

Fx = (fx - min(xRange)) / kd + 2;
Fz = (fz - min(zRange)) / kd + 2;

bPar = zeros(length);
dPar = zeros(length(2), length(1), length(2), length(1));

% constraints for curvatures
parfor i = 1:length(2)
    Fi = Fx - i;

    Ni = B0Spline.B(Fi);
    Niu = B0Spline.DB(Fi) / kd;
    Niuu = B0Spline.DDB(Fi) / kd^2;

    kStart = max(i - 3, 1);
    kEnd = min(i + 3, length(2));
```

```
bSlice = zeros(length(1), 1);
dSlice = zeros(length(1), length(2), length(1));

for j = 1:length(1)
    Fj = Fz - j;

    Nj = B0Spline.B(Fj);
    Njv = B0Spline.DB(Fj) / kd;


    Njvv = B0Spline.DDB(Fj) / kd^2;

    Njiuu = Nj * Niuu;
    Njvvi = Njvv * Ni;
    Njviu = Njv * Niu;

    Hij = Fv2 .* Njiuu - Fuv2 .* Njviu + Fu2 .* Njvvi;

    L = BHG .* Hij;

    bSlice(j) = I.integrate(L);

    lStart = max(j - 3, 1);
    lEnd = min(j + 3, length(1));

    for k = kStart:kEnd
        Fk = Fx - k;

        Nk = B0Spline.B(Fk);
        Nku = B0Spline.DB(Fk) / kd;
        Nkuu = B0Spline.DDB(Fk) / kd^2;

        for l = lStart:lEnd
            Fl = Fz - l;

            Nl = B0Spline.B(Fl);
            Nlv = B0Spline.DB(Fl) / kd;
            Nlvv = B0Spline.DDB(Fl) / kd^2;

            Nlkuu = Nl * Nkuu;
            Nlvvk = Nlvv * Nk;
            Nlvku = Nlv * Nku;

            Hkl = Fv2 .* Nlkuu - Fuv2 .* Nlvku + Fu2 .* Nlvvk;

            K = ((Njiuu .* Nlvvk + Njvvi .* Nlkuu) / 2) - Njviu .* Nlvku;

            Q = ABG4 .* Hij .* Hkl - AG4 .* K;

            dSlice(j, k, l) = I.integrate(Q);
        end
    end
end
```

```
    bPar(:, i) = bSlice;
    dPar(i, :, :, :) = dSlice;
end

b = zeros(length(1) * length(2) + 3, 1);
b(1:length(1)*length(2), 1) = reshape(bPar, length(1) * length(2), 1);

D = zeros(length(1) * length(2) + 3, length(1) * length(2));

for i = 1:length(2)

    Fi = Fx - i;

    Ni = B0Spline.B(Fi);
    Niu = B0Spline.DB(Fi) / kd;

    kStart = max(i - 3, 1);
    kEnd = min(i + 3, length(2));

    for j = 1:length(1)
        Fj = Fz - j;

        Nj = B0Spline.B(Fj);
        Njv = B0Spline.DB(Fj) / kd;

        Nij = Nj * Ni;
        Niuj = Nj * Niu;
        Nijv = Njv * Ni;

        row = (i - 1) * length(1) + j;

        D(end - 2, row) = Nij(idxJ00, idxI00);
        D(end - 1, row) = Niuj(idxJ00, idxI00);
        D(end, row) = Nijv(idxJ00, idxI00);

        lStart = max(j - 3, 1);
        lEnd = min(j + 3, length(1));

        for k = kStart:kEnd
            col = (k - 1) * length(1);

            for l = lStart:lEnd
                D(row, col + l) = dPar(i, j, k, l);
            end
        end
    end
end
```

```
x = D \ b;

    Cy = reshape(x, size(Cy));

    elapsedTime = toc - curTime;

    display(strcat('iteration: ', num2str(iter), ', time: ',
num2str(elapsedTime), 's', ', errorDiff: ', num2str(errorDiff), ', condition: ',
num2str(cond(D), '%e')));
end
```