TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Constructing curves using BI-ARCS

Citation for published version (APA):
Haan, de, S., Scholten, I., & Rienstra, S. W. (1996). *Constructing curves using BI-ARCS*. (IWDE report; Vol. 9607). Technische Universiteit Eindhoven.

Document status and date:
Published: 01/01/1996

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 12. Jul. 2024

Technische
Universiteit
Eindhoven

Instituut Wiskundige Dienstverlening Eindhoven

REPORT IWDE 96 - 07

CONSTRUCTING CURVES USING BI-ARCS

Siebren de Haan
Ingeborg Scholten
Sjoerd Rienstra

December 1996

Den Dolech 2
Postbus 513
5600 MB Eindhoven

Constructing curves using BI-ARCS

Siebren de Haan Ingeborg Scholten Sjoerd Rienstra

December 20, 1996

Summary

Bending tubes into smooth shapes is an actual item for Philips Lighting. A method to bend a glass tube is to pull the tube with constant linear and angular speed through a hot spot where the glass is weakened. A bent tube of almost arbitrary shape can be obtained by a combination of successive circle segments. To get a smooth shape there are some restrictions on finding the circles to use.
Besides demanding to have a smooth shape it will be the cheapest way to use as less circles as possible to get this shape; the loss of accuracy in shape against the profit by using less circles can be weighed.
Programs in C++ that construct this kind of shapes are written and explained.

Contents

1 Introduction

Philips Lighting is developing a new kind of brake- and rear-lights. These lights will be made of thin tubes which have to be bent in specified plane curves. Philips has a computerized bending-machine to do this but it can only make a curve existing of arcs of a circle. The object of this report is to describe a plane curve that fits the given data points (defined by about 50 to 100 x,y co-ordinates of points) in a proper way by arcs of circles. Something to take into account is that the co-ordinates of the curve contain some error; this means that the curve does not have to go through all points exactly.
Section 2 first gives some background information about the bi-arcs that will be used. The way to make a curve between two points using bi-arcs is described. In Section 3 the problem is defined. It is divided into the initial problem and the main problem. The initial problem concerns the construction of an *initial guess* for a curve consisting of $(K-1)$ bi-arcs when N data-points are given. The main problem is to construct a *well-fitting* curve using $(K-1) < (N-1)$ bi-arcs when N data-points are given. The analysis and results of the initial problem are described in Section 4.1. The analysis and results of the main problem are described in Section 4.2. In Section 5 the results are illustrated for a real-life case; discussion about those results is presented in Section 5.1. Section 6 gives information about the program in C++ that we wrote and finally, in Section 7 conclusions are drawn and recommendations are given.
For PC-DOS, the input and output for our program is controlled by a program `BIARCMNU`. This program is written by Sjoerd Rienstra, member of IWDE of the Faculty of Mathematics and Computer Science of the Eindhoven University of Technology.

2 Construction of a bi-arc

Throughout this report the definition of a bi-arc will be very important. We follow the construction used in [4]. In several other publications different constructions are made which are equivalent to our construction of a bi-arc, see for instances [3].
We start with describing the construction of a bi-arc between two points determined by their x and y coordinates.
A bi-arc is a curve between two points that is made of two joint circular arcs in such a way that the tangent is continuous in the joining point. A complete description of each arc can be given when the position of the starting point (or end point) (x, y), the radius of the circle r and the angle of the tangent

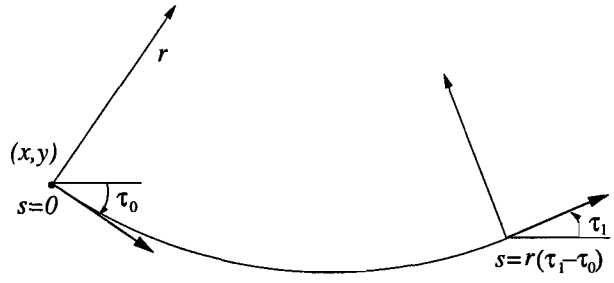

Figure 1: A convex arc.

vector in the starting and end point are given.
Let τ_0 be the angle of the tangent vector in the starting point and τ_1 be the angle of the vector in the end point of the arc. The difference between τ_0 and τ_1 determines whether the arc is concave or convex. Let $\tau(s)$ be the angle of the direction vector of the arc in a point on the arc defined by s with s the arc length, that is

$$\tau(s) = \tau_0 + \frac{s}{r}, \tag{1}$$

where s runs from 0 to $r(\tau_1 - \tau_0)$. Suppose now $\tau_1 - \tau_0 > 0$, then the arc C_1 is convex and is given by

$$C_1(s) = \begin{pmatrix} x \\ y \end{pmatrix} + r \begin{pmatrix} -\sin(\tau_0) + \sin(\tau(s)) \\ \cos(\tau_0) - \cos(\tau(s)) \end{pmatrix}, \tag{2}$$

where $\tau(s)$ is given by equation (1), x and y are the co-ordinates of the starting point, r is the radius and τ_0 and τ_1 are the tangents directions in the starting point and end point. See Figure 1.
When $\tau_1 - \tau_0 < 0$ the arc C_2 is concave and is given by

$$C_2(s) = \begin{pmatrix} x \\ y \end{pmatrix} + r \begin{pmatrix} \sin(\tau_0) - \sin(\tau(s)) \\ -\cos(\tau_0) + \cos(\tau(s)) \end{pmatrix}, \tag{3}$$

where $\tau(s)$ is given by equation (1). See Figure 2.
Equations (2) and (3) differ only in the sign of the radius r. Normally the radius of a circle is a positive number; however, when we define the radius of a concave arc to be minus the radius of the circle from which the arc originates, we have a unique representation of an arc. Note that the sign of r depends on whether τ_0 is larger than τ_1.
We now construct a bi-arc that connects two points with given tangent vectors. First we translate and rotate the points such that the starting point is the origin and the tangent vector in the first point is the unit x-vector. After the construction of the bi-arc we have to rotate and translate the solution to

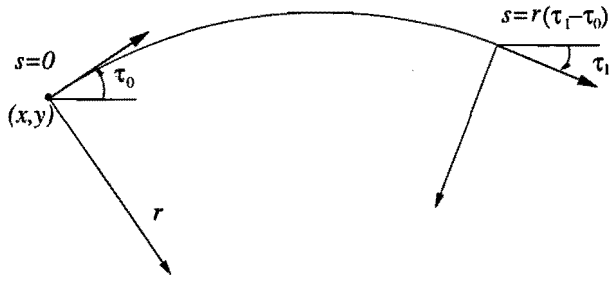

Figure 2: A concave arc.

the original co-ordinates. Let x and y be the co-ordinates of the rotated and translated end point.
Instead of using the coordinates of the point (x, y) we use the length of the vector (x, y) and the angle between the vector and the x-axis. Let l be the length of the vector (x, y) and α the angle between the vector and the x-axis. Let W be the tangent vector in the end point. We chose the name W for the tangent vector in the end point in order to keep the same nomenclature as in the C++ program.
Let the bi-arc between the origin and the point (x, y) consist of C_1 and C_2. Let C_1 be the first circular part and C_2 the second part of the bi-arc. Suppose that the first bi-arc has a radius r_A and an end point with tangent θ. Then, using the above notation for an arc we have

$$C_1(s) = r_A \begin{pmatrix} \sin\tau(s) \\ 1 - \cos\tau(s) \end{pmatrix}, \tag{4}$$

with $\tau(s) = sr_A^{-1}$. The second arc starts at the point

$$C_1(r_A\theta) = r_A \begin{pmatrix} \sin\theta \\ 1 - \cos\theta \end{pmatrix}, \tag{5}$$

and has a radius of r_B, and a prescribed tangent vector W in the end point. Using the above construction, we obtain for the second arc:

$$C_2(s) = r_A \begin{pmatrix} \sin\theta \\ 1 - \cos\theta \end{pmatrix} + r_B \begin{pmatrix} -\sin\theta + \sin\tau(s) \\ \cos\theta - \cos\tau(s) \end{pmatrix}, \tag{6}$$

with $\tau(s) = \theta + sr_B^{-1}$.
When for $s = r_B(W - \theta)$ the vector $C_2(s)$ is equal to (x, y) we have a bi-arc that connects the origin with the point (x, y) and which has a tangent vector with a direction equal to W in that point. This results in the following equation, where (x, y) is expressed using the length l and the angle α

$$l \cdot \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} = r_A \begin{pmatrix} \sin\theta \\ 1 - \cos\theta \end{pmatrix} + r_B \begin{pmatrix} -\sin\theta + \sin W \\ \cos\theta - \cos W \end{pmatrix} \tag{7}$$

Note that the unknowns are the angle θ and the radii r_A and r_B. We have three unknowns and two equations. By choosing a suitable value of θ we have two equations for the two unknowns. What we mean by suitable is discussed at the end of this section. Except for the case $W = 0$ we have uniquely solvable set of equations.
For $W = 0$ we have the following equation

$$l \cdot \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} = (r_A - r_B) \begin{pmatrix} \sin\theta \\ 1 - \cos\theta \end{pmatrix} \tag{8}$$

which has only a solution for $\theta = 2\alpha$. The relation between the radii is

$$r_A = r_B + \frac{l}{2\sin\alpha} \tag{9}$$

This relation still does not determine the radii uniquely, however by taking the radius of r_A equal to $-r_B$ the radii are uniquely determined. This choice results in a bi-arc with the smallest curvature which is defined for a circle as the inverse of radius (actually the inverse of the absolute value of the radius). When $W \neq 0$ the solution of equation (7) for given θ is

$$r_A = \frac{\sin\left(\frac{W+\theta}{2} - \alpha\right)}{2\sin\frac{\theta}{2}\sin\frac{W}{2}} l \tag{10}$$

$$r_B = \frac{\sin\left(\alpha - \frac{\theta}{2}\right)}{2\sin\frac{W-\theta}{2}\sin\frac{W}{2}} l \tag{11}$$

To prevent cusps at the joining point the two arc lengths should be positive. If $W > 0$ this implies that $2\alpha - W < \theta < 2\alpha$, if $W < 0$ this implies that $2\alpha < \theta < 2\alpha - W$.
In Figure 3 a bi-arc connecting two points A and B is drawn. Note that the tangent in A is parallel to the x-axis. When we choose

$$\theta = 2\alpha - \frac{1}{2}W \tag{12}$$

we have a curve with no cusp. In this case the discontinuity in the curvature is minimized. One can also choose θ such that the curvature of the bi-arc is minimized.

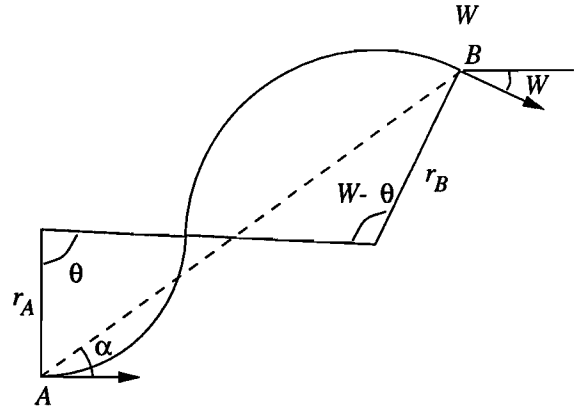

Figure 3: A bi-arc.

3 Problem definition

The problem we solve is:

> **Construct a curve of $(K-1)$ bi-arcs that minimizes the distance to N given data points and the curvature.**

This problem is divided into an initial problem and a main problem. The initial problem is to obtain a guess for a curve close to the data. The main problem is to obtain a curve that minimizes a function; this curve is initiated by the starting guess. This minimization function is explained in one of the following sections.

Definition of the initial problem

In the initial problem we search for a first guess for the curve. This means that the position of the K points, interconnected by the bi-arcs, has to be chosen in a smart way. The first of these K points is the first of the N data points, the last of the K points is the last of the N data points. In these K points the tangents are chosen. Taking these tangents into consideration, a curve has to be constructed consisting of $(K-1)$ bi-arcs. By considering all possible values of θ for every bi-arc, we search for the curve with minimal curvature in every bi-arc.

Definition of the main problem

The main problem is to construct a curve, again consisting of $(K-1)$ bi-arcs; however, this curve has to minimize

- a function of the distance of all given points to that curve
- the curvature along the whole curve.

4 Analysis and result of the problem

This section first describes the analysis of the initial problem; the results are given using an example. The following section concerns the main problem. The result of the main problem is visualized with the same example data.

4.1 The initial problem

We now analyze the initial problem. Having defined bi-arcs, the first step we make is to decide through which points the curve will go, and to determine the tangents in these points of the curve. There are several ways to choose the co-ordinates of the K points. Some possibilities are:

1. Choose K out of the N given points at regular intervals.

2. Choose more points in a region where the curvature is large and less points in a region where the curvature is small.

3. Construct a tube with radius ε around all but one of the N points. If that one point is outside the tube, it has to be taken into account; if that point lies inside the tube, it does not give much information and can be deleted out of the dataset. Repeat this until only K points are left. This is shown in Figure 4.

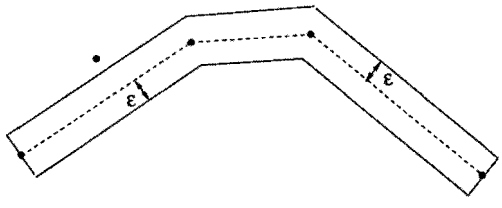

Figure 4: Choosing K points using tubes.

4. Draw a straight line between successive points. The total length of this line from the first point to the last point will be divided into $K-1$ equal parts. The K points lie on the straight line with regular intervals. This method is shown in Figure 5.

In all possibilities the first point is the first point of the N given points and the K-th point is the last point of the N given points.
We implemented both the second and the last method to select K points. Using the second method, the starting-points can be chosen by giving indices of the data-points or by giving the co-ordinates of the points.

Figure 5: Choosing K points to use to construct a curve.

We illustrate the second method by an example. We use the following example data points given in the table below.

i	x	y
1	1.000000	0.000000
2	0.000000	2.000000
3	1.000000	4.000000
4	2.000000	6.000000
5	3.000000	5.000000
6	4.000000	4.000000
7	5.500000	3.000000

The points through which the curve will go are chosen to be the points with indices $1, 4, 7$, so $K = 3$. Knowing the tangents in the curve points, we can construct a bi-arc between every two successive curve points. We decided to give the tangent in a point z_i the value of the direction between the point before (i.e. z_{i-1}) and the point after this point (i.e. z_{i+1}), except for the first and the last point. In these points the tangents are determined by using the point itself and its neighbor. The tangent of the first point of the example curve is parallel to the line between the first and second curve point; the tangent of the second point is parallel to the line between the first and third (or last) point; the tangent in the last point is parallel to the line between the last but one and the last point. This is shown in Figure 6.
Knowing these tangents, we can construct a curve with least mean curvature. As said before by choosing θ in the middle of its allowed range we obtain a bi-arc with minimal jump in the curvature; r_A and r_B are chosen corresponding to this. We consider two definitions of mean curvature for one bi-arc:

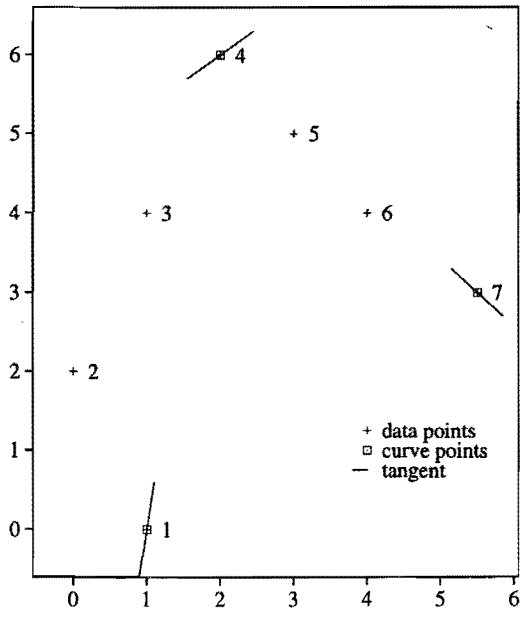

Figure 6: The data points and the tangents in the curve points.

$$1. \quad |\frac{\theta}{r_A}| + |\frac{W-\theta}{r_B}|$$

$$2. \quad \sqrt{\frac{|\theta|}{r_A^2} + \frac{|W-\theta|}{r_B^2}}$$

We extend this definition of curvature for one bi-arc to a curve of bi-arcs. This curve consists of arcs with radii r_i and angles θ_i, and the definitions become

$$1. \quad \sum |\frac{\theta_i}{r_i}|$$

$$2. \quad \sqrt{\sum \frac{|\theta_i|}{r_i^2}}$$

The first definition of the mean curvature is also called the strain energy. This definition suggest that the mean curvature is proportional to the integral of the square of the curvature along the curve, which in our case is equal to the inverse of the radius. The second definition can be seen as the integral of the third power of the curvature along the curve. See [8].
To show the differences between the two definitions of mean curvature we show a graph of the resulting minimized curve for the example dataset. These two graphs are shown in Figure 7.

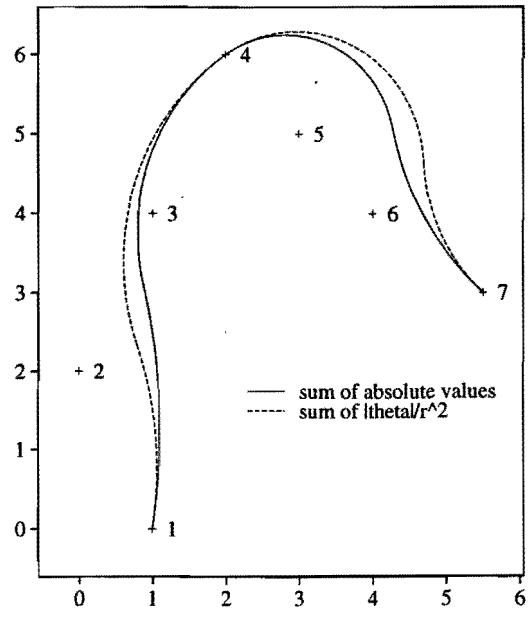

Figure 7: The resulting curves for the two different definitions of curvature.

From Figure 7 we see that when the mean curvature is defined as the sum of absolute value of the angles divided by the squared radius, a smoother solution is obtained than when the curvature is defined as the sum of absolute values. The best choice depends on the demands of the user.

4.2 The main problem

In this section the way we solved the main problem is described and the results are shown in figures, using the example data from the previous section. The main problem is to construct a curve that goes through K points such that the distance of the data-points to the curve is minimized.
Three definitions of distance of a data set to a curve are considered:

1. $\frac{1}{N}\sum d_i$
2. $\sqrt{\frac{1}{N}\sum (d_i)^2}$
3. $\max\ d_i$

where d_i is the smallest distance of point i, with $(1 \leq i \leq N)$ to the curve. Using the initial guess discussed in the previous section, we can vary the position and the tangents of the curve points to find a curve that minimizes

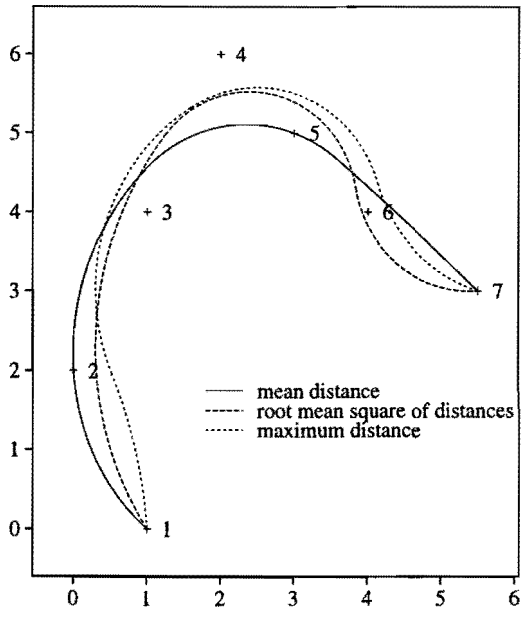

Figure 8: The resulting curves for the three different definitions of distance

the distance. The three different definitions of distance result in the three curves shown in Figure 8.

5 Results of the project

For the main problem, many minimization functions are possible. Our program can use two definitions of curvature and three definitions of the distance of a data set to a curve. These definitions can be combined to prescribe the minimization function. Depending on the chosen function, the program in C++, using the 'Numerical Recipes' routine `Amoeba`, searches for a curve with minimum function-value, given a certain initial guess. The final result may be plotted by the public domain package `GNUPLOT`.
In the next figure we illustrate a curve that is originally described by $N = 53$ data-points. In Figure 9, this curve is fit by just 3 bi-arcs. The initial guess for the ideal curve is formed using the i^{th} and j^{th} data point, where $i = 26..30$ and $j = 16..19$. The result for these different starting curves of the value of the minimization function is shown in Table 1. The minimization function in this case is the mean distance.
The constructed curve has the following properties:

Mean distance	0.061942
Root mean square of distances	0.086923
Maximal distance	0.304040
$\sqrt{\sum \lvert\theta_i/r\rvert}$	0.253664
$\sqrt{\sum \lvert\theta_i\rvert/r^2}$	0.181054

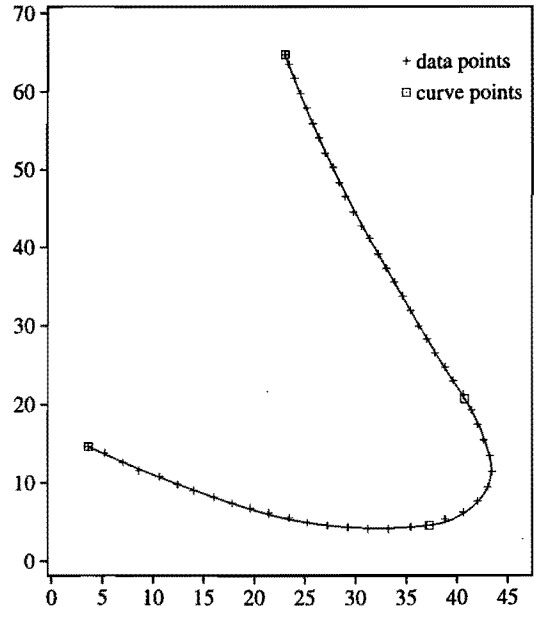

Figure 9: Resulting curve; minimization of the mean distance.

$j\downarrow^{i\rightarrow}$	26	27	28	29	30
16		0.104694	0.095740	0.064818	0.099097
17	0.080639	0.082372	0.068117	0.061942	0.066411
18	0.077762	0.074229	0.066217	0.063847	0.064660
19	0.111227	0.120433	0.117885	0.129475	

Table 1: Results for different starting curves, minimizing the mean distance.

Results by using other minimization functions are similar.

5.1 Discussion

Already mentioned in Section 5, many different minimization functions can be used. All these functions give slightly different resulting curves. By

interactive use of the program, the user will find the function and initial guesses that give the best results. Some criteria to decide are stated below:

- the maximum distance of a point to the curve
- the mean of the distances
- the root mean square of the distances
- the mean curvature in one or other definition.

Values of these criteria are standard output of the program in C++. Knowing these values using different minimization functions, the user can decide which function is optimal to use.

6 C++ program

The elementary C++ program `BIARC` needs input like the data-points, which type of minimization, how to construct a first guess, etc. Machines which use the DOS or WINDOWS operation system the input is controlled by a menu-bases program `BIARCMNU` which constructs an input file for the program `BIARC`. The program `BIARCMNU` is a combination of DOS calls and Turbo Pascal.
The user can use the program `BIARC` also as a stand-alone executable. On UNIX based machines this program should be compiled using a `makefile`; on DOS machines using the `BIARC.PRJ` file and Turbo C++. In the stand-alone case the user has to generate the input file. Furthermore the file containing the data points should be `punten.dat`. The contents of the input file must be:

line 1	:	value of the variable `keuzekrom`, (1, 2)
line 2	:	value of the variable `keuzeafst`, (1, 2, 3)
line 3	:	value of the variable `keuzecomb`, (1, 2)
line 4	:	filename, for example `curve`
line 5	:	level of tolerance, for example 0.01 or 0.001
line 6	:	weight of the curvature in the minimization function ($0 \leq w \leq 1$)
line 7	:	number of points through which the curve has to be constructed
line 8	:	value of the variable `keuzeprint`, (0, 1)
line 9	:	value of the variable `keuzeselect`, (0, 1, 2)
line 10-?	:	index/ x-co-ordinate y-co-ordinate of the points through which the curve has to be constructed

The meaning of the variables is:

`keuzekrom`	1 =	curvature is equal to the sum of the absolute values of the curvatures
	2 =	curvature is equal to the square root of the sum of the absolute of the angles divided by the square of the radius
`keuzeafst`	1 =	distance is equal to the mean the distance from the data-points to the curve
	2 =	distance is equal to the square root of the mean of the squares of the distance from the data-points to the curve
	3 =	distance is equal to the maximum distance
`keuzecomb`	1 =	minimizing distance
	2 =	minimizing $(1-w)\times$distance $+ w\times$ curvature
`keuzeprint`	0 =	generate a plot with only the curve and data points
	1 =	generate a plot with the curve, data points and the triangles which visualize the initial guess used in the algorithm `Amoeba`, see [7].
`keuzeselect`	0 =	K points are chosen at regular intervals (using a straight line)
	1 =	K points are chosen by giving the index of these points in the original data
	2 =	K points are chosen by giving their co-ordinates

When giving the index or co-ordinates in the input file, results will be achieved faster when these indexes or co-ordinates are given in a 'logical' order: the first point in the input file is the first point of the curve, the second point in the input file is the second point in the curve, etcetera.
The C++ program uses the data in the input file to calculate the best curve. The output is written in the following files, depending on the choice of the filename, say `curve`:

`curve.plt`	:	commands used by `GNUPLOT`
`curve.dat`	:	data-points
`curve.cv1`	:	points of the first BI-ARCS
`curve.cv2`	:	points of the second BI-ARCS
`curve.pnt`	:	starting/ending points of the BI-ARCS
`curve.res`	:	description BI-ARCS by their arc radius, arc degrees, length of arc and sign for the direction of the arc

When the program has found a solution, it calculates the distance and curvature for all implemented definitions. When the program `BIARCMNU` is used, this program again takes over and plots the result via a call to `GNUPLOT`. In the other case the user has to start `GNUPLOT` and give the following command `load 'curve.plt'` at the `GNUPLOT` prompt in order to see the result of the curve fitting.

7 Conclusion and recommendations

Bending tubes into smooth shapes using arcs of circles can be realized. Depending on the number and specification of the arcs used, the tube will be bent in a shape that has a better fit through given points.
A C++ program is written to calculate the best specifications for the arcs to be used, given the number of arcs to be used. `Amoeba` is the minimizing procedure from Numerical Recipes [7]. Other optimization-procedures could have been used. However, these would have been more difficult to implement and probably it would take more time to run the program. `Amoeba` is robust, although not fast. However, the results are satisfactory, therefore only this procedure is implemented.
To achieve a curve with the shape the user wants, he/she has to apply trial and error using the program in C++. Choices have to be made about several things:

- how many arcs have to be used
- how to choose the points through which an initial curve can be constructed
- what function has to be minimized
- what tolerance has to be taken into account

The curve that is found will depend on all these choices.
It should be noted that for this type of nonlinear problems no algorithm exists that returns a universally best result. Therefore, a trial and error strategy is always in order.

References

[1] D.S. Meek, D.J. Walton *Approximation of discrete data by G1 arc splines*, Computer Aided Design 1992, Vol. 24, No. 6, Pg 301-306.

[2] J. Hoschek, G. Seemann *Spherical splines*, Rairo-mathematical modeling and numerical analysis 1992, Vol. 26, No. 1, Pg 1-22.

[3] J. Hoschek *Circular Splines*, Computer Aided Design 1992, Vol. 24, No. 11, Pg 611-618.

[4] D.S. Meek, D.J. Walton *A note on planar minimax arc splines*, Computers and Graphics 1992, Vol. 16, No. 4, Pg 431-433.

[5] D.S. Meek, D.J. Walton *Approximating smooth planar curves by arc splines*, Journal of computational and applied mathematics 1995, Vol. 59, No. 2, Pg. 221-231.

[6] S.N. Yang, W.C. Du *Numerical methods for approximating digitized-curves by piecewise circular arcs*, Journal of computational and applied mathematics 1996, Vol. 66, No. 1-2, Pg. 557-569.

[7] W.H. Press, W.T. Vettering, S.A. Teukolsky, B.P. Flannery *Numerical Recipes in C*, 1992, 2nd Ed., Cambridge University Press.

[8] J.W.Wesselink *Variational Modeling of Curves and Surfaces*, 1996, PhD thesis, Eindhoven University of Technology.