Plotting performance map

Problem presented by

Jens Dahl Poulsen and Lars Voigt

HV-Turbo A/S

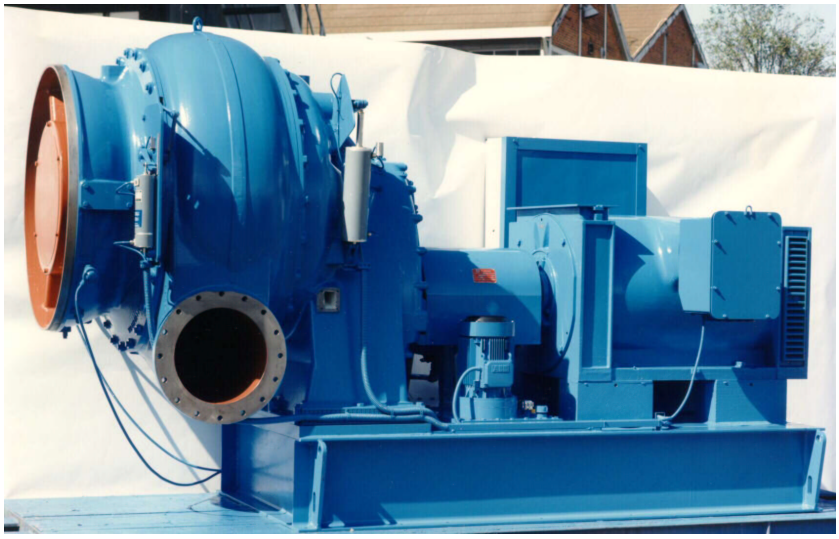

Figure 1: A compressor from HV-Turbo. The inlet is to the left and the outlet is the pipe below.

HV-TURBO A/S would like a program which can plot performance maps for radial blowers automatically. The performance map has flow along the horizontal axis and pressure along the vertical axis and shows the isentropic efficiency as function of flow and pressure.

Input data for the program are measured dimensionless values of flow, pressure and efficiency in several combinations of inlet guide vane and diffuser settings.

The output of the program should be a flow- pressure-plot containing curves which indicates combinations of flow and pressure for each inlet guide vane position and diffuser position, respectively. Furthermore the plot should contain contours of efficiency.

Finally it should be possible to give a combination of flow and pressure as input and get interpolated values of inlet guide vane position and diffuser position based on the performance map.

Study Group contributers

Jens Gravesen, Nikolaj Nordkvist, Dorthe Almind Pedersen,
Lisbeth Aagaard Pedersen, Peter Røgen, and Stefan Wolff.
Report prepared by Jens Gravesen.

1 Introduction

For a number of inlet (θ) and diffuser (ϕ) positions we are given values of normalized flow (x), normalized pressure (y), and efficiency (η). If we for a fixed value of (θ, ϕ) plot y as a function of x we obtain the possible modes of operation of the compressor with the given inlet and diffuser position. These curves are called *characteristics* of the compressor. In Section 2 we address the first task, which is to obtain the characteristics from the measurements. In the same section we also find the efficiency as a function of x.

In Section 3 we combine the characteristic maps from Section 2 to produce what HV-Turbo call the "envelopes".

In Section 4 we avoid the "envelopes" and describe a method that produces the performance map directly from the curves found in Section 2.

2 Characteristics

We now look at a fixed position (θ, ϕ) of the inlet and the diffuser, and we are given a sequence of values

$$(x_1, y_1, \eta_1), (x_2, y_2, \eta_2), \ldots, (x_n, y_n, \eta_n) \quad \text{with } x_1 < x_2 < \cdots < x_n.$$

We want to find functions $y(x)$ and $\eta(x)$ such that

$$y(x_i) = y_i \quad \text{and} \quad \eta(x_i) = \eta_i \quad \text{for } i = 1, \ldots, n, \tag{1}$$

or as measurements always have some errors, we replace the interpolation problem (1) with the approximation problem

$$y(x_i) \approx y_i \quad \text{and} \quad \eta(x_i) \approx \eta_i \quad \text{for } i = 1, \ldots, n. \tag{2}$$

We formulate both problems as least square problems

$$\text{minimize } \sum_{i=1}^{n}(y(x_i)-y_i)^2 \quad \text{and} \quad \text{minimize } \sum_{i=1}^{n}(\eta(x_i)-\eta_i)^2 \tag{3}$$

As it stands this is straight forward, we just have to pick a suitable class of functions in which to minimize. The only complication is that the characteristic $y(x)$ needs to be decreasing and convex, i.e., we have the side conditions

$$y'(x) < 0 \quad \text{and} \quad y''(x) < 0. \tag{4}$$

We have tried different classes of functions and report the result in Table .1 below. It is worth noticing that the logarithmic case with only three coefficients does

	Parabola	Logarithm	Cubic	Spline 2	Spline 3
# coeff.	3	3	4	4	5
y	5.4%	4.0%	4.8%	2.1%	0.4%
η	5.2%	—	1.4%	1.5%	0.6%

Table .1: The number of coefficients and the maximal relative error for different types of approximation.

surprisingly well. In the following subsections we give more information and present a plot of the the worst case for each class of functions.

2.1 Parabolas

We were informed that the characteristics in some cases resemble parabolas so we tried that first.

$$y(x) = ax^2 + bx + c. \tag{5}$$

If we ignore the side conditions (4) then the interpolation problem (1) is an over determined set of equations

$$\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}. \tag{6}$$

The least square solution is simply the solution to the 3×3 system found by multiplying both sides of the equation with the transpose of the 'x_i' matrix:

$$\begin{bmatrix} x_1^2 & x_2^2 & \dots & x_n^2 \\ x_1 & x_2 & \dots & x_n \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} x_1^2 & x_2^2 & \dots & x_n^2 \\ x_1 & x_2 & \dots & x_n \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \tag{7}$$

or

$$\begin{bmatrix} \sum_{i=1}^n x_i^4 & \sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_i^2 \\ \sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i & n \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n x_i^2 y_i \\ \sum_{i=1}^n x_i y_i \\ \sum_{i=1}^n y_i \end{bmatrix}. \tag{8}$$

The case of the efficiency η is of course the same. In Figure 2 the maximal relative error for the 28 datasets are plotted. For both $y(x)$ and $\eta(x)$ the worst case

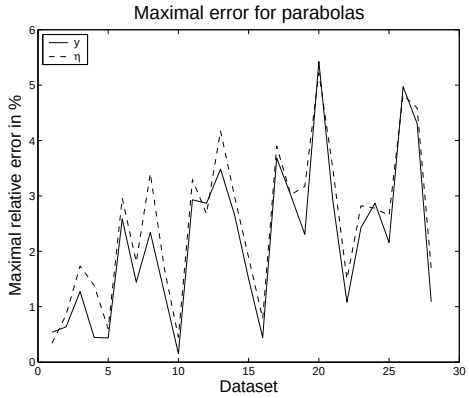

Figure 2: The maximal relative error for the 28 datasets using parabolas.

was dataset no. 20, where $(\theta, \phi) = (8, 8)$. The resulting parabolas are shown in Figure 3. We have in both cases that the error is around 5%.

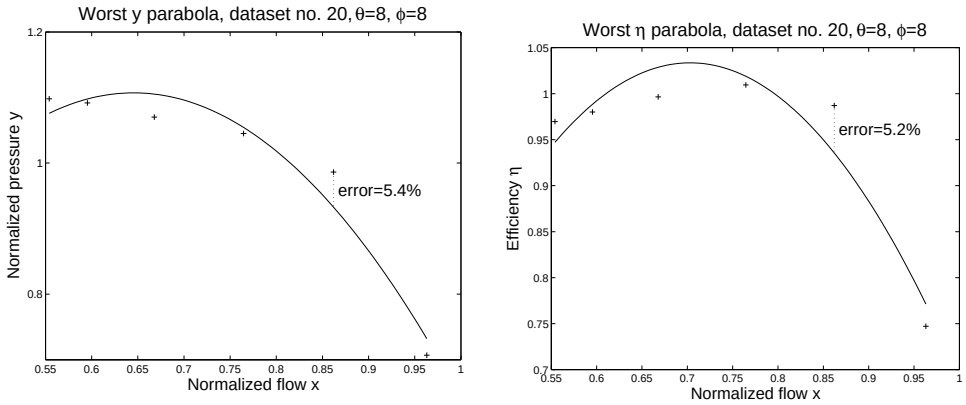

Figure 3: The worst approximating parabola. To the left $y(x)$ to the right $\eta(x)$. Observe that $y(x)$ is not monotonic decreasing and that the the dataset is non convex.

2.2 Logarithms

In order to ensure that the side conditions (4) are satisfied we can use scaled translated versions of the logarithm:

$$y(x) = a\log(b - x) + c, \tag{9}$$

$$b > x_{\max} \tag{10}$$

The conditions (4) are now automatically satisfied, but now the least square problem (3) becomes non linear. We first tried the function `lsqcurvefit` from Matlabs Optimization Toolbox [2], but ran into problems. First dataset no. 20 had problems and we needed to increase the option `MaxFunEvals` to 800 in order for the method to converge. Secondly, even when the function reported "Optimization terminated successfully: Relative function value changing with less than OPTIONS.TolFun" the result was obviously wrong. In Figure 4 the maximal relative error for the 28 datasets are plotted together with the worst case, and this is

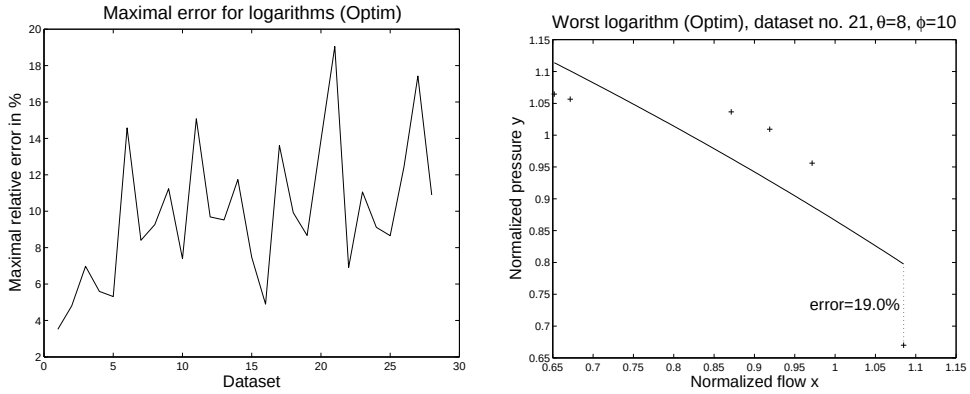

Figure 4: To the left the maximal relative error for the 28 datasets using logarithm and the optimization toolbox. To the right the worst case.

certainly not the optimal logarithm function.

In order to do better we note that if we fix b in (9) then the least square problem becomes linear again. We obtain the over determined linear system

$$\begin{bmatrix} \log(b - x_1) & 1 \\ \log(b - x_2) & 1 \\ \vdots & \vdots \\ \log(b - x_n) & 1 \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{11}$$

where the least square solution is the solution to the 2×2-system

$$\begin{bmatrix} \log(b-x_1) & \log(b-x_2) & \dots & \log(b-x_n) \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} \log(b-x_1) & 1 \\ \log(b-x_2) & 1 \\ \vdots & \vdots \\ \log(b-x_n) & 1 \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix}$$

$$= \begin{bmatrix} \log(b-x_1) & \log(b-x_2) & \dots & \log(b-x_n) \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{12}$$

or

$$\begin{bmatrix} \sum_{i=1}^n (\log(b-x_i))^2 & \sum_{i=1}^n \log(b-x_i) \\ \sum_{i=1}^n \log(b-x_i) & n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n y_i \log(b-x_i) \\ \sum_{i=1}^n y_i \end{bmatrix} \tag{13}$$

In this manner a and c becomes functions of b and the least square problem (3) becomes a question of minimizing the following function of b:

$$\sum_{i=1}^{n} \big(a(b)\log(b-x_i) + c(b) - y_i\big)^2$$

Instead of using an advanced minimization procedure we simply evaluated the function for say 1000 values of b in the interval $]x_n, x_n+1]$ and pick the smallest result. In Figure 5 the results are summarized.

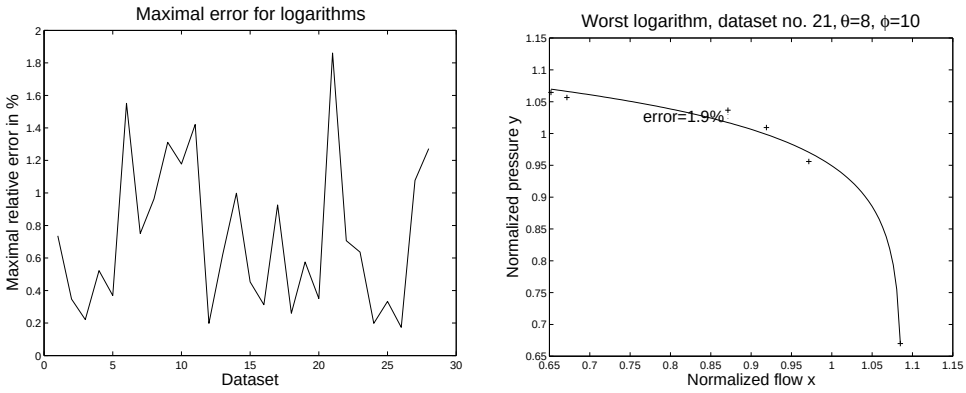

Figure 5: To the left the maximal relative error for the 28 datasets using logarithm. To the right the worst case.

2.3 Cubic Bézier

We now express both $y(x)$ and $\eta(x)$ as a cubic polynomial in *Bézier form*, see [1], i.e.,

$$y(x) = \sum_{k=0}^{3} c_k B_k^3(t) \quad \text{and} \quad \eta(x) = \sum_{k=0}^{3} d_k B_k^3(t) \tag{14}$$

where

$$x = x_1(1-t) + x_n t \quad \text{or} \quad t = \frac{x_n - x}{x_n - x_1} \tag{15}$$

and

$$B_k^n(t) = \binom{n}{k}(1-t)^{n-k}t^k \tag{16}$$

are the *Bernstein polynomials*, see [1]. They can be found by the recursion

$$\mathbf{B}^0(t) = [1], \tag{17}$$
$$\mathbf{B}^{n+1}(t) = (1-t)[\mathbf{B}^n(t), 0] + t[0, \mathbf{B}^n(t)], \tag{18}$$

where $\mathbf{B}^n(t) = [B_0^n(t), \ldots, B_n^n(t)]$. In order to formulate the side conditions (4) we need the first two derivatives of $y(x)$ or $y(t)$, see [1],

$$y'(t) = 3\sum_{k=0}^{2}(c_{k+1} - c_k)B_k^2(t), \tag{19}$$
$$y''(t) = 6\sum_{k=0}^{1}(c_{k+2} - 2c_{k+1} + c_k)B_k^1(t). \tag{20}$$

We have $B_0^1(t) = 1 - t$ and $B_1^1(t) = t$ so $y''(t) < 0$ for $t \in [0, 1]$ if and only if

$$c_{k+2} - 2c_{k+1} + c_k < 0 \quad \text{for } k = 0, 1. \tag{21}$$

The first derivative is negative if all the control points $c_{k+1} - c_k$, $k = 0, 1, 2$, are negative, but the other implication is not true. In order to get a tighter condition we use *subdivision*, see [1], i.e., we write the first and second half of the derivative on the form (19). The control points for the first half are

$$c_1 - c_0, \quad \frac{(c_1 - c_0) + (c_2 - c_1)}{2} = \frac{c_2 - c_0}{2}, \quad \frac{c_2 - c_0}{4} + \frac{c_3 - c_1}{4},$$

and for the second half

$$\frac{c_2 - c_0}{4} + \frac{c_3 - c_1}{4}, \quad \frac{(c_2 - c_1) + (c_3 - c_2)}{2} = \frac{c_3 - c_1}{2}, \quad c_3 - c_2.$$

The first derivative is negative if these numbers are negative and this happens if and only if

$$c_1 - c_0 < 0, \quad c_2 - c_0 < 0, \quad c_3 - c_1 < 0, \quad c_3 - c_2 < 0. \tag{22}$$

So for y we now have the following problem

$$\text{minimize} \sum_{i=1}^{n} \left(y_i - \sum_{k=0}^{3} c_k B_k^3(t_i) \right)^2, \tag{23}$$

under the side conditions

$$\begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{24}$$

So we minimize a quadratic functional under linear side conditions. This can be solved by *quadratic programming*, see [2, 3]. In the case of the efficiency η we have no side conditions and we end up with a linear problem. The interpolation problem (1) gives the over determined linear system

$$\begin{bmatrix} B_0^3(t_1) & B_1^3(t_1) & B_2^3(t_1) & B_3^3(t_1) \\ B_0^3(t_2) & B_1^3(t_2) & B_2^3(t_2) & B_3^3(t_2) \\ \vdots & \vdots & \vdots & \vdots \\ B_0^3(t_n) & B_1^3(t_n) & B_2^3(t_n) & B_3^3(t_n) \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

and as before the least square solution is the solution to the 4×4 system we obtain by multiplication with the transpose.

The result can be found in Figure 6 where the maximal error for the 28 datasets are plotted. For both $y(x)$ and $\eta(x)$ the worst case was dataset no. 20, where $(\theta, \phi) = (8, 8)$. The resulting cubics are shown in Figure 7.

2.4 Quadratic B-spline

Here we express $y(x)$ and $\eta(x)$ as a quadratic B-spline, see [1], i.e., a C^1 function consisting of K pieces of parabolas. Such functions can be written

$$y(x) = \sum_{k=1}^{2+K} c_k N_k^2(\mathbf{u}, t) \quad \text{and} \quad \eta(x) = \sum_{k=1}^{2+K} d_k N_k^2(\mathbf{u}, t) \tag{25}$$

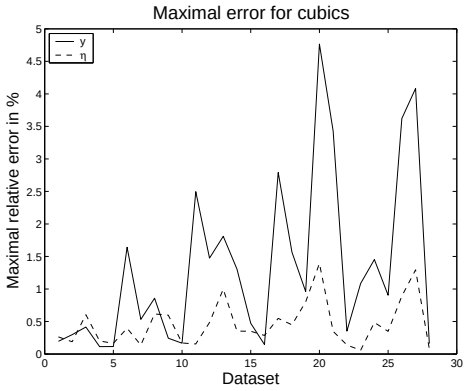

Figure 6: The maximal relative error for the 28 datasets using cubics.

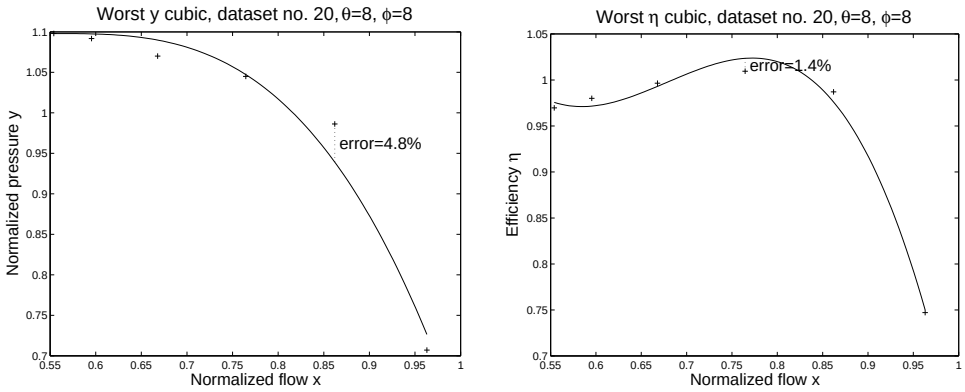

Figure 7: The worst approximating cubic. To the left $y(x)$ to the right $\eta(x)$

where $N_k^2(\mathbf{u}, t)$ are the quadratic *B-splines* on the *knot sequence*

$$\mathbf{u} = u_0, u_1, \ldots, u_{4+K}. \tag{26}$$

An important special case is the *uniform* B-spline with $u_k = k - 2$, i.e.,

$$\mathbf{u} = -2, -1, 0, \ldots, K, K+1, K+2. \tag{27}$$

Each piece of parabola is defined on the interval $[u_{k-1}, u_k]$, $k = 3, \ldots, 2 + K$. The relation between x and t is given by

$$x = x_1 \frac{u_{2+K} - t}{u_{2+K} - u_2} + x_n \frac{t - u_2}{u_{2+K} - u_2} \quad \text{or} \quad t = u_2 \frac{x_n - x}{x_n - x_1} + u_{2+K} \frac{x - x_1}{x_n - x_1}. \tag{28}$$

If we have the uniform B-spline with $u_k = k - 2$ then

$$x = x_1 \frac{K - t}{K} + x_n \frac{t}{K} \quad \text{or} \quad t = K \frac{x - x_1}{x_n - x_1}. \tag{29}$$

The B-splines can be found by the recursion, see [1],

$$N_k^0(\mathbf{u}, t) = \begin{cases} 1 & \text{if } t \in [u_{k-1}, u_k[\\ 0 & \text{otherwise} \end{cases} \quad k = 1, 2, \ldots, 4 + K \tag{30}$$

$$N_k^{r+1}(\mathbf{u}, t) = \frac{t - u_{k-1}}{u_{k+r} - u_{k-1}} N_k^r(\mathbf{u}, t) + \frac{u_{k+r+1} - t}{u_{k+r+1} - u_k} N_{k+1}^r(\mathbf{u}, t) \tag{31}$$

$$k = 1, \ldots, 3 + K - r, \quad r = 0, 1. \tag{32}$$

In order to formulate the side conditions (4) we need the first two derivatives of $y(x)$ or $y(t)$, see [1],

$$y'(t) = \sum_{k=1}^{2+K} c'_k N_k^1(\mathbf{u}', t) \quad \text{and} \quad y''(t) = \sum_{k=2}^{2+K} c''_k N_k^0(\mathbf{u}'', t) \tag{33}$$

where the control points of the derivatives are given by

$$c'_k = 2 \frac{c_k - c_{k-1}}{u_{k+1} - u_{k-1}} \quad \text{and} \quad c''_k = \frac{c'_k - c'_{k-1}}{u_k - u_{k-1}}, \tag{34}$$

and the knot sequences are

$$\mathbf{u}' = u_1, \ldots, u_{3+K} \quad \text{and} \quad \mathbf{u}'' = u_2, \ldots, u_{2+K}. \tag{35}$$

In the special case of a uniform B-spline, $u_{k+1} - u_k = 1$ for all k, we have

$$c'_k = c_k - c_{k-1} \quad \text{and} \quad c''_k = c'_k - c'_{k-1} = c_k - 2c_{k-1} + c_{k-2}. \tag{36}$$

In any case, the derivative is a piecewise linear function that interpolates the control points c'_k and the second derivative is piecewise constant equal to the control points c''_k. So $y', y'' < 0$ if and only if

$$\begin{gathered} c_k - c_{k-1} < 0 \\ (u_k - u_{k-2})c_k - (u_k - u_{k-2} + u_{k+1} - u_{k-1})c_{k-1} + (u_{k+1} - u_{k-1})c_{k-2} < 0 \end{gathered} \tag{37}$$

for all k, and in the case of a uniform B-spline

$$c_k - c_{k-1} < 0 \quad \text{and} \quad c_k - 2c_{k-1} + c_{k-2} < 0 \quad \text{for all } k. \tag{38}$$

We have tried to use a quadratic uniform B-spline with two or three segments. So for y we now have the following problem

$$\text{minimize} \sum_{i=1}^{n} \left(y_i - \sum_{k=1}^{K+2} c_k N_k^2(t_i) \right)^2, \tag{39}$$

under the side conditions

$$\begin{bmatrix} -1 & 1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -1 & 1 & 0 \\ 0 & \dots & 0 & 0 & -1 & 1 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{K+2} \end{bmatrix} < \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{40}$$

Just as in the previous case this can be solved by quadratic programming. In the case of the efficiency η we have no side conditions and we end up with a linear problem. The interpolation problem (1) gives the over determined linear system

$$\begin{bmatrix} N_1^2(t_1) & N_2^2(t_1) & \dots & N_{K+2}^2(t_1) \\ N_1^2(t_2) & N_2^2(t_2) & \dots & N_{K+2}^2(t_2) \\ \vdots & \vdots & & \vdots \\ N_1^2(t_n) & N_2^2(t_n) & \dots & N_{K+2}^2(t_n) \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{K+2} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

and as before the least square solution is the solution to the $(K+2) \times (K+2)$ system we obtain by multiplication with the transpose. The results can be found in Figure 8 where the maximal error for the 28 datasets are plotted. The resulting splines are shown in Figure 9.

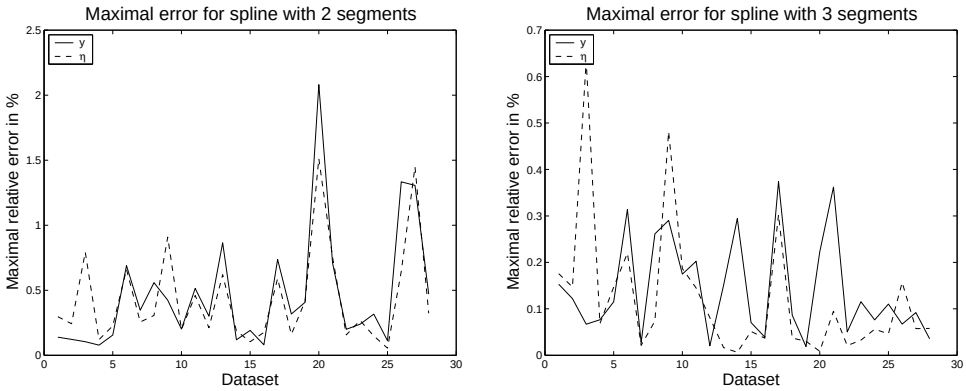

Figure 8: The maximal relative error for the 28 datasets using uniform quadratic splines. The result with two segments is shown to the left and the result with three segments to the right.

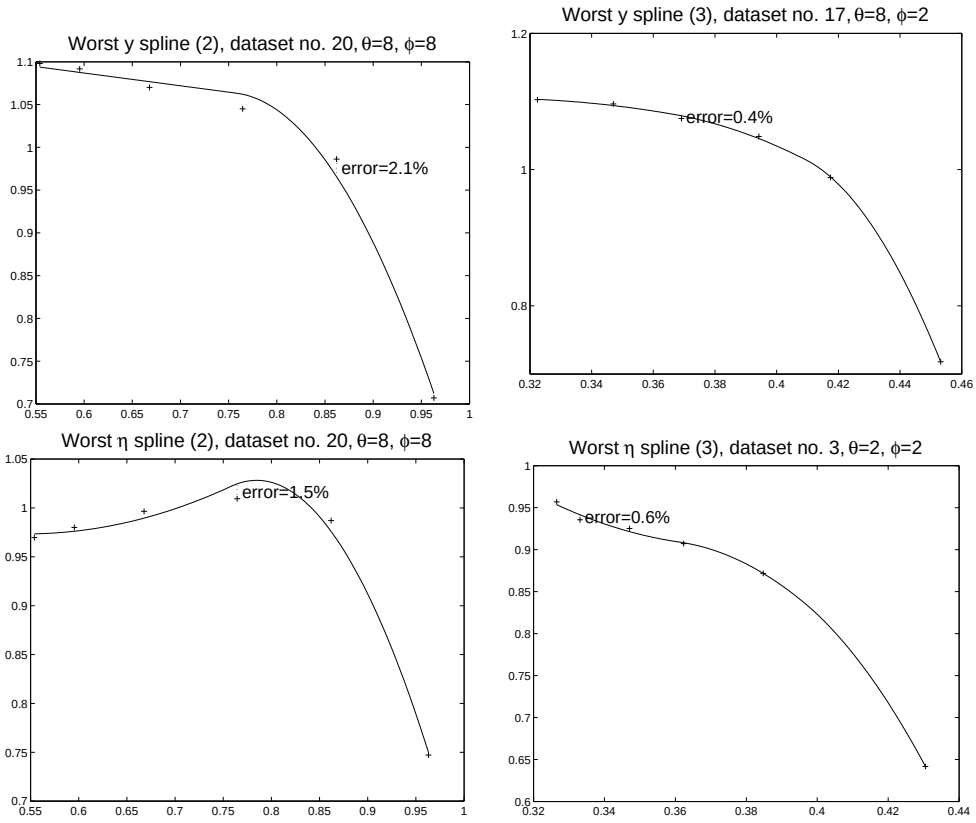

Figure 9: The worst approximating uniform quadratic spline. To the left two segments and to the right three segments. On top $y(x)$ and below $\eta(x)$.

3 The "Envelope"

For a fixed value θ of the inlet position we now have a number of diffuser positions $\phi_1, \ldots, \phi_n$ and corresponding characteristics and efficiencies

$$y_i(x), \quad \eta_i(x), \quad x \in [x_{i,\min}, x_{i,\max}], \quad i = 1, \ldots, n. \tag{41}$$

At this point HV-Turbo find what they call the *"envelope"* of these curves. To a given x value they look at the curves defined over this point and pick the largest $y_i(x)$. If we extend y_i to be zero outside its original domain, i.e.,

$$\widehat{y}_i(x) = \begin{cases} y_i(x) & \text{for } x \in [x_{i,\min}, x_{i,\max}], \\ 0 & \text{otherwise} \end{cases} \quad i = 1, \ldots, n. \tag{42}$$

then we have

$$\widehat{y}(x) = \max\{\widehat{y}_1(x), \ldots, \widehat{y}_n(x)\}. \tag{43}$$

By interpolating the discrete set of curves $\widehat{y}_i$ we obtain a one-parameter family of curves and we can now for each x take the maximum from this family. We could also determine the ordinary envelope of this family, but it would in general not be the same as the max-curve above. Don't get fooled by names!

For the interpolation we simply use linear interpolation between the data $\mathbf{c}$ defining the functions $y_k(x) = y(\mathbf{c}_k, x)$ and $y_{k+1}(x) = y(\mathbf{c}_{k+1}, x)$, i.e.,

$$x_{\min}(\phi) = \frac{\phi_{k+1} - \phi}{\phi_{k+1} - \phi_k} x_{k,\min} + \frac{\phi - \phi_k}{\phi_{k+1} - \phi_k} x_{k+1,\min}, \tag{44}$$

$$x_{\max}(\phi) = \frac{\phi_{k+1} - \phi}{\phi_{k+1} - \phi_k} x_{k,\max} + \frac{\phi - \phi_k}{\phi_{k+1} - \phi_k} x_{k+1,\max}, \tag{45}$$

$$y_\phi(x) = \begin{cases} y\left(\frac{\phi_{k+1}-\phi}{\phi_{k+1}-\phi_k}\mathbf{c}_k + \frac{\phi-\phi_k}{\phi_{k+1}-\phi_k}\mathbf{c}_{k+1}, x\right) & x \in [x_{\min}(\phi), x_{\max}(\phi)], \\ 0 & x \notin [x_{\min}(\phi), x_{\max}(\phi)], \end{cases} \tag{46}$$

$$\widehat{y}(x) = \max_\phi\{y_\phi(x)\}. \tag{47}$$

Observe that this linear interpolation preserves any of the side conditions (10), (24), or (40), we have used. Also note that by noticing which ϕ that gives the maximum we obtain a function $\phi(x)$, i.e.,

$$y_{\phi(x)}(x) = \widehat{y}(x). \tag{48}$$

The result can be seen in Figures 10–14.

We can perform the same linear interpolation for the efficiency η,

$$\eta_\phi(x) = \frac{\phi_{k+1} - \phi}{\phi_{k+1} - \phi_k} \eta_k(x) + \frac{\phi - \phi_k}{\phi_{k+1} - \phi_k} \eta_{k+1}(x), \quad x \in [x_{\min}(\phi), x_{\max}(\phi)], \tag{49}$$

and get the efficiency along the max-curve,

$$\eta(x) = \eta_{\phi(x)}(x). \tag{50}$$

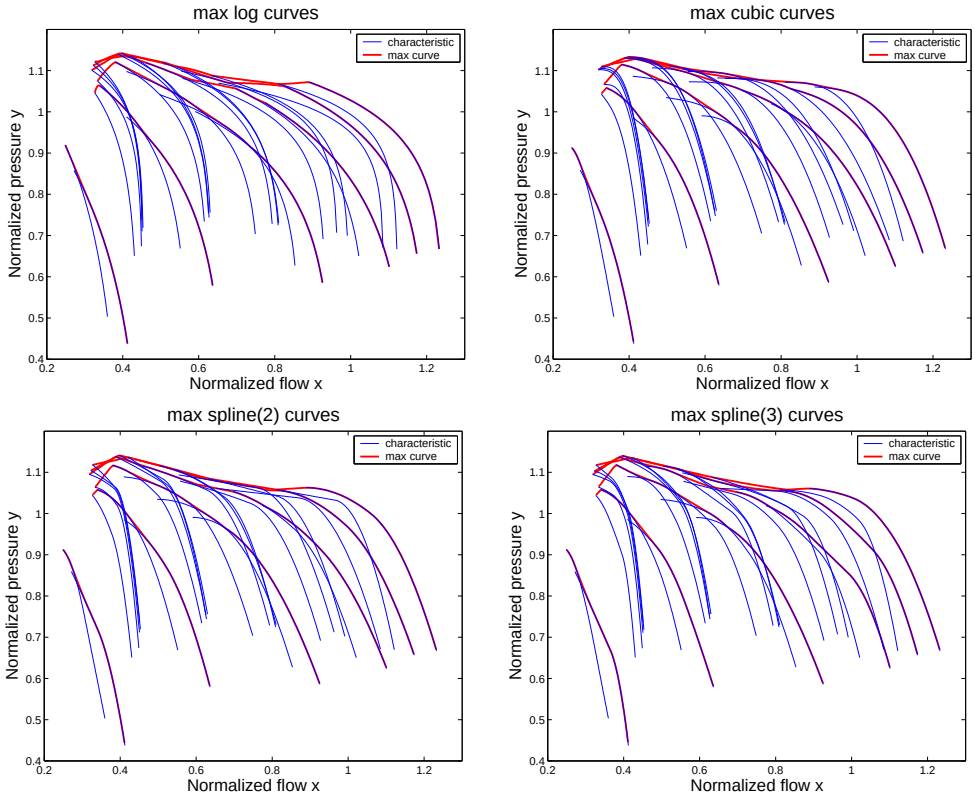

Figure 10: All the curves and their max for the different classes of functions.

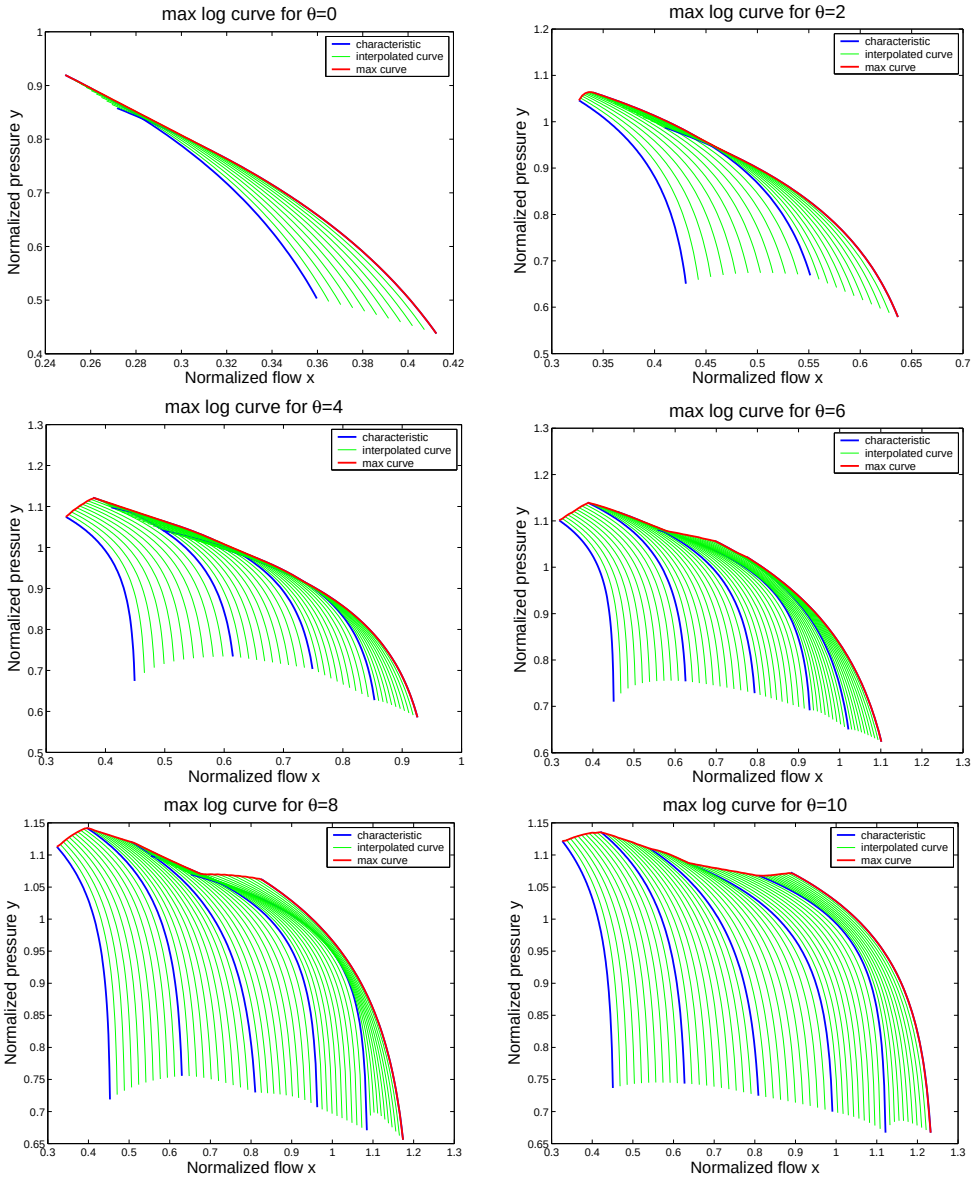

Figure 11: The maximum of the logarithmic curves

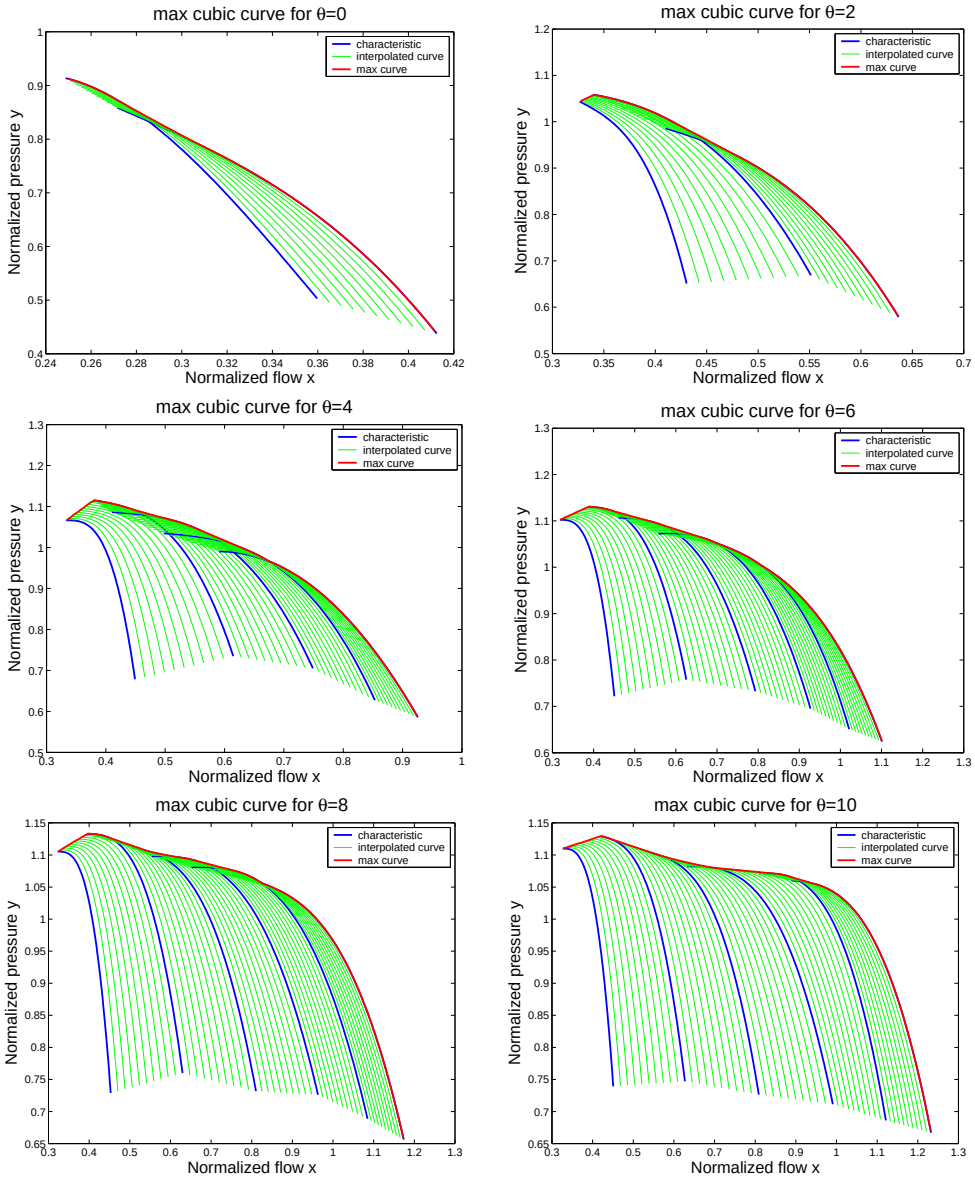

Figure 12: The maximum of the cubic curves

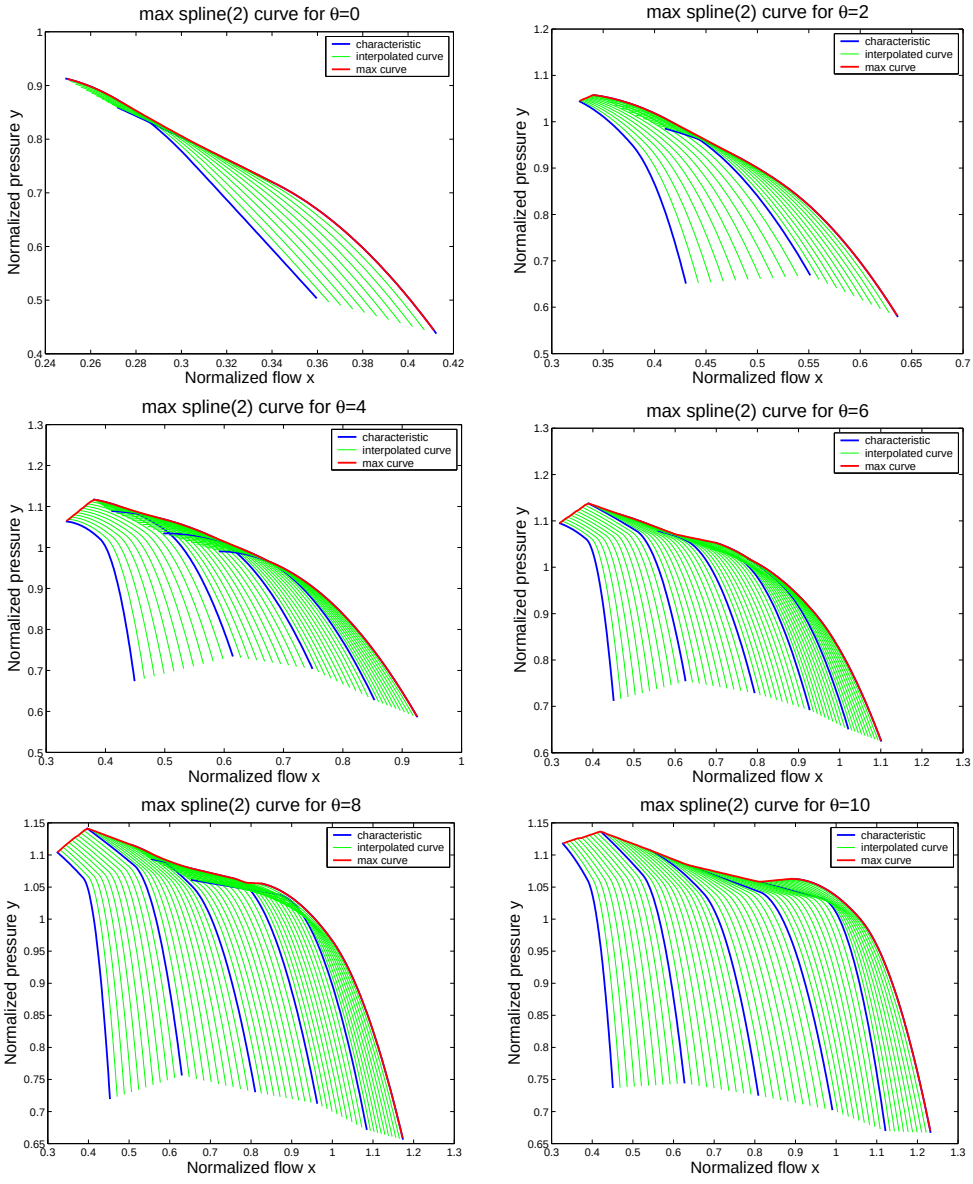

Figure 13: The maximum of the spline curves with two segments

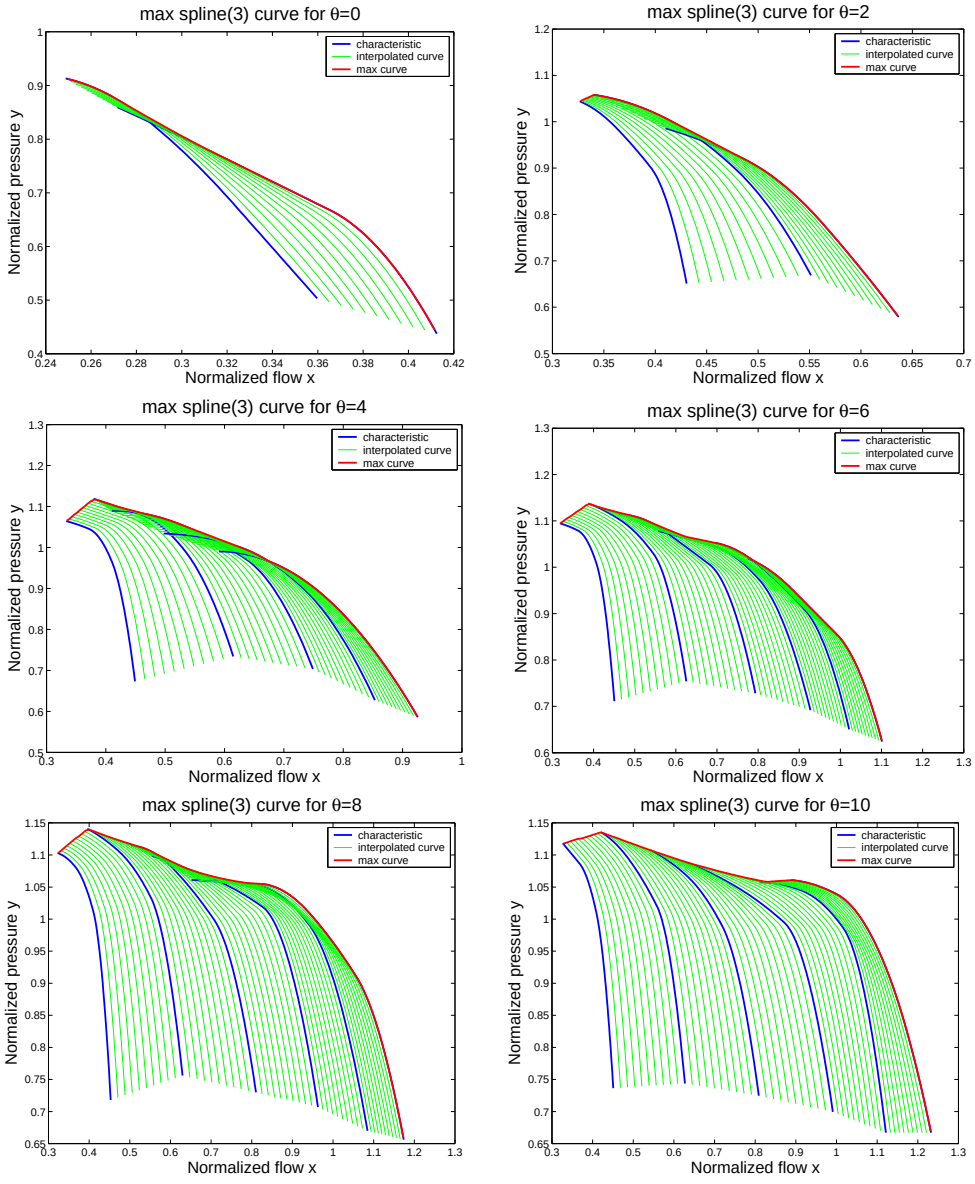

Figure 14: The maximum of the spline curves with three segments

4 The Performance Map

Presently HV-Turbo uses the max-curves from the previous section together with the functions $\phi(x)$ and $\eta(x)$ to map the maximal efficiency as functions of (x, y), the assumption being that the maximal efficiency is found along the max-curves.

It seems more natural for a given point (x_0, y_0) to search for the maximal efficiency among all positions (θ, ϕ) whose characteristic satisfies $y(x_0) = y_0$.

In the previous section we interpolated between discrete ϕ values, but now we want to interpolate between the discrete θ values too. So we are given a number of inlet and diffuser positions $(\theta_1, \phi_1), \ldots, (\theta_n, \phi_n)$ and corresponding characteristics and efficiencies calculated by one of the methods in Section 2,

$$y_i(x), \quad \eta_i(x), \quad x \in [x_{i,\min}, x_{i,\max}], \quad i = 1, \ldots, n. \tag{51}$$

If we for the function $y_i(x)$ put

$$t = K\frac{x_{i,\max} - x}{x_{i,\max} - x_{i,\min}} \quad \text{and} \quad x_i(t) = x_{i,\min}(K - t) + x_{i,\max}t \tag{52}$$

then we get curves

$$(x_i(t), y_i(t)), \quad t \in [0, K], \quad i = 1, \ldots, n. \tag{53}$$

We now want maps $f(\theta, \phi, t)$, $g(\theta, \phi, t)$, and $h(\theta, \phi, t)$, such that

$$\begin{aligned} f(\theta_i, \phi_i, t) &= x_i(t), \\ g(\theta_i, \phi_i, t) &= y_i(t), \qquad i = 1, \ldots, n. \\ h(\theta_i, \phi_i, t) &= \eta_i(t), \end{aligned} \tag{54}$$

Both $x_i(t)$, $y_i(t)$, and $\eta_i(t)$ are defined by data $(x_{i,\min}, x_{i,\max})$, $\mathbf{y}_i$, and $\boldsymbol{\eta}_i$ respectively. so if put $\mathbf{c}_i = (x_{i,\min}, x_{i,\max}, \mathbf{y}_i, \boldsymbol{\eta}_i)$, then we can write

$$(x_i(t), y_i(t), \eta_i(t)) = (x, y, \eta)(\mathbf{c}_i, t). \tag{55}$$

We solve the problem (54) by interpolating the data linearly. For the dataset provided by HV-Turbo the points (θ_i, ϕ_i) were as in Figure 15, were we also have drawn a triangulation of these points.

For each triangle $T_\ell = [(\theta_{i_\ell}, \phi_{i_\ell}), (\theta_{j_\ell}, \phi_{j_\ell}), (\theta_{k_\ell}, \phi_{k_\ell})]$, $\ell = 1, \ldots, L$, we have data $\mathbf{c}_{i_\ell}$, $\mathbf{c}_{j_\ell}$, and $\mathbf{c}_{k_\ell}$ defined in each corner. We now perform linear interpolation in the triangle

$$\text{if} \qquad (\theta, \phi) = u(\theta_{i_\ell}, \phi_{i_\ell}) + v(\theta_{j_\ell}, \phi_{j_\ell}) + w(\theta_{k_\ell}, \phi_{k_\ell}), \tag{56}$$

$$\text{where} \qquad 1 = u + v + w, \tag{57}$$

$$\text{then} \qquad \mathbf{c}(\theta, \phi) = u\mathbf{c}_{i_\ell} + v\mathbf{c}_{j_\ell} + w\mathbf{c}_{k_\ell}. \tag{58}$$

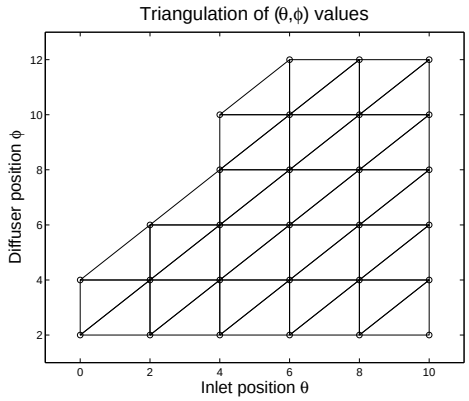

Figure 15: Triangulation of the given values of inlet and diffuser position.

The numbers (u, v, w) are called the *barycentric coordinates* of (θ, ϕ) with respect to the triangle T_ℓ. At this point we have three functions

$$(\widetilde{x}, \widetilde{y}, \widetilde{\eta})(\theta, \phi, t) = (x, y, \eta)(\mathbf{c}(\theta, \phi), t) \tag{59}$$

and we define

$$\widehat{\eta}(x, y) = \max\{\widetilde{\eta}(\theta, \phi, t) \mid \widetilde{x}(\theta, \phi, t) = x \wedge \widetilde{y}(\theta, \phi, t) = y\} \tag{60}$$

$$= \widetilde{\eta}\left(\widehat{\theta}(x, y), \widehat{\phi}(x, y), t(x, y)\right), \tag{61}$$

We now take a number of parameter values $t_1, \ldots, t_M \in [0, 1]$ and a number of barycentric coordinates (u_i, v_i, w_i), $i = 1, \ldots, \widehat{N}$. We choose the following set of barycentric coordinates

$$\left\{ \left(\frac{\alpha}{N}, \frac{\beta}{N}, \frac{\gamma}{N} \right) \;\middle|\; \alpha, \beta, \gamma \in \mathbb{N}_0 \wedge \alpha + \beta + \gamma = N \right\}, \tag{62}$$

and then $\widehat{N} = (N + 1)(N + 2)/2$. For each triangle we get data $\mathbf{c}_{\ell,i} = u_i \mathbf{c}_{i_\ell} + v_i \mathbf{c}_{j_\ell} + w_i \mathbf{c}_{k_\ell}$ and this gives us values

$$(x_{\ell,i,j}, y_{\ell,i,j}, \eta_{\ell,i,j}) = (x, y, \eta)(\mathbf{c}_{\ell,i}, t_j). \tag{63}$$

The next step is to choose a grid in the (x, y)-plane. If the grid-size is $R \times S$ then we put

$$\widehat{x}_0 = \min_{\ell,i,j}\{x_{\ell,i,j}\}, \qquad \widehat{x}_1 = \max_{\ell,i,j}\{x_{\ell,i,j}\}, \tag{64}$$

$$\widehat{y}_0 = \min_{\ell,i,j}\{y_{\ell,i,j}\}, \qquad \widehat{y}_1 = \max_{\ell,i,j}\{y_{\ell,i,j}\}, \tag{65}$$

and

$$x_r = \frac{(2R-2r+1)\widehat{x}_0 + (2r-1)\widehat{x}_1}{2R}, \quad r = 1, \ldots, R, \tag{66}$$

$$y_s = \frac{(2S-2s+1)\widehat{y}_0 + (2s+1)\widehat{y}_1}{2S}, \quad s = 1, \ldots, S, \tag{67}$$

$$\widehat{\eta}_{r,s} = \max\left\{ \eta_{\ell,i,j} \;\middle|\; |x_{\ell,i,j} - x_r| \le \frac{1}{2R} \wedge |y_{\ell,i,j} - y_s| \le \frac{1}{2S} \right\} \tag{68}$$

$$= \eta_{\ell_{r,s},i_{r,s},j_{r,s}}, \tag{69}$$

$$\widehat{\theta}_{r,s} = u_{i_{r,s}}\theta_{i_{\ell_{r,s}}} + v_{i_{r,s}}\theta_{j_{\ell_{r,s}}} + w_{i_{r,s}}\theta_{k_{\ell_{r,s}}}, \tag{70}$$

$$\widehat{\phi}_{r,s} = u_{i_{r,s}}\phi_{i_{\ell_{r,s}}} + v_{i_{r,s}}\phi_{j_{\ell_{r,s}}} + w_{i_{r,s}}\phi_{k_{\ell_{r,s}}}. \tag{71}$$

In practice we initialize $\widehat{\eta}_{r,s} = \widehat{\theta}_{r,s} = \widehat{\phi}_{r,s} = 0$, and run through all triples ℓ, i, j. For each triple we determine the r and s that minimizes $|x_{\ell,i,j} - x_r|$ and $|y_{\ell,i,j} - y_s|$ respectively, if $\eta_{\ell,i,j} > \widehat{\eta}_{r,s}$ then we put $\widehat{\eta}_{r,s} = \eta_{\ell,i,j}$, $\widehat{\theta}_{r,s} = \theta_{\ell,i,j}$, and $\widehat{\phi}_{r,s} = \phi_{\ell,i,j}$. We could in principle have that $x_{\ell,i,j} = x_r + \frac{1}{2R} = x_{r+1} - \frac{1}{2R}$ so according to (68) we should choose both r and $r+1$, and similar for y. It is an unlikely event and it is not important so in the program (Appendix A.1) we let Matlab choose just one of the possibilities.

We have of course that $\widehat{\eta}(x_r, y_s) \approx \widehat{\eta}_{r,s}$ and similar for $\widehat{\theta}$ and $\widehat{\phi}$, so we can now plot the iso-lines in the (x, y)-plane for these three functions. The results when $N = 64$, $M = 128$, and $R = S = 64$ are shown in Figures 16–19. The contour plots looks somewhat ragged, but it would of course be possible to smooth the data before we plot it or find the contours. It can perhaps help to replace the piecewise linear interpolation with a smoother interpolation or approximation, but then we have to be careful and make sure that the side conditions (10), (24), or (40) are satisfied. It might be caused by undersampling in some regions of the xy-plane. We sample evenly along the characteristics and if we look at Figures 10–14 it is clear that we have more samples for high values of y. This is of course not hard to change, we only need to take more samples at the right (lower) end of the characteristics.

5 Conclusion

At the moment HV-Turbo uses a manual method to plot the characteristics and performance maps for a compressor. The company asked if it is possible to replace the manual method with an automated procedure. Furthermore, HV-Turbo asked if it to a given combination of flow and pressure is possible to find the optimal inlet guide vane position and diffuser position. We have demonstrated that this indeed is the case.

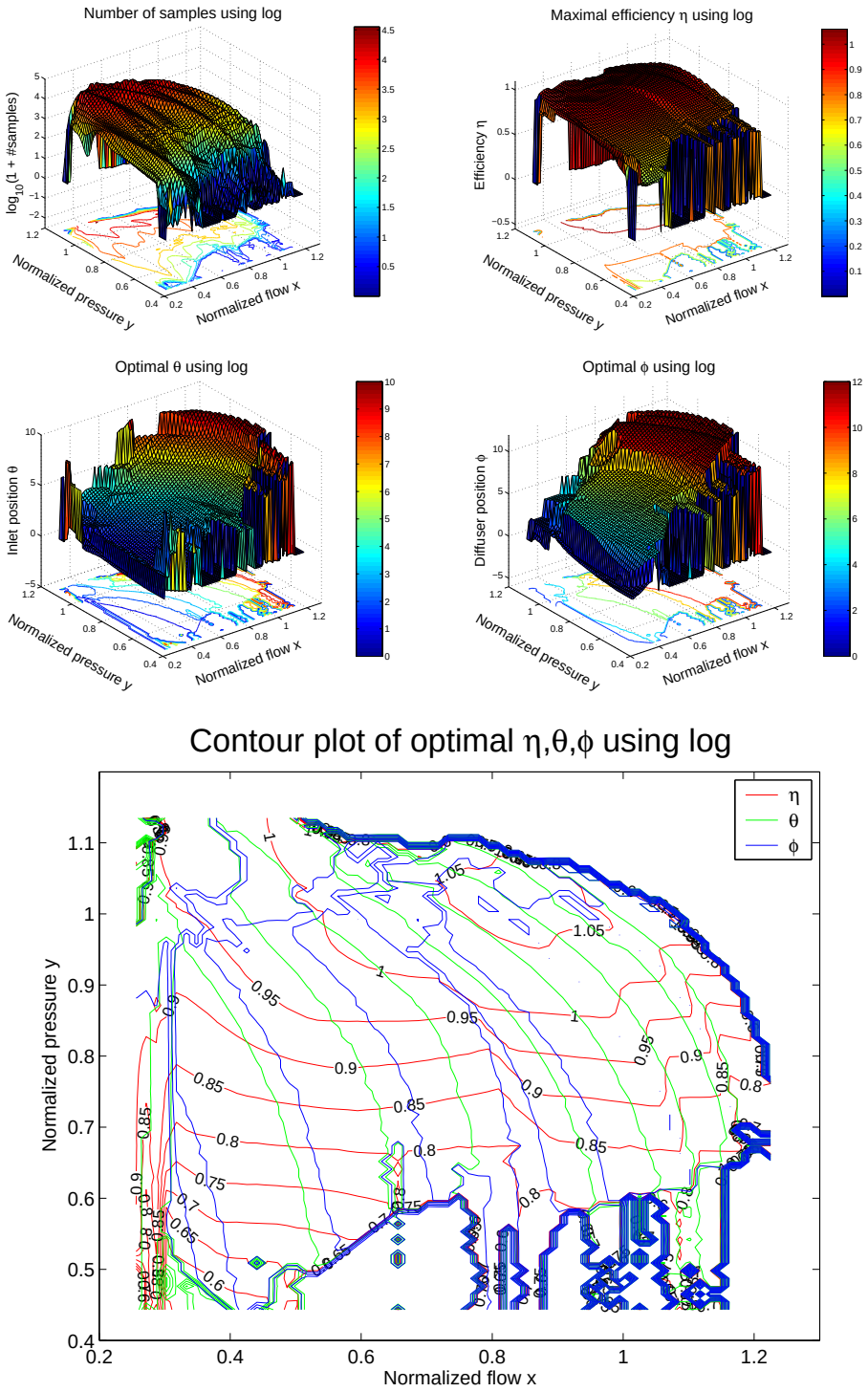

Figure 16: The number of samples (log-plot) and the optimal η, θ, and ϕ using log.

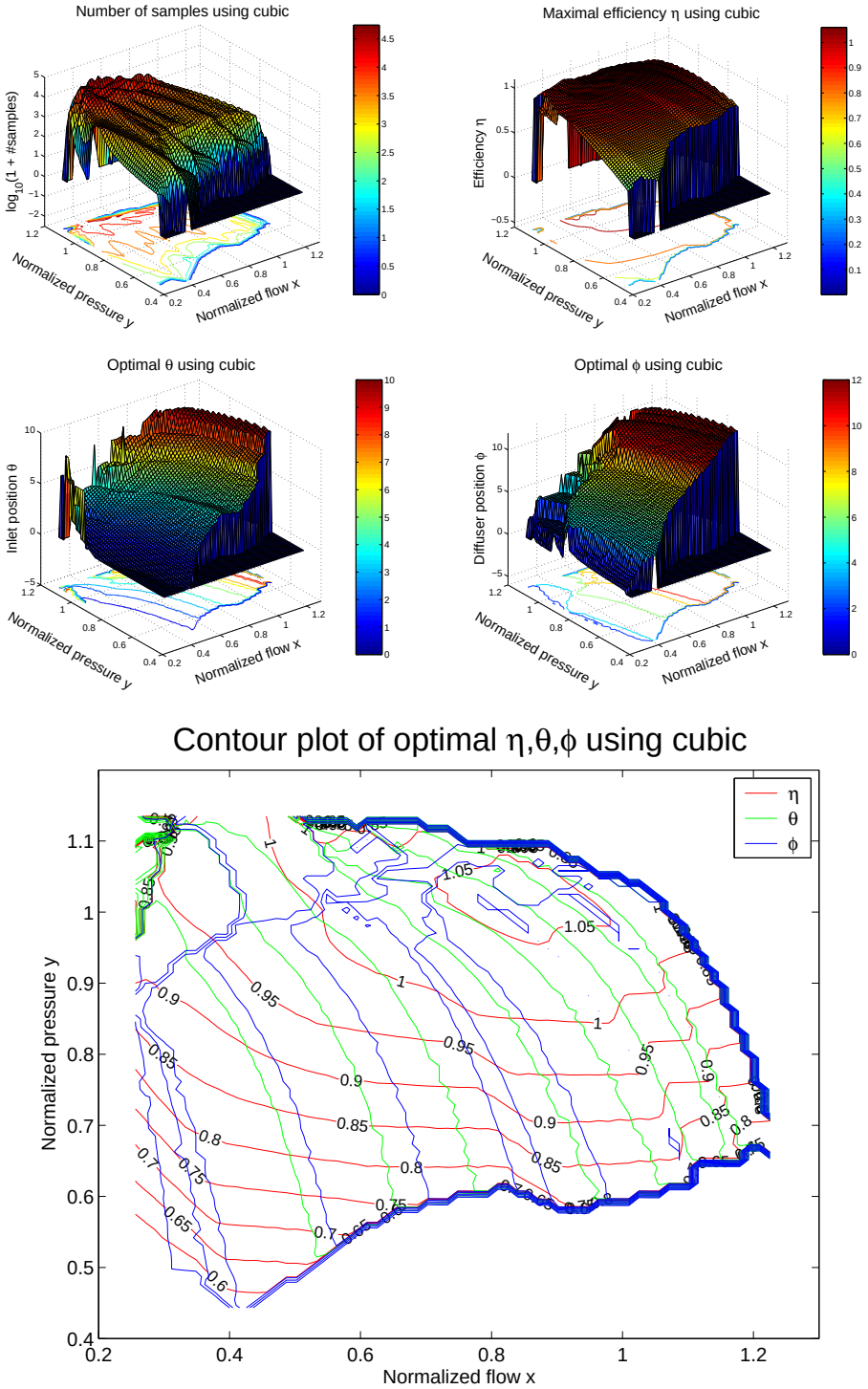

Figure 17: The number of samples (log-plot) and the optimal η, θ, and ϕ using cubic.

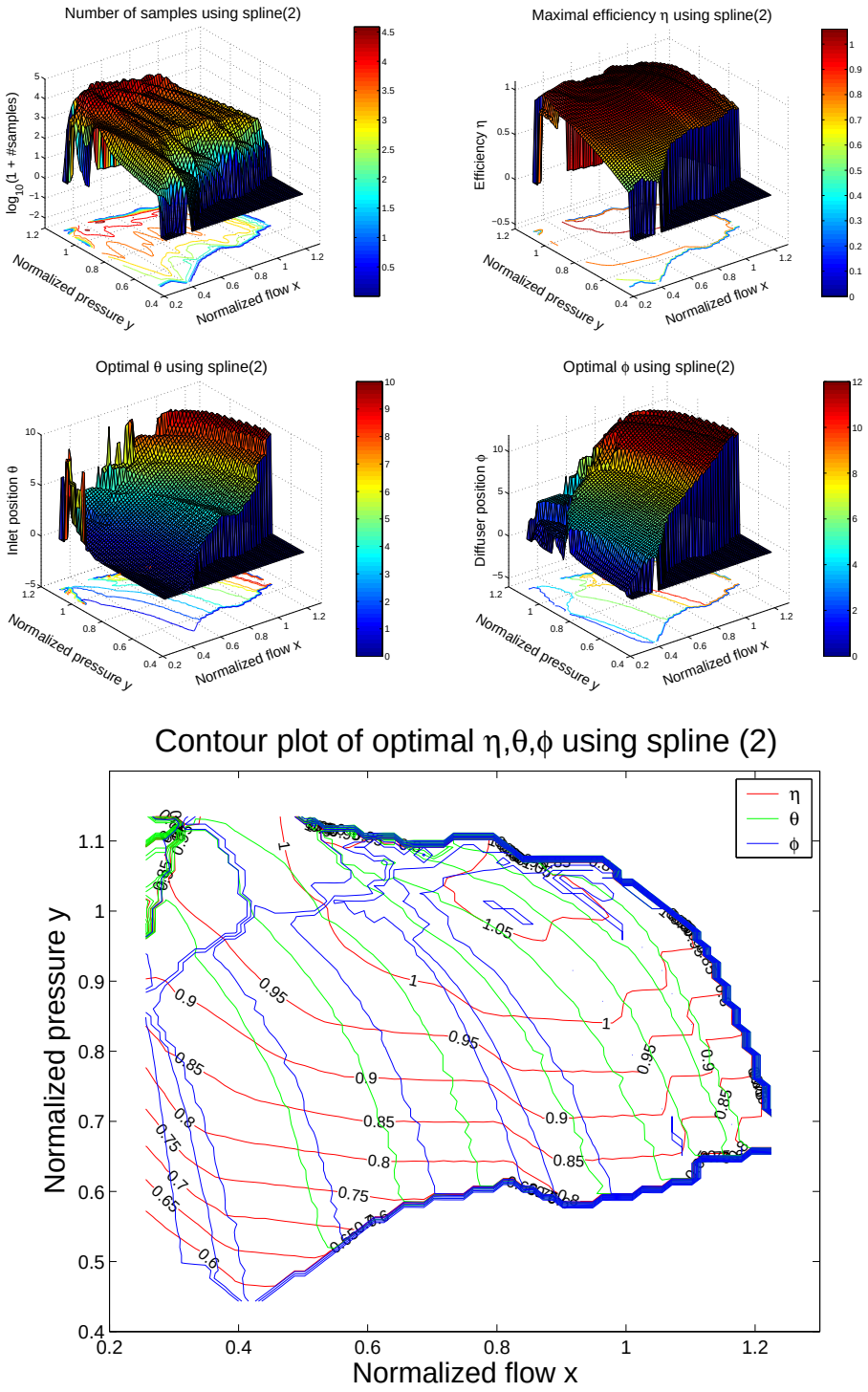

Figure 18: The number of samples (log-plot) and the optimal η, θ, and ϕ using B-splines with two segments.

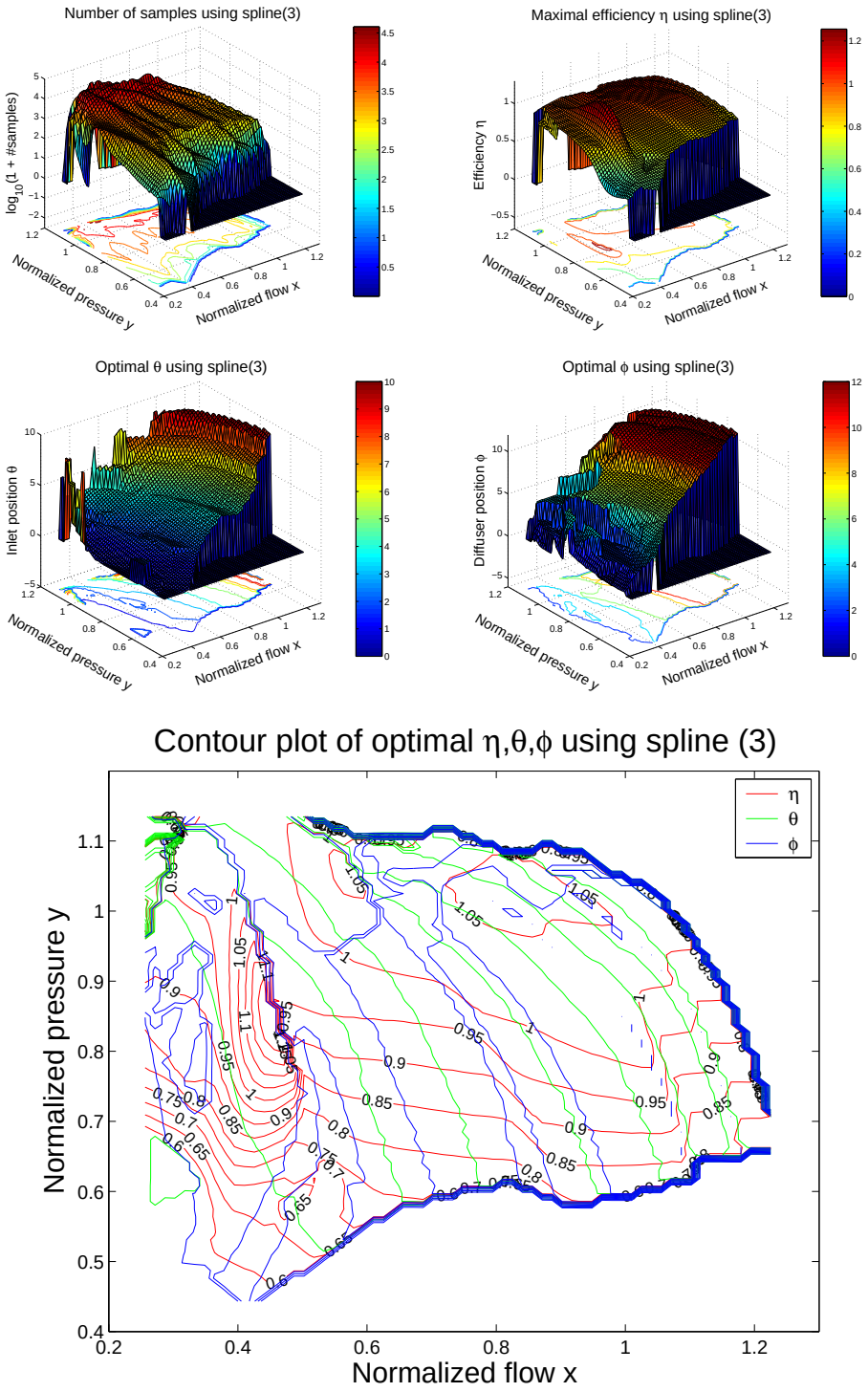

Figure 19: The number of samples (log-plot) and the optimal η, θ, and ϕ using B-splines with three segments.

We have given a number of suggestions for plotting the compressor characteristics, but the choice between these suggestions will depend on further testing and evaluation by experts within the company.

One step in the manual procedure for producing the performance map was the construction of "envelopes", called max-curves in this report, and we have given a method for constructing these curves.

Instead of mimicking the present manual procedure, using the max-curves, for producing the performance map, we have chosen to go directly for the optimal efficiency. We have given a method for plotting the maximal efficiency as a function of flow and pressure together with the corresponding inlet guide vane position and diffuser position. This could of course also be the basis for a method for *controlling* the compressor.

We have used a simple linear interpolation in the $\theta\phi$-plane and it might be necessary to used some kind of spline approximation instead, remembering the side conditions. In this case the $\log$-method for the characteristic would give a non-linear (and non-quadratic) optimization problem.

In any case, more extensive testing is needed in order to establish the right level and method for sampling. Finally, everything should be validate against physical experiments.

References

[1] Jens Gravesen, *Differential Geometry and Design of Shape and Motion*, DTU 2002, http://www.mat.dtu.dk/people/J.Gravesen/cagd.pdf

[2] *Optimization Toolbox User's Guide*, Version 2, The Mathworks, 2002

[3] Nick Gould and Philippe Toint, *A Quadratic Programming Page*, http://www.numerical.rl.ac.uk/qp/qp.html

A Matlab files

The Matlab file below produces all the plots in the report.

A.1 hvturbo.m

```
%
% Load the data set
%

load esgi51;

NoP=28; % Total Number of inlet/diffuser Positions
NoM=6;  % Number of Measurements for each position

theta=zeros(NoP,1);
phi=zeros(NoP,1);
x=zeros(NoP,NoM);
y=zeros(NoP,NoM);
eta=zeros(NoP,NoM);
xmin=zeros(NoP,1);
xmax=zeros(NoP,1);

xp=zeros(NoP,100);
yp=zeros(NoP,100);
etap=zeros(NoP,100);

for no=0:NoP-1
  theta(no+1)=esgi51(5*no+1,1);
  phi(no+1)=esgi51(5*no+2,1);
  x(no+1,:)=esgi51(5*no+3,1:NoM);
  y(no+1,:)=esgi51(5*no+4,1:NoM);
  eta(no+1,:)=esgi51(5*no+5,1:NoM);
  xmin(no+1)=x(no+1,1);
  xmax(no+1)=x(no+1,end);

  xp(no+1,:)=linspace(xmin(no+1),xmax(no+1));

end

old=1;
j=1;
clear thp;
```

```
for no=1:NoP
  if theta(no)~=theta(old)
    thp(1,j)={theta(old)};
    thp(2,j)={[old:no-1]};
    thp(3,j)={min(xmin(old:no-1))};
    thp(4,j)={max(xmax(old:no-1))};
    old=no;
    j=j+1;
  end
end
thp(1,j)={theta(no)};
thp(2,j)={[old:no]};
thp(3,j)={min(xmin(old:no))};
thp(4,j)={max(xmax(old:no))};

%
% Setup for the performance map
%
etacntr=linspace(0.6,1.2,13);
N=64;    % #triangle=(N+1)(N+2)/2 (=1326 for N=50)
M=128;   % #t-values
PR=64;  % x-grid
PS=64;  % y-grid

Puvw=zeros((N+1)*(N+2)/2,3);
k=0;
for i=0:N
  for j=0:i
    k=k+1;
    Puvw(k,:)=[i/N,j/N,(N-i-j)/N];
  end
end
Pt=linspace(0,1,M+1);
Px=((2*PS+1)*min(xmin)-max(xmax)+ ...
    2*[1:PS]*(max(xmax)-min(xmin)))/(2*PS); % x is column number
Py=((2*PR+1)*min(min(y))-max(max(y))+ ...
    2*[1:PR]*(max(max(y))-min(min(y))))/(2*PR);
Peta=zeros(PR,PS);
Ptheta=zeros(PR,PS);
Pphi=zeros(PR,PS);
Pn=zeros(PR,PS);

%
```

```
% collect triangles in theta-phi plane
%
tn=0;
for i=1:length(thp)-1
  tmp1=thp{2,i};
  tmp2=thp{2,i+1};
  for j=1:length(tmp1)-1
    tn=tn+1;
    Ptri(tn,:)=[tmp1(j),tmp2(j),tmp2(j+1)];
    tn=tn+1;
    Ptri(tn,:)=[tmp1(j),tmp2(j+1),tmp1(j+1)];
  end
  j=j+1;
  if length(tmp1)<length(tmp2)
    tn=tn+1;
    Ptri(tn,:)=[tmp1(j),tmp2(j),tmp2(j+1)];
  end;
end;

fn=0;
fn=fn+1;
figure(fn);
hold off;
plot(theta,phi,'ko');
axis([min(theta)-1,max(theta)+1,min(phi)-1,max(phi)+1]);
hold on
for i=1:length(Ptri)
  plot([theta(Ptri(i,:));theta(Ptri(i,1))],...
       [phi(Ptri(i,:));phi(Ptri(i,1))],'k-');
end
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
xlabel('Inlet position \theta','fontsize',18);
ylabel('Diffuser position \phi','fontsize',18);
title('Triangulation of (\theta,\phi) values','Fontsize',20);
print -depsc  -r600 theta-phi

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Parabolas y = a*x^2 + b*x +c
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
ycoeff=zeros(NoP,3);
yres=zeros(NoP,NoM);
etacoeff=zeros(NoP,3);
etares=zeros(NoP,NoM);

for no=1:NoP

  X=[x(no,:).^2; x(no,:); ones(1,NoM)];

  ycoeff(no,:)=y(no,:)/X;
  yres(no,:)=y(no,:)-ycoeff(no,:)*X;

  etacoeff(no,:)=eta(no,:)/X;
  etares(no,:)=eta(no,:)-etacoeff(no,:)*X;

  X=[xp(no,:).^2; xp(no,:); ones(size(xp(no,:)))];
  yp(no,:)=ycoeff(no,:)*X;
  etap(no,:)=etacoeff(no,:)*X;

end

%
% Worst y of parabola:
%
err=abs(yres)./y;
[m,k]=max(err);
[m,l]=max(max(err));
k=k(l);
[k,l,theta(k),phi(k),err(k,l)]

yerr=max(err');

fn=fn+1;
figure(fn);
hold off;
plot(x(k,:),y(k,:),'k+');
hold on;
plot([x(k,l),x(k,l)],[y(k,l),y(k,l)-yres(k,l)],'k:');
plot(xp(k,:),yp(k,:),'k-');
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
s=sprintf('Worst y parabola, dataset no. %d, \\theta=%d, \\phi=%d', ...
          k, theta(k), phi(k));
```

```
xlabel('Normalized flow x','fontsize',18);
ylabel('Normalized pressure y','fontsize',18);
title(s,'Fontsize',20);
text(x(k,l),y(k,l)-.5*yres(k,l), ...
     sprintf(' error=%3.1f%%',100*err(k,l)), 'Fontsize',18);
print -depsc  -r600 y-parabola

%
% Worst eta of parabola:
%
err=abs(etares)./eta;
[m,k]=max(err);
[m,l]=max(max(err));
k=k(l);
[k,l,theta(k),phi(k),err(k,l)]

etaerr=max(err');

fn=fn+1;
figure(fn);
hold off;
plot(x(k,:),eta(k,:),'k+');
hold on;
plot([x(k,l),x(k,l)],[eta(k,l),eta(k,l)-etares(k,l)],'k:');
plot(xp(k,:),etap(k,:),'k-');
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
s=sprintf('Worst \\eta parabola, dataset no. %d, \\theta=%d, \\phi=%d', ...
          k, theta(k), phi(k));
xlabel('Normalized flow x','fontsize',18);
ylabel('Efficiency \eta','fontsize',18);
title(s,'Fontsize',20);
text(x(k,l),eta(k,l)-.5*etares(k,l), ...
     sprintf(' error=%3.1f%%',100*err(k,l)),'Fontsize',18);
print -depsc -r600 eta-parabola



fn=fn+1;
figure(fn);
hold off;
tmp=1:NoP;
plot(tmp,100*yerr,'k-',tmp,100*etaerr,'k--');
```

```
legend('y','\eta',2);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
title('Maximal error for parabolas','Fontsize',20);
xlabel('Dataset','fontsize',18);
ylabel('Maximal relative error in %','fontsize',18);
print -depsc -r600 parabola-err;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Logarithms y = a*log(b-x) + c ;  b>max(x)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ycoeff=zeros(NoP,3);
yres=zeros(NoP,NoM);

safety=0.0001;
options = optimset('MaxFunEvals',800);

for no=1:NoP
  [abc, R,res,exitflag,output,lambda,jacobian] = ...
      lsqcurvefit('abclog',[2 2 -1], x(no,:), y(no,:), ...
                  [-inf,xmax(no)+safety],[],options);
  ycoeff(no,:)=abc;
  yres(no,:)=y(no,:)-(abc(1)*log(abc(2)-x(no,:))+abc(3));
  yp(no,:)=abc(1)*log(abc(2)-xp(no,:))+abc(3);
end

%
% Worst y of log (Optim):
%
err=abs(yres)./y;
[m,k]=max(err);
[m,l]=max(max(err));
k=k(l);
[k,l,theta(k),phi(k),err(k,l)]

yerr=max(err');

fn=fn+1;
figure(fn);
```

```
hold off;
plot(x(k,:),y(k,:),'k+');
hold on;
plot([x(k,l),x(k,l)],[y(k,l),y(k,l)-yres(k,l)],'k:');
plot(xp(k,:),yp(k,:),'k-');
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
s=sprintf('Worst logarithm (Optim), dataset no. %d, \\theta=%d, \\phi=%d', ...
          k, theta(k), phi(k));
xlabel('Normalized flow x','fontsize',18);
ylabel('Normalized pressure y','fontsize',18);
title(s,'Fontsize',20);
text(x(k,l),y(k,l)-.5*yres(k,l),sprintf('error=%3.1f%% ',100*err(k,l)), ...
     'Fontsize',18,'HorizontalAlignment','Right');
print -depsc  -r600 y-log-matlab;

fn=fn+1;
figure(fn);
hold off;
plot(100*yerr,'k-');
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
title('Maximal error for logarithms (Optim)','Fontsize',20);
xlabel('Dataset','fontsize',18);
ylabel('Maximal relative error in %','fontsize',18);
print -depsc -r600 log-err-matlab

N=1000;
for no=1:NoP
  b=linspace(xmax(no)+safety,xmax(no)+1,N);
  a=zeros(N);
  c=zeros(N);
  res=zeros(N,NoM);
  for i=1:N
    X=[log(b(i)-x(no,:)); ones(1,NoM)];
    ac=y(no,:)/X;
    a(i)=ac(1,1);
    c(i)=ac(1,2);
    res(i,:)=y(no,:)-(a(i)*log(b(i)-x(no,:))+c(i));
  end
  [m,k]=min(sum(res'.^2));
  ycoeff(no,:)=[a(k), b(k), c(k)];
  yres(no,:)=res(k,:);
```

```
  yp(no,:)=ycoeff(no,1)*log(ycoeff(no,2)-xp(no,:))+ycoeff(no,3);
end

%
% Worst y of log
%
err=abs(yres)./y;
[m,k]=max(err);
[m,l]=max(max(err));
k=k(l);
[k,l,theta(k),phi(k),err(k,l)]

yerr=max(err');

fn=fn+1;
figure(fn);
hold off;
plot(x(k,:),y(k,:),'k+');
hold on;
plot([x(k,l),x(k,l)],[y(k,l),y(k,l)-yres(k,l)],'k:');
plot(xp(k,:),yp(k,:),'k-');
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
s=sprintf('Worst logarithm, dataset no. %d, \\theta=%d, \\phi=%d', ...
          k, theta(k), phi(k));
xlabel('Normalized flow x','fontsize',18);
ylabel('Normalized pressure y','fontsize',18);
title(s,'Fontsize',20);
text(x(k,l),y(k,l)-.5*yres(k,l),sprintf('error=%3.1f%% ',100*err(k,l)),
     'Fontsize',18,'HorizontalAlignment','Right');
print -depsc  -r600 y-log;

fn=fn+1;
figure(fn);
hold off;
plot(100*yerr,'k-');
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
title('Maximal error for logarithms','Fontsize',20);
xlabel('Dataset','fontsize',18);
ylabel('Maximal relative error in %','fontsize',18);
print -depsc -r600 log-err
```

```
%
% "Envelope" of log
%

N=50;  % the number of interpolated curves we use to find the max
kk=10; % the number of interpolated curves we plot

R=0;
for i=1:length(thp)
  tmp=thp{2,i};
  d=thp{3,i};       % the minimal x
  e=thp{4,i};       % the maximal x
  xx=linspace(d,e); % the x values we want
  yy=zeros((N+1)*length(tmp),length(xx)); % all y-values initialized to zero

  fn=fn+1;
  figure(fn);
  hold off;
  k1=tmp(1);

  OldHandle=plot(xp(k1,:),yp(k1,:),'b-','LineWidth',2); % The first given curve
  hold on;

  a1=ycoeff(k1,1);
  b1=ycoeff(k1,2);
  c1=ycoeff(k1,3);
  d1=xmin(k1);
  e1=xmax(k1);

  for j=2:length(tmp)
    k0=k1;
    k1=tmp(j);
    plot(xp(k1,:),yp(k1,:),'b-','LineWidth',2); % A given curve
    a0=a1;
    b0=b1;
    c0=c1;
    d0=d1;
    e0=e1;

    a1=ycoeff(k1,1);
    b1=ycoeff(k1,2);
    c1=ycoeff(k1,3);
    d1=xmin(k1);
```

```
    e1=xmax(k1);

    for k=1:kk-1
      a=((kk-k)*a0+k*a1)/kk;
      b=((kk-k)*b0+k*b1)/kk;
      c=((kk-k)*c0+k*c1)/kk;
      d=((kk-k)*d0+k*d1)/kk;      % xmin
      e=((kk-k)*e0+k*e1)/kk;      % xmax
      xt=linspace(d,e);
      yt=a*log(b-xt)+c;
      NewHandle=plot(xt,yt,'g-'); % An interpolated curve
    end

    for k=0:N
      a=((N-k)*a0+k*a1)/N;
      b=((N-k)*b0+k*b1)/N;
      c=((N-k)*c0+k*c1)/N;
      d=((N-k)*d0+k*d1)/N;     % xmin
      e=((N-k)*e0+k*e1)/N;     % xmax
      ide=find(d<=xx & xx<=e);
      dxe=xx(ide);             % the x values where the curve is defined
      R=R+1;
      yy(R,ide)=a*log(b-dxe)+c;% the y values
    end
  end
  YMax(1,i)={xx};
  YMax(2,i)={max(yy)};
  MaxHandle=plot(xx,max(yy),'r-','LineWidth',2);
  legend([OldHandle;NewHandle;MaxHandle], ...
         'characteristic','interpolated curve','max curve');
  xlabel('Normalized flow x','fontsize',18);
  ylabel('Normalized pressure y','fontsize',18);
  s=sprintf('max log curve for \\theta=%d',thp{1,i});
  title(s,'Fontsize',20)
  print('-depsc', '-r600', sprintf('logmax-%02d', thp{1,i}));
end

fn=fn+1;
figure(fn);
hold off;
MaxHandle=plot(YMax{1,1}, YMax{2,1}, 'r-','LineWidth',2);
hold on
for i=2:length(thp)
```

```
  plot(YMax{1,i}, YMax{2,i}, 'r-','LineWidth',2);
end
for k=1:NoP
  OldHandle=plot(xp(k,:),yp(k,:),'b-');
end
axis([0.2 1.3 0.4 1.2]);
legend([OldHandle;MaxHandle],'characteristic','max curve');
xlabel('Normalized flow x','fontsize',18);
ylabel('Normalized pressure y','fontsize',18);
s=sprintf('max log curves');
title(s,'Fontsize',20)
print('-depsc', '-r600', sprintf('log-max'));

%
% We save the result for the performance map
%
logycoeff=ycoeff;
logetacoeff=etacoeff;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Cubic y = sum c_k B^3_k(t)
%       x = x_1(1-t)+x_n t
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ycoeff=zeros(NoP,4);
yres=zeros(NoP,NoM);
etacoeff=zeros(NoP,4);
etares=zeros(NoP,NoM);

A= [-1  1  0 0
    -1  0  1 0
     0 -1  0 1
     0  0 -1 1
     1 -2  1 0
     0  1 -2 1];
b=zeros(6,1);

for no=1:NoP

  t=(x(no,:)-xmin(no))/(xmax(no)-xmin(no));
```

```
  B=bernstein(3,t);

  etacoeff(no,:)=eta(no,:)/B;
  etares(no,:) =eta(no,:)-etacoeff(no,:)*B;

  H=B*B';
  f=-B*y(no,:)';

  options=optimset('LargeScale','off');
  [coeff,fval]= quadprog(H,f,A,b,[],[],[],[],[],options);
  ycoeff(no,:) = coeff';
  yres(no,:) = y(no,:)-coeff'*B;

  tt=(xp(no,:)-xmin(no))/(xmax(no)-xmin(no));
  yp(no,:)=coeff'*bernstein(3,tt);
  etap(no,:)=etacoeff(no,:)*bernstein(3,tt);
end

%
% Worst y:
%
err=abs(yres)./y;
[m,k]=max(err);
[m,l]=max(max(err));
k=k(l);
[k,l,theta(k),phi(k),err(k,l)]

yerr=max(err');

fn=fn+1;
figure(fn);
hold off;
plot(x(k,:),y(k,:),'k+');
hold on;
plot([x(k,l),x(k,l)],[y(k,l),y(k,l)-yres(k,l)],'k:');
plot(xp(k,:),yp(k,:),'k-');
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
s=sprintf('Worst y cubic, dataset no. %d, \\theta=%d, \\phi=%d', ...
          k, theta(k), phi(k));
xlabel('Normalized flow x','fontsize',18);
ylabel('Normalized pressure y','fontsize',18);
title(s,'Fontsize',20);
```

```
text(x(k,l),y(k,l)-.5*yres(k,l), ...
     sprintf(' error=%3.1f%%',100*err(k,l)),'Fontsize',18);
print -depsc  -r600 y-cubic

%
% Worst eta:
%
err=abs(etares)./eta;
[m,k]=max(err);
[m,l]=max(max(err));
k=k(l);
[k,l,theta(k),phi(k),err(k,l)]

etaerr=max(err');

fn=fn+1;
figure(fn);
hold off;
plot(x(k,:),eta(k,:),'k+');
hold on;
plot([x(k,l),x(k,l)],[eta(k,l),eta(k,l)-etares(k,l)],'k:');
plot(xp(k,:),etap(k,:),'k-');
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
s=sprintf('Worst \\eta cubic, dataset no. %d, \\theta=%d, \\phi=%d', ...
          k, theta(k), phi(k));
xlabel('Normalized flow x','fontsize',18);
ylabel('Efficiency \eta','fontsize',18);
title(s,'Fontsize',20);
text(x(k,l),eta(k,l)-.5*etares(k,l), ...
     sprintf(' error=%3.1f%%',100*err(k,l)), 'Fontsize',18);
print -depsc -r600 eta-cubic

fn=fn+1;
figure(fn);
hold off;
tmp=1:NoP;
plot(tmp,100*yerr,'k-',tmp,100*etaerr,'k--');
legend('y','\eta',2);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
title('Maximal error for cubics','Fontsize',20);
xlabel('Dataset','fontsize',18);
```

```
ylabel('Maximal relative error in %','fontsize',18);
print -depsc -r600 cubic-err;

%
% "Envelope" of cubic
%

N=50;  % the number of interpolated curves we use to find the max
kk=10; % the number of interpolated curves we plot
R=0;
for i=1:length(thp)
  tmp=thp{2,i};
  d=thp{3,i}; % the minimal x
  e=thp{4,i}; % the maximal x
  xx=linspace(d+0.001,e-0.001); % the x values we want
  yy=zeros((N+1)*length(tmp),length(xx)); % all y-values initialized to

  fn=fn+1;
  figure(fn);
  hold off;
  k1=tmp(1);

  OldHandle=plot(xp(k1,:),yp(k1,:),'b-','LineWidth',2); % The first give
  hold on;

  a1=xmin(k1);
  b1=xmax(k1);
  c1=ycoeff(k1,:);

  for j=2:length(tmp)
    k0=k1;
    k1=tmp(j);
    plot(xp(k1,:),yp(k1,:),'b-','LineWidth',2); % A given curve
    a0=a1;
    b0=b1;
    c0=c1;

    a1=xmin(k1);
    b1=xmax(k1);
    c1=ycoeff(k1,:);

    for k=1:kk-1
      a=((kk-k)*a0+k*a1)/kk;      % xmin
```

```
      b=((kk-k)*b0+k*b1)/kk;      % xmax
      c=((kk-k)*c0+k*c1)/kk;
      xt=linspace(a,b);
      tt=(xt-a)/(b-a);
      yt=c*bernstein(3,tt);
      NewHandle=plot(xt,yt,'g-'); % An interpolated curve
    end

    for k=0:N
      a=((N-k)*a0+k*a1)/N;      % xmin
      b=((N-k)*b0+k*b1)/N;      % xmax
      c=((N-k)*c0+k*c1)/N;
      ide=find(a<=xx & xx<=b);
      dxe=xx(ide);              % the x values where the curve is defined
      tt=(dxe-a)/(b-a);         % the corresponding t values
      R=R+1;
      yy(R,ide)=c*bernstein(3,tt); % the y values
    end
  end
  YMax(1,i)={xx};
  YMax(2,i)={max(yy)};
  MaxHandle=plot(xx,max(yy),'r-','LineWidth',2);
  legend([OldHandle;NewHandle;MaxHandle], ...
         'characteristic','interpolated curve','max curve');
  xlabel('Normalized flow x','fontsize',18);
  ylabel('Normalized pressure y','fontsize',18);
  s=sprintf('max cubic curve for \\theta=%d',thp{1,i});
  title(s,'Fontsize',20)
  print('-depsc', '-r600', sprintf('cubicmax-%02d', thp{1,i}));
end

fn=fn+1;
figure(fn);
hold off;
MaxHandle=plot(YMax{1,1}, YMax{2,1}, 'r-','LineWidth',2);
hold on
for i=2:length(thp)
  plot(YMax{1,i}, YMax{2,i}, 'r-','LineWidth',2);
end
for k=1:NoP
  OldHandle=plot(xp(k,:),yp(k,:),'b-');
end
axis([0.2 1.3 0.4 1.2]);
```

```
legend([OldHandle;MaxHandle],'characteristic','max curve');
xlabel('Normalized flow x','fontsize',18);
ylabel('Normalized pressure y','fontsize',18);
s=sprintf('max cubic curves');
title(s,'Fontsize',20)
print('-depsc', '-r600', sprintf('cubic-max'));

%
% Performance map for cubic and log (eta is the same)
%

% first log:
Peta=zeros(PR,PS);
Ptheta=zeros(PR,PS);
Pphi=zeros(PR,PS);
Pn=zeros(PR,PS);
% data: xmin,xmax, ycoeff, etacoeff theta phi
CC=[xmin,xmax, logycoeff, etacoeff, theta, phi];

for l=1:length(Ptri)
  for i=1:length(Puvw)
    C=Puvw(i,:)*CC(Ptri(l,:),:);
    % evaluate the curves for t in Pt :
    xli=C(1)*(1-Pt)+C(2)*Pt;
    yli=C(3)*log(C(4)-xli)+C(5);
    Btj=bernstein(3,Pt);
    etali=C(6:9)*Btj;
    for j=1:length(Pt)
      [tmp,s]=min(abs(Px-xli(j))); % x is column number
      [tmp,r]=min(abs(Py-yli(j)));
      Pn(r,s)=Pn(r,s)+1;
      if(etali(j)>Peta(r,s))
        Peta(r,s)=etali(j);
        Ptheta(r,s)=C(10);
        Pphi(r,s)=C(11);
      end
    end
  end
end

fn=fn+1;
figure(fn);
hold off;
```

```
Z=log10(1+Pn);
BOT=floor(min(min(Z)));
TOP=ceil(max(max(Z)));
h=surfc(Px,Py,Z);
colorbar;
for i=2:length(h)
  newz=get(h(i),'Zdata')+0.5*(BOT-TOP);
  set(h(i),'Zdata', newz);
end
ZMIN=1.5*BOT-0.5*TOP;
axis([0.2 1.3 0.4 1.2 ZMIN TOP]);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
title('Number of samples using log','Fontsize',20);
xlabel('Normalized flow x','fontsize',18, ...
       'Position',[0.2 0.2 ZMIN],'Rotation',20);
ylabel('Normalized pressure y','fontsize',18, ...
       'Position',[-0.1 0.3 ZMIN],'Rotation',-30);
zlabel('log_{10}(1 + #samples)','fontsize',18);
print -depsc -r600 log-samples;


fn=fn+1;
figure(fn);
hold off;
Z=Peta;
BOT=floor(10*min(min(Z)))/10;
TOP=ceil(10*max(max(Z)))/10;
h=surfc(Px,Py,Z);
colorbar;
for i=2:length(h)
  newz=get(h(i),'Zdata')+0.5*(BOT-TOP);
  set(h(i),'Zdata', newz);
end
ZMIN=1.5*BOT-0.5*TOP;
axis([0.2 1.3 0.4 1.2 ZMIN TOP]);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
title('Maximal efficiency \eta using log','Fontsize',20);
xlabel('Normalized flow x','fontsize',18, ...
       'Position',[0.2 0.2 ZMIN],'Rotation',20);
ylabel('Normalized pressure y','fontsize',18, ...
       'Position',[-0.1 0.3 ZMIN],'Rotation',-30);
```

```
zlabel('Efficiency \eta','fontsize',18);
print -depsc -r600 log-eta;

fn=fn+1;
figure(fn);
hold off;
Z=Ptheta;
BOT=floor(min(min(Z)));
TOP=ceil(max(max(Z)));
h=surfc(Px,Py,Z);
colorbar;
for i=2:length(h)
  newz=get(h(i),'Zdata')+0.5*(BOT-TOP);
  set(h(i),'Zdata', newz);
end
ZMIN=1.5*BOT-0.5*TOP;
axis([0.2 1.3 0.4 1.2 ZMIN TOP]);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
title('Optimal \theta using log','Fontsize',20);
xlabel('Normalized flow x','fontsize',18, ...
       'Position',[0.2 0.2 ZMIN],'Rotation',20);
ylabel('Normalized pressure y','fontsize',18, ...
       'Position',[-0.1 0.3 ZMIN],'Rotation',-30);
zlabel('Inlet position \theta','fontsize',18);
print -depsc -r600 log-theta;

fn=fn+1;
figure(fn);
hold off;
Z=Pphi;
BOT=floor(min(min(Z)));
TOP=ceil(max(max(Z)));
h=surfc(Px,Py,Z);
colorbar;
for i=2:length(h)
  newz=get(h(i),'Zdata')+0.5*(BOT-TOP);
  set(h(i),'Zdata', newz);
end
ZMIN=1.5*BOT-0.5*TOP;
axis([0.2 1.3 0.4 1.2 ZMIN TOP]);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
```

```
title('Optimal \phi using log','Fontsize',20);
xlabel('Normalized flow x','fontsize',18, ...
       'Position',[0.2 0.2 ZMIN],'Rotation',20);
ylabel('Normalized pressure y','fontsize',18, ...
       'Position',[-0.1 0.3 ZMIN],'Rotation',-30);
zlabel('Diffuser position \phi','fontsize',18);
print -depsc -r600 log-phi;

fn=fn+1;
figure(fn);
hold off;
[cs,heta]=contour(Px,Py,Peta,etacntr,'r');
clabel(cs,heta);
hold on
[cs,hth]=contour(Px,Py,Ptheta, ...
            linspace(0,floor(10*max(max(Ptheta)))/10,10),'g');
[cs,hph]=contour(Px,Py,Pphi, ...
            linspace(0,floor(10*max(max(Pphi)))/10,10),'b');
axis([0.2 1.3 0.4 1.2]);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
legend([heta(1) hth(1) hph(1)],'\eta','\theta','\phi');
xlabel('Normalized flow x','fontsize',12);
ylabel('Normalized pressure y','fontsize',12);
title('Contour plot of optimal \eta,\theta,\phi using log', ...
      'Fontsize',20);
print -depsc -r600 log-contour;

%
% then cubic:
%
Peta=zeros(PR,PS);
Ptheta=zeros(PR,PS);
Pphi=zeros(PR,PS);
Pn=zeros(PR,PS);
% data: xmin,xmax, ycoeff, etacoeff theta phi
CC=[xmin,xmax, ycoeff, etacoeff, theta, phi];

for l=1:length(Ptri)
  for i=1:length(Puvw)
    C=Puvw(i,:)*CC(Ptri(l,:),:);
    % evaluate the curves for t in Pt :
    xli=C(1)*(1-Pt)+C(2)*Pt;
```

```
    Btj=bernstein(3,Pt);
    yli=C(3:6)*Btj;
    etali=C(7:10)*Btj;
    for j=1:length(Pt)
      [tmp,s]=min(abs(Px-xli(j))); % x is column number
      [tmp,r]=min(abs(Py-yli(j)));
      Pn(r,s)=Pn(r,s)+1;
      if(etali(j)>Peta(r,s))
        Peta(r,s)=etali(j);
        Ptheta(r,s)=C(11);
        Pphi(r,s)=C(12);
      end
    end
  end
end

fn=fn+1;
figure(fn);
hold off;
Z=log10(1+Pn);
BOT=floor(min(min(Z)));
TOP=ceil(max(max(Z)));
h=surfc(Px,Py,Z);
colorbar;
for i=2:length(h)
  newz=get(h(i),'Zdata')+0.5*(BOT-TOP);
  set(h(i),'Zdata', newz);
end
ZMIN=1.5*BOT-0.5*TOP;
axis([0.2 1.3 0.4 1.2 ZMIN TOP]);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
title('Number of samples using cubic','Fontsize',20);
xlabel('Normalized flow x','fontsize',18, ...
       'Position',[0.2 0.2 ZMIN],'Rotation',20);
ylabel('Normalized pressure y','fontsize',18, ...
       'Position',[-0.1 0.3 ZMIN],'Rotation',-30);
zlabel('log_{10}(1 + #samples)','fontsize',18);
print -depsc -r600 cubic-samples;

fn=fn+1;
figure(fn);
hold off;
```

```
Z=Peta;
BOT=floor(10*min(min(Z)))/10;
TOP=ceil(10*max(max(Z)))/10;
h=surfc(Px,Py,Z);
colorbar;
for i=2:length(h)
  newz=get(h(i),'Zdata')+0.5*(BOT-TOP);
  set(h(i),'Zdata', newz);
end
ZMIN=1.5*BOT-0.5*TOP;
axis([0.2 1.3 0.4 1.2 ZMIN TOP]);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
title('Maximal efficiency \eta using cubic','Fontsize',20);
xlabel('Normalized flow x','fontsize',18, ...
       'Position',[0.2 0.2 ZMIN],'Rotation',20);
ylabel('Normalized pressure y','fontsize',18, ...
       'Position',[-0.1 0.3 ZMIN],'Rotation',-30);
zlabel('Efficiency \eta','fontsize',18);
print -depsc -r600 cubic-eta;

fn=fn+1;
figure(fn);
hold off;
Z=Ptheta;
BOT=floor(min(min(Z)));
TOP=ceil(max(max(Z)));
h=surfc(Px,Py,Z);
colorbar;
for i=2:length(h)
  newz=get(h(i),'Zdata')+0.5*(BOT-TOP);
  set(h(i),'Zdata', newz);
end
ZMIN=1.5*BOT-0.5*TOP;
axis([0.2 1.3 0.4 1.2 ZMIN TOP]);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
title('Optimal \theta using cubic','Fontsize',20);
xlabel('Normalized flow x','fontsize',18, ...
       'Position',[0.2 0.2 ZMIN],'Rotation',20);
ylabel('Normalized pressure y','fontsize',18, ...
       'Position',[-0.1 0.3 ZMIN],'Rotation',-30);
zlabel('Inlet position \theta','fontsize',18);
```

```
print -depsc -r600 cubic-theta;

fn=fn+1;
figure(fn);
hold off;
Z=Pphi;
BOT=floor(min(min(Z)));
TOP=ceil(max(max(Z)));
h=surfc(Px,Py,Z);
colorbar;
for i=2:length(h)
  newz=get(h(i),'Zdata')+0.5*(BOT-TOP);
  set(h(i),'Zdata', newz);
end
ZMIN=1.5*BOT-0.5*TOP;
axis([0.2 1.3 0.4 1.2 ZMIN TOP]);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
title('Optimal \phi using cubic','Fontsize',20);
xlabel('Normalized flow x','fontsize',18, ...
       'Position',[0.2 0.2 ZMIN],'Rotation',20);
ylabel('Normalized pressure y','fontsize',18, ...
       'Position',[-0.1 0.3 ZMIN],'Rotation',-30);
zlabel('Diffuser position \phi','fontsize',18);
print -depsc -r600 cubic-phi;

fn=fn+1;
figure(fn);
hold off;
[cs,heta]=contour(Px,Py,Peta,etacntr,'r');
clabel(cs,heta);
hold on
[cs,hth]=contour(Px,Py,Ptheta, ...
            linspace(0,floor(10*max(max(Ptheta)))/10,10),'g');
[cs,hph]=contour(Px,Py,Pphi, ...
            linspace(0,floor(10*max(max(Pphi)))/10,10),'b');
axis([0.2 1.3 0.4 1.2]);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
legend([heta(1) hth(1) hph(1)],'\eta','\theta','\phi');
xlabel('Normalized flow x','fontsize',12);
ylabel('Normalized pressure y','fontsize',12);
title('Contour plot of optimal \eta,\theta,\phi using cubic', ...
```

```
      'Fontsize',20);
print -depsc -r600 cubic-contour;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Quadratic uniform B-spline y = sum c_k n^2_k(t)
%                              x = (x_1*(K-t)+x_n*t)/K
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% We try 2 and 3 segments

for K=2:3
  ycoeff=zeros(NoP,K+2);
  yres=zeros(NoP,NoM);
  etacoeff=zeros(NoP,K+2);
  etares=zeros(NoP,NoM);

  u=[-2:K+2]; % knot sequence

  A=zeros(2*K+1,K+2);
  for k=1:K+1
    A(k,k)=-1;
    A(k,k+1)=1;
  end
  for k=1:K
    A(K+1+k,k)=1;
    A(K+1+k,k+1)=-2;
    A(K+1+k,k+2)=1;
  end
  b=zeros(2*K+1,1);

  for no=1:NoP

    t=K*(x(no,:)-xmin(no))/(xmax(no)-xmin(no));
    N=bspline(2,u,t);

    etacoeff(no,:)=eta(no,:)/N;
    etares(no,:) =eta(no,:)-etacoeff(no,:)*N;

    H=N*N';
    f=-N*y(no,:)';
```

```
  options=optimset('LargeScale','off');
  [coeff,fval]= quadprog(H,f,A,b,[],[],[],[],[],options);
  ycoeff(no,:) = coeff';
  yres(no,:) = y(no,:)-coeff'*N;

  tt=K*(xp(no,:)-xmin(no))/(xmax(no)-xmin(no));
  yp(no,:)=coeff'*bspline(2,u,tt);
  etap(no,:)=etacoeff(no,:)*bspline(2,u,tt);
end

%
% Worst y:
%
err=abs(yres)./y;
[m,k]=max(err);
[m,l]=max(max(err));
k=k(l);
[k,l,theta(k),phi(k),err(k,l)]

yerr=max(err');

fn=fn+1;
figure(fn);
hold off;
plot(x(k,:),y(k,:),'k+');
hold on;
plot([x(k,l),x(k,l)],[y(k,l),y(k,l)-yres(k,l)],'k:');
plot(xp(k,:),yp(k,:),'k-');
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
s=sprintf('Worst y spline (%d), dataset no. %d, \\theta=%d, \\phi=%d',
          K, k, theta(k), phi(k));
xlabel('Normalized flow x','fontsize',18);
ylabel('Normalized pressure y','fontsize',18);
title(s,'Fontsize',20);
text(x(k,l),y(k,l)-.5*yres(k,l), ...
     sprintf(' error=%3.1f%%',100*err(k,l)), 'Fontsize',18);
print('-depsc', '-r600', sprintf('y-spline-%d',K));

%
% Worst eta:
%
```

```
err=abs(etares)./eta;
[m,k]=max(err);
[m,l]=max(max(err));
k=k(l);
[k,l,theta(k),phi(k),err(k,l)]

etaerr=max(err');

fn=fn+1;
figure(fn);
hold off;
plot(x(k,:),eta(k,:),'k+');
hold on;
plot([x(k,l),x(k,l)],[eta(k,l),eta(k,l)-etares(k,l)],'k:');
plot(xp(k,:),etap(k,:),'k-');
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
s=sprintf('Worst \\eta spline (%d), dataset no. %d, \\theta=%d, \\phi=%d', ..
          K, k, theta(k), phi(k));
xlabel('Normalized flow x','fontsize',18);
ylabel('Efficiency \eta','fontsize',18);
title(s,'Fontsize',20);
text(x(k,l),eta(k,l)-.5*etares(k,l), ...
     sprintf(' error=%3.1f%%',100*err(k,l)),'Fontsize',18);
print('-depsc', '-r600', sprintf('eta-spline-%d',K));

fn=fn+1;
figure(fn);
hold off;
tmp=1:NoP;
plot(tmp,100*yerr,'k-',tmp,100*etaerr,'k--');
legend('y','\eta',2);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
s=sprintf('Maximal error for spline with %d segments', K);
title(s,'Fontsize',20);
xlabel('Dataset','fontsize',18);
ylabel('Maximal relative error in %','fontsize',18);
print('-depsc', '-r600', sprintf('spline%d-err',K));

%
% "Envelope" of quadratic spline
%
```

```
N=50;  % the number of interpolated curves we use to find the max
kk=10; % the number of interpolated curves we plot
R=0;
for i=1:length(thp)
  tmp=thp{2,i};
  d=thp{3,i}; % the minimal x
  e=thp{4,i}; % the maximal x
  xx=linspace(d+0.001,e-0.001); % the x values we want
  yy=zeros((N+1)*length(tmp),length(xx)); % all y-values initialized t

  fn=fn+1;
  figure(fn);
  hold off;
  k1=tmp(1);
  OldHandle=plot(xp(k1,:),yp(k1,:),'b-','LineWidth',2);
  hold on;

  a1=xmin(k1);
  b1=xmax(k1);
  c1=ycoeff(k1,:);

  for j=2:length(tmp)
    k0=k1;
    k1=tmp(j);
    plot(xp(k1,:),yp(k1,:),'b-','LineWidth',2); % A given curve
    a0=a1;
    b0=b1;
    c0=c1;

    a1=xmin(k1);
    b1=xmax(k1);
    c1=ycoeff(k1,:);

    for k=1:kk-1
      a=((kk-k)*a0+k*a1)/kk;      % xmin
      b=((kk-k)*b0+k*b1)/kk;      % xmax
      c=((kk-k)*c0+k*c1)/kk;
      xt=linspace(a,b);
      tt=K*(xt-a)/(b-a);
      yt=c*bspline(2,u,tt);
      NewHandle=plot(xt,yt,'g-'); % An interpolated curve
    end
```

```
    for k=0:N
      a=((N-k)*a0+k*a1)/N;      % xmin
      b=((N-k)*b0+k*b1)/N;      % xmax
      c=((N-k)*c0+k*c1)/N;
      ide=find(a<=xx & xx<=b);
      dxe=xx(ide);              % the x values where the curve is defined
      tt=K*(dxe-a)/(b-a);         % the corresponding t values
      R=R+1;
      yy(R,ide)=c*bspline(2,u,tt); % the y values
    end
  end
  YMax(1,i)={xx};
  YMax(2,i)={max(yy)};
  MaxHandle=plot(xx,max(yy),'r-','LineWidth',2);
  legend([OldHandle;NewHandle;MaxHandle], ...
         'characteristic','interpolated curve','max curve');
  xlabel('Normalized flow x','fontsize',18);
  ylabel('Normalized pressure y','fontsize',18);
  s=sprintf('max spline(%d) curve for \\theta=%d',K,thp{1,i});
  title(s,'Fontsize',20)
  print('-depsc', '-r600', sprintf('spline%dmax-%02d', K, thp{1,i}));
end

fn=fn+1;
figure(fn);
hold off;
MaxHandle=plot(YMax{1,1}, YMax{2,1}, 'r-','LineWidth',2);
hold on
for i=2:length(thp)
  plot(YMax{1,i}, YMax{2,i}, 'r-','LineWidth',2);
end
for k=1:NoP
  OldHandle=plot(xp(k,:),yp(k,:),'b-');
end
axis([0.2 1.3 0.4 1.2]);
legend([OldHandle;MaxHandle],'characteristic','max curve');
xlabel('Normalized flow x','fontsize',18);
ylabel('Normalized pressure y','fontsize',18);
s=sprintf('max spline(%d) curves',K);
title(s,'Fontsize',20)
print('-depsc', '-r600', sprintf('spline%d-max',K));
```

```
%
% Performance map for spline
%

Peta=zeros(PR,PS);
Ptheta=zeros(PR,PS);
Pphi=zeros(PR,PS);
Pn=zeros(PR,PS);
% data: xmin,xmax, ycoeff, etacoeff theta phi
CC=[xmin,xmax, ycoeff, etacoeff, theta, phi];

for l=1:length(Ptri)
  for i=1:length(Puvw)
    C=Puvw(i,:)*CC(Ptri(l,:),:);
    % evaluate the curves for t in Pt :
    xli=C(1)*(1-Pt)+C(2)*Pt;
    Ntj=bspline(2,u,K*Pt); % we define the spline on [0,K] !
    yli=C(3:4+K)*Ntj;
    etali=C(5+K:6+2*K)*Ntj;
    for j=1:length(Pt)
      [tmp,s]=min(abs(Px-xli(j))); % x is column number
      [tmp,r]=min(abs(Py-yli(j)));
      Pn(r,s)=Pn(r,s)+1;
      if(etali(j)>Peta(r,s))
        Peta(r,s)=etali(j);
        Ptheta(r,s)=C(7+2*K);
        Pphi(r,s)=C(8+2*K);
     end
    end
  end
end

fn=fn+1;
figure(fn);
hold off;
Z=log10(1+Pn);
BOT=floor(min(min(Z)));
TOP=ceil(max(max(Z)));
h=surfc(Px,Py,Z);
colorbar;
for i=2:length(h)
  newz=get(h(i),'Zdata')+0.5*(BOT-TOP);
  set(h(i),'Zdata', newz);
```

```
end
ZMIN=1.5*BOT-0.5*TOP;
axis([0.2 1.3 0.4 1.2 ZMIN TOP]);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
s=sprintf('Number of samples using spline(%d)',K);
title(s,'Fontsize',20);
xlabel('Normalized flow x','fontsize',18, ...
       'Position',[0.2 0.2 ZMIN],'Rotation',20);
ylabel('Normalized pressure y','fontsize',18, ...
       'Position',[-0.1 0.3 ZMIN],'Rotation',-30);
zlabel('log_{10}(1 + #samples)','fontsize',18);
print('-depsc', '-r600', sprintf('spline%d-samples',K));

fn=fn+1;
figure(fn);
hold off;
Z=Peta;
BOT=floor(10*min(min(Z)))/10;
TOP=ceil(10*max(max(Z)))/10;
h=surfc(Px,Py,Z);
colorbar;
for i=2:length(h)
  newz=get(h(i),'Zdata')+0.5*(BOT-TOP);
  set(h(i),'Zdata', newz);
end
ZMIN=1.5*BOT-0.5*TOP;
axis([0.2 1.3 0.4 1.2 ZMIN TOP]);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
s=sprintf('Maximal efficiency \\eta using spline(%d)',K);
title(s,'Fontsize',20);
xlabel('Normalized flow x','fontsize',18, ...
       'Position',[0.2 0.2 ZMIN],'Rotation',20);
ylabel('Normalized pressure y','fontsize',18, ...
       'Position',[-0.1 0.3 ZMIN],'Rotation',-30);
zlabel('Efficiency \eta','fontsize',18);
print('-depsc', '-r600', sprintf('spline%d-eta',K));

fn=fn+1;
figure(fn);
hold off;
Z=Ptheta;
```

```
BOT=floor(min(min(Z)));
TOP=ceil(max(max(Z)));
h=surfc(Px,Py,Z);
colorbar;
for i=2:length(h)
  newz=get(h(i),'Zdata')+0.5*(BOT-TOP);
  set(h(i),'Zdata', newz);
end
ZMIN=1.5*BOT-0.5*TOP;
axis([0.2 1.3 0.4 1.2 ZMIN TOP]);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
s=sprintf('Optimal \\theta using spline(%d)',K);
title(s,'Fontsize',20);
xlabel('Normalized flow x','fontsize',18, ...
       'Position',[0.2 0.2 ZMIN],'Rotation',20);
ylabel('Normalized pressure y','fontsize',18, ...
       'Position',[-0.1 0.3 ZMIN],'Rotation',-30);
zlabel('Inlet position \theta','fontsize',18);
print('-depsc', '-r600', sprintf('spline%d-theta',K));

fn=fn+1;
figure(fn);
hold off;
Z=Pphi;
BOT=floor(min(min(Z)));
TOP=ceil(max(max(Z)));
h=surfc(Px,Py,Z);
colorbar;
for i=2:length(h)
  newz=get(h(i),'Zdata')+0.5*(BOT-TOP);
  set(h(i),'Zdata', newz);
end
ZMIN=1.5*BOT-0.5*TOP;
axis([0.2 1.3 0.4 1.2 ZMIN TOP]);
axesobj=findobj('type','axes');
set(axesobj,'fontsize',12);
s=sprintf('Optimal \\phi using spline(%d)',K);
title(s,'Fontsize',20);
xlabel('Normalized flow x','fontsize',18, ...
       'Position',[0.2 0.2 ZMIN],'Rotation',20);
ylabel('Normalized pressure y','fontsize',18, ...
       'Position',[-0.1 0.3 ZMIN],'Rotation',-30);
```

```
  zlabel('Diffuser position \phi','fontsize',18);
  print('-depsc', '-r600', sprintf('spline%d-phi',K));

  fn=fn+1;
  figure(fn);
  hold off;
  [cs,heta]=contour(Px,Py,Peta,etacntr,'r');
  clabel(cs,heta);
  hold on
  [cs,hth]=contour(Px,Py,Ptheta, ...
              linspace(0,floor(10*max(max(Ptheta)))/10,10),'g');
  [cs,hph]=contour(Px,Py,Pphi, ...
              linspace(0,floor(10*max(max(Pphi)))/10,10),'b');
  axis([0.2 1.3 0.4 1.2]);
  axesobj=findobj('type','axes');
  set(axesobj,'fontsize',12);
  legend([heta(1) hth(1) hph(1)],'\eta','\theta','\phi');
  xlabel('Normalized flow x','fontsize',18);
  ylabel('Normalized pressure y','fontsize',18);
  s=sprintf('Contour plot of optimal \\eta,\\theta,\\phi using spline (%d)',K);
  title(s,'Fontsize',20);
  print('-depsc', '-r600', sprintf('spline%d-contour',K));
end
```

A.2 abclog.m

```
function  [Y,J]=logfuncjac(beta,x)

a=beta(1);
b=beta(2);
c=beta(3);

Y=a*log(b-x)+c;

if nargout>1
    J=[log(b-x)', a./(b-x'), ones(length(x),1)]; % The Jacobian
end
```

A.3 bernstein.m

```
function b=bernstein(deg,t)

zero=zeros(1,length(t));
b=ones(deg+1,length(t));
u=repmat(t,deg+1,1);

for i=1:deg
  b=(1-u(1:i+1,:)).*[b(1:i,:); zero]+u(1:i+1,:).*[zero; b(1:i,:)];
end;
```

A.4 bspline.m

```
function n=bspline(deg,u,t)

  n=zeros(length(u)-1,length(t)); % deg to long

  K=length(u)-2*deg-1;
  L=length(t);
  k=1;
  for i=1:L
    while u(k)-t(i)<0
      k=k+1;
    end
    % now t(i)\in [u(k),u(k+1)[
    n(k-1,i)=1;
  end
  for r=1:deg
    n=(repmat(t,K+2*deg-r,1)-repmat(u(1:K+2*deg-r)',1,L))./ ...
      (repmat(u(1+r:K+2*deg)',1,L)-repmat(u(1:K+2*deg-r)',1,L)).* ...
      n(1:K+2*deg-r,:) + ...
      (repmat(u(r+2:K+2*deg+1)',1,L)-repmat(t,K+2*deg-r,1))./ ...
      (repmat(u(r+2:K+2*deg+1)',1,L)-repmat(u(2:K+2*deg-r+1)',1,L)).* ..
      n(2:K+2*deg-r+1,:);
  end
```