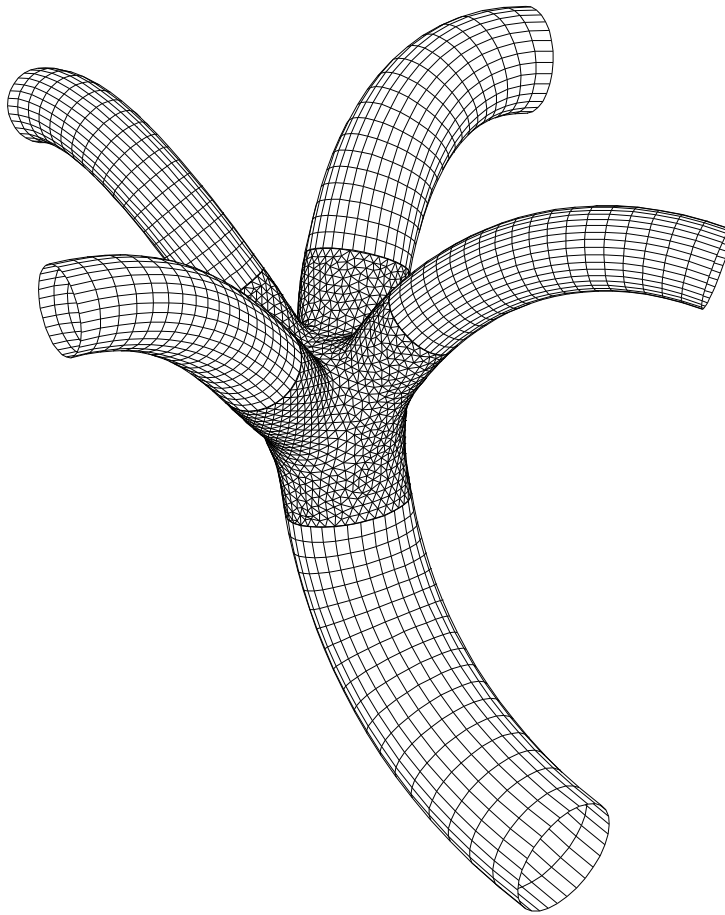


# Geometry and Algorithms for COMPUTER AIDED DESIGN



Erich Hartmann

Department of Mathematics  
Darmstadt University of Technology  
October 2003



# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>7</b>
1.1	Methods for DISPLAYING objects . . . . .	7
1.2	On the contents . . . . .	8
1.3	Software for displaying curves and surfaces . . . . .	9
1.4	On literature . . . . .	10
<b>2</b>	<b>TOOLS</b>	<b>11</b>
2.1	Structure of a CAD-program . . . . .	11
2.1.1	Global constants: The file "geoconst.pas" . . . . .	11
2.1.2	Globale types: The file "geotype.pas" . . . . .	11
2.1.3	Globale variables: The file "geovar.pas" . . . . .	11
2.1.4	Capability of the graphics software . . . . .	12
2.2	Functions on $\mathbb{R}$ , operations with vectors . . . . .	14
2.2.1	Functions on $\mathbb{R}$ . . . . .	14
2.2.2	Operations with vectors . . . . .	14
2.3	Routines to Analytic Geometry . . . . .	17
2.3.1	Polarangle and quadratic equation . . . . .	17
2.3.2	Intersection line–line, circle–line, circle–circle . . . . .	17
2.3.3	Equation of a plane . . . . .	18
2.3.4	Intersection line–plane . . . . .	18
2.3.5	Intersection of three planes . . . . .	18
2.3.6	Intersection of two planes . . . . .	19
2.3.7	$\xi$ - $\eta$ -coordinates of a point in a plane . . . . .	19
2.3.8	Coordinates with respect of a new 3D-coordinate system . . . . .	19
2.4	Numeric: GAUSS-elimination, NEWTON-iteration . . . . .	19
<b>3</b>	<b>PARALLEL/CENTRAL-PROJECTION</b>	<b>21</b>
3.1	Orthographic Projection . . . . .	21
3.1.1	The projection formulae . . . . .	21
3.1.2	Procedures for orthographic projection . . . . .	22
3.2	Central projection . . . . .	23
3.2.1	The projection formulae . . . . .	23
3.2.2	Procedures for central projection . . . . .	24
<b>4</b>	<b>CURVES</b>	<b>27</b>
4.1	Planar Curves . . . . .	27
4.1.1	Definition and Representations of Planar Curves . . . . .	27
4.1.2	Arc length and curvature of a planar curve . . . . .	28
4.1.3	Offset Curves . . . . .	30
4.1.4	The normalform of a planar curve . . . . .	30

4.2	Displaying Parametric Curves in $\mathbb{R}^2$ . . . . .	32
4.3	Displaying implicit curves . . . . .	33
4.3.1	Marching algorithm . . . . .	33
4.3.2	Raster algorithm . . . . .	35
4.4	Intersection of two planar curves . . . . .	37
4.4.1	Intersection of a parametric curve and an implicit curve . . . . .	38
4.4.2	Intersection of two implicit curves . . . . .	38
4.4.3	Intersection of two parametric curves . . . . .	39
4.5	Foot points on planar curves . . . . .	39
4.5.1	Foot point on a parametric curve, curve inversion . . . . .	39
4.5.2	Foot point on an implicit curve . . . . .	40
4.5.3	Stable first order foot point algorithms . . . . .	41
4.6	Bézier-curves . . . . .	42
4.7	Applications of the normalform . . . . .	43
4.7.1	Offset curves of implicit curves . . . . .	44
4.7.2	Bisector curves . . . . .	45
4.7.3	Numerical Parameterization of Curves . . . . .	46
4.8	Space Curves . . . . .	46
<b>5</b>	<b>SURFACES</b> . . . . .	<b>49</b>
5.1	Definition and Representations of Surfaces . . . . .	49
5.2	The First and Second Fundamental Forms of a Surface . . . . .	50
5.2.1	The first fundamental form, arc length . . . . .	50
5.2.2	The second fundamental form, curvature . . . . .	50
5.3	Offset surfaces . . . . .	52
5.4	Normalform of a surface . . . . .	52
5.4.1	Definition of the normalform . . . . .	52
5.4.2	On the first and second derivatives of the normalform of a surface . . . . .	53
5.4.3	$C^n$ -contact, $G^n$ -contact . . . . .	55
5.4.4	$G^2$ -continuity theorems . . . . .	56
5.4.5	The curvature of an intersection curve . . . . .	57
5.5	Normalform of an implicit surface . . . . .	58
5.6	Normalform of a parametric surface . . . . .	59
5.7	Foot point on a parametric surface, surface inversion . . . . .	60
5.8	Stable first order foot point algorithms for surfaces . . . . .	61
5.8.1	Foot point algorithm for parametric surfaces . . . . .	61
5.8.2	Foot point algorithm for implicit surfaces . . . . .	61
5.9	The normalform of a space curve . . . . .	62
5.9.1	Definition of the normalform . . . . .	62
5.9.2	Foot point algorithm and evaluation of the normalform of a space curve . . . . .	63
5.9.3	Determining foot points on an intersection curve . . . . .	64
5.10	Applications of normalforms . . . . .	64
5.10.1	Approximation of a set of intersecting surfaces . . . . .	64
5.10.2	Approximation of intersecting pipe surfaces . . . . .	65
5.10.3	Numerical parametrization of implicit surfaces . . . . .	65
<b>6</b>	<b>HIDDENLINE-ALGORITHM FOR NON CONVEX POLYHEDRONS</b> . . . . .	<b>67</b>
6.1	The Hiddenline-Algorithm . . . . .	67
6.2	Auxiliary procedures for the hiddenline algorithm . . . . .	71
6.2.1	The procedure <code>aux_polyhedron</code> . . . . .	71
6.2.2	The procedures <code>aux_quadrangle</code> , <code>aux_cylinder</code> , <code>aux_torus</code> and displaying a parametrized surface . . . . .	71

6.3	Intersection of faces, intersection of two polyhedrons . . . . .	74
6.3.1	Intersection of faces bounded by planar polygons in $\mathbb{R}^3$ . . . . .	74
6.3.2	Intersection of two polyhedrons . . . . .	76
6.4	IS line–polygonal patch, IS polygon–polyhedron . . . . .	78
6.4.1	Intersection of a line segment and a planar polygonal patch . . . . .	78
6.4.2	Intersection points of a polygon and a polyhedron . . . . .	79
<b>7</b>	<b>TRIANGULATION OF IMPLICIT SURFACES</b>	<b>81</b>
7.1	The triangulation algorithm (marching method) . . . . .	81
7.1.1	The idea of the algorithm . . . . .	81
7.1.2	The procedure <code>surfacepoint</code> . . . . .	83
7.1.3	The data structure . . . . .	83
7.1.4	Step S0 . . . . .	84
7.1.5	Step S1 . . . . .	84
7.1.6	Step S2 . . . . .	84
7.1.7	Step S3 . . . . .	85
7.2	Examples . . . . .	87
7.2.1	Sphere . . . . .	87
7.2.2	Cylinder . . . . .	88
7.2.3	Torus . . . . .	88
7.2.4	Set of two intersecting surfaces . . . . .	88
7.2.5	$G^2$ -continuous blending of three cylinders . . . . .	89
7.3	Sample program: <code>trisample.p</code> . . . . .	91
7.4	Ray tracing of triangulated surfaces with POV-Ray . . . . .	92
<b>8</b>	<b>INTERSECTIONS: CURVE – SURFACE, SURFACE – SURFACE</b>	<b>93</b>
8.1	Intersection Curve – Surface . . . . .	93
8.1.1	IS parametric curve – implicit surface . . . . .	93
8.1.2	IS implicit curve – implicit surface . . . . .	93
8.1.3	IS implicit curve – parametric surface . . . . .	94
8.1.4	IS parametric curve – parametric surface . . . . .	94
8.2	Intersection surface – surface . . . . .	94
8.2.1	IS of two implicit surfaces . . . . .	94
8.2.2	IS of an implicit and a parametric surface . . . . .	96
8.2.3	IS of two parametric surfaces . . . . .	96
8.3	Contour lines of surfaces . . . . .	98
8.3.1	Contour line of an implicit surface . . . . .	98
8.3.2	Contour line of a parametric surface . . . . .	99
8.3.3	Contour line of a quadric surface . . . . .	99
8.4	Hiddenline–algorithm for simple cases . . . . .	101
8.4.1	Fast normal–test . . . . .	101
8.4.2	Ray–test for implicit surfaces . . . . .	101
<b>9</b>	<b>CONICS AND QUADRICS</b>	<b>105</b>
9.1	Conics . . . . .	105
9.1.1	Ellipse . . . . .	105
9.1.2	Hyperbola . . . . .	106
9.1.3	Parabola . . . . .	107
9.1.4	Equation of a conic . . . . .	108
9.1.5	Conics in space . . . . .	108
9.2	Quadrics: ellipsoid, cylinder . . . . .	108
9.2.1	Ellipsoid . . . . .	108

9.2.2	Cylinder . . . . .	109
<b>10</b>	<b>SURFACES OF REVOLUTION, RULED SURFACES, ...</b>	<b>111</b>
10.1	Surfaces of revolution . . . . .	111
10.2	Ruled surfaces . . . . .	112
10.3	Developable connections between curves . . . . .	113
10.4	Coons patches . . . . .	115
10.5	Canal surface, embankment surface . . . . .	115
10.5.1	Canal surfaces . . . . .	117
10.5.2	Embankment surfaces . . . . .	118
<b>11</b>	<b>BLENDING CURVES AND SURFACES</b>	<b>119</b>
11.1	$G^n$ -blending . . . . .	119
11.2	Implicit $G^n$ -blending curves and surfaces . . . . .	119
11.2.1	Parabolic functional splines . . . . .	120
11.2.2	Elliptic functional splines . . . . .	124
11.3	Parametric $G^n$ -blending curves and surfaces . . . . .	127
11.3.1	$G^n$ -blending of parametric curves . . . . .	127
11.3.2	$G^n$ -blending of parametric surfaces . . . . .	129
11.4	Blending with rolling ball contact curves . . . . .	133
<b>12</b>	<b>PROPERTIES OF BÉZIER CURVES</b>	<b>137</b>
12.1	Properties of the Bernstein polynomials . . . . .	137
12.2	The Casteljau-algorithm . . . . .	137
12.3	Derivative of a Bézier curve . . . . .	139
12.4	Degree elevation of a Bézier curve . . . . .	139
12.5	Bézier spline curves . . . . .	140
12.5.1	Division of a Bézier curve . . . . .	140
12.5.2	Smoothness conditions . . . . .	141
12.6	Rational Bézier curves . . . . .	144
12.6.1	Rational curves and projective curves . . . . .	144
12.7	Rational Bézier curves . . . . .	144
12.8	Conics as rational Bézier curves . . . . .	145
<b>13</b>	<b>BÉZIER-SURFACES</b>	<b>147</b>
13.1	Tensor product Bézier surfaces . . . . .	147
13.2	The Casteljau algorithm . . . . .	148
13.3	Degree elevation . . . . .	148
13.4	Derivatives of a Bézier surface . . . . .	148
13.5	Bézier triangular surfaces . . . . .	149
<b>14</b>	<b>B-SPLINE CURVES</b>	<b>151</b>
14.1	The B-Spline base functions . . . . .	151
14.2	The de-Boor algorithm . . . . .	153
<b>A</b>	<b>THE REAL PROJECTIVE PLANE</b>	<b>155</b>
A.1	The real affine plane . . . . .	155
A.2	The real projective plane . . . . .	155
A.3	Isomorphism between the two models . . . . .	156
A.4	Collineations of the real projective plane . . . . .	157
A.5	Projective equivalency of the conics . . . . .	158

# Chapter 1

## INTRODUCTION

Essential tasks of Computer Aided Design are:

- 1) *designing* objects like parts of machines, car bodies, ... and
- 2) *displaying* designed or basic objects like spheres, cylinders, tori, ...

### 1.1 Methods for DISPLAYING objects

The aim of the displaying process is to generate a plane image of a 3 dimensional object. There are essentially two methods for the projection of objects onto a plane:

#### I) Parallel projection.

Parallel projection projects points of objects like curves or surfaces onto a plane by **parallel** rays. If the rays are orthogonal to the image plane the method is called *orthographic projection*. Other cases are called *skew parallel projections*. In case of a horizontal image plane one gets a *bird's eye-view*.

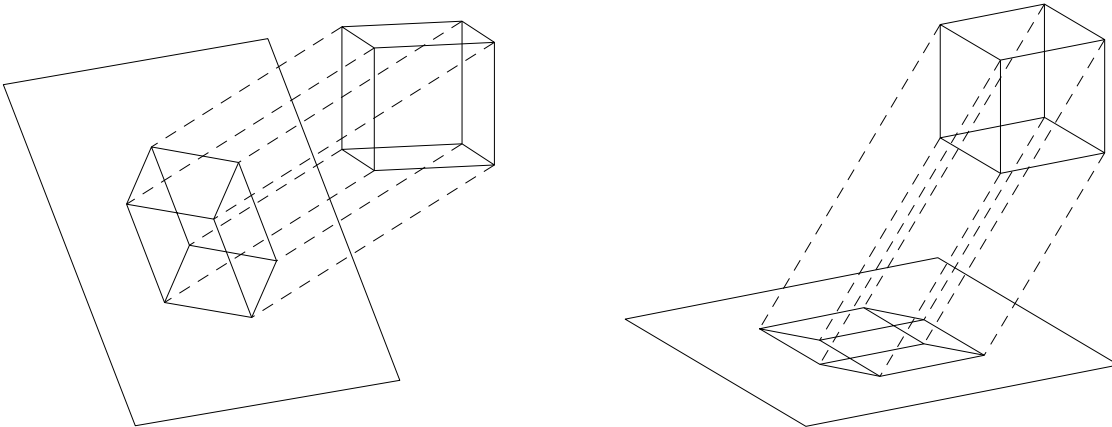


Figure 1.1: Orthogonal parallel projection and bird's eye-view of a cube

#### II) Central projection.

One gets a more realistic image, using projection rays which contain a common point Z, the centre or eye-point)

The advantage of a central projection is the existence of several additional parameters (coordinates of the centre and the distance to the image plane). The disadvantage: A central projection is no linear mapping and needs more CPU-time for processing.

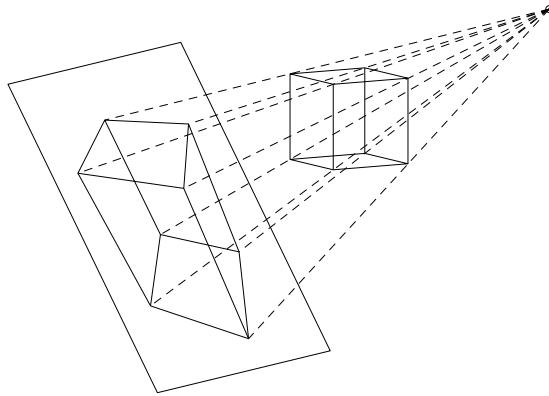


Figure 1.2: Central projection of a cube

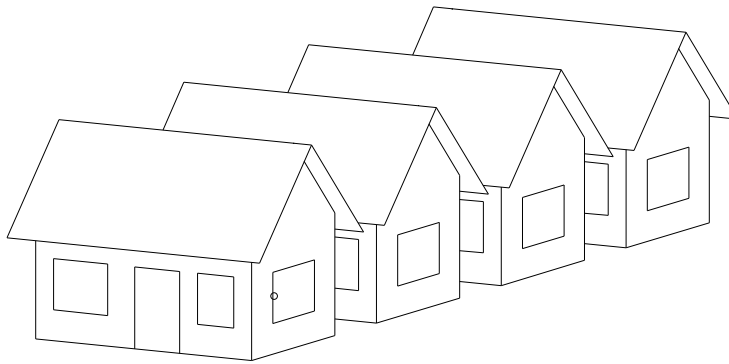


Figure 1.3: Parallel projection of a row of houses

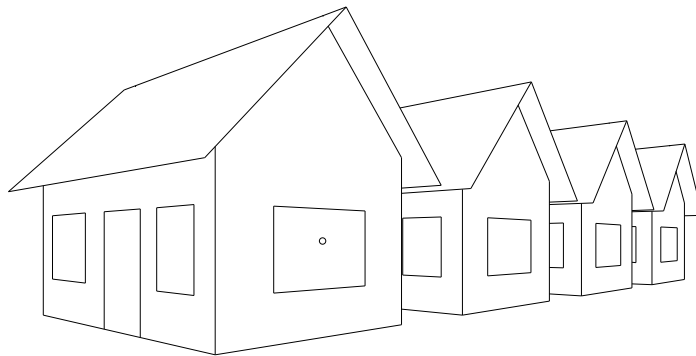


Figure 1.4: Central projection of a row of houses

The advantage of the parallel projection is its linearity and the invariance of proportions of line segments..

## 1.2 On the contents

Chapter 2: *Tools* defines the minimal needs which should be provided by the graphic system actually used. The data structure is introduced and its application demonstrated at a simple example



(ngon). The language for programming is PASCAL. But all procedures can be translated easily (by hand or automatically) into another language (for example “p2c” translates Pascal code to C-code automatically). Besides these system dependent procedures Chapter 1 contains a lot of procedures solving basic problems of Analytical Geometry.

The PASCAL code of the procedures is available on

<http://www.mathematik.tu-darmstadt.de/~ehartmann>

(see section on software below.)

Chapter 3 provides the routines for the orthographic and central projection.

Chapter 4 and Chapter 5 introduces basic results from Differential Geometry of curves and surfaces.

Chapter 6 and Chapter 7 gives methods for displaying parametric and implicit surfaces.

Chapter 8 deals with the intersection of curves and surfaces.

Chapter 9 describes the representation of conics and quadrics like ellipsoid and cylinder.

Chapter 10 introduces some special surfaces of practical use (surfaces of revolution, ruled surfaces, Coons patches, embankment surfaces, and canal surfaces).

Chapter 11 shows how to construct  $G^n$ -continuous transition surfaces (blending surfaces).

Chapter 12, Chapter 13 and Chapter 14 provide some material on Bézier curves and surfaces and B-spline curves.

The appendix explains some techniques and statements of projective planes necessary for rational Bézier curves.

## 1.3 Software for displaying curves and surfaces

**Installation** of the system on a LINUX-PC:

1. download of `cdg0egv.tgz` from

<http://www.mathematik.tu-darmstadt.de/~ehartmann>

2. `tar xvzf cdg0egv.tgz` generates the volume `cdg0egv` including subvolumes `examples`, `include`, `tools`, `units`.

FREE-PASCAL compiler has to be installed and `gv` is needed.

3. Go to subvolume `example` and run `make`. (The necessary units, the tool `pldv` and the example `ngon` will be generated.)

4. Compiling a program, for example `tori.h.p`: `make tori.h`

Run of the program: just type `tori.h`.

Continuing after drawing: `QUIT gv` .

Instant terminating after drawing: `CTRL C` .

5. `pldv` applied to a PLD-file (for example: `ngon.pld`, see `graph_on(..)` in Section 2.1.4)

a) shows the picture on the screen: `pldv ngon.pld -a`

b) generates a Postscript-file: `pldv ngon.pld -a -pps`

b) generates an EPS-file: `pldv ngon.pld -a -peps`

(All options of `pldv` are to be seen by just typing `pldv`.)

6. The informations on a) point coordinates and b) points in a face necessary for the hiddenline algorithm can be stored in an OFF-file by the procedure `write_nangles_to_offfile` contained in `parsurf.h.off.p` and used as input for the free software *GEOMVIEW* available at <http://geomview.org>.

**Remark:**

`pldv` in its original version was written by Andreas Görg.

## 1.4 On literature

Recommended **Books** on:

- *Differential Geometry*:
  - do Carmo, P.M. (1976): *Differential Geometry of Curves and Surfaces*, Prentice-Hall, Inc. Englewood Cliffs, New Jersey.
  - Lipschutz, M.M. (1969): *Theory and Problems of Differential Geometry*, Schaum's Outline Series, McGraw-Hill, New York.
- *Computer Aided Geometric Design*:
  - Farin, G. (1990): *Curves and Surfaces for Computer-Aided Geometric Design, A Practical Guide*, Academic Press
  - Hoschek, J., Lasser, D. (1989): *Fundamentals of Computer Aided Geometric Design* AK Peters.

### Remark:

Some text and figures of this lecture notes are taken from

- Hartmann, E. (1998): *A Marching Method for the Triangulation of Surfaces*. The Visual Computer 14, 95–108.
- Hartmann, E. (1998): *The Normalform of a Planar Curve and its Application to Curve Design*. in Mathematical Methods for Curves and Surfaces II, M. Daehlen, T. Lyche, L. Schumaker eds., Vanderbilt Univ. Press, Nashville.
- Hartmann, E. (1999): *On the curvature of curves and surfaces defined by normalforms*. Comp. Aided Geom. Des. 16, 355–376.
- Hartmann, E. (2000): *Numerical Parameterization of Curves and Surfaces*. Comp. Aided Geom. Des. 17, 251–266.
- Hartmann, E. (2001): *Implicit  $G^n$ -blending of vertices*. Comp. Aided Geom. Des. 18, 267–285.
- Hartmann, E. (2001):  *$G^n$ -continuous connections between normal ringed surfaces*. Comp. Aided Geom. Des. 18, 751–770.
- Hartmann, E. (2001): *Parametric  $G^n$ -blending curves and surfaces*. The Visual Computer 17, 1–13.
- Hartmann, E. (2001): *The normalform of a space curve and its application to surface design*. The Visual Computer 17, 445–456.

# Chapter 2

## TOOLS

Essential tools for solving CAD problems are basic operations for vectors (sum, scalar product, ...) and routines for simple tasks like the intersection of a line and a circle. Before introducing such basic procedures we discuss the minimal demand on the graphics system.

### 2.1 Structure of a CAD-program

First of all we define the global constants, types and variables used in any program. Essential are the types `vt2d`, `vt3d`, `vts2d`, `vts3d` which define vectors and arrays of vectors.

#### 2.1.1 Global constants: The file "geoconst.pas"

The file "geoconst.pas" contains the constant `array_size` which will be used in 2.1.2, the reals  $\pi$ ,  $2\pi$ ,  $\frac{\pi}{2}$  and `eps1`, ... , `eps8` used for error bounds . The constants `black`,... described colors.

```
array_size= 1000; {...20000 for hiddenline-alg.}
pi= 3.14159265358;      pi2= 6.2831853;      pih= 1.5707963;
eps1=0.1;      eps2=0.01;      eps3=0.001;      eps4=0.0001;
eps5=0.00001;  eps6=0.000001;  eps7=0.0000001; eps8=0.00000001;
default=-1; black=0; blue=1; green=2; cyan=3; red=4; magenta=5; brown=6;
lightgray=7; darkgray=8; lightblue=9; lightgreen=10; lightcyan=11;
lightred=12; lightmagenta=13; yellow=14; white=15;
```

#### 2.1.2 Globale types: The file "geotype.pas"

```
r_array = array[0..array_size] of real;
i_array = array[0..array_size] of integer;
b_array = array[0..array_size] of boolean;
vt2d    = record x,y: real; end;
vt3d    = record x,y,z: real; end;
vts2d   = array[0..array_size] of vt2d;
vts3d   = array[0..array_size] of vt3d;
matrix3d= array[1..3,1..3] of real;
```

#### 2.1.3 Globale variables: The file "geovar.pas"

```
null2d:vt2d; null3d:vt3d;      {Nullvectors}
{**for area_2d and curve2d:}
origin2d:vt2d;
```

```

{**for parallel and central projection:}
  u_angle,v_angle,          {projection angles}
  rad_u,rad_v,              {rad(u), rad(v)}
  sin_u,cos_u,sin_v,cos_v:real; {sin(u),cos(u), ...}
  e1vt,e2vt,n0vt:vt3d;      {base vectors and}
                              {normal vector of the image plane}

{**for Central-Projection:}
  mainpt,                   {mainpoint}
  centre:vt3d;              {centre}
  distance:real;            {distance mainpoint - centre}

```

### 2.1.4 Capability of the graphics software

In order to ease the transportation of the system to another one all procedures for drawing rely on the following 9 machine dependent procedures. For a LINUX-PC the realization is contained in package **cdg0egv** (see Introduction).

1. **graph\_on(ipl)**, *ipl:integer*, starts the graphics-software and sets the vectors **null2d**, **null3d** to default;  
*ipl = 0* : Display on screen, only,  
*ipl ≠ 0* : any application of **draw\_area(...)** (see below) generates a file **name.pld** that contains all drawing devices of the actual drawing, including color and line width.  
**pldv name.pld** shows the drawing stored in **name.pld** on the screen. **pldv name.pld -pps** generates a *POSTSCRIPT*-file which can be printed.  
**pldv name.pld -peps** generates an *ENCAPSULATED POSTSCRIPT*-file which can be included into a *TEX*-file.
2. **draw\_area(width,height,x0,y0,scalefactor)** clears the screen and defines a drawing area provided with an orthogonal coordinate system.  
**origin2d = (x0,y0)** is a global variable.  
 All length are assumed to be in mm (reals !)  
 1 mm should be 1 mm on the screen and on the paper, if **scalefactor=1**.  
**scalefactor= s** scales the drawing by factor *s*.

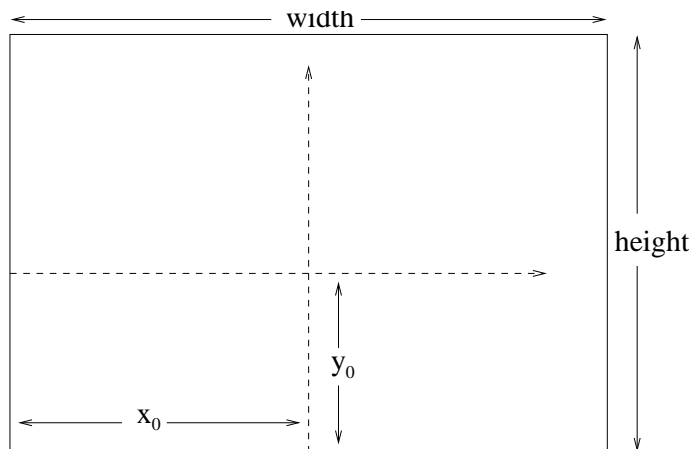


Figure 2.1: Origin of the coordinate system for the drawing area

3. **draw\_end** closes a drawing. If there is need for an additional drawing **draw\_area** has to be applied again
4. **graph\_off** closes the graphic software definitely.

#### DRAWING DEVICES:

5. **pointc2d(x,y,style)**,  $x,y:\text{real}$ ;  $\text{style}:\text{integer}$ ,  
marks the point(x,y) by o if  $\text{style} = 0$ , + if  $\text{style} = 1$ , ... .  
For  $\text{style} = 10$  oder 50 oder 100 the marks are smaller circles (scaled)  
**point2d(p,style)**,  $p:\text{vt2d}$ ;  $\text{style}:\text{integer}$ ,  
like **pointc2d** using the type **vt2d** for the point.
6. **linec2d(x1,y1,x2,y2,style)**,  $x1,y1,x2,y2:\text{real}$ ;  $\text{style}:\text{integer}$ ,  
draws the line segment  $(x1,y1)(x2,y2)$  as:  
————, if  $\text{style} = 0$ , - - - - -, if  $\text{style} = 1$ , - · - · - · -, if  $\text{style} = 2$ , .....  
The following devices are defined by **linec2d**.
  - (a) **line2d(p1,p2,style)**,  $p1,p2:\text{vt2d}$ ;  $\text{style}:\text{integer}$ ,  
as **linec2d** using type **vt2d** for the bounding points.
  - (b) **arrowc2d(x1,y1,x2,y2,style)**,  $x1,y1,x2,y2:\text{real}$ ;  $\text{style}:\text{integer}$   
draws an arrow from  $(x1,y1)$  to  $(y2,y2)$ .
  - (c) **arrow2d(p1,p2,style)**,  $p1,p2:\text{vt2d}$ ;  $\text{style}:\text{integer}$   
draws an arrow from  $p_1$  to  $p_2$ .
  - (d) **curve2d(p,n1,n2,style)**,  $p:\text{vts2d}$ ;  $\text{style}:\text{integer}$ ,  
draws a polyline with vertices  $p_{n1}, \dots, p_{n2}$ .

All length and coordinates must be given in mm !

7. **new\_color(color)**,  $\text{color}:\text{integer}$ ,  
sets a new color. For example:  $\text{color} = \text{red}$ .  $\text{color} = \text{default}$  is equivalent to black.
8. **new\_linewidth(factor)**,  $\text{factor}:\text{real}$ ,  
changes the line width.  $\text{factor}=1$  means normal line width.

#### STRUCTURE of a program:

After putting all global constants, types, variables and procedures into a *unit* **geograph** a program is of the following simple form:

```

program name;
uses geograph;
const ...
type ...
var ...:vts2d;
    ...:integer;
    ...:real;
    ...
    {$i procs.pas}      {additional procedures}
{*****}
begin {main program}
graph_on(...);
...
{drawing:}

```

```

draw_area(...);
...
...
draw_end;
....
graph_off;
end.

```

## 2.2 Functions on $\mathbb{R}$ , operations with vectors

Now we provide PASCAL-functions and procedures for some often used real functions and operations. Because of the simplicity of these functions and procedures we give only heads. The complete source is contained in the file `proc_ag.pas` (see volume `include`).

### 2.2.1 Functions on $\mathbb{R}$

1.  $r \rightarrow \text{sign}(a)$  ( $-1$ , if  $a < 0$ , else  $+1$ )  
`function sign(a:real):integer;`
2.  $a, b \rightarrow \max\{a, b\}$  (maximum of  $a, b$ )  
 $a, b \rightarrow \min\{a, b\}$  (minimum of  $a, b$ )  
`function max(a,b:real):real; function min(a,b:real):real;`

### 2.2.2 Operations with vectors

1.  $x, y \rightarrow \mathbf{v} = (x, y)$ ,  
 $x, y, z \rightarrow \mathbf{v} = (x, y, z)$   
`procedure put2d(x,y:real; var v:vt2d);`  
`procedure put3d(x,y,z:real; var v:vt3d);`  
 $\mathbf{v} = (x, y, z) \rightarrow x, y, z$   
`procedure get3d(v:vt3d; var x,y,z:real);`
2.  $r, \mathbf{v} \rightarrow r\mathbf{v}$  (scaling)  
`procedure scale2d(r:real; v:vt2d; var vs:vt2d);`  
`procedure scale3d(r:real; v:vt3d; var vs:vt3d);`  
 $r_1, r_2, (x, y) \rightarrow (r_1x, r_2y)$  bzw.  
 $r_1, r_2, r_3, (x, y, z) \rightarrow (r_1x, r_2y, r_3z)$  (scaling single coordinates)  
`procedure scaleco2d(r1,r2:real; v:vt2d; var vs:vt2d);`  
`procedure scaleco3d(r1,r2,r3:real; v:vt3d; var vs:vt3d);`
3.  $\mathbf{v}_1, \mathbf{v}_2 \rightarrow \mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2$  (sum of two vectors)  
`procedure sum2d(v1,v2:vt2d; var vs:vt2d);`  
`procedure sum3d(v1,v2:vt3d; var vs:vt3d);`  
 $\mathbf{v}_1, \mathbf{v}_2 \rightarrow \mathbf{v} = \mathbf{v}_1 - \mathbf{v}_2$  (difference of two vectors)  
`procedure diff2d(v1,v2:vt2d; var vd:vt2d);`  
`procedure diff3d(v1,v2:vt3d; var vd:vt3d);`
4.  $r_1, \mathbf{v}_1, r_2, \mathbf{v}_2 \rightarrow \mathbf{v} = r_1\mathbf{v}_1 + r_2\mathbf{v}_2$  (linear combination of vectors)  
`procedure lcomb2vt2d(r1:real; v1:vt2d; r2:real; v2:vt2d; var vlc:vt2d);`  
`procedure lcomb2vt3d(r1:real; v1:vt3d; r2:real; v2:vt3d; var vlc:vt3d);`  
and analogously linear combinations of 3 and 4 vectors:  
`lcomb3vt2d(r1,v1, r2,v2, r3,v3, vlc);`  
`lcomb3vt3d(r1,v1, r2,v2, r3,v3, vlc);`

- ```

lcomb4vt2d(r1,v1, r2,v2, r3,v3, r4,v4, vlc);
lcomb4vt3d(r1,v1, r2,v2, r3,v3, r4,v4, vlc);

```
5.  $\mathbf{v} = (x, y) \rightarrow |x| + |y|$  bzw.  $\mathbf{v} = (x, y, z) \rightarrow |x| + |y| + |z|$   
`function abs2d(v:vt2d):real;      function abs3d(v:vt3d):real;`
  6.  $\mathbf{v} = (x, y) \rightarrow \|\mathbf{v}\| = \sqrt{x^2 + y^2}$  bzw.  
 $\mathbf{v} = (x, y, z) \rightarrow \|\mathbf{v}\| = \sqrt{x^2 + y^2 + z^2}$   
`function length2d(v:vt2d):real;      function length3d(v:vt3d):real;`
  7.  $\mathbf{v} \rightarrow \mathbf{v}/\|\mathbf{v}\|$   
`procedure normalize2d(var v:vt2d);    procedure normalize3d(var v:vt3d);`
  8.  $\mathbf{p}, \mathbf{q} \rightarrow \|\mathbf{p} - \mathbf{q}\|$      $\mathbf{p}, \mathbf{q} \rightarrow \|\mathbf{p} - \mathbf{q}\|^2$   
`function distance2d(p,q:vt2d):real;`  
`function distance3d(p,q:vt3d):real;`  
`function distance2d.square(p,q:vt2d):real;`  
`function distance3d.square(p,q:vt3d):real;`
  9.  $\mathbf{v}_1, \mathbf{v}_2 \rightarrow \mathbf{v}_1 \cdot \mathbf{v}_2$     (scalar product)  
`function scalarp2d(v1,v2:vt2d):real;`  
`function scalarp3d(v1,v2:vt3d):real;`
  10.  $\mathbf{v}_1, \mathbf{v}_2 \rightarrow \mathbf{v}_1 \times \mathbf{v}_2$  (vector product)  
`procedure vectorp(v1,v2:vt3d; var vp:vt3d);`
  11.  $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \rightarrow |\mathbf{v}_1 \mathbf{v}_2 \mathbf{v}_3|$   
 $(\mathbf{v}_1 \cdot (\mathbf{v}_2 \times \mathbf{v}_3), 3 \times 3\text{-determinant})$   
`function determ3d(v1,v2,v3:vt3d):real;`
  12.  $\cos \varphi, \sin \varphi, \mathbf{p} = (x, y) \rightarrow \mathbf{p}_r = (x \cos \varphi - y \sin \varphi, x \sin \varphi + y \cos \varphi)$   
(rotation around the origin, angle: $\varphi$ )  
`procedure rotor2d(cos_rota,sin_rota:real; p:vt2d; var pr:vt2d);`
  13.  $\cos \varphi, \sin \varphi, \mathbf{p}_0, \mathbf{p} \rightarrow \mathbf{p}_r$   
(rotation around point  $\mathbf{p}_0$ , angle: $\varphi$ )  
`rotp02d(cos_rota,sin_rota,p0,p, pr);`
  14.  $\cos \varphi, \sin \varphi, \mathbf{p} \rightarrow \mathbf{p}_r$   
(rotation around x-axis, y-axis, z-axis respect. )  
`procedure rotorx(cos_rota,sin_rota,p, pr);`  
`procedure rotory(cos_rota,sin_rota,p, pr);`  
`procedure rotorz(cos_rota,sin_rota,p, pr);`
  15.  $\cos \varphi, \sin \varphi, \mathbf{p}_0, \mathbf{p} \rightarrow \mathbf{p}_r$   
(rotation around an axis through  $\mathbf{p}_0$  parallel to a coordinate axis in  $\mathbb{R}^3$ )  
`procedure rotp0x(cos_rota,sin_rota,p0,p, pr);`  
`procedure rotp0y(cos_rota,sin_rota,p0,p, pr);`  
`procedure rotp0z(cos_rota,sin_rota,p0,p, pr);`
  16. Change of numbers and vectors resp.:  
 $a \leftrightarrow b$     bzw.     $\mathbf{v}_1 \leftrightarrow \mathbf{v}_2$   
`procedure change1d(var a,b:real);`  
`procedure change2d(var v1,v2:vt2d);`  
`procedure change3d(var v1,v2:vt3d);`

**Example 2.1** The following program draws a regular  $n$ -gon and (on demand) all possible edges. The points of the  $n$ -gon are on a circle with midpoint  $(0,0)$  and radius  $r$ . The coordinates of the points are  $x_i = r \cos(i\Delta\varphi)$ ,  $y_i = r \sin(i\Delta\varphi)$  mit  $\Delta\varphi = 2\pi/n$ ,  $i = 0, \dots, n-1$ . (For the program: Point  $P_{i+1}$  is generated by rotating  $P_i$ ).

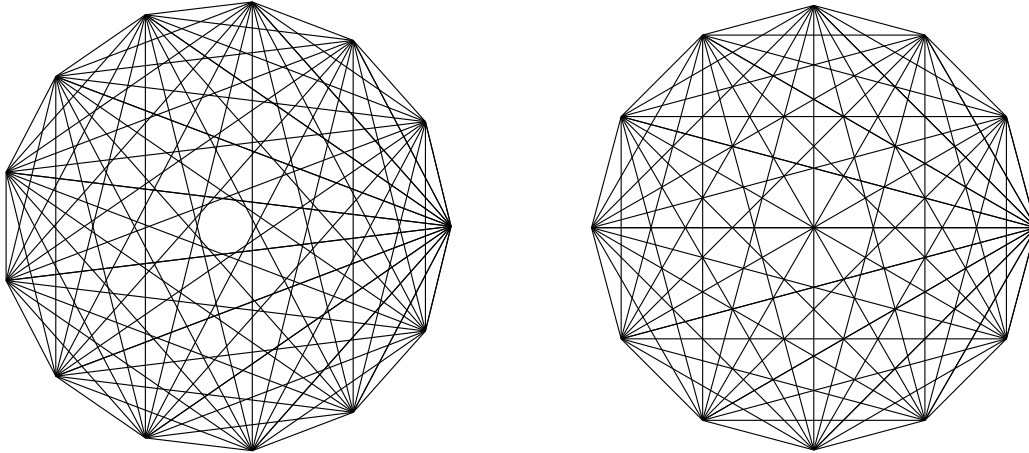


Figure 2.2:  $n$ -gon with all possible edges (Example 2.1)

```

{*****}
{*** regular n-gon ***}
{*****}
program ngon;
uses geograph;
var p : vts2d;
    n,icon,i,j,iand: integer;
    r,dw,cdw,sdw: real;
{*****}
begin {main program}
  graph_on(0);
  repeat
    writeln('*** n-gon ***');
    writeln('n ? radius r of the corresponding circle ?');      readln(n,r);
    writeln('Connect any pair of points ? (yes=1)');  readln(icon);
{coordinates of the points:}
    put2d(r,0, p[0]); dw:= pi2/n; cdw:= cos(dw); sdw:= sin(dw);
    for i:= 0 to n-1 do rotor2d(cdw,sdw,p[i], p[i+1]);
    draw_area(2*r+20,2*r+20,r+10,r+10,1);
{drawing:}  new_color(yellow);
    if icon=1 then
      for i:= 0 to n-1 do
        for j:= i+1 to n do
          line2d(p[i],p[j],0)
    else
      curve2d(p,0,n,0);
    draw_end;
    writeln('Additional drawing? (yes:1, no:0)');  readln(iand);
  until iand=0;
  graph_off;
end.

```



**Problem 2.1** Write a program that generates the following pictures.

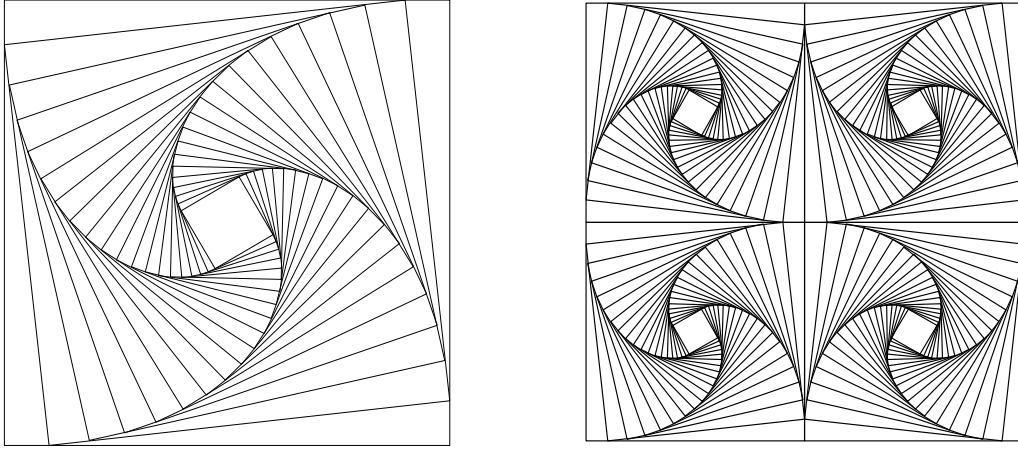


Figure 2.3: To problem 2.1

## 2.3 Routines to Analytic Geometry

### 2.3.1 Polarangle and quadratic equation

a) For the change of rectangular coordinates of  $\mathbb{R}^2$  in polar coordinates we use `polar_angle` which determines the polarangle of point  $(x,y)$ :

```
function polar_angle(x,y:real):real;
```

b) Real solutions of a quadratic equation  $ax^2 + bx + c = 0$ :

(The solutions are ordered by size. `ns` is the number of real solutions)

```
procedure equation_degree2(a,b,c:real; var x1,x2:real; var ns:integer);
```

The sources of these and the following procedures are contained in file `proc_ag.pas`.

### 2.3.2 Intersection line–line, circle–line, circle–circle

a) Intersection line–line :

Procedure `is_line_line` uses CRAMERs rule.

```
procedure is_line_line(a1,b1,c1, a2,b2,c2:real; var xs,ys:real; var nis:integer);
```

```
{IS-point (xs,ys) (nis=1) of the lines a1*x+b1*y=c1, a2*x+b2*y=c2.
```

```
 If the lines are parallel: nis<>1.}
```

b) Intersection circle–line:

circle :  $(x - x_m)^2 + (y - y_m)^2 = r^2$ ,  $r > 0$ .

line :  $ax + by = c$ ,  $(a, b) \neq (0, 0)$

With substitution  $\xi = x - x_m$ ,  $\eta = y - y_m$  we get

$a\xi + b\eta = c'$  with  $c' = c - ax_m - by_m$  and

$\xi^2 + \eta^2 = r^2$ .

If  $r^2(a^2 + b^2) - c'^2 > 0$  we get the solutions

$$\xi_{1/2} = (ac' \pm b\sqrt{r^2(a^2 + b^2) - c'^2}) / (a^2 + b^2), \quad \eta_{1/2} = (bc' \mp a\sqrt{r^2(a^2 + b^2) - c'^2}) / (a^2 + b^2)$$

and hence

$$x_{1/2} = x_m + \xi_{1/2}, \quad y_{1/2} = y_m + \eta_{1/2}.$$

```

procedure is_circle_line(xm,ym,r, a,b,c:real; var x1,y1,x2,y2:real; var nis:integer);
{Intersection circle-line:  $\text{sqr}(x-xm)+\text{sqr}(y-ym)=r*r$ ,  $a*x+b*y=c$ ,
  IS-points: (x1,y1),(x2,y2).  $x1 \leq x2$ , nis: number of intersection points.}

```

Intersection of the unitcircle ( $x^2 + y^2 = 1$ ) with a line:

```

procedure is_unitcircle_line(a,b,c:real; var x1,y1,x2,y2:real; var nis:integer);

```

In both procedures we have  $x_1 \leq x_2$ .

c) Intersection circle-circle :

1. circle :  $(x - x_1)^2 + (y - y_1)^2 = r_1^2$ ,  $r_1 > 0$ ,
2. circle :  $(x - x_2)^2 + (y - y_2)^2 = r_2^2$ ,  $r_2 > 0$ ,  $(x_1, y_1) \neq (x_2, y_2)$ .

This system is equivalent to:

$$(x - x_1)^2 + (y - y_1)^2 = r_1^2, \quad ax + by = c \quad \text{with}$$

$$a = 2(x_2 - x_1), \quad b = 2(y_2 - y_1) \quad \text{and} \quad c = r_2^2 - x_1^2 - y_1^2 - r_2^2 + x_2^2 + y_2^2.$$

That means: The intersection points of the circles and the intersection points of the first circle and the line  $ax + by = c$  are the same.

```

procedure is_circle_circle(xm1,ym1,r1,xm2,ym2,r2:real; var x1,y1,x2,y2:real; var nis:integer);
{Intersection circle--circle.  $x1 \leq x2$ . nis: number of intersection points.}

```

### 2.3.3 Equation of a plane

Given: 3 points  $P_i : \mathbf{p}_i$ ,  $i = 1, 2, 3$ .

Sought: equation  $\mathbf{n} \cdot \mathbf{x} = d$ , that means the normal vector  $\mathbf{n}$  and  $d$ .

Solution:  $\mathbf{n} = (\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)$  and  $d = \mathbf{n} \cdot \mathbf{p}_1$ .

plane\_equ determines  $\mathbf{n}$  and  $d$ . The boolean variable error is set true, in case of  $\mathbf{n} \approx \mathbf{0}$ .

```

procedure plane_equ(p1,p2,p3:vt3d; var nv:vt3d; var d:real; var error:boolean);
{Determines the equation  $\text{nv}*\mathbf{x}=d$  of the plane containing p1,p2,p3.
  error=true: no plane.}

```

### 2.3.4 Intersection line-plane

Given: line  $\mathbf{x}(t) = \mathbf{p} + t\mathbf{r}$ , plane  $\mathbf{n} \cdot \mathbf{x} = d$ .

Sought: Intersection point  $\mathbf{p}_{is}$ .

Solution:  $\mathbf{p}_{is} = \mathbf{p} - ((\mathbf{n} \cdot \mathbf{p} - d)/\mathbf{n} \cdot \mathbf{r})\mathbf{r}$ .

is\_line\_plane determines the intersection point, if it exists.

```

procedure is_line_plane(p,rv,nv:vt3d; d:real; var pis:vt3d; var nis:integer);
{Intersection line-plane. Line: point p, direction r. plane:  $\text{nv}*\mathbf{x} = d$ .
  nis=0: no IS-point ,nis=1: IS--point exists, nis=2: line in plane.}

```

### 2.3.5 Intersection of three planes

Given: Three planes  $\varepsilon_i : \mathbf{n}_i \cdot \mathbf{x} = d_i$ ,  $i = 1, 2, 3$ ,  $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3$  linear independent.

Sought: Intersection point  $\mathbf{p}_{is} : \varepsilon_1 \cap \varepsilon_2 \cap \varepsilon_3$ .

From  $\mathbf{p}_{is} = \xi(\mathbf{n}_2 \times \mathbf{n}_3) + \eta(\mathbf{n}_3 \times \mathbf{n}_1) + \zeta(\mathbf{n}_1 \times \mathbf{n}_2)$

we get

$$\mathbf{p}_{is} = (d_1(\mathbf{n}_2 \times \mathbf{n}_3) + d_2(\mathbf{n}_3 \times \mathbf{n}_1) + d_3(\mathbf{n}_1 \times \mathbf{n}_2))/\mathbf{n}_1 \cdot (\mathbf{n}_2 \times \mathbf{n}_3).$$

(If the normals are not linearly independent, there exists an intersection line or two planes are parallel.)

is\_3\_planes determines the intersection. error= true, if the intersection consists not of exactly one point.

```

procedure is_3_planes(nv1:vt3d; d1:real; nv2:vt3d; d2:real; nv3:vt3d; d3:real;
  var pis:vt3d; var error:boolean);
{Intersection of the planes  $\text{nv1}*\mathbf{x}=d1$ ,  $\text{nv2}*\mathbf{x}=d2$ ,  $\text{nv3}*\mathbf{x}=d3$ .
  error= true if not exactly ONE intersection point.}

```

### 2.3.6 Intersection of two planes

Given: two planes  $\varepsilon_i : \mathbf{n}_i \cdot \mathbf{x} = d_i, \quad i = 1, 2, \quad \mathbf{n}_1, \mathbf{n}_2$  linear independent.

Sought:  $\varepsilon_1 \cap \varepsilon_2 : \mathbf{x} = \mathbf{p} + t\mathbf{r}$ .

The direction of the intersection line is  $\mathbf{r} = \mathbf{n}_1 \times \mathbf{n}_2$ . We get a point  $P : \mathbf{p}$  of the intersection line by intersecting  $\varepsilon_1, \varepsilon_2$  with the plane  $\varepsilon_3 : \mathbf{x} = s_1\mathbf{n}_1 + s_2\mathbf{n}_2$ .

$$P : \mathbf{p} = \frac{d_1\mathbf{n}_2^2 - d_2(\mathbf{n}_1 \cdot \mathbf{n}_2)}{\mathbf{n}_1^2\mathbf{n}_2^2 - (\mathbf{n}_1 \cdot \mathbf{n}_2)^2}\mathbf{n}_1 + \frac{d_2\mathbf{n}_1^2 - d_1(\mathbf{n}_1 \cdot \mathbf{n}_2)}{\mathbf{n}_1^2\mathbf{n}_2^2 - (\mathbf{n}_1 \cdot \mathbf{n}_2)^2}\mathbf{n}_2$$

`is_plane_plane` determines the direction  $\mathbf{r}$  and a point of the intersection line. `error=true`, if the planes are parallel.

```
procedure is_plane_plane(nv1:vt3d; d1:real; nv2:vt3d; d2:real;
                        var p,rv:vt3d; var error:boolean);
{Intersection of the planes nv1*x=d1, nv2*x=d2. IS--line: x = p + t*rv .
 error= true: Intersection is no line.}
```

### 2.3.7 $\xi$ - $\eta$ -coordinates of a point in a plane

Given: plane  $\varepsilon : \mathbf{x} = \mathbf{p}_0 + \xi\mathbf{v}_1 + \eta\mathbf{v}_2$  and point  $P : \mathbf{p}$  in  $\varepsilon$ .

Sought:  $\xi, \eta$  such that  $\mathbf{p} = \mathbf{p}_0 + \xi\mathbf{v}_1 + \eta\mathbf{v}_2$ .

By scalar multiplication of  $\mathbf{p}$  by  $\mathbf{v}_1, \mathbf{v}_2$  we get the linear system:

$$(\mathbf{p} - \mathbf{p}_0) \cdot \mathbf{v}_1 = \xi\mathbf{v}_1^2 + \eta\mathbf{v}_1 \cdot \mathbf{v}_2, \quad (\mathbf{p} - \mathbf{p}_0) \cdot \mathbf{v}_2 = \xi\mathbf{v}_1 \cdot \mathbf{v}_2 + \eta\mathbf{v}_2^2,$$

`ptco_plane3d` determines  $\xi, \eta$  using CRAMERs rule. `error=true`, if the determinant of the linear system is  $\approx 0$ . (If  $P$  is not in  $\varepsilon$  and  $\xi, \eta$  is the solution of the linear system above, point  $P'$ :  $\mathbf{p}' = \mathbf{p}_0 + \xi\mathbf{v}_1 + \eta\mathbf{v}_2$  is the footpoint of the perpendicular line from  $P$  to plane  $\varepsilon$ .)

```
procedure ptco_plane3d(p0,v1,v2,p:vt3d; var xi,eta:real; var error:boolean);
{v1,v2 linear independent, p-p0,v1,v2 are linear dependent.
 We get xi,eta such that p = p0 + xi*v1 + eta*v2.}
```

### 2.3.8 Coordinates with respect of a new 3D-coordinate system

Given: new origin  $B_0 : \mathbf{b}_0$ , new basis  $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$  and point  $P : \mathbf{p}$ .

Sought:  $\xi, \eta, \zeta$  such that  $\mathbf{p} = \mathbf{b}_0 + \xi\mathbf{b}_1 + \eta\mathbf{b}_2 + \zeta\mathbf{b}_3$ .

From CRAMERs rule we get:

$$\xi = \det(\mathbf{p} - \mathbf{b}_0, \mathbf{b}_2, \mathbf{b}_3) / \det(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$$

$$\eta = \det(\mathbf{b}_1, \mathbf{p} - \mathbf{b}_0, \mathbf{b}_3) / \det(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$$

$$\zeta = \det(\mathbf{b}_1, \mathbf{b}_2, \mathbf{p} - \mathbf{b}_0) / \det(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$$

```
procedure newcoordinates3d(p,b0,b1,b2,b3: vt3d; var pnw: vt3d);
{Determines the coordinates of p with respect of basis b1,b2,b3 and origin b0.}
```

## 2.4 Numeric: GAUSS-elimination, NEWTON-iteration

For solving a larger **system of linear equations**  $A\mathbf{x} = \mathbf{b}$  one uses an appropriate version of GAUSS-elimination. A description of the method and a suitable PASCAL-code is contained in *Meyberg, Vachenaer: Höhere Mathematik 1, Springer-Verlag, 1999.*

For solving a **system of non linear equations** one usually apply the NEWTON-iteration. Here a short description of the method:

Given: Function  $\mathbf{F} : D \rightarrow \mathbb{R}^n, D \subseteq \mathbb{R}^n$ , and a starting point  $\mathbf{x}_0$  for the iteration.

Wanted: A point  $\mathbf{x}^*$  in the vicinity of  $\mathbf{x}_0$  with  $\mathbf{F}(\mathbf{x}^*) = 0$ .

**The algorithm:**

For  $\nu = 0, 1, 2, \dots$

(1) solve the linear system

$$\mathbf{F}'(\mathbf{x}_\nu)\mathbf{d}_\nu = -\mathbf{F}(\mathbf{x}_\nu), \quad \text{where } \mathbf{F}' := \left(\frac{\partial F_i}{\partial x_k}\right) \text{ and } \mathbf{F} = (F_1, F_2, \dots, F_n), \quad \mathbf{x} = (x_1, x_2, \dots, x_n).$$

(2)  $\mathbf{x}_{\nu+1} = \mathbf{x}_\nu + \mathbf{d}_\nu$

(3) If  $\|\mathbf{x}_{\nu+1} - \mathbf{x}_\nu\|$  is small enough (or another termination) we set  $\mathbf{x}^* = \mathbf{x}_{\nu+1}$ .

# Chapter 3

## PARALLEL/CENTRAL- PROJECTION

For displaying a 3-dimensional object one uses usually either

- a) orthographic projection (parallel rays orthogonal to the image plane) or
- b) central projection (rays which have a point, the centre, in common).

### 3.1 Orthographic Projection

#### 3.1.1 The projection formulae

In order to describe the orthographic projection of a point onto the image plane  $\varepsilon_0$  we introduce a suitable orthogonal coordinate system  $(O; x, y, z)$  such that the origin is contained in  $\varepsilon_0$ . (A translation of  $\varepsilon_0$  has no effect on the image but on the coordinates.)  $\varepsilon_0$  and the orthographic projection are defined uniquely by the normal vector  $\mathbf{n}_0$  of  $\varepsilon_0$ . We choose  $\mathbf{n}_0$  such that  $|\mathbf{n}_0| = 1$  and its direction points in the opposite direction of the projection rays ( $\mathbf{n}_0$  points to the "sun"). For the description of  $\mathbf{n}_0$  we use the angles  $u, v$  which are sphere coordinates ( $u$  is "geographic longitude" and  $v$  "geographic latitude"):

$$\mathbf{n}_0 = (\cos u \cos v, \sin u \cos v, \sin v), \quad 0 \leq u \leq 2\pi, \quad -\pi/2 \leq v \leq \pi/2.$$

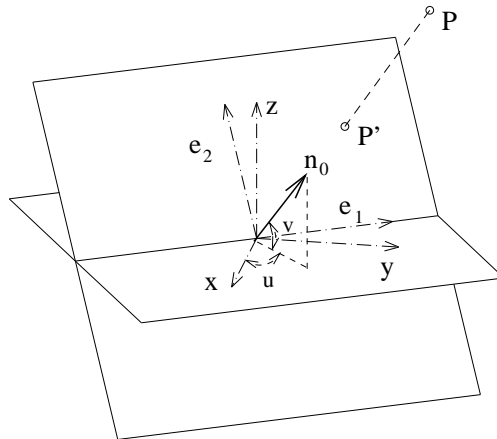


Figure 3.1: Parallel projection of a point  $P$

An image of a point will be described by coordinates corresponding to a rectangular coordinate system  $(O; x_e, y_e)$  of the plane  $\varepsilon_0$ . The origins of the image plane and the object space coincide and in case of  $|v| < \pi/2$  the  $y_e$ -axis is the image of the  $z$ -axis. Hence the  $x_e$ -axis is contained in the intersection line of  $\varepsilon_0$  and the  $x$ - $y$ -plane. The vectors

$$\mathbf{e}_1 = (-\sin u, \cos u, 0), \quad \mathbf{e}_2 = (-\cos u \sin v, -\sin u \sin v, \cos v)$$

are an orthonormal basis in  $\varepsilon_0$  and  $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{n}_0\}$  is an orthonormal basis of object space  $\mathbb{R}^3$ .

The coordinates  $x', y'$  of the image of a point  $P : \mathbf{p} = (x, y, z)$  are

$$\begin{aligned} x' &= \mathbf{e}_1 \cdot \mathbf{p} = -x \sin u + y \cos u \\ y' &= \mathbf{e}_2 \cdot \mathbf{p} = -(x \cos u + y \sin u) \sin v + z \cos v. \end{aligned}$$

Hence an orthographic projection is a linear mapping.

### 3.1.2 Procedures for orthographic projection

The reals  $\sin u, \cos u, \sin v, \cos v$  and the normal vector  $\mathbf{n}_0$  of the image plane will be determined by procedure `init_parallel_projection` right after the input of the angles  $u, v$  and stored in global variables. The following procedures are necessary for an orthographic projection and are contained in file `proc_pp.pas` (see volume include).

```

procedure init_parallel_projection;
begin
  writeln('*** PARALLEL-PROJECTION ***');
  writeln;
  writeln('Projection angles u, v ? (in degree)');
  readln(u_angle, v_angle);
  rad_u:= u_angle*pi/180;      rad_v:= v_angle*pi/180;
  sin_u:= sin(rad_u)      ;    cos_u:= cos(rad_u)      ;
  sin_v:= sin(rad_v)      ;    cos_v:= cos(rad_v)      ;
{normal vector of the image plane:}
  n0vt.x:= cos_u*cos_v;  n0vt.y:= sin_u*cos_v;  n0vt.z:= sin_v;
end; { init_parallel_projection }
{*****}
procedure pp_vt3d_vt2d(p:vt3d; var pp:vt2d);
  {Determines the image of a point}
{*****}
procedure pp_point(p:vt3d; style:integer);
  {Projects a point and marks the image with respect to style}
{*****}
procedure pp_line(p1,p2:vt3d ; style:integer);
  {Projects the line segment p1,p2 with respect to style}
{*****}
procedure pp_arrow(p1,p2:vt3d; style:integer);
  {Projects an arrow}
{*****}
procedure pp_axes(al:real);
  {Projects the coordinate axes, al: length of an axis}
{*****}
procedure pp_vts3d_vts2d(var p:vts3d; n1,n2:integer; var pp:vts2d);
  {Determines the array pp of projections of an array p of points.}
{*****}
procedure pp_curve(var p:vts3d; n1,n2,style:integer);
  {Projects the 3d-ngon p[n1]...p[n2]}
{*****}

```

### 3.2 Central projection

One gets a more realistic picture by using central projection, i.e. the projection rays meet in a point  $Z$ , the centre or eyepoint. I.g. central projection does not respect proportions and the images of parallel lines meet in a point (see fig 3.2).

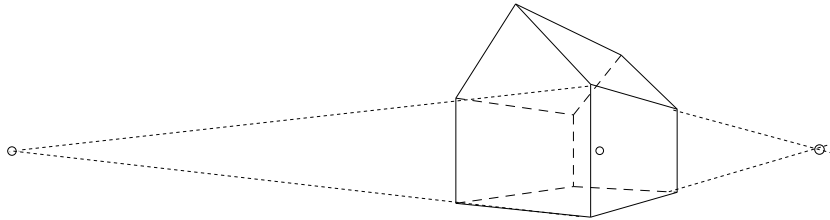


Figure 3.2: central projection of a house

We will see that many procedures written for orthographic projection can be changed easily to central projection.

#### 3.2.1 The projection formulae

Let  $\varepsilon$  be a plane and  $Z : \mathbf{z}$  a point not contained in  $\varepsilon$ . Function  $\Phi$  which maps a point  $P : \mathbf{p}$  onto the intersection point  $P'$  of line  $\overline{ZP}$  with plane  $\varepsilon$  (if it exists) is called **central projection from  $Z$  onto  $\varepsilon$** .

$Z$  is called **centre** or **eyepoint** of central projection  $\Phi$  (see fig. 3.3). The footpoint  $H : \mathbf{h}$  of the perpendicular through  $Z$  onto the image plane  $\varepsilon_0$  is called **mainpoint** of  $\Phi$ . With the normal vector

$$\mathbf{n}_0 := (\cos u \cos v, \sin u \cos v, \sin v), \quad u \in [0, 2\pi], \quad v \in [-\pi/2, \pi/2],$$

we get the equation of plane  $\varepsilon_0$  as:  $(\mathbf{x} - \mathbf{h}) \cdot \mathbf{n}_0 = 0$ . For centre  $Z : \mathbf{z}$  and mainpoint  $H$  we get the relation  $\mathbf{z} = \mathbf{h} + \delta \mathbf{n}_0$  with  $\delta > 0$ . The distance  $\delta$  between  $Z$  and plane  $\varepsilon_0$  is called **distance of  $\Phi$** . All points of  $\mathbb{R}^3$  which have no image (with respect of  $\Phi$ ) are contained in the *vanishing plane*  $\varepsilon_v$  parallel to  $\varepsilon_0$  that contains centre  $Z$ .

The central projection  $\Phi$  is uniquely determined by the angles  $u, v$ , the mainpoint  $H$  and the distance

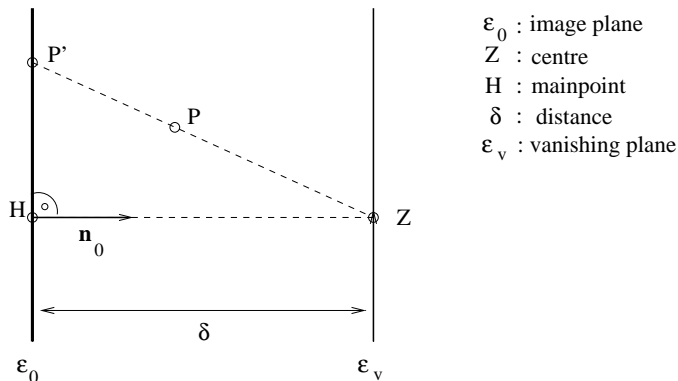


Figure 3.3: mainpoint, distance and vanishing plane

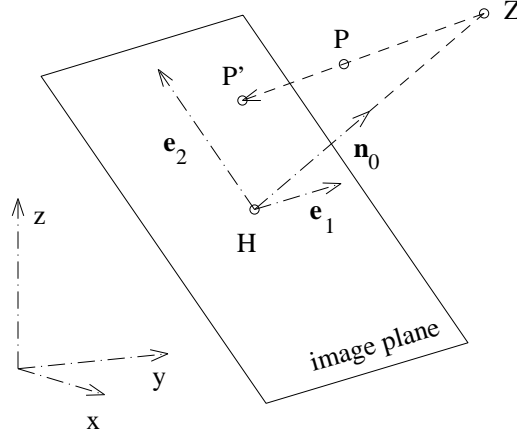


Figure 3.4: central projection of a point

$\delta$ . A point  $P : \mathbf{p}$  not contained in the vanishing plane  $\varepsilon_v$  will be mapped onto  $P' := \overline{ZP} \cap \varepsilon_0$ :

$$P' : \mathbf{p}' = \mathbf{z} + \frac{(\mathbf{h} - \mathbf{z}) \cdot \mathbf{n}_0}{(\mathbf{p} - \mathbf{z}) \cdot \mathbf{n}_0} (\mathbf{p} - \mathbf{z}) = \mathbf{h} + \delta \mathbf{n}_0 + \frac{\delta}{\delta - (\mathbf{p} - \mathbf{h}) \cdot \mathbf{n}_0} (\mathbf{p} - \mathbf{h} - \delta \mathbf{n}_0)$$

Because of  $P' \in \varepsilon_0$  we have additionally  $(\mathbf{p}' - \mathbf{h}) \cdot \mathbf{n}_0 = 0$ .

The coordinate system in image plane  $\varepsilon_0$  is chosen such that  $H$  coincides with the origin and (analogously to the orthographic projection)

$$\mathbf{e}_1 := (-\sin u, \cos u, 0), \quad \mathbf{e}_2 := (-\cos u \sin v, -\sin u \sin v, \cos v)$$

are the base vectors.  $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{n}_0\}$  is a  $ON$ -base of  $\mathbb{R}^3$  (see fig. 3.4).

There are reals  $x', y'$  (the coordinates of  $P'$  with respect to base  $\{\mathbf{e}_1, \mathbf{e}_2\}$ ) such that  $\mathbf{p}' = \mathbf{h} + x' \mathbf{e}_1 + y' \mathbf{e}_2$ .

From the two representations of  $\mathbf{p}'$  above and the orthogonality of  $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{n}_0\}$  we get

$$x' = \frac{\mathbf{e}_1 \cdot (\mathbf{p} - \mathbf{h})}{1 - (\mathbf{p} - \mathbf{h}) \cdot \mathbf{n}_0 / \delta} \quad y' = \frac{\mathbf{e}_2 \cdot (\mathbf{p} - \mathbf{h})}{1 - (\mathbf{p} - \mathbf{h}) \cdot \mathbf{n}_0 / \delta}$$

(In case of  $\mathbf{h} = 0$  and  $\delta \rightarrow \infty$  we get the formulae for the orthographic projection !)

Mathematically the projection is defined on  $\mathbb{R}^3 \setminus \varepsilon_v$  but in praxis applied only on points in front of the vanishing plane  $\varepsilon_v$ . (A point  $P : \mathbf{p}$  is in front of  $\varepsilon_v$  if  $(\mathbf{p} - \mathbf{h}) \cdot \mathbf{n}_0 < \delta$ .)

### 3.2.2 Procedures for central projection

The essential procedures used for central projection are listed below. One should keep in mind that `mainpt`, `centre`, `n0vt` and `distance` are global variables (see `geovar.pas`). The central projection of a point is performed in two steps:

- (1) Coordinate transformation into system  $(H; \mathbf{e}_1, \mathbf{e}_2, \mathbf{n}_0)$  with origin  $H$  and base  $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{n}_0\}$ .

$$\mathbf{p} = (x, y, z) \rightarrow \bar{\mathbf{p}} = (\bar{x}, \bar{y}, \bar{z}) \text{ with } \bar{x} = (\mathbf{p} - \mathbf{h}) \cdot \mathbf{e}_1, \bar{y} = (\mathbf{p} - \mathbf{h}) \cdot \mathbf{e}_2, \bar{z} = (\mathbf{p} - \mathbf{h}) \cdot \mathbf{n}_0,$$

- (2) Central projection within the system  $(H; \mathbf{e}_1, \mathbf{e}_2, \mathbf{n}_0)$  onto  $\bar{x} - \bar{y}$ -plane:

$$\bar{\mathbf{p}} = (\bar{x}, \bar{y}, \bar{z}) \rightarrow \left( \frac{\bar{x}}{1 - \bar{z}/\delta}, \frac{\bar{y}}{1 - \bar{z}/\delta} \right)$$



```

procedure init_centralparallel_projection(ind : integer);
begin
  if ind=1 then begin
    writeln('*** CENTRAL-projection ***');
    writeln('mainpoint ?'); readln(mainpt.x,mainpt.y,mainpt.z);
    writeln('distance ?'); readln(distance); end
  else begin
    writeln('*** PARALLEL-projection ***');
    mainpt:= null3d; distance:= 1000000000; end;
    writeln('Angles u, v ? (in degree)'); readln(u_angle,v_angle);
    rad_u:= u_angle*pi/180; rad_v:= v_angle*pi/180;
    sin_u:= sin(rad_u); cos_u:= cos(rad_u);
    sin_v:= sin(rad_v); cos_v:= cos(rad_v);
  {base e1,e2 and normal n0 of image plane:}
    e1vt.x:= -sin_u; e1vt.y:= cos_u; e1vt.z:= 0;
    e2vt.x:= -cos_u*sin_v; e2vt.y:=-sin_u*sin_v; e2vt.z:= cos_v;
    n0vt.x:= cos_u*cos_v; n0vt.y:= sin_u*cos_v; n0vt.z:= sin_v;
  {centre:}
    lcomb2vt3d(1,mainpt, distance,n0vt, centre);
  end; { init_central_parallel_projection }
  {*****}
  procedure transf_to_e1e2n0_base(p : vt3d; var pm : vt3d);
  {Determines the coordinates with respect to the system with the mainpoint as
  origin and base e1,e2,n0.}
  {*****}
  procedure cp_vt3d_vt2d(p: vt3d; var pp : vt2d);
  {central projection (coordinates) of a point}
  var xe,ye,ze,cc : real; pm : vt3d;
  begin
    diff3d(p,mainpt, pm);
    xe:= scalarp3d(pm,e1vt); {coordinates of p with respect of system:}
    ye:= scalarp3d(pm,e2vt); {mainpoint,e1,e2,n0}
    ze = scalarp3d(pm,n0vt);
    cc:= 1-ze/distance;
    if cc>eps6 then begin pp.x:= xe/cc; pp.y:= ye/cc; end {projection}
    else
      writeln('point not in front of the vanishing plane !!!');
  end; {cp_vt3d_vt2d}

```

The following procedures can be taken literally from parallel projection. We just replace `pp_` by `cp_`. They are contained in file `proc_cp.pas` (see volume include).

```

procedure cp_point(p: vt3d; style: integer);
  {projects and marks a point}
  {*****}
procedure cp_line(p1,p2 : vt3d ; style : integer);
  {projects the line segment p1 p2}
  {*****}
procedure cp_arrow(p1,p2 : vt3d; style : integer);
  {projects an arrow}
  {*****}
procedure cp_axes(al : real);
  {projects the coordinate axes}
  {*****}
procedure cp_vts3d_vts2d(p: vts3d; n1,n2 : integer; var pp : vts2d);
  {determines the projection of the array of points: p[i] , i= n1...n2.}
  {*****}

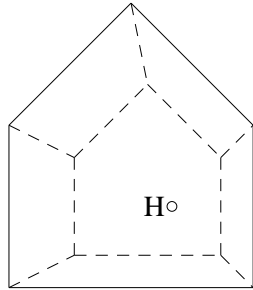
```

```

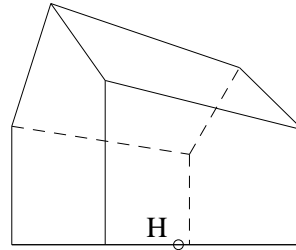
procedure cp_curve(p: vts3d; n1,n2,style : integer);
  {projects a 3D-ngon}
  {*****}

```

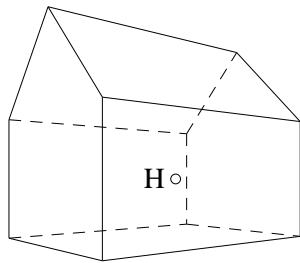
Examples:



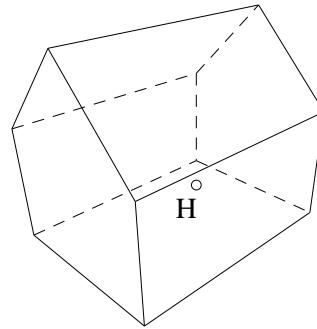
$u=0^\circ, v=0^\circ$



$u=50^\circ, v=0^\circ$



$u=50^\circ, v=0^\circ$



$u=45^\circ, v=30^\circ$

Figure 3.5: Central projections of a house

# Chapter 4

## CURVES

### 4.1 Planar Curves

#### 4.1.1 Definition and Representations of Planar Curves

**Definition 4.1** A planar curve  $\Gamma$  is the image of a real interval  $I$  by a locally injective  $C^0$  function  $\mathbf{x}$  into  $\mathbb{R}^2$ :

$$\Gamma := \{\mathbf{x}(t) \mid t \in I\}.$$

**Definition 4.2** A regular planar curve  $\Gamma$  is the image of a real interval  $I$  by a  $C^1$  function  $\mathbf{x}$  into  $\mathbb{R}^2$ :

$$\Gamma := \{\mathbf{x}(t) \mid t \in I\} \quad \text{and} \quad \dot{\mathbf{x}}(t) \neq \mathbf{0} \text{ for } t \in I.$$

$\dot{\mathbf{x}}(t)$  is a tangent vector at curve point  $\mathbf{x}(t)$ .

$\mathbf{x}$  is a regular parametric representation of  $\Gamma$ . The representation is of class  $C^n$  if  $\mathbf{x}(t)$  is  $C^n$ .

**Example 4.1**  $\mathbf{x}(t) = (t, t^2)^\top$ ,  $t \in [0, 1]$  (arc of the unit parabola  $y = x^2$ ). The parametric representation is regular.

The parametric representation of a regular curve is not unique! The parabola arc above for instance can be represented by

b)  $\mathbf{x}(t) = (t^2, t^4)^\top$ ,  $t \in [0, 1]$ ,    c)  $\mathbf{x}(t) = (\frac{2t}{1+t}, (\dots)^2)^\top$ ,  $t \in [0, 1]$ , too.

Representation b) is not regular (because of  $\dot{\mathbf{x}}(0) = (0, 0)^\top$ ), c) is regular.

**Definition 4.3** A regular implicit plane curve is a non empty subset  $\Gamma \subset \mathbb{R}^2$  of the form

$$f : \mathbb{R}^2 \supset D \rightarrow \mathbb{R}, \quad \Gamma := \{\mathbf{x} \in D \mid f(\mathbf{x}) = 0\}, \quad \nabla f(\mathbf{x}) \neq \mathbf{0} \text{ for } \mathbf{x} \in \Gamma,$$

where  $f$  is a  $C^1$ -function. For a curve point  $\mathbf{x}$  vector  $\nabla f(\mathbf{x})$  is a normal vector.

$f(\mathbf{x}) = 0$  is a regular implicit representation of  $\Gamma$ . The representation is of class  $C^n$  if  $f$  is  $C^n$ .

**Example 4.2**  $\Gamma := \{\mathbf{x} \in \mathbb{R}^2 \mid f(\mathbf{x}) := x^2 + y^2 - 1 = 0\}$  (unit circle).

Obviously, the implicit representation of a curve is not unique, too. For instance

b)  $f(\mathbf{x}) := \sqrt{x^2 + y^2} - 1 = 0$ ,    c)  $f(\mathbf{x}) := (x^2 + y^2 - 1)^2 = 0$

represent the unit circle implicitly, too. Representation b) is regular, c) is not regular.

**Definition 4.4** *The representation of a curve is called explicit if one of the coordinates  $x, y$  is a function of the remaining coordinate.*

**Example 4.3**  $y = x^2, x \in [0, 1]$  is an explicit representation of an arc of the unit parabola.

The following (theoretical) results show that the representations can be changed locally. The proofs rely on the *implicit function theorem*.

**Lemma 4.1** *Let  $\Gamma$  be a curve with a regular  $C^n$ -continuous implicit (parametric) representation and  $\mathbf{x}_0$  a point of  $\Gamma$ . Then there exists locally a  $C^n$ -continuous explicit representation of  $\Gamma$  at  $\mathbf{x}_0$ . Hence there exists locally a regular parametric (implicit) representation, which is  $C^n$ -continuous, too.*

**Proof:** a) Let be  $f(x, y) = 0$  with  $\nabla f(\mathbf{x}_0) \neq \mathbf{0}$  a regular implicit representation of curve  $\Gamma$ . Without loss of generality we assume  $f_y(\mathbf{x}_0) \neq 0$ . The implicit function theorem guarantees in a vicinity of  $\mathbf{x}_0$  the existence of a  $C^n$ -continuous function  $y(x)$  such that  $f(x, y(x)) = 0$  and  $y'(x) = -f_x(x, y)/f_y(x, y), \dots$ . Hence  $y = y(x)$  is locally an explicit  $C^n$ -representation.  
 b) Let be  $\mathbf{x}(t) = (x(t), y(t))^\top$  a regular parameterization of  $\Gamma$  with  $\mathbf{x}(t_0) = \mathbf{x}_0$ . With out loss of generality we assume  $\dot{x}(t_0) \neq 0$ . Hence equation  $x - x(t) = 0$  can be solved in a vicinity of  $t_0$  for  $t$ . With solution  $t(x)$  we get the explicit representation  $y = y(t(x))$ , which is  $C^n$ -continuous, too.  
 From an explicit representation one derives easily both a parametric and an implicit representation. For example: From  $y = y(x)$  we get the parametric representation  $(x, y(x))^\top$  and the implicit representation  $y - y(x) = 0$ .  $\square$

### 4.1.2 Arc length and curvature of a planar curve

**Definition 4.5** *Let  $\Gamma : \mathbf{x} = \mathbf{x}(t), t \in [a, b]$ , be a regular curve.  $s(t) := \int_a^t \sqrt{\dot{x}^2 + \dot{y}^2} dt$  is the arc length of the arc between the points  $\mathbf{x}(a)$  and  $\mathbf{x}(t)$ . Because of  $\dot{s}(t) = \sqrt{\dot{x}^2 + \dot{y}^2} > 0$  function  $s(t)$  is strictly monotone increasing and the equation  $s - s(t) = 0$  can be solved for  $t$  (theoretically) and  $s$  used for a standard representation, the arc length representation:  $\mathbf{x} = \bar{\mathbf{x}}(s) := \mathbf{x}(t(s))$  with its characteristic property  $\|\frac{d\bar{\mathbf{x}}}{ds}\| = 1$ .*

*In order to indicate the usage of the arc length as parameter one denotes the derivatives by a prime:  $\bar{\mathbf{x}}' = \frac{d\bar{\mathbf{x}}}{ds}$ .*

**Definition 4.6** *Let  $\mathbf{x}(s)$  be the arc length parameterization of a regular  $C^2$ -continuous curve. An essential property is  $\mathbf{x}' \cdot \mathbf{x}'' = 0$ , which is a direct consequence of  $\mathbf{x}' \cdot \mathbf{x}' = 1$ .*

*The curvature vector  $\mathbf{x}''$  describes the change of the unit tangent vector  $\mathbf{t} := \mathbf{x}'$  with respect to the arc length and is perpendicular to the tangent vector. If  $\mathbf{n}(t)$  is any unit normal vector, one gets the oriented curvature  $k = \mathbf{x}'' \cdot \mathbf{n}$ .*

*(For example  $\mathbf{n}(t) := (-\dot{y}(t), \dot{x}(t))^\top / \|\dots\|$ . The usage of  $\mathbf{x}'' / \|\mathbf{x}''\|$  as unit normal is possible if the curvature is not 0.)*

*A point with  $k = 0$  is called inflection point.*

**Lemma 4.2** *For the change of the unit normal we get:  $\mathbf{n}' = -k\mathbf{x}'$ .*

**Proof:** Let be  $\mathbf{n} := (-y', x')^\top$  the unit normal (parameterized by the arc length). Hence  $\mathbf{n}' = (-y'', x'')^\top$  is a tangent vector and  $\mathbf{n}' \cdot \mathbf{x}' = -x'y'' + y'x'' = -\mathbf{x}'' \cdot \mathbf{n} = -k$ .

The choice  $\mathbf{n} := (y', -x')^\top$  yields the same result.  $\square$

**Remark:**

When dealing with normalforms (see Section 4.1.4.2) one should remember the statement of the last lemma.

**Lemma 4.3** a) Let  $\Gamma : \mathbf{x} = \mathbf{x}(t) = (x(t), y(t))^\top, t \in [a, b]$ , be a regular (parametric)  $C^2$ -continuous curve with unit normal  $\mathbf{n}(t)$ . The curvature vector  $\mathbf{x}''$  and the curvature  $k$  at point  $\mathbf{x}(t)$  are:

$$\mathbf{x}'' = \frac{\ddot{\mathbf{x}}\dot{\mathbf{x}}^2 - (\dot{\mathbf{x}} \cdot \ddot{\mathbf{x}})\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|^4} = \frac{\ddot{\mathbf{x}} \cdot \mathbf{n}}{\|\dot{\mathbf{x}}\|^2} \mathbf{n}, \quad k = \frac{\ddot{\mathbf{x}} \cdot \mathbf{n}}{\|\dot{\mathbf{x}}\|^2} = \frac{\det(\dot{\mathbf{x}}, \ddot{\mathbf{x}})}{\|\dot{\mathbf{x}}\|^3}, \quad \text{for } \mathbf{n} = \frac{(-\dot{y}, \dot{x})^\top}{\|\dot{\mathbf{x}}\|}.$$

b) Let  $\Gamma : y = f(x), x \in [a, b]$ , be a regular (explicit)  $C^2$ -continuous curve. The curvature vector  $\mathbf{x}''$  and the curvature  $k$  at point  $(x, f(x))$  are:

$$\mathbf{x}'' = \frac{f''(x)}{(1 + f'(x)^2)^{3/2}} \frac{(-f'(x), 1)^\top}{\sqrt{1 + f'(x)^2}}, \quad k(x) = \frac{f''(x)}{(1 + f'(x)^2)^{3/2}}.$$

c) Let  $\Gamma : f(x, y) = 0, (x, y) \in D \subset \mathbb{R}^2$  be a regular (implicit)  $C^2$ -continuous curve. The curvature vector  $\mathbf{x}''$  and the curvature  $k$  at point  $(x, y)$  are:

$$\mathbf{x}'' = \frac{-f_{xx}f_y^2 + 2f_{xy}f_xf_y - f_{yy}f_x^2}{(f_x^2 + f_y^2)^{3/2}} \frac{\nabla f}{\|\nabla f\|}, \quad k = \frac{-f_{xx}f_y^2 + 2f_{xy}f_xf_y - f_{yy}f_x^2}{(f_x^2 + f_y^2)^{3/2}}.$$

**Proof:** a) Differentiating  $\mathbf{x}' = \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|}$  for the arc length  $s$  we get

$$\frac{d\mathbf{x}'}{ds} = \frac{d\mathbf{x}'}{dt} \frac{1}{ds/dt} = \frac{\ddot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|^2} - \frac{(\dot{\mathbf{x}} \cdot \ddot{\mathbf{x}})\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|^4} = \frac{\ddot{\mathbf{x}} \cdot \mathbf{n}}{\|\dot{\mathbf{x}}\|^2} \mathbf{n},$$

which is result a).

b) Applying a) to the parametric curve  $\mathbf{x}(t) := (t, f(t))^\top$  we get result b).

c) Because the representation of the implicit curve is regular  $f(x, y) = 0$  can be solved locally for  $x$  or  $y$ . Without loss of generality we assume:  $f_y(x_0, y_0) \neq 0, (x_0, y_0) \in \Gamma$ . The implicit function theorem guarantees the existence of a function  $y(x)$  such that in a vicinity of  $x_0$  the equation  $f(x, y(x)) = 0$  holds. Differentiating implicitly yields:  $y'(x) = -f_x(x, y(x))/f_y(x, y(x))$ . Analogously we get  $y'' = -(f_{xx}f_y^2 - 2f_{xy}f_xf_y + f_{yy}f_x^2)/f_y^3$ . Using result b) yields result c).  $\square$

**Example 4.4** a) For the unit parabola  $y = f(x) = x^2$  we get:

$$\mathbf{x}'' = \frac{(-2x, 1)}{\sqrt{1 + 4x^2}} \frac{2}{(1 + 4x^2)^{3/2}}, \quad k(x) = \frac{2}{(1 + 4x^2)^{3/2}}.$$

b) For an arbitrary circle  $f(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_m)^2 - r^2 = 0$  we get:

$$\nabla f(\mathbf{x}) = 2(\mathbf{x} - \mathbf{x}_m), \quad \|\nabla f(\mathbf{x})\| = 2r, \quad f_{xx} = f_{yy} = 2, f_{xy} = 0, \quad \text{and} \quad \mathbf{x}'' = \mathbf{x}_m - \mathbf{x}, \quad k = 1/r.$$

**Definition 4.7** Let  $\Gamma$  be a regular  $C^2$ -curve and  $\mathbf{x}_0 \in \Gamma$  not an inflection point.

a) The circle  $c_{\mathbf{x}_0} : (\mathbf{x} - \mathbf{x}_0 - \frac{\mathbf{x}''(\mathbf{x}_0)}{(\mathbf{x}''(\mathbf{x}_0))^2})^2 - \frac{1}{(\mathbf{x}''(\mathbf{x}_0))^2} = 0$  is called osculating circle, its midpoint centre of curvature at  $\mathbf{x}_0$ .

b) The parabola  $p_{\mathbf{x}_0} : \mathbf{x} = \mathbf{x}_0 + s\mathbf{x}'(\mathbf{x}_0) + \frac{s^2}{2}\mathbf{x}''(\mathbf{x}_0)$  is called the osculating parabola at point  $\mathbf{x}_0$ .

$c_{\mathbf{x}_0}$  and  $p_{\mathbf{x}_0}$  are tangent to  $\Gamma$  in  $\mathbf{x}_0$  (i.e.  $\Gamma, c_{\mathbf{x}_0}$  and  $p_{\mathbf{x}_0}$  have point  $\mathbf{x}_0$  and the tangent at this point in common) and have at  $\mathbf{x}_0$  the same curvature vector as  $\Gamma$ .

### 4.1.3 Offset Curves

**Definition 4.8** Let  $\Gamma : \mathbf{x} = \mathbf{x}(t) = (x(t), y(t))^\top, t \in [a, b]$ , be a regular (parametric)  $C^2$ -continuous curve,  $\mathbf{n}(t) = \frac{(\dot{y}(t), -\dot{x}(t))^\top}{\sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}}$  (unit normal) and fixed  $d \in \mathbb{R}$ .

Then  $\Gamma_d : \mathbf{x} = \mathbf{x}(t) + d\mathbf{n}(t)$  is called offset curve of  $\Gamma$  with distance  $d$ .

The offset curve  $\Gamma_d$  depends only on curve  $\Gamma$  as a point set in  $\mathbb{R}^2$  and distance  $d$ .

**Lemma 4.4** Let  $\Gamma : \mathbf{x} = \mathbf{x}_0(t) = (x(t), y(t))^\top, t \in [a, b]$ , be a regular  $C^n$ -continuous curve,  $n \geq 3$ , with unit normal  $\mathbf{n}_0(t)$ , the oriented curvature  $k_0(t) = \mathbf{x}'' \cdot \mathbf{n}$  at point  $\mathbf{x}(t)$  ( $\mathbf{x}'$ : unit tangent vector,  $\mathbf{x}''$ : curvature vector) and  $d \in \mathbb{R}$  such that  $1 - dk_0(t) > 0$  then the offset curve  $\Gamma_d : \mathbf{x} = \mathbf{x}_0(t) + d\mathbf{n}_0(t)$  (s. definition above) is regular of class  $C^{n-1}$ , and the curvature is  $k_d(t) = \frac{k_0(t)}{1 - dk_0(t)}$ .

**Proof:** Without loss of generality we assume  $\Gamma_0$  is parameterized by arc length. From Lemma 4.2 we get  $\mathbf{n}'_0(s) = -k_0(s)\mathbf{x}'_0(s)$ . Differentiating the parametric representation  $\Gamma_d : \mathbf{x} = \mathbf{x}_0(s) + d\mathbf{n}_0(s)$  of the offset curve yields

$$\dot{\mathbf{x}} = \mathbf{x}'_0(s) + d\mathbf{n}'_0(s) = (1 - dk_0(s))\mathbf{x}'_0(s) \quad \text{and} \quad \ddot{\mathbf{x}} = -dk'_0(s)\mathbf{x}'_0(s) + (1 - dk_0(s))\mathbf{x}''_0(s).$$

The unit normal vector of the offset curve agrees with  $\mathbf{n}_0$ . Hence the curvature of offset curve  $\Gamma_d$  is

$$\frac{\ddot{\mathbf{x}} \cdot \mathbf{n}_0}{\|\dot{\mathbf{x}}\|^2} = \frac{(1 - dk_0)\mathbf{x}'_0 \cdot \mathbf{n}_0}{(1 - dk_0)^2} = \frac{k_0}{1 - dk_0} \quad \text{which proves the statement of the lemma.} \quad \square$$

### 4.1.4 The normalform of a planar curve

#### 4.1.4.1 Definition of the normalform

In order to get a normalized implicit representation (comparable with the normalized arc length parametric representation) we extend the idea of the HESSE-normalform for lines to curves.

**Definition 4.9** The implicit representation  $h(\mathbf{x}) = \mathbf{n} \cdot \mathbf{x} - d = 0, \|\mathbf{n}\| = 1$  of a line  $g$  in  $\mathbb{R}^2$  is called HESSE normalform of line  $g$ .

( $h(\mathbf{x})$  describes the oriented distance of point  $\mathbf{x}$  to the line  $g$ .)

**Definition 4.10** An implicit representation  $h(\mathbf{x}) = 0$  of a curve  $\Gamma$  in  $\mathbb{R}^2$  with  $\|\nabla h\| = 1$  in a vicinity of  $\Gamma$  is called normalform of curve  $\Gamma$ . (We regard  $h = 0$  and  $-h = 0$  as the same normalforms.)

$h$  is called normalform function or, because of its geometric meaning, oriented distance function.

**Example 4.5**  $h(x, y) := \sqrt{x^2 + y^2} - r = 0$  is the normalform of the circle  $x^2 + y^2 = r^2$ .

If the given curve is smooth enough, then its normalform always exists (theoretically).

**Theorem 4.1** For any  $C^n$ -continuous,  $n \geq 2$ , planar curve  $\Gamma$  there exists in a vicinity  $V$  of  $\Gamma$  the oriented distance function  $h$  and  $h$  is of class  $C^n$ , too. Hence  $\Gamma$  has an implicit representation  $h(\mathbf{x}) = 0$  such that  $\|\nabla h\| = 1$  on  $V$ .

On  $V$  function  $h$  has the properties: 1)  $h(\mathbf{x} + \delta \nabla h(\mathbf{x})) = h(\mathbf{x}) + \delta$ , 2)  $\nabla h(\mathbf{x} + \delta \nabla h(\mathbf{x})) = \nabla h(\mathbf{x})$ .

**Proof:** Let the  $C^n$ -curve  $\Gamma$ ,  $n \geq 2$ , be parameterized by the arc length  $s$ :

$\mathbf{x} = \mathbf{x}_0(s) = (x(s), y(s))^\top$ ,  $s \in [a, b]$ , with unit normal  $\mathbf{n}_0(s) := (-y'(s), x'(s))^\top$ , curvature  $k_0(s)$  and

$\Gamma_d$ :  $\mathbf{x} = \mathbf{x}_0(s) + d\mathbf{n}_0(s)$  its offset curve with (oriented) distance  $d$ .

The vector equation  $F(s, d, x, y) := \mathbf{x}_0(s) + d\mathbf{n}_0(s) - \mathbf{x} = 0$  can be solved for  $s$  and  $d$  if  $\det(F_s, F_d) \neq 0$ .

With  $\mathbf{n}'_0(s) = -k_0(s)\mathbf{x}'_0(s)$  (see Lemma 4.2) we get

$$\det(F_s, F_d) = \det(\mathbf{x}'_0 + d\mathbf{n}'_0, \mathbf{n}_0) = \det((1 - dk_0)\mathbf{x}'_0, \mathbf{n}_0) = (1 - dk_0) \neq 0$$

in a vicinity  $V : -v \leq d \leq v$ ,  $v > 0$  of curve  $\Gamma$ .

Due to the implicit function theorem there exist functions  $s(x, y)$  and  $d(x, y)$  such that the following equation is fulfilled in the vicinity  $V$  of curve  $\Gamma$ :

$\mathbf{x}_0(s(x, y)) + d(x, y)\mathbf{n}_0(s(x, y)) = \mathbf{x}$ . Differentiating this equation for  $x, y$  yields

$$\nabla s \mathbf{x}'_0^\top + \nabla d \mathbf{n}_0^\top + d \nabla s \mathbf{n}'_0^\top = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad (\text{recognize: } \begin{pmatrix} a1 \\ a2 \end{pmatrix} \begin{pmatrix} b1 \\ b2 \end{pmatrix}^\top = \begin{pmatrix} a1b1 & a2b1 \\ a1b2 & a2b2 \end{pmatrix}).$$

Multiplying this matrix equation by  $\mathbf{n}_0$  and respecting  $\mathbf{x}'_0^\top \mathbf{n}_0 = \mathbf{n}'_0^\top \mathbf{n}_0 = 0$ ,  $\mathbf{n}_0^\top \mathbf{n}_0 = 1$  we get  $\nabla d = \mathbf{n}_0$ . Hence function  $d(x, y)$  is of class  $C^n$ , too.

In order to distinguish function  $d(x, y)$  and distance parameter  $d$  we set  $h(x, y) := d(x, y)$ .

For points  $\mathbf{x}$  with the same foot point  $\mathbf{x}_0$  have the same gradient  $\nabla h(\mathbf{x})$ . Hence we get the essential properties of the distance function  $h$ :

$$1) h(\mathbf{x} + \delta \nabla h(\mathbf{x})) = h(\mathbf{x}) + \delta, \quad 2) \nabla h(\mathbf{x} + \delta \nabla h(\mathbf{x})) = \nabla h(\mathbf{x}). \quad \square$$

The proof of the uniqueness of the normalform function  $h$  is omitted.

Properties 1), 2) of the last theorem show that  $h$  and  $\nabla h$  can be evaluated numerically at point  $\mathbf{x}$  by determining the foot point  $\mathbf{x}_0$  and its unit normal  $\mathbf{n}_0$  of point  $\mathbf{x}$  on curve  $\Gamma$ :

$$h(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_0) \cdot \mathbf{n}_0, \quad \nabla h(\mathbf{x}) = \mathbf{n}_0.$$

#### 4.1.4.2 On the first and second derivatives of the distance function

Let be  $\Gamma$  a  $C^2$ -continuous planar curve with normalform  $h = 0$  and  $\mathbf{x} = \mathbf{c}(s)$  its arc length parameterization with unit normal  $\mathbf{n}(s) = \nabla h(\mathbf{c}(s))$ . Hence  $h(\mathbf{c}(s)) = 0$ . Differentiating this equation yields

$$(1) \quad \nabla h \cdot \mathbf{c}' = 0, \quad (2) \quad \mathbf{c}'^\top H_h \mathbf{c}' + \nabla h \cdot \mathbf{c}'' = 0.$$

From the second equation we get the following result:

The curvature of curve  $\Gamma$  is:

$$k = \nabla h \cdot \mathbf{c}'' = -\mathbf{c}'^\top H_h \mathbf{c}'.$$

- Differentiating  $(\nabla h)^2 = 1$  yields  $H_h \nabla h = \mathbf{0}$ .

Hence one eigenvalue of the Hessian matrix  $H_h$  is  $\lambda_1 = 0$  with eigenvector  $\nabla h$ . There exists a second eigenvalue  $\lambda_2 = \kappa$  with the tangent vector  $\mathbf{t} := (-h_y, h_x)^\top$  as eigenvector and  $\mathbf{t}^\top H_h \mathbf{t} = \kappa$ . Hence  $\kappa$  is the negative curvature of curve  $\Gamma$ :

$$-k = \kappa = \mathbf{t}^\top H_h \mathbf{t} = h_y^2 h_{xx} - 2h_x h_y h_{xy} + h_x^2 h_{yy}.$$

- The *characteristic polynomial* of  $H_h$  is  $\lambda^2 - (h_{xx} + h_{yy})\lambda$  and

$$\det(H_h) = 0, \quad \kappa = h_{xx} + h_{yy} \quad \text{and} \quad H_h^2 = \kappa H_h.$$

- The Hessian matrix  $H_h$  of the normalform function  $h$  at a curve point can be expressed by the gradient  $\nabla h$  (unit normal) and the curvature  $k$  (The three coefficients of  $H_h$  are the solution of the linear system  $H_h \nabla h = \mathbf{0}$ ,  $\mathbf{t}^\top H_h \mathbf{t} = -k$ ):

$$H_h = -k \begin{pmatrix} h_y^2 & -h_x h_y \\ -h_x h_y & h_x^2 \end{pmatrix} = -k(I - \nabla h \nabla h^\top),$$

with  $2 \times 2$  unit matrix  $I$ .  $\nabla h^\top$  is the transpose of the column vector  $\nabla h$ .

### 4.1.4.3 Normalform of an implicit curve

Let  $\Gamma : f(x, y) = 0$  be a regular implicit curve with continuous second derivatives of function  $f$  and  $h$  its normalform function.

For a *curve point* we get

1.  $h = 0$ ,
2.  $\nabla h = \frac{\nabla f}{\|\nabla f\|}$ ,
3.  $-k = \frac{f_y^2 f_{xx} - 2f_x f_y f_{xy} + f_x^2 f_{yy}}{(f_x^2 + f_y^2)^{3/2}}$ , (see Lemma 4.3),  
 $H_h = -k \begin{pmatrix} h_y^2 & -h_x h_y \\ -h_x h_y & h_x^2 \end{pmatrix} = \frac{f_y^2 f_{xx} - 2f_x f_y f_{xy} + f_x^2 f_{yy}}{\|\nabla f\|^5} \begin{pmatrix} f_y^2 & -f_x f_y \\ -f_x f_y & f_x^2 \end{pmatrix}$ .

For a point  $\mathbf{x}$  in the vicinity of the curve with foot point  $\mathbf{x}_0 \in \Gamma$  we get:

1.  $\nabla h(\mathbf{x}) = \nabla h(\mathbf{x}_0)$ ,
2.  $h(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_0) \cdot \nabla h(\mathbf{x}_0)$
3.  $H_h(\mathbf{x}) = \frac{-k(\mathbf{x}_0)}{1 - h(\mathbf{x})k(\mathbf{x}_0)} \begin{pmatrix} h_y^2(\mathbf{x}_0) & -h_x(\mathbf{x}_0)h_y(\mathbf{x}_0) \\ -h_x(\mathbf{x}_0)h_y(\mathbf{x}_0) & h_x^2(\mathbf{x}_0) \end{pmatrix} = \frac{H_h(\mathbf{x}_0)}{1 - h(\mathbf{x})k(\mathbf{x}_0)}$ .

### 4.1.4.4 Normalform of a parametric curve

Let  $\Gamma : \mathbf{x} = \mathbf{x}(t) = (x(t), y(t))^\top$  be a regular parametric curve with continuous second derivatives and  $h$  its normalform function.

For a *curve point*  $\mathbf{x}(t) = (x(t), y(t))^\top$  we get

$$h(\mathbf{x}) = 0, \quad \nabla h(\mathbf{x}) = \frac{(-\dot{y}, \dot{x})^\top}{\|\dot{\mathbf{x}}\|} \quad \text{and} \quad \text{from } k = (\ddot{\mathbf{x}} \cdot \nabla h) / \|\dot{\mathbf{x}}\|^2 = \frac{-\ddot{x}\dot{y} + \dot{x}\ddot{y}}{\|\dot{\mathbf{x}}\|^3} \text{ we get}$$

$$H_h(\mathbf{x}) = \frac{\ddot{x}\dot{y} - \dot{x}\ddot{y}}{\|\dot{\mathbf{x}}\|^5} \begin{pmatrix} \dot{x}^2 & \dot{x}\dot{y} \\ \dot{x}\dot{y} & \dot{y}^2 \end{pmatrix}.$$

For a point in the vicinity of curve  $\Gamma$ : apply the analogous part of an implicit curve (see above).

## 4.2 Displaying Parametric Curves in $\mathbb{R}^2$

For *displaying* a parametric curve we move through the parameter interval by not necessary equidistant steps, calculate a sequence of points and draw the polygon by procedure `curve2d`.

**Example 4.6** a) *ellipse*:  $(a \cos t, b \sin t)^\top$ ,  $0 \leq t \leq 2\pi$   
 b) *cycloids*:  $\mathbf{c}(t) = (x(t), y(t))^\top$ ,  $0 \leq t \leq 2\pi$  with

$$\begin{aligned} x(t) &= (a + b) \cos t - \lambda a \cos((a + b)t/a) \\ y(t) &= (a + b) \sin t - \lambda a \sin((a + b)t/a) \quad b > 0, \quad a + b > 0, \quad \lambda > 0. \end{aligned}$$

(see Fig. 4.1.)

In general parameter  $t$  is not the arc length and the distances of neighboured points may vary in a wide range. This behaviour can be omitted if function  $\mathbf{c}(t)$  is differentiable and one chooses the step length dependent of the “velocity”  $\dot{\mathbf{c}}(t)$ . In order to get distances  $\approx s$  we consider the Taylor expansion for  $\mathbf{c}(t)$

$$\mathbf{c}(t_{i+1}) \approx \mathbf{c}(t_i) + \dot{\mathbf{c}}(t_i)\Delta t_i, \quad \Delta t_{i+1} = t_{i+1} - t_i.$$



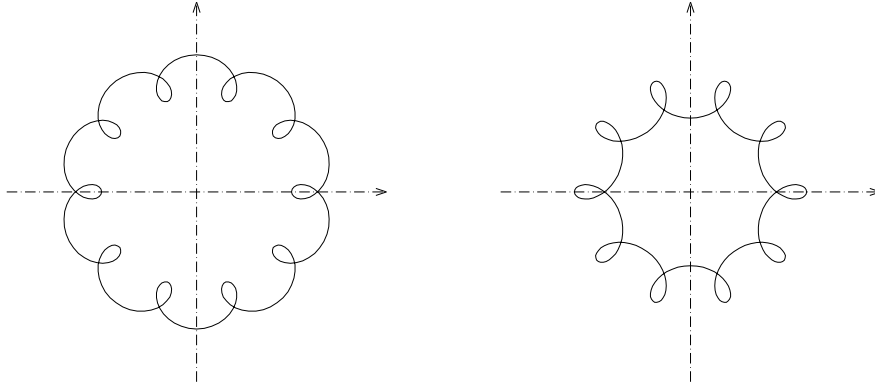


Figure 4.1: cycloids

Hence we get  $\|\mathbf{c}(t_{i+1}) - \mathbf{c}(t_i)\| \approx s$ , if  $\Delta t_i = s/\|\dot{\mathbf{c}}(t_i)\|$  and  $\|\dot{\mathbf{c}}(t_i)\| \neq 0$ .

This velocity dependent step length produces a *uniformly* distributed sequence of points.

We get a *curvature dependent* distribution of the points from the following considerations:

We demand that amount  $|\alpha|$  of the angle between the tangent vectors  $\dot{\mathbf{c}}(t_i), \dot{\mathbf{c}}(t_{i+1})$  of two neighboring points should be nearly constant. From

$$\dot{\mathbf{c}}(t_{i+1}) \approx \dot{\mathbf{c}}(t_i) + \ddot{\mathbf{c}}(t_i)\Delta t_i, \quad \Delta t_{i+1} = t_{i+1} - t_i, \quad \text{and} \quad \left| \frac{\det(\dot{\mathbf{c}}(t_{i+1}), \dot{\mathbf{c}}(t_i))}{\dot{\mathbf{c}}(t_{i+1}) \cdot \dot{\mathbf{c}}(t_i)} \right| = |\tan \alpha| \approx |\alpha|$$

for a "small" angle  $\alpha$  we get

$$|\alpha| \approx \left| \frac{\det(\dot{\mathbf{c}}(t_i), \ddot{\mathbf{c}}(t_i)) \Delta t_i}{\dot{\mathbf{c}}(t_i)^2 + \dot{\mathbf{c}}(t_i) \cdot \ddot{\mathbf{c}}(t_i)\Delta t_i} \right| \approx \left| \frac{\det(\dot{\mathbf{c}}(t_i), \ddot{\mathbf{c}}(t_i)) \Delta t_i}{\dot{\mathbf{c}}(t_i)^2} \right| = \|k(t_i)\dot{\mathbf{c}}(t_i)\Delta t_i\|$$

Hence we choose

$$|\Delta t_i| = \max\left(\min\left(\frac{|\alpha_0/k(t_i)|}{\|\dot{\mathbf{c}}(t_i)\|}, \frac{s_{max}}{\|\dot{\mathbf{c}}(t_i)\|}, \frac{s_{min}}{\|\dot{\mathbf{c}}(t_i)\|}\right), \|\dot{\mathbf{c}}(t_i)\| \neq 0\right),$$

where  $\alpha_0$  is the desired approximate angle between two neighboring tangent vectors (if  $k(t_i)$  is large enough) and  $s_{max}$  an upper boundary for the step size (if  $k(t_i)$  is too small)  $s_{min}$  a lower boundary for the step size (if  $k(t_i)$  is too large).

## 4.3 Displaying implicit curves

### 4.3.1 Marching algorithm

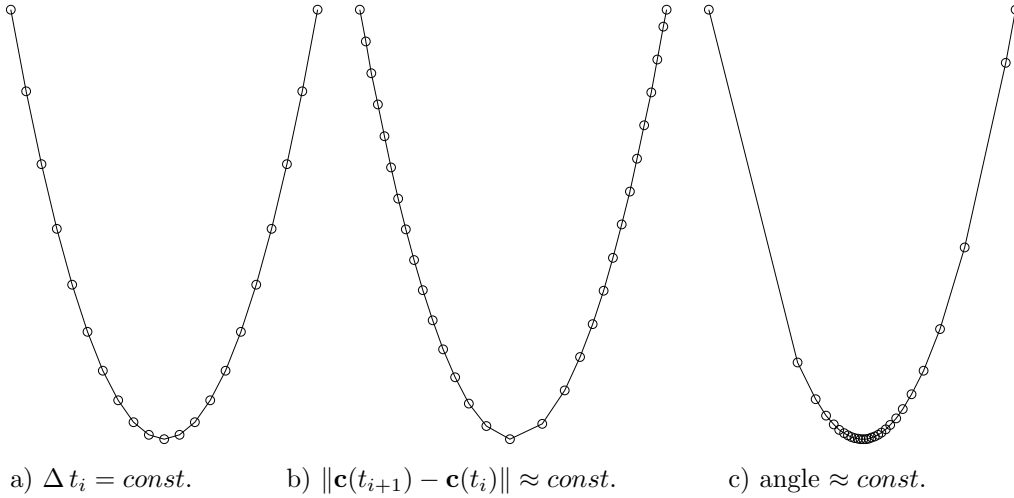
The calculation of a sequence of points of an implicit curve is not trivial. A rather simple method is the following one:

The **marching method** starts with a point in the vicinity of the curve and calculates a sequence of points of the same arc. The algorithm is fast and rather robust. Essentially there are two sub-problems to be solved:

- (1) Find a curve point when a starting point in the vicinity of the curve is given.
- (2) Generate a sequence of curve points.

Solution of subproblem (1): procedure **curvepoint**

(CP1) Choose a starting point  $Q_0 : \mathbf{q}_0 = (x_0, y_0)$  in the vicinity of the curve,

Figure 4.2: point distribution on curve  $\mathbf{c}(t) = (t, t^2)$ 

(CP2) repeat  $\mathbf{q}_{j+1} = \mathbf{q}_j - \frac{f(\mathbf{q}_j)}{\nabla f(\mathbf{q}_j)^T} \nabla f(\mathbf{q}_j)$  (Newton step for  $f(\mathbf{q}_j + t \nabla f(\mathbf{q}_j))$ .)  
 until  $\|\mathbf{q}_{j+1} - \mathbf{q}_j\|$  small enough.  
 curve point =  $\mathbf{q}_{j+1}$ .

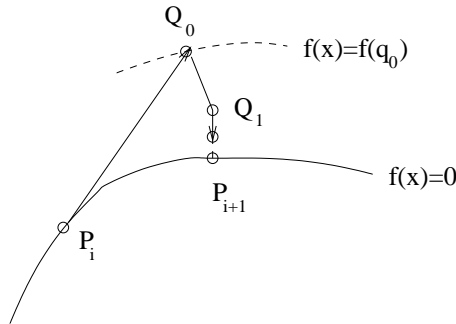


Figure 4.3: marching method for an implicit curve

**Performance** of the marching method:

- C1) Choose a starting point  $Q_0 : \mathbf{q}_0 = (x_0, y_0)$  and a step length  $s$  (approximate distance of neighbored points).
- C2) First curve point  $P_1 : \mathbf{p}_1 = \text{curvepoint}(\mathbf{q}_0)$ .
- C3) curve point  $P_{i+1} : \mathbf{p}_{i+1}$ :  
 $\mathbf{p}_{i+1} = \text{curvepoint}(\mathbf{p}_i + s \mathbf{t}_i)$ , with unit tangent  $\mathbf{t}_i = (-f_y(\mathbf{p}_i), f_x(\mathbf{p}_i))^T / \|\dots\|$  at point  $P_i$ .
- C4) Repeat 3) until the desired number of points is reached or  $P_{i+1} \approx P_0$  (closed curve).

**Remark:**

The marching method is rather robust and can even be used for tracing curves with singular points (i.e. points with  $\nabla f = 0$ ). But one should take care of the direction of the unit tangent when passing a singular point (see procedure `implicit_curvepts` in program `cassini.p`).

**Example 4.7** a) circle:  $x^2 + y^2 - r^2 = 0$ , hyperbola:  $xy - 1 = 0$   
 b) Cassini-curves:  $(x^2 + y^2)^2 - 2c^2(x^2 - y^2) - (a^4 - c^4) = 0$ ,  $a > 0$ ,  $c > 0$ .  
 For  $a = c$  we get lemniscates.

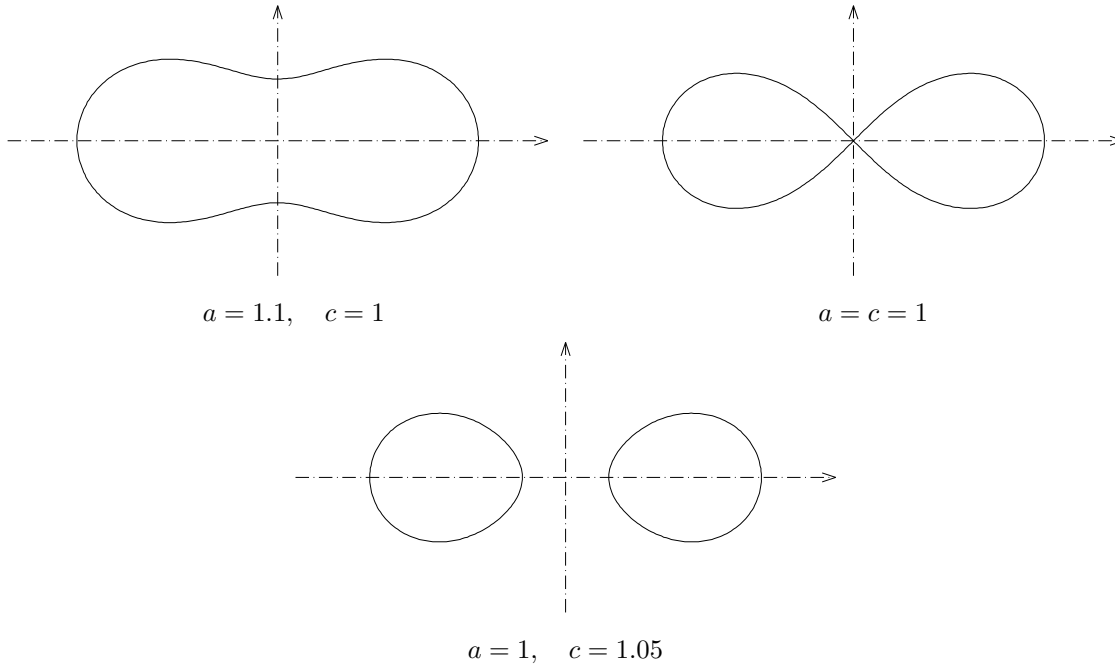


Figure 4.4: Cassini-curves

For a *curvature dependent* distribution of the points we take the second order Taylor expansion at point  $\mathbf{p}_k$  of the arc length parametric representation  $\mathbf{c}(s)$  and get a curvature dependent starting point for curve point  $P_{i+1}$ :

$$\mathbf{q}_0 = \mathbf{p}_i + s\mathbf{c}'(0) + \frac{s^2}{2}\mathbf{c}''(0),$$

with unit tangent vector  $\mathbf{c}'(0)$  and curvature vector  $\mathbf{c}''(0)$  at point  $P_i$ . Step length  $s$  should be chosen such that:

$$s = \max(\min(|\alpha_0/k_i|, s_{max}), s_{min}),$$

where  $\alpha_0$  is the desired approximate angle between two neighboring tangent vectors (if  $k_i$  is large enough) and  $s_{max}$  an upper boundary for the step size (if  $k_i$  is too small)  $s_{min}$  a lower boundary for the step size (if  $k_i$  is too large).

### 4.3.2 Raster algorithm

**Idea** of the algorithm:

We rasterize the domain where we suppose curve points. For any grid point we estimate its distance to the curve and, if the point is “close” to the curve, determine a curve point applying procedure `curvepoint` of the previous subsection. The displayed collection of generated curve points give an impression of the shape of the implicit curve and it looks like a curve if the raster is narrow enough.

**Performance:**

Given: implicit curve  $\Gamma : f(x, y) = 0$ .

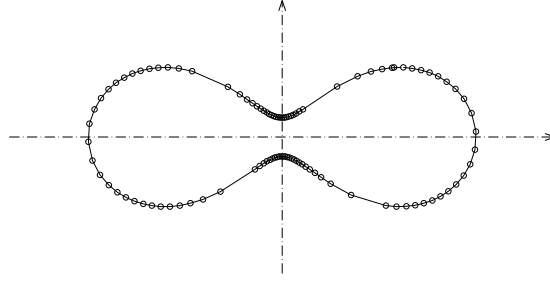


Figure 4.5: curvature dependent generation of points on an implicit curve,  $\alpha_0 = 0.2$

- 1) Choose an x-interval  $[x_{min}, x_{max}]$ , a y-interval  $[y_{min}, y_{max}]$  and the numbers  $n_x, n_y$  for their subdivision with distances  $d_x = (x_{max} - x_{min})/n_x$ ,  $d_y = (y_{max} - y_{min})/n_y$  respectively.  $d_{max} = \max(d_x, d_y)$  is the maximal raster distance.
- 2) For any raster point  $(x_i, y_k)$  we calculate  $f(x_i, y_k)$  and  $\nabla f(x_i, y_k)$ . In case of  $\nabla f(x_i, y_k) = (0, 0)$ , we ignore this (singular) grid point. otherwise we estimate the distance of grid point  $(x_i, y_k)$  to the curve by

$$\delta_{ik} := |f(x_i, y_k)| / \|\nabla f(x_i, y_k)\|, \text{ distance of 1. order.}$$

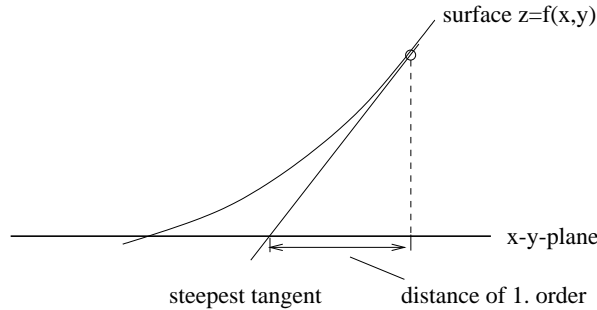


Figure 4.6: Distance of 1. order of a point to the implicit curve

- 3) For a point  $(x_i, y_k)$  with  $\delta_{ik} < \gamma d_{max}$ ,  $\gamma > 0.5$  we apply procedure `curvepoint` and get a curve point in the vicinity.

**Remark:**

The algorithm can be accelerated if one does not estimate the distance of any grid point but allows distance dependent jumps. One should secure that the jumps are not too large. Otherwise there may occur gaps.

**Example 4.8**  $f(x, y) = 0.004 + 0.11x - 0.177y - 0.174x^2 + 0.224xy - 0.303y^2 - 0.168x^3 + 0.327x^2y - 0.087xy^2 - 0.013y^3 + 0.235x^4 - 0.667x^3y + 0.745x^2y^2 - 0.029xy^3 + 0.072y^4$   
(aus TAU'94).

**Example 4.9** a)  $f(x, y) = (8x^4 - 4x^2y^2 + y^4)y^2 - (x^2 + y^2)^4$ ,  
b)  $f(x, y) = (3x^2 - y^2)^2y^2 - (x^2 + y^2)^4$ .

**Remark:**

The raster algorithm may be used for determining intersection points of implicit curves, too. For given curves  $f_1(x, y) = 0, f_2(x, y) = 0$ , one applies the raster algorithm to the implicit "curve"  $f(x, y) := f_1(x, y)^2 + f_2(x, y)^2 = 0$ .

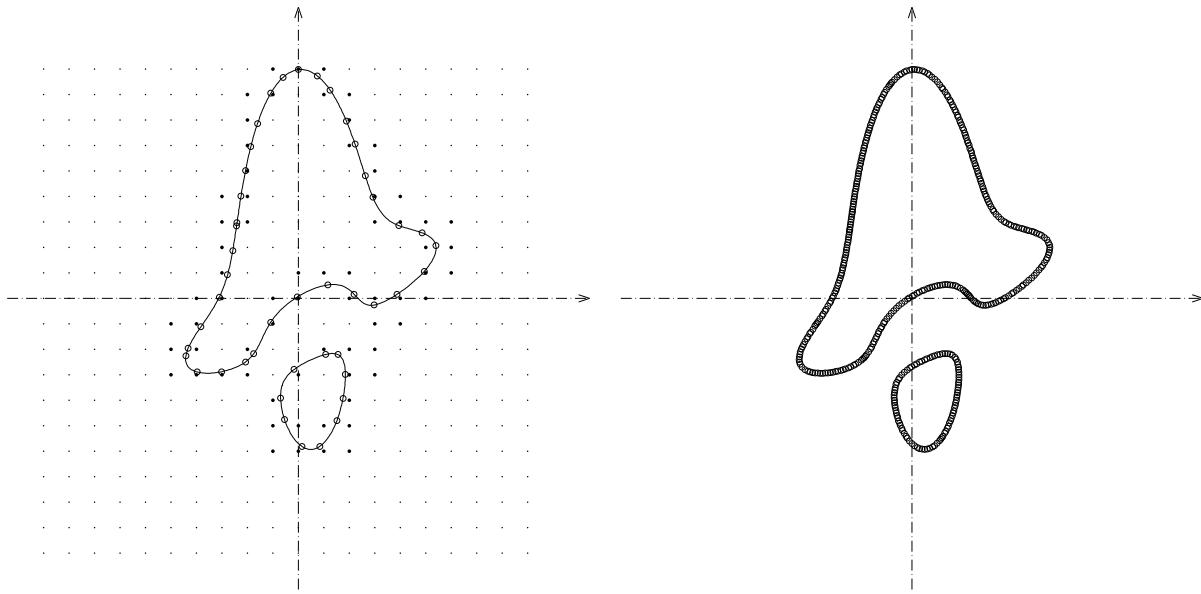


Figure 4.7: Raster algorithmus, left:  $n_x = n_y = 20$ , right:  $n_x = n_y = 200$

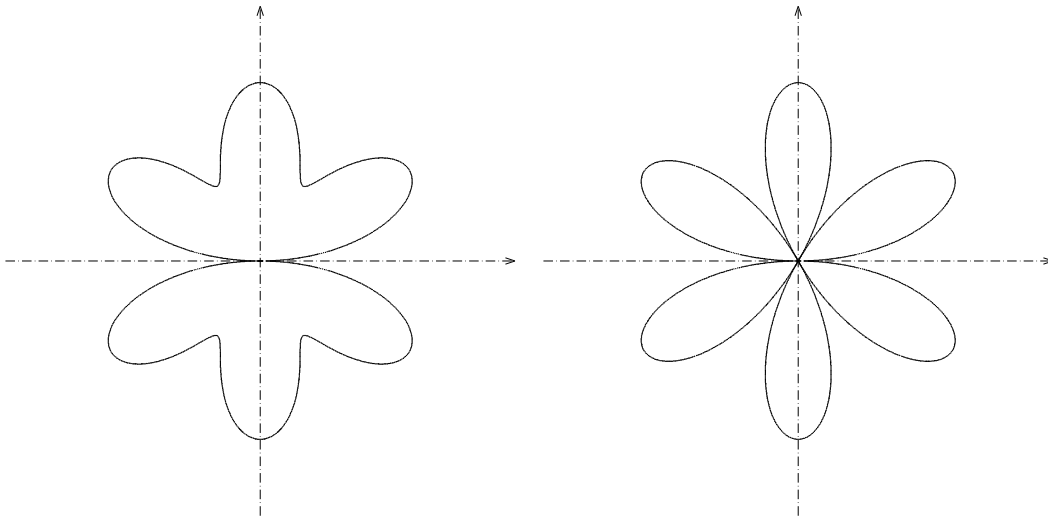


Figure 4.8: To example 4.9.  $n_x = n_y = 500$ .

**Example 4.10**  $f_1(x, y) = (3x^2 - y^2)^2 y^2 - (x^2 + y^2)^4$ ,  $f_2(x, y) = (x - 0.5)^2 + y^2 - 1 = 0$

## 4.4 Intersection of two planar curves

Determining intersection points of two curves usually means solving a system of non linear equations, which can be done using a Newton-iteration for suitable starting points.

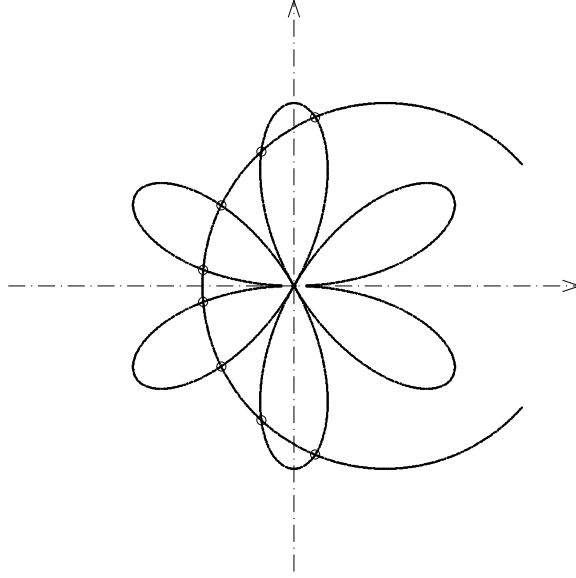


Figure 4.9: To example 4.10

#### 4.4.1 Intersection of a parametric curve and an implicit curve

Given: curves  $\Gamma_1 : \mathbf{x} = \mathbf{c}(t)$ ,  $a \leq t \leq b$  and  $\Gamma_2 : f(\mathbf{x}) = 0$ ,  $\mathbf{x} \in D \subseteq \mathbb{R}^2$ .

Wanted: Intersection points  $\Gamma_1 \cap \Gamma_2$ .

The **algorithm**:

- (1) Determining a starting point:  
Choose a subdivision  $t_1, t_2, \dots$  of the interval  $[a, b]$  (eventually equidistant).  
Look for changes of the sign of  $f(\mathbf{x}_i)$  for points  $\mathbf{x}_i := \mathbf{c}(t_i) \in \Gamma_1$ ,  $i = 1, 2, \dots$ . Is there a change near  $s_0 := t_i$ , use  $s_0$  as starting parameter for the following Newton-iteration.
- (2) Determine a solution  $t^*$  of  $F(t) := f(\mathbf{c}(t))$  applying the **Newton-iteration** to case  $n = 1$  with starting parameter  $t := s_0$  from (1). Point  $\mathbf{x}^* := \mathbf{c}(t^*)$  is an intersection point
- (3) Apply (2) to all observed changes of the sign.

**Problem 4.1** Determine the intersection points of the curves

$$\Gamma_1 : \mathbf{x} = \mathbf{c}(t) := (t, t^3)^\top, \quad 0 \leq t \leq 3,$$

$$\Gamma_2 : f(x, y) := (x - 1)^2 + (y - 1)^2 - 10 = 0$$

#### 4.4.2 Intersection of two implicit curves

Given:  $\Gamma_1 : f(\mathbf{x}) = 0$  and  $\Gamma_2 : g(\mathbf{x}) = 0$ ,  $\mathbf{x} \in D \subseteq \mathbb{R}^2$ .

Wanted: Intersection points  $\Gamma_1 \cap \Gamma_2$ .

The **algorithm**:

- (1) Determining a starting point:  
Determining points  $\mathbf{p}_1, \mathbf{p}_2, \dots$  on one component of  $\Gamma_1$  using the marching algorithm with  $\approx$  equal steplength. Look for changes of the sign of  $g(\mathbf{p}_i)$  for  $i = 1, 2, \dots$ . Is there a change near point  $\mathbf{x}_0 := \mathbf{p}_i$  use  $\mathbf{x}_0$  as starting point for the following Newton-iteration.

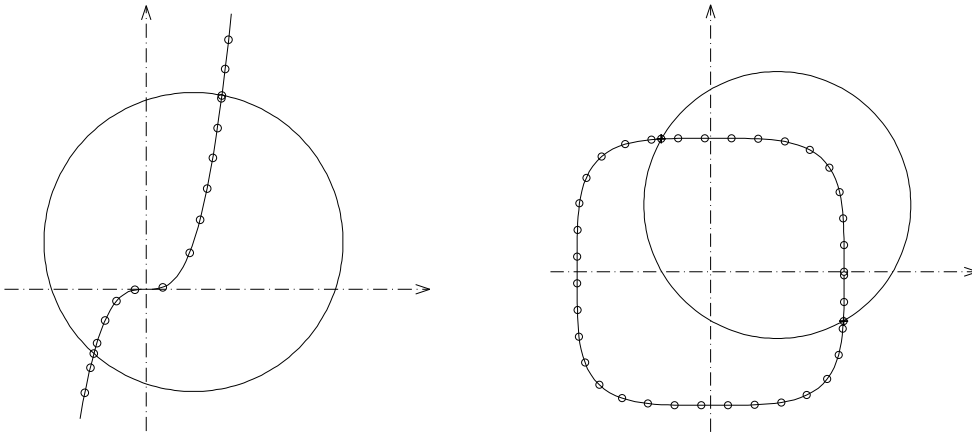


Figure 4.10: Left: to problem 4.1, right: to problem 4.2

- (2) Determine a solution  $\mathbf{x}^*$  of vector equation  $\mathbf{F}(\mathbf{x}) := (f(\mathbf{x}), g(\mathbf{x}))^\top = (0, 0)^\top$  applying a **Newton-iteration** to  $n = 2$  and starting point  $\mathbf{x}_0$  from (1). Point  $\mathbf{x}^*$  is an intersection point of the given curves.
- (3) Apply (2) to all changes of the sign of  $g(\mathbf{p}_i)$ ,  $i = 1, 2, \dots$

**Problem 4.2** Determine the intersection points of the curves

$$\Gamma_1 : f(x, y) := x^4 + y^4 - 1 = 0 \text{ and}$$

$$\Gamma_2 : g(x, y) := (x - 0.5)^2 + (y - 0.5)^2 - 1 = 0.$$

### 4.4.3 Intersection of two parametric curves

Given: curves  $\Gamma_1 : \mathbf{x} = \mathbf{c}_1(s)$ ,  $s \in [a, b]$ , and  $\Gamma_2 : \mathbf{x} = \mathbf{c}_2(t)$ ,  $t \in [c, d]$ .

Wanted: Intersection points  $\Gamma_1 \cap \Gamma_2$ .

The **algorithm**:

- (1) Determining start parameters:  
Choose subdivisions of the intervals  $[a, b]$ ,  $[c, d]$  (eventually equidistant) and calculate the corresponding polygons  $P_1, P_2, \dots \in \Gamma_1$  and  $Q_1, Q_2, \dots \in \Gamma_2$ . Look for pairs  $P_i, Q_k$  of points with minimal distance.  $s_i, t_k$  will be used as starting parameters for the following Newton-iteration.
- (2) Determine for start parameters from (1) a solution of the vector equation  $\mathbf{F}(s, t) := \mathbf{c}_1(s) - \mathbf{c}_2(t) = (0, 0)^\top$  applying a Newton-Iteration. (A single step of the iteration can be considered geometrically as the intersection of two tangents.)
- (3) Apply (2) to all Startparameters from (1).

## 4.5 Foot points on planar curves

### 4.5.1 Foot point on a parametric curve, curve inversion

Given: curve  $\Gamma : \mathbf{x} = \mathbf{c}(t)$ ,  $t \in [a, b]$ , point  $P : \mathbf{p}$ .

Wanted: Intersection point  $L$  of the line  $l$  through point  $P$  perpendicular to curve  $\Gamma$ . If  $L$  is unique we call  $L$  the *foot point* of  $P$  on curve  $\Gamma$ .

We assume that  $\mathbf{c}(t)$  is  $C^2$  and foot point  $L$  is unique.

If  $\Gamma$  is a line  $\mathbf{x} = \mathbf{q} + t\mathbf{r}$ :  $L : \mathbf{l} = \mathbf{q} + \frac{(\mathbf{p}-\mathbf{q}) \cdot \mathbf{r}}{\mathbf{r} \cdot \mathbf{r}} \mathbf{r}$ .

In general we seek a point  $\mathbf{c}(t)$  on  $\Gamma$  with:  $f(t) := (\mathbf{c}(t) - \mathbf{p}) \cdot \dot{\mathbf{c}}(t) = 0$ .

The **algorithm**:

(1) Determining a starting parameter:

Generate a nearly equidistant polygon  $Q_1, Q_2, \dots$  and look for a point  $L_0 := Q_i$  with minimal distance to  $P$ . Let  $t_0$  be its parameter, i.e.  $L_0 : \mathbf{c}(t_0)$ .

(2) Apply a Newton-iteration to equation  $f(t) := (\mathbf{c}(t) - \mathbf{p}) \cdot \dot{\mathbf{c}}(t) = 0$  with starting parameter from (1).

$$f'(t) = (\dot{\mathbf{c}}(t))^2 + (\mathbf{c}(t) - \mathbf{p}) \cdot \ddot{\mathbf{c}}(t).$$

(Second derivatives are used !) Or: Determine foot points on tangents successively (only 1. derivatives are necessary.)

(2') Let be  $L_i$  the  $i$ -th approximation of the foot point and  $t_i$  its corresponding parameter.  $\mathbf{x} = \mathbf{c}(t_i) + \Delta t \dot{\mathbf{c}}(t_i)$  is the tangent line at point  $L_i$  (linear approximation of  $\mathbf{c}(t)$  !). Determine the foot point of  $P$  on the tangent line. For the foot point on the tangent line we get  $\Delta t = (\mathbf{p} - \mathbf{c}(t_i)) \dot{\mathbf{c}}(t_i) / \dot{\mathbf{c}}(t_i)^2$ . Hence the improved parameter is  $t_{i+1} = t_i + \Delta t$  and  $L_{i+1} : \mathbf{c}(t_{i+1})$ .

(3') Repeat (2') until  $\|\mathbf{c}(t_{i+1}) - \mathbf{c}(t_i)\|$  is small enough. (Or another termination.)

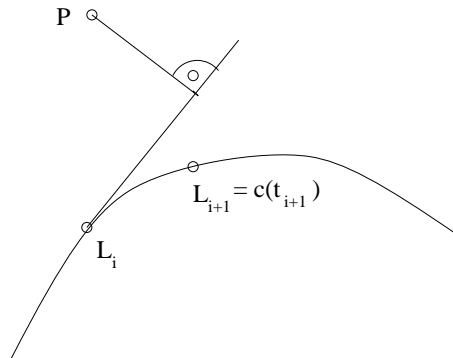


Figure 4.11: Step (2') of the foot point algorithm

**Remark:** The foot point algorithm can be used for determining the unknown parameter  $t_0$  of a curve point  $P_0 : \mathbf{p}_0 \in \Gamma : \mathbf{c}(t)$ . This task is called *curve inversion*.

### 4.5.2 Foot point on an implicit curve

Given: curve  $\Gamma : f(\mathbf{x}) = 0, \mathbf{x} \in D \subseteq \mathbb{R}^2$  and point  $P : \mathbf{p}$ .  $f \in C^2, \nabla f \neq 0$ .

Wanted: Foot point  $L$  of point  $P$  on curve  $\Gamma$ .

The **algorithm**:

(1) Applying procedure `curvepoint` (see section 4.3.1) we get a first approximation  $L_0 : \mathbf{x}_0$  for the foot point.



(2) The foot point is the solution of the system

$$(\mathbf{x} - \mathbf{p}) \cdot (f_y(\mathbf{x}), -f_x(\mathbf{x})) = 0, \quad f(\mathbf{x}) = 0.$$

(Where  $(f_y(\mathbf{x}), -f_x(\mathbf{x}))$  is a tangent vector at curve point  $\mathbf{x}$ .) This system will be solved by a Newton-iteration with starting point  $\mathbf{x}_0$  from (1).

(Second derivatives are needed !)

Or: Determine foot points on tangents successively (first derivatives are used only).

(2') Repeat  $\mathbf{t}_{i+1} = \mathbf{p} - \frac{(\mathbf{p} - \mathbf{x}_i) \cdot \nabla f(\mathbf{x}_i)}{\nabla f(\mathbf{x}_i)^2} \nabla f(\mathbf{x}_i)^\top$  (foot point on tangent line),  
 $\mathbf{x}_{i+1} = \text{curvepoint}(\mathbf{t}_{i+1})$ .  
 until  $\|\mathbf{x}_{i+1} - \mathbf{x}_i\|$  is "small" enough.  
 $L = \mathbf{x}_{i+1}$ .

**Remark:**

While using the simple first order foot point algorithms there may occur problems considering convergence. Improved and rather stable first order algorithms are contained in the next subsection.

### 4.5.3 Stable first order foot point algorithms

For applying the normalform and its first two derivatives it is essential to have stable algorithms for determining foot points on curves. Here we give algorithms for parametric and implicit curves which use first order derivatives only. The heart of the algorithms is the combination of calculating foot points on tangents and approximate foot points on tangent parabolas. The curvature of the curves is respected indirectly by the tangent parabolas.

#### 4.5.3.1 Foot point algorithm for parametric curves

Let  $\Gamma : \mathbf{x}(t) = \mathbf{c}(t)$  be a smooth planar curve,  $\mathbf{p}$  a point in the vicinity of  $\Gamma$  and  $t_0$  the parameter of a starting point for the *foot point algorithm*:

**repeat**

$\mathbf{p}_i = \mathbf{c}(t_i)$

$\Delta t = \frac{(\mathbf{p} - \mathbf{p}_i) \cdot \dot{\mathbf{c}}(t_i)}{\dot{\mathbf{c}}(t_i)^2} \quad \mathbf{q}_i = \mathbf{p}_i + \Delta t \dot{\mathbf{c}}(t_i)$  (foot point on tangent)

$\mathbf{p}_{i+1} = \mathbf{c}(t_i + \Delta t)$ ,  $\mathbf{f}_1 := \mathbf{q}_i - \mathbf{p}_i$ ,  $\mathbf{f}_2 := \mathbf{p}_{i+1} - \mathbf{q}_i$ ,

if  $\|\mathbf{q}_i - \mathbf{p}_i\| > \varepsilon$  then (one Newton step for the foot point on the tangent parabola  $\mathbf{x} = \mathbf{p}_i + \alpha \mathbf{f}_1 + \alpha^2 \mathbf{f}_2$ )

$a_0 := (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{f}_1$ ,  $a_1 := 2\mathbf{f}_2 \cdot (\mathbf{p} - \mathbf{p}_i) - \mathbf{f}_1^2$ ,  $a_2 := -3\mathbf{f}_1 \cdot \mathbf{f}_2$ ,  $a_3 := -2\mathbf{f}_2^2$

$\alpha := 1 - \frac{a_0 + a_1 + a_2 + a_3}{a_1 + 2a_2 + 3a_3}$

if  $0 < \alpha < \alpha_{max}$  then (prevent extreme cases)

$t_{i+1} = t_i + \alpha \Delta t$ ,  $\mathbf{p}_{i+1} = \mathbf{c}(t_{i+1})$

**until**  $\|\mathbf{p}_i - \mathbf{p}_{i+1}\| < \varepsilon$ .

*foot point*  $\mathbf{f} = \mathbf{p}_{i+1}$ .

Suitable values for the boundaries are  $\varepsilon = 10^{-6}$  and  $\alpha_{max} = 20$ .

#### 4.5.3.2 Foot point algorithm for implicit curves

Let  $\Gamma : f(\mathbf{x}) = 0$  be a smooth planar implicit curve. We use procedure `curvepoint` of subsection 4.3.1

Let  $\mathbf{p}$  be a point in the vicinity of curve  $\Gamma$ . The following algorithm determines the *foot point* of  $\mathbf{p}$  on  $\Gamma$ :

(FP0)  $\mathbf{p}_0 = \text{curvepoint}(\mathbf{p})$

(FP1) **repeat**

$\mathbf{q}_i = \mathbf{p} - \frac{(\mathbf{p} - \mathbf{p}_i) \cdot \nabla f(\mathbf{p}_i)}{\|\nabla f(\mathbf{p}_i)\|^2} \nabla f(\mathbf{p}_i)$  (foot point on tangent line),

$\mathbf{p}_{i+1} = \text{curvepoint}(\mathbf{q}_i)$ ,  $\mathbf{f}_1 := \mathbf{q}_i - \mathbf{p}_i$ ,  $\mathbf{f}_2 := \mathbf{p}_{i+1} - \mathbf{q}_i$ ,

if  $\|\mathbf{q}_i - \mathbf{p}_i\| > \varepsilon$  then (one Newton step for the foot point on the tangent parabola  $\mathbf{x} = \mathbf{p}_i + \alpha \mathbf{f}_1 + \alpha^2 \mathbf{f}_2$ )

$a_0 := (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{f}_1$ ,  $a_1 := 2\mathbf{f}_2 \cdot (\mathbf{p} - \mathbf{p}_i) - \mathbf{f}_1^2$ ,  $a_2 := -3\mathbf{f}_1 \cdot \mathbf{f}_2$ ,  $a_3 := -2\mathbf{f}_2^2$

$\alpha := 1 - \frac{a_0 + a_1 + a_2 + a_3}{a_1 + 2a_2 + 3a_3}$

if  $0 < \alpha < \alpha_{max}$  then (prevent extreme cases)

$\mathbf{q}_i = \mathbf{p}_i + \alpha \mathbf{f}_1 + \alpha^2 \mathbf{f}_2$ ,  $\mathbf{p}_{i+1} = \text{curvepoint}(\mathbf{q}_i)$

**until**  $\|\mathbf{p}_i - \mathbf{p}_{i+1}\| < \varepsilon$ .

foot point  $\mathbf{f} = \mathbf{p}_{i+1}$ .

## 4.6 Bézier–curves

Bézier–curves play an important role in CAD. This section provides just an introduction. More material can be found in Chapter 12 and 13 and in books like FARIN: *Curves and Surfaces for CAGD* and HOSCHEK & LASSER: *Fundamentals of computer aided geometric design*.

Numerical simple planar curves are the following ones:  $\mathbf{x}(t) = (x(t), y(t))^T$ ,  $t_1 \leq t \leq t_2$ , where  $x(t)$  and  $y(t)$  are polynomes. For  $x(t) := a_0 + a_1 t + a_2 t^2 + \dots + a_n t^n$  and  $y(t) := b_0 + b_1 t + b_2 t^2 + \dots + b_n t^n$ , we get

$$\begin{aligned} \mathbf{x}(t) &= (a_0, b_0)^T + (a_1, b_1)^T t + \dots + (a_n, b_n)^T t^n. \\ &= \mathbf{a}_0 + \mathbf{a}_1 t + \dots + \mathbf{a}_n t^n \end{aligned}$$

with vectors  $\mathbf{a}_i := (a_i, b_i)^T$ .

I.g. there is no simple correlation between the points  $A_i : \mathbf{a}_i$  and the shape of the curve.

This disadvantage can be omitted by representing the polynomes  $x(t), y(t)$  in **Bernstein–base**  $\{B_0^n(t), B_1^n(t), \dots, B_n^n(t)\}$ :

$$B_i^n(t) := \binom{n}{i} t^i (1-t)^{n-i}, \quad 0 \leq i \leq n$$

(instead of the monom base  $\{1, t, t^2, \dots, t^n\}$ ).

Now  $n > 0$  is fixed and the vectors  $\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_n$  are the vertices of a polygon.

$\mathbf{x}(t) := \mathbf{b}_0 B_0^n(t) + \mathbf{b}_1 B_1^n(t) + \dots + \mathbf{b}_n B_n^n(t)$ ,  $0 \leq t \leq 1$ , is a **Bézier–curve** of (maximal) degree  $n$ . The polygon  $\mathbf{b}_0, \dots, \mathbf{b}_n$  is the **control polygon** of the Bézier–curve.

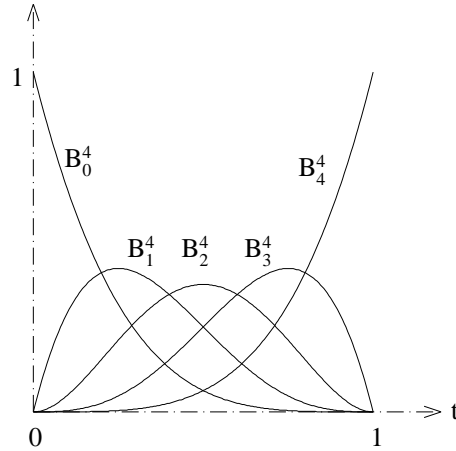
**Properties of the Bernstein–polynomes:**

(1)  $B_0^n(t) + B_1^n(t) + \dots + B_n^n(t) = 1$ ,

(2)  $B_0^n(0) = 1$ ,  $B_i^n(0) = 0$  for  $i > 0$ ,  $B_n^n(1) = 1$ ,  $B_i^n(1) = 0$  for  $i < n$ .

(3) Any Bernstein–polynom  $B_i^n$  has exactly one maximum: at  $t = i/n$ . This is the reason for the essential property:

A slight change of point  $\mathbf{b}_i$  effects the shape of the curve in the vicinity of  $\mathbf{x}(i/n)$  only.

Figure 4.12: Bernstein-polynoms  $B_i^4$ **Properties of a Bézier-curve:**

- (1)  $\mathbf{b}_0$  is the first point,  $\mathbf{b}_n$  is the last point,
- (2)  $\mathbf{b}_1 - \mathbf{b}_0$  is a tangent vector at the first point,  $\mathbf{b}_0 = \mathbf{x}(0)$ ,  $\mathbf{b}_n - \mathbf{b}_{n-1}$  is a tangent vector at the last point  $\mathbf{b}_n = \mathbf{x}(1)$ .
- (3) The polygon represents  $\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_n$  approximately the shape of the curve.

The evaluation of the polynoms  $x(t), y(t)$  can be done in a HORNER-like scheme (see FARIN: Curves and Surfaces, procedure `hornbez`):

```
function bezier_comp(degree: integer; coeff : r_array; t: real) : real;
{determines one component of a Bezier-curve (see FARIN)}
var i,n_choose_i : integer; fact,t1,aux : real;
begin
  t1:= 1-t; fact:=1; n_choose_i:= 1;
  aux:= coeff[0]*t1;
  for i:= 1 to degree-1 do
    begin
      fact:= fact*t;
      n_choose_i:= n_choose_i*(degree-i+1) div i;
      aux:= (aux + fact*n_choose_i*coeff[i])*t1;
    end;
  aux:= aux + fact*t*coeff[degree] ;
  bezier_comp:= aux;
end; bezier_comp
{*****}
```

**Remark:**

The procedure `bezier_comp` can be applied to a Bézier-curve  $(x(t), y(t), z(t))^T$  in  $\mathbb{R}^3$ , too.

## 4.7 Applications of the normalform

The normalform and their first two derivatives of a  $C^2$ -curve can be evaluated by using the foot point algorithms of section 4.5. Hence nearly any curve can be treated as an implicit one if necessary. This is an essential advantage for blending curves and for displaying bisector curves or offset curves of implicit curves.

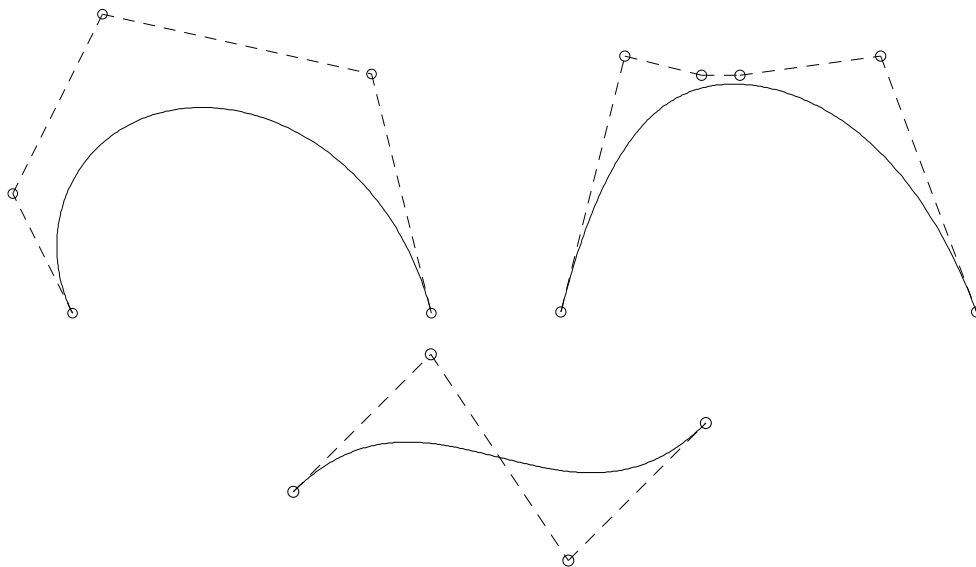


Figure 4.13: Bézier-curves and their control-polygons

### 4.7.1 Offset curves of implicit curves

Offset curves of a (planar) curve are just level curves of its normalform function  $h = c$  and can be traced by the marching algorithm for implicit curves (see section 4.3.1).

**Example 4.11** *Figure 4.14 shows the implicit curve  $f(\mathbf{x}) = x^4 + y^4 - 1 = 0$  (fat) and some offset curves. Recognize: some of the offset curves inside of  $f(x, y) = 0$  have singularities.*

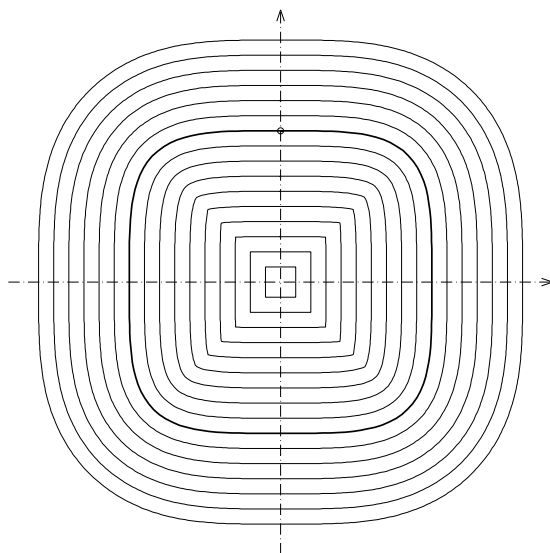


Figure 4.14: Offset curves of the implicit curve  $f(\mathbf{x}) = x^4 + y^4 - 1 = 0$

### 4.7.2 Bisector curves

The *bisector curves* or *Voronoi curves* of two curves  $\Gamma_1, \Gamma_2$  consist of the points, which have the same distances to both the curves  $\Gamma_1$  and  $\Gamma_2$ .

Let be  $\Gamma_1, \Gamma_2$  two regular  $C^2$  implicit or parametric curves and  $h_1 = 0, h_2 = 0$  their normalforms. Then the equations

$$h_1(\mathbf{x}) - h_2(\mathbf{x}) = 0, \quad h_1(\mathbf{x}) + h_2(\mathbf{x}) = 0$$

are implicit representations of the bisectors of  $\Gamma_1$  and  $\Gamma_2$ . Hence bisectors can be traced by the marching algorithm for implicit curves.

**Example 4.12** *Abb. 4.15 shows bisectors of two Bézier curves.*

The normalform of a point  $\mathbf{p}$  is just  $h(\mathbf{x}) = \|\mathbf{x} - \mathbf{p}\| = 0$ . Hence the bisectors involving a point can be treated in the same way.

**Example 4.13** *Figure 4.16 shows bisectors of*

- a) the ellipse  $x^2 + \frac{y^2}{4} = 1$  and point  $(-0.5, 1)$
- b) the curve  $x^4 + y^4 = 1$  and point  $(-0.4, 0.6)$ .

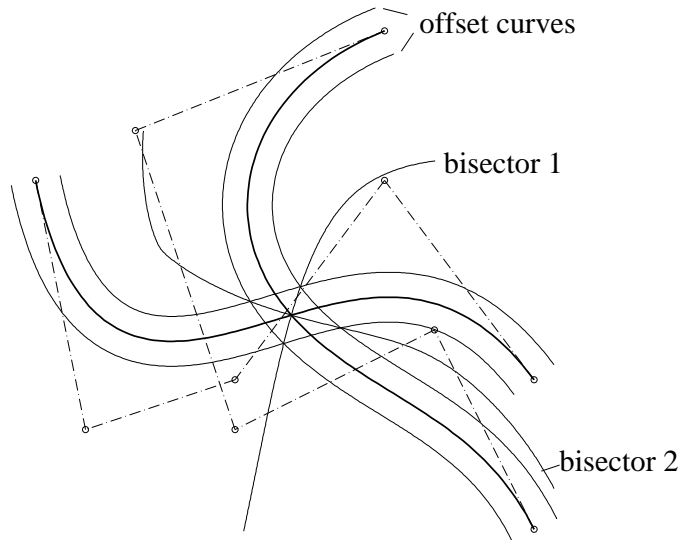


Figure 4.15: bisectors of two Bézier curves (fat)

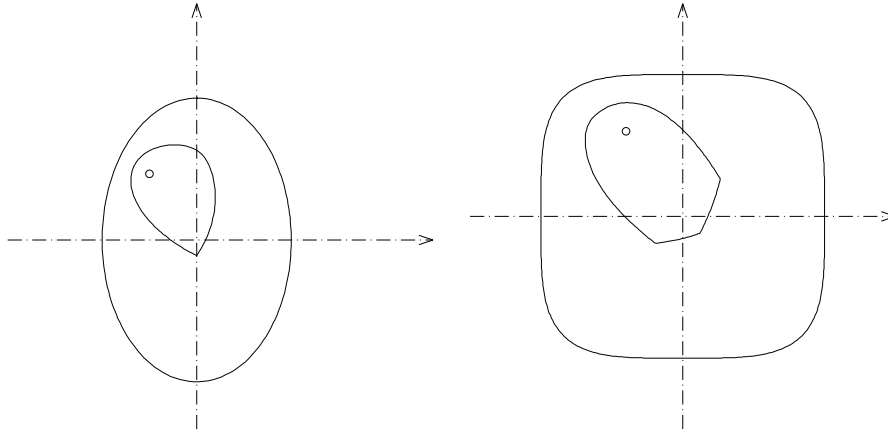


Figure 4.16: Bisectors of a point and a) an ellipse b) curve of 4-th degree

### 4.7.3 Numerical Parameterization of Curves

The length of a curve or any curve integral is defined for *parametric* curves, only. In order to determine such quantities for implicit curves we introduce a numeric parametrization of an implicit curve.

Let  $f(x, y) = 0$  be a regular implicit curve with normalform  $h(x, y) = 0$  and  $\Gamma : \mathbf{x} = \mathbf{c}(t), t \in [a, b]$  a parametric curve in the vicinity of curve  $f = 0$ . (For the application below  $\Gamma$  is chosen as a Bézier curve which is a rough approximation of an arc of  $f = 0$ .) Both curves are supposed to be of class  $C^2$ . Then the parametric curve

$$\Gamma_0 : \mathbf{x} = \mathbf{c}_0(t) := \mathbf{c}(t) - h(\mathbf{c}(t))\nabla h(\mathbf{c}(t)), \quad t \in [a, b]$$

is an arc of the given curve  $f = 0$ , because of  $h(\mathbf{c} - h(\mathbf{c})\nabla h(\mathbf{c})) = h(\mathbf{c}) - h(\mathbf{c}) = 0$  (see basic properties of the normalform function  $h$  above).

Differentiating yields  $\dot{\mathbf{c}}_0 = \dot{\mathbf{c}} - (\nabla h \nabla h^\top) \dot{\mathbf{c}} - h H_h \dot{\mathbf{c}} = (I - h H_h)(I - \nabla h \nabla h^\top) \dot{\mathbf{c}}$ , where  $\mathbf{x}^\top$  assigns the transpose of the column vector  $\mathbf{x}$ ,  $I$  the  $2 \times 2$  unit matrix and  $H_h$  the Hessian matrix of  $h$ . In the equation above  $h, \nabla h$  and  $H_h$  are evaluated at point  $\mathbf{c}(t)$ .

Vector  $(I - \nabla h \nabla h^\top) \dot{\mathbf{c}}$  is a tangent vector of the curve  $h = h(\mathbf{c}(t)) = \text{const}$  (offset curve of  $h = 0$ ) and can be expressed by  $(\dot{\mathbf{c}} \cdot \mathbf{r})\mathbf{r}$  with  $\mathbf{r} := (-h_y, h_x)$ . Hence

$$H_h(\mathbf{c})(I - \nabla h(\mathbf{c})\nabla h(\mathbf{c})^\top) \dot{\mathbf{c}} = H_h(\mathbf{c})(\dot{\mathbf{c}} \cdot \mathbf{r})\mathbf{r} = \kappa(\mathbf{c})(\dot{\mathbf{c}} \cdot \mathbf{r})\mathbf{r}.$$

Because  $\nabla h(\mathbf{c}) = \nabla h(\mathbf{c}_0)$  and  $\kappa(\mathbf{c}) = \frac{\kappa(\mathbf{c}_0)}{1+h(\mathbf{c})\kappa(\mathbf{c}_0)}$  (see Lemma 4.4, Theorem 4.1) we get

$$\dot{\mathbf{c}}_0 = \frac{(\dot{\mathbf{c}} \cdot \mathbf{r}(\mathbf{c}_0))\mathbf{r}(\mathbf{c}_0)}{1 + h(\mathbf{c})\kappa(\mathbf{c}_0)},$$

$\mathbf{c}, \dot{\mathbf{c}}$  and  $\mathbf{c}_0$  evaluated for parameter  $t$ .

$\mathbf{c}_0(t)$  is the foot point of  $\mathbf{c}(t)$  on curve  $h = 0$  and  $h(\mathbf{c})$  the oriented distance  $\nabla h(\mathbf{c}_0) \cdot (\mathbf{c} - \mathbf{c}_0)$  of point  $\mathbf{c}(t)$  to curve  $h = 0$ .

Figure 4.17 gives a geometric interpretation of the formula for the derivative  $\dot{\mathbf{c}}_0$ .

## 4.8 Space Curves

The definition of a space curve is analogously to the definition of a planar curve. An implicit representation as zero set of *one* equation is possible in rare cases. For example:  $x^2 + y^2 = 0$

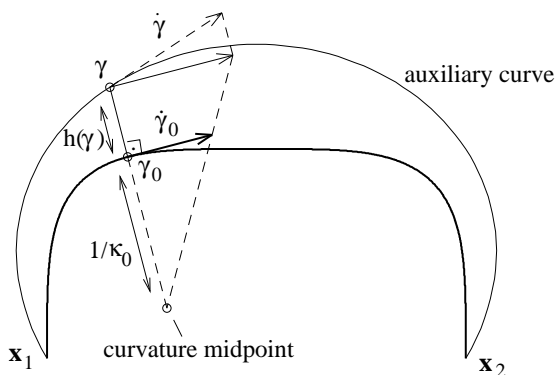


Figure 4.17: The derivative  $\dot{\mathbf{c}}_0$  of the parameterized curve

describes the z-axis. Curves as the solution of *two* equations can be considered as the intersection curves of two implicit surfaces, which will be discussed later.

**Definition 4.11** A space curve  $\Gamma$  is the image of a real interval  $I$  by a locally injective  $C^0$  function  $\mathbf{x}$  into  $\mathbb{R}^3$ :

$$\Gamma := \{\mathbf{x}(t) \mid t \in I\}.$$

**Definition 4.12** A regular space curve  $\Gamma$  is the image of a real interval  $I$  by a  $C^1$  function  $\mathbf{x}$  into  $\mathbb{R}^3$ :

$$\Gamma := \{\mathbf{x}(t) \mid t \in I\} \quad \text{and} \quad \dot{\mathbf{x}}(t) \neq \mathbf{0} \text{ for } t \in I.$$

$\dot{\mathbf{x}}(t)$  is a tangent vector at curve point  $\mathbf{x}(t)$ .

$\mathbf{x}$  is a regular parametric representation of  $\Gamma$ . The representation is of class  $C^n$  if  $\mathbf{x}(t)$  is  $C^n$ .

**Example 4.14**  $\mathbf{x}(t) = (\cos(t), \sin(t), t)^\top$ ,  $t \in [0, 1]$  (arc of a helix). The parametric representation is regular.

For a regular space curve there exists (theoretically) always an *arc length representation* analogously to the planar case, too: The characteristic property is  $\|\frac{d\mathbf{x}}{ds}\| = 1$ .

If  $\mathbf{x}(t)$  is  $C^2$  we have the essential property:  $\mathbf{x}' \cdot \mathbf{x}'' = 0$ .

The *curvature vector*  $\mathbf{x}''$  describes the change of the unit *tangent vector*  $\mathbf{t} := \mathbf{x}'$  with respect of the arc length. Its amount is called *curvature*  $k$  and if  $k \neq 0$  its unit direction denoted by  $\mathbf{n}$  and called *principal normal*:

$$\mathbf{x}'' = k\mathbf{n}, \quad \|\mathbf{n}\| = 1, \quad \mathbf{t} \cdot \mathbf{n} = 0.$$

A point with  $k = 0$  is called *inflection point*.

**Definition 4.13** For a non inflection point  $\mathbf{x}(s)$  of a  $C^3$  curve we define:

$\mathbf{b}(s) := \mathbf{t}(s) \times \mathbf{n}(s)$  is called *binormal*, plane with equation  $(\mathbf{x} - \mathbf{x}(s)) \cdot \mathbf{b}(s) = 0$  its *osculating plane*. (The osculating plane at a curve point contains the second order Taylor expansion of the curve.)

A measure for the change of the osculating plane is the *torsion*  $\tau$ . We observe:

$$\mathbf{b} \perp \mathbf{b}' = (\mathbf{t} \times \mathbf{n})' = \mathbf{t}' \times \mathbf{n} + \mathbf{t} \times \mathbf{n}' = \mathbf{t} \times \mathbf{n}' \perp \mathbf{t} \quad (\mathbf{t}' \text{ and } \mathbf{n} \text{ are parallel !})$$

and define  $\tau$  by  $\mathbf{b}' = -\tau\mathbf{n}$ .

The orthonormal frame  $\{\mathbf{t}, \mathbf{n}, \mathbf{b}\}$  is called *Frenet frame*.

From  $\mathbf{t}' = k\mathbf{n}$ ,  $\mathbf{b}' = -\tau\mathbf{n}$  and  $\mathbf{n}' = (\mathbf{b} \times \mathbf{t})' = \mathbf{b}' \times \mathbf{t} + \mathbf{b} \times \mathbf{t}' = -\tau\mathbf{n} \times \mathbf{t} + k\mathbf{b} \times \mathbf{n} = -k\mathbf{t} + \tau\mathbf{b}$  we get the *Frenet formulae*:

$$\mathbf{t}' = k\mathbf{n}, \quad \mathbf{n}' = -k\mathbf{t} + \tau\mathbf{b}, \quad \mathbf{b}' = -\tau\mathbf{n}.$$

Hence in case of  $\tau = 0$  for all curve points the curve is planar.

In practice curves are not given by their arc length parameterization. How to compute curvature, torsion,... in general is the content of the following lemma.

**Lemma 4.5** *Let  $\Gamma : \mathbf{x} = \mathbf{x}(t)$   $t \in [a, b]$  be a  $C^3$ -continuous regular space curve. For points where  $\dot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t)$  are linearly independent the following formulae are valid:*

$$\mathbf{t} = \frac{\dot{\mathbf{x}}(t)}{\|\dot{\mathbf{x}}(t)\|}, \quad \mathbf{b} = \frac{\dot{\mathbf{x}}(t) \times \ddot{\mathbf{x}}(t)}{\|\dot{\mathbf{x}}(t) \times \ddot{\mathbf{x}}(t)\|}, \quad \mathbf{n} = \mathbf{b} \times \mathbf{t}, \quad k = \frac{\|\dot{\mathbf{x}}(t) \times \ddot{\mathbf{x}}(t)\|}{\|\dot{\mathbf{x}}(t)\|^3}, \quad \tau = \frac{(\dot{\mathbf{x}}(t) \times \ddot{\mathbf{x}}(t)) \cdot \dot{\dot{\mathbf{x}}}(t)}{\|\dot{\mathbf{x}}(t) \times \ddot{\mathbf{x}}(t)\|^2}.$$

**Proof:** Analogously to the plane case we get for the curvature vector  $\mathbf{x}'' = \frac{\ddot{\mathbf{x}}^2 - (\dot{\mathbf{x}} \cdot \ddot{\mathbf{x}})\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|^4}$ . Applying formula:  $\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = \mathbf{b}(\mathbf{a} \cdot \mathbf{c}) - \mathbf{c}(\mathbf{a} \cdot \mathbf{b})$  yields

$$\mathbf{x}'' = \frac{\dot{\mathbf{x}} \times (\ddot{\mathbf{x}} \times \dot{\mathbf{x}})}{\|\dot{\mathbf{x}}\|^4} \quad \text{and} \quad k = \|\mathbf{x}''\| = \frac{\|\dot{\mathbf{x}}\| \|\ddot{\mathbf{x}} \times \dot{\mathbf{x}}\|}{\|\dot{\mathbf{x}}\|^4} = \frac{\|\ddot{\mathbf{x}} \times \dot{\mathbf{x}}\|}{\|\dot{\mathbf{x}}\|^3} \quad (\dot{\mathbf{x}} \perp (\ddot{\mathbf{x}} \times \dot{\mathbf{x}}) !).$$

$$\text{Hence } \mathbf{n} = \frac{\mathbf{x}''}{k} = \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|} \times \frac{\ddot{\mathbf{x}} \times \dot{\mathbf{x}}}{\|\ddot{\mathbf{x}} \times \dot{\mathbf{x}}\|} \quad \text{and} \quad \mathbf{b} = \frac{\dot{\mathbf{x}} \times \ddot{\mathbf{x}}}{\|\dot{\mathbf{x}} \times \ddot{\mathbf{x}}\|}.$$

$$\text{From } \mathbf{b}' = \dot{\mathbf{b}}/\|\dot{\mathbf{x}}\| \text{ we get } \mathbf{b}' = \frac{(\dot{\mathbf{x}} \times \dot{\mathbf{x}})(\dot{\mathbf{x}} \times \ddot{\mathbf{x}}) - (\dot{\mathbf{x}} \times \ddot{\mathbf{x}})((\dot{\mathbf{x}} \times \dot{\mathbf{x}}) \cdot (\dot{\mathbf{x}} \times \ddot{\mathbf{x}}))}{\|\dot{\mathbf{x}}\| \|\dot{\mathbf{x}} \times \ddot{\mathbf{x}}\|^3} \text{ and}$$

$$-\tau = \mathbf{b}' \cdot \mathbf{n} = \mathbf{b}' \cdot \left( \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|} \times \frac{\ddot{\mathbf{x}} \times \dot{\mathbf{x}}}{\|\ddot{\mathbf{x}} \times \dot{\mathbf{x}}\|} \right) = \frac{\dot{\mathbf{x}} \times \ddot{\mathbf{x}}}{\|\dot{\mathbf{x}}\| \|\ddot{\mathbf{x}} \times \dot{\mathbf{x}}\|} \cdot \left( \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|} \times \frac{\ddot{\mathbf{x}} \times \dot{\mathbf{x}}}{\|\ddot{\mathbf{x}} \times \dot{\mathbf{x}}\|} \right) = \frac{(\dot{\mathbf{x}} \times \ddot{\mathbf{x}}) \cdot (\dot{\mathbf{x}} \times (\ddot{\mathbf{x}} \times \dot{\mathbf{x}}))}{\|\ddot{\mathbf{x}} \times \dot{\mathbf{x}}\|^2 \|\dot{\mathbf{x}}\|^2}.$$

$$\text{Applying the identity: } (\mathbf{a} \times \mathbf{b}) \cdot (\mathbf{c} \times \mathbf{d}) = (\mathbf{a} \cdot \mathbf{c})(\mathbf{b} \cdot \mathbf{d}) - (\mathbf{b} \cdot \mathbf{c})(\mathbf{a} \cdot \mathbf{d}) \text{ yields: } -\tau = -\frac{(\dot{\mathbf{x}} \times \ddot{\mathbf{x}}) \cdot \dot{\mathbf{x}}}{\|\ddot{\mathbf{x}} \times \dot{\mathbf{x}}\|^2}. \quad \square$$

For *displaying* a space curve we use the methods and formulae for equally and curvature dependent distribution of points developed for planar curves.

The *normalform of a space curve* will be introduced later (see Section 5.9).



# Chapter 5

## SURFACES

### 5.1 Definition and Representations of Surfaces

**Definition 5.1** A surface  $\Phi$  is the image of a rectangle  $D := [a, b, ] \times [c, d] \subset \mathbb{R}^2$  by a locally injective  $C^0$  function  $\mathbf{x}$  into  $\mathbb{R}^3$ :

$$\Phi := \{\mathbf{x}(u, v) \mid (u, v) \in D\}.$$

For  $(u_0, v_0) \in D$  curves  $\mathbf{x} = \mathbf{x}(u_0, v)$  and  $\mathbf{x} = \mathbf{x}(u, v_0)$  are the parameter curves at point  $\mathbf{x}(u_0, v_0)$ .

**Definition 5.2** A regular surface  $\Phi$  is the image of a rectangle  $D := [a, b, ] \times [c, d] \subset \mathbb{R}^2$  by a locally injective  $C^1$  function  $\mathbf{x}$  into  $\mathbb{R}^3$ :

$$\Phi := \{\mathbf{x}(u, v) \mid (u, v) \in D\}, \quad \mathbf{x}_u \times \mathbf{x}_v \neq \mathbf{0}.$$

$\mathbf{x}_u, \mathbf{x}_v$  are the tangent vectors of the parameter curves and  $\mathbf{x}_u \times \mathbf{x}_v$  is a normal vector (orthogonal to the tangent plane).

**Example 5.1**  $\mathbf{x}(u, v) = (u, v, u^2 + v^2)^\top$ ,  $(u, v) \in D := [0, 1] \times [0, 1]$  (patch of the paraboloid  $z = x^2 + y^2$ ). The parametric representation is regular.

**Definition 5.3** A regular implicit surface is a non empty subset  $\Phi \subset \mathbb{R}^3$  of the form

$$f : \mathbb{R}^2 \supset D \rightarrow \mathbb{R}, \quad \Phi := \{\mathbf{x} \in D \mid f(\mathbf{x}) = 0\}, \quad \nabla f(\mathbf{x}) \neq \mathbf{0} \text{ for } \mathbf{x} \in \Phi,$$

where  $f$  is a  $C^1$ -function. For a surface point  $\mathbf{x}$  vector  $\nabla f(\mathbf{x})$  is a normal vector.

$f(\mathbf{x}) = 0$  is a regular implicit representation of  $\Phi$ . The representation is of class  $C^n$  if  $f$  is  $C^n$ .

**Example 5.2**  $\Phi := \{\mathbf{x} \in \mathbb{R}^3 \mid f(\mathbf{x}) := x^2 + y^2 + z^2 - 1 = 0\}$  (unit sphere).

Obviously, the parametric/implicit representation of a surface is not unique.

**Definition 5.4** The representation of a surface is called explicit if one of the coordinates  $x, y, z$  is a function of the remaining coordinates.

The following (theoretical) results show that the representations can be changed locally. The proofs rely on the *implicit function theorem*.

**Lemma 5.1** a) Let  $\Phi$  be a regular  $C^n$ -continuous parametric (implicit) surface and  $\mathbf{x}_0$  a point of  $\Phi$ . Then there exists a local  $C^n$ -continuous implicit (parametric) representation of  $\Phi$  at point  $\mathbf{x}_0$  which is regular, too.

b) For any surface with a regular parametric or regular implicit representation there exists for any point  $\mathbf{x}_0$  locally an explicit representation which is  $C^n$  if the given representation is  $C^n$ .

From an explicit representation one derives easily both a parametric and an implicit representation. (For example: From  $z = Z(x, y)$  we get the parametric representation  $(x, y, Z(x, y))$  and the implicit representation  $Z(x, y) - z = 0$ .)

## 5.2 The First and Second Fundamental Forms of a Surface

### 5.2.1 The first fundamental form, arc length

**Definition 5.5** Let  $\Phi : \mathbf{x} = \mathbf{x}(u, v)$ ,  $u, v \in D \subset \mathbb{R}^2$  be a regular surface and  $u = u(t), v = v(t), t \in [t_1, t_2]$  a regular curve in  $D$ . Hence  $\Gamma : \mathbf{x} = \mathbf{c}(t) := \mathbf{x}(u(t), v(t)), t \in [t_1, t_2]$  is a regular curve on the surface. For the arc length of curve  $\Gamma$  we get:

$$s(t) := \int_{t_1}^t \sqrt{\dot{\mathbf{c}}^2} dt = \int_{t_1}^t \sqrt{E\dot{u}^2 + 2F\dot{u}\dot{v} + G\dot{v}^2} dt,$$

where

$$E(u, v) := \mathbf{x}_u^2, \quad F(u, v) := \mathbf{x}_u \cdot \mathbf{x}_v, \quad G(u, v) := \mathbf{x}_v^2$$

are the coefficients of the first fundamental form and  $\frac{ds}{dt} = \sqrt{E\dot{u}^2 + 2F\dot{u}\dot{v} + G\dot{v}^2} \neq 0$ . Hence  $E, F, G$  are the coefficients of a strongly positive definite quadratic form:  $I(\xi, \eta) := E\xi^2 + 2F\xi\eta + G\eta^2$ .

**Remark:**

The answer of any question about distances on a surface needs the 1.fundamental form.

### 5.2.2 The second fundamental form, curvature

Let  $\Phi : \mathbf{x} = \mathbf{x}(u, v)$ ,  $u, v \in D \subset \mathbb{R}^2$  be a regular surface and  $\Gamma : \mathbf{x} = \mathbf{c}(s) := \mathbf{x}(u(s), v(s)), s \in [s_1, s_2]$  a regular curve on surface  $\Phi$  parameterized by its arc length  $s$ . Hence

$\mathbf{c}' = \mathbf{x}_u u' + \mathbf{x}_v v'$  is the unit tangent vector and

$\mathbf{c}'' = \mathbf{x}_{uu} u'^2 + \mathbf{x}_{vv} v'^2 + 2\mathbf{x}_{uv} u'v' + \mathbf{x}_{uu} u'v'^2 + \mathbf{x}_{vv} v'^2$  its curvature vector.

$\mathbf{c}''$  is orthogonal to the tangent vector  $\mathbf{c}'$  and can be split into its normal and tangential component (with respect of surface  $\Phi$ ):

$\mathbf{c}'' = k_n \mathbf{n} + k_g \mathbf{m}$  where  $\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|}$  is the surface unit normal and  $\mathbf{m} := \mathbf{c}' \times \mathbf{n}$  (tangential to the surface).  $k_n$  is called *normal curvature* and  $k_g$  *geodesic curvature* of the surface curve  $\Gamma$ .

For measuring the curvature of the surface at a surface point  $\mathbf{x}(u, v)$  one considers the intersection curves of the surface with the planes orthogonal to the surface at point  $\mathbf{x}(u, v)$ . For such *normal sections* curvature vector  $\mathbf{c}''$  is parallel to the surface normal, i.e.  $k_g = 0$ . For its curvature we get  $k = k_n = \mathbf{c}'' \cdot \mathbf{n} = Lu'^2 + 2Mu'v' + Nv'^2$  where

$$L(u, v) := \mathbf{n} \cdot \mathbf{x}_{uu}, \quad M(u, v) := \mathbf{n} \cdot \mathbf{x}_{uv}, \quad N(u, v) := \mathbf{n} \cdot \mathbf{x}_{vv}.$$

$L, M, N$  are called the coefficients of the *second fundamental form*.

$L, M, N$  can be considered as the coefficients of a quadratic form:  $II(\xi, \eta) := L\xi^2 + 2M\xi\eta + N\eta^2$ .

For an *arbitrary parameterization* we get from  $u' = \frac{du}{dt} \frac{dt}{ds}$ ,  $v' = \frac{dv}{dt} \frac{dt}{ds}$ ,  $\frac{ds}{dt} = \sqrt{E\dot{u}^2 + 2F\dot{u}\dot{v} + G\dot{v}^2}$  the formula for the *normal curvature* of the surface in tangential direction  $\mathbf{c}' = \mathbf{x}_u du + \mathbf{x}_v dv$ :

$$k_n = \frac{Ldu^2 + 2Mdudv + Ndv^2}{Edu^2 + 2Fdudv + Gdv^2} = \frac{II(du, dv)}{I(du, dv)}.$$

A point with constant normal curvature is called an *umbilic point* or a *flat point* if  $k_n = 0$ .

If the normal curvature is not constant (at a point) there exist a maximum and a minimum. For determining these extrem curvatures and the tangential direction of the corresponding normal section we differentiate function (subscript “n” is omitted)  $k(\xi, \eta) := II(\xi, \eta)/I(\xi, \eta)$  and we get from

$$k_\xi = 2 \frac{(L\xi + M\eta)I - (E\xi + F\eta)II}{I^2} = 2 \frac{(L\xi + M\eta) - (E\xi + F\eta)k}{I} = 0,$$

$$k_\eta = 2 \frac{(M\xi + N\eta)I - (F\xi + G\eta)II}{I^2} = 2 \frac{(M\xi + N\eta) - (F\xi + G\eta)k}{I} = 0,$$

the linear system

$$\begin{aligned} (L - kE)\xi + (M - kF)\eta &= 0 \\ (M - kF)\xi + (N - kG)\eta &= 0 \end{aligned}$$

which has non trivial solutions if its determinant is 0. This yields the quadratic equation for  $k$ :

$$k^2 - \frac{EN - 2FM + GL}{EG - F^2}k + \frac{LN - M^2}{EG - F^2} = 0.$$

From the theorem of VIETA we get the *main curvatures*:

$$k_{1/2} = H \pm \sqrt{H^2 - K}$$

where

$$H := \frac{1}{2} \frac{EN - 2FM + GL}{EG - F^2} = \frac{k_1 + k_2}{2} \text{ is the mean curvature and}$$

$$K := \frac{LN - M^2}{EG - F^2} = k_1 k_2 \text{ the Gaussian curvature at the surface point of consideration.}$$

The tangential direction of the normal section with main curvature  $k_i$  is  $\mathbf{t}_i := \xi_i \mathbf{x}_u + \eta_i \mathbf{x}_v$ , with  $(\xi_i, \eta_i)$  a non trivial solution of

$$\begin{aligned} (L - k_i E)\xi + (M - k_i F)\eta &= 0 \\ (M - k_i F)\xi + (N - k_i G)\eta &= 0 \end{aligned}$$

In case of  $k_1 \neq k_2$  the vectors  $\mathbf{t}_1, \mathbf{t}_2$  define the *main or principle directions*.

In order to prove that  $\mathbf{t}_1, \mathbf{t}_2$  are **perpendicular**, we assume (without loss of generality)  $F = \mathbf{x}_u \cdot \mathbf{x}_v = 0$ . Hence in case of  $M \neq 0$  we get from  $(\xi_1, \eta_1) = (-M, L - k_1 E)$ ,  $(\xi_2, \eta_2) = (-M, L - k_2 E)$ , the equation  $\mathbf{t}_1 \cdot \mathbf{t}_2 = \xi_1 \xi_2 E + \eta_1 \eta_2 G = \dots = 0$  using  $k_1 k_2 = (LN - M^2)/EG$ ,  $k_1 + k_2 = (EN + GL)/EG$ . In case of  $M = 0$  we have  $k_1 = L/E$ ,  $k_2 = N/G$  and  $(\xi_1, \eta_1) = (1, 0)$ ,  $(\xi_2, \eta_2) = (0, 1)$  which yields  $\mathbf{t}_1 \cdot \mathbf{t}_2 = 0$ .

For theoretical considerations occasionally we assume (without loss of generality) that  $\mathbf{x}_u, \mathbf{x}_v$  are parallel to the principal directions: We assume  $\mathbf{x}_u = \mathbf{t}_1$ ,  $\mathbf{x}_v = \mathbf{t}_2$ . In this case the parameterlines are called *lines of curvature*. Essential advantages of such a parametrization are:

$F = 0$  (because principal directions are mutually orthogonal) and

$M = 0$ ,  $k_1 = L/E$ ,  $k_2 = N/G$ , because  $(1, 0), (0, 1)$  are solutions of the linear systems above containing  $k_1$  and  $k_2$  respectively.

**Definition 5.6** *The surface points can be classified with respect of their Gaussian curvature:*

- 1) If  $K > 0$ , the point is called elliptical (all normalcurvatures are not 0 and have the same sign),
- 2) if  $K < 0$ , the point is called hyperbolic (there are positiv and negativ normal curvatures),
- 3) If  $K = 0$ , the point is called parabolic (one main curvature is 0).

**Example 5.3** *Any point of*

- 1) an ellipsoid or a paraboloid is elliptical ,
- 2) a hyperboloid of one sheet is hyperbolic,
- 3) a cylinder or a cone is parabolic. (see Figure 5.1)

For derivatives of offset surfaces (see Section 5.3) the derivatives of the surface unit normal  $\mathbf{n}$  are necessary:

From  $\mathbf{n}^2 = 1$  we know  $\mathbf{n} \cdot \mathbf{n}_u = 0$  and  $\mathbf{n} \cdot \mathbf{n}_v = 0$ . Hence  $\mathbf{n}_u, \mathbf{n}_v$  are linear combinations of the tangent vectors  $\mathbf{x}_u, \mathbf{x}_v$ . Differentiating the equations  $\mathbf{n} \cdot \mathbf{x}_u = 0$ ,  $\mathbf{n} \cdot \mathbf{x}_v = 0$  yields

$$L = \mathbf{n} \cdot \mathbf{x}_{uu} = -\mathbf{n}_u \cdot \mathbf{x}_u, \quad M = \mathbf{n} \cdot \mathbf{x}_{uv} = -\mathbf{n}_u \cdot \mathbf{x}_v = -\mathbf{n}_v \cdot \mathbf{x}_u, \quad N = \mathbf{n} \cdot \mathbf{x}_{vv} = -\mathbf{n}_v \cdot \mathbf{x}_v.$$

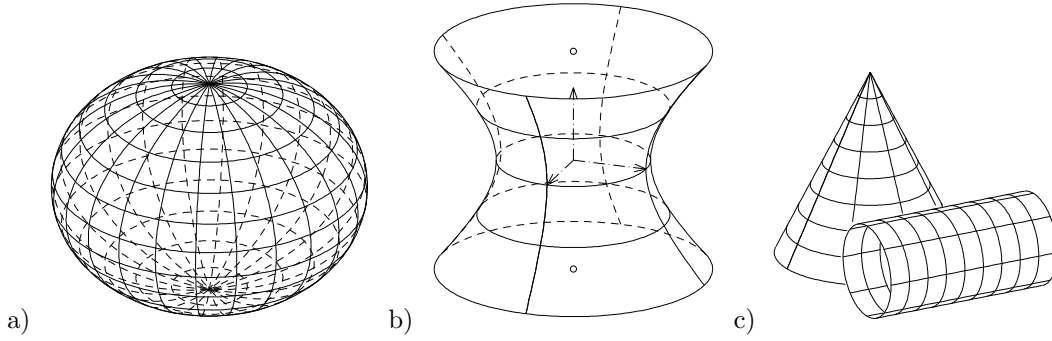


Figure 5.1: a) ellipsoid, b) hyperboloid of one sheet, c) cylinder, cone

These equations allow to determine the components of  $\mathbf{n}_u, \mathbf{n}_v$  with respect of the vectors  $\mathbf{x}_u, \mathbf{x}_v$ :

$$\mathbf{n}_u = \frac{MF - LG}{EG - F^2} \mathbf{x}_u + \frac{LF - ME}{EG - F^2} \mathbf{x}_v, \quad \mathbf{n}_v = \frac{NF - MG}{EG - F^2} \mathbf{x}_u + \frac{MF - NE}{EG - F^2} \mathbf{x}_v.$$

(Weingarten equations).

In case of a curvature line parametrization we have:  $\mathbf{n}_u = -\frac{L}{E} \mathbf{x}_u = -k_1 \mathbf{x}_u$ ,  $\mathbf{n}_v = -\frac{N}{G} \mathbf{x}_v = -k_2 \mathbf{x}_v$ .

### 5.3 Offset surfaces

**Definition 5.7** Let  $\Phi_0 : \mathbf{x} = \mathbf{x}_0(u, v)$  be a regular  $C^2$ -surface and  $\mathbf{n}_0(u, v)$  the unit normal and  $d \in \mathbb{R}$  fixed. Then surface  $\Phi_d : \mathbf{x} = \mathbf{x}_0(u, v) + d\mathbf{n}_0(u, v)$  is called offset surface with distance  $d$ . The offset surface  $\Phi_d$  depends only on the surface  $\Phi_0$  as a point set in  $\mathbb{R}^3$  and distance  $d$ .

Analogously to the curve case we get

**Lemma 5.2** Let  $\Phi_0 : \mathbf{x} = \mathbf{x}_0(u, v)$  be a regular  $C^n$ -surface,  $n \geq 2$ ,  $k_{0,1}(u, v), k_{0,2}(u, v)$  the main curvatures and  $d \in \mathbb{R}$  such that  $(1 - dk_{0,1}(u, v))(1 - dk_{0,2}(u, v)) > 0$  then the offset surface  $\Phi_d : \mathbf{x} = \mathbf{x}_d := \mathbf{x}_0(u, v) + d\mathbf{n}_0(u, v)$  is a regular  $C^{n-1}$  surface,  $\mathbf{n}_d(u, v) = \mathbf{n}_0(u, v)$  and the main curvatures are

$$k_{d,1} = \frac{k_{0,1}}{1 - dk_{0,1}}, \quad k_{d,2} = \frac{k_{0,2}}{1 - dk_{0,2}}.$$

**Proof:** Let be  $\mathbf{x}_0(u, v)$  a curvature line parameterization. Hence:  $\mathbf{n}_{0,u} = -k_1 \mathbf{x}_{0,u}$ ,  $\mathbf{n}_{0,v} = -k_2 \mathbf{x}_{0,v}$  and  $\mathbf{x}_{0,du} \times \mathbf{x}_{0,dv} = (\mathbf{x}_{0,u} + d\mathbf{n}_{0,u}) \times (\mathbf{x}_{0,v} + d\mathbf{n}_{0,v}) = (1 - dk_{0,1})(1 - dk_{0,2}) \mathbf{x}_{0,u} \times \mathbf{x}_{0,v}$ .

For  $(1 - dk_{0,1}(u, v))(1 - dk_{0,2}(u, v)) > 0$  the offset surface  $\Phi_d$  is regular and of class  $C^{n-1}$ .

Obviously, a normal plane  $\varepsilon_n$  of  $\Phi_0$  at point  $\mathbf{x}_0(u, v)$  is a normal plane of  $\Phi_d$  and the intersection  $\Phi_d \cap \varepsilon_n$  is an offset curve of  $\Phi_0 \cap \varepsilon_n$ . Hence for corresponding normal curvatures we get  $k_{dn} = \frac{k_{0n}}{1 - dk_{0n}}$  (see Lemma 4.4). Because the stationary points of  $k_{dn}(\xi, \eta)$  and  $k_{0n}(\xi, \eta)$  agree, we get  $k_{d,1} = \frac{k_{0,1}}{1 - dk_{0,1}}$ ,  $k_{d,2} = \frac{k_{0,2}}{1 - dk_{0,2}}$  and  $\mathbf{t}_{d,1} = \mathbf{t}_{0,1}$ ,  $\mathbf{t}_{d,2} = \mathbf{t}_{0,2}$  (directions of main curvatures).  $\square$

### 5.4 Normalform of a surface

#### 5.4.1 Definition of the normalform

**Definition 5.8** The implicit representation  $h(\mathbf{x}) = \mathbf{n} \cdot \mathbf{x} - d = 0, \|\mathbf{n}\| = 1$  of a plane  $\varepsilon$  in  $\mathbb{R}^3$  is called HESSE normalform of plane  $\varepsilon$ .

( $h(\mathbf{x})$  describes the oriented distance of point  $\mathbf{x}$  to the plane  $\varepsilon$ .)

**Definition 5.9** An implicit representation  $h(\mathbf{x}) = 0$  of a surface  $\Phi$  with  $\|\nabla h\| = 1$  in a vicinity of  $\Phi$  is called normalform of surface  $\Phi$ . (We regard  $h = 0$  and  $-h = 0$  as the same normalforms.)  $h$  is called normalform function or, because of its geometric meaning, oriented distance function.

**Example 5.4**  $h(x, y, z) := \sqrt{x^2 + y^2 + z^2} - r = 0$  is the normalform of the sphere  $x^2 + y^2 + z^2 = r^2$ .

If the given surface is smooth enough, then its normalform always exists (theoretically).

**Theorem 5.1** For any  $C^n$ -continuous,  $n \geq 2$ , surface  $\Phi$  there exists in a vicinity  $V$  of  $\Phi$  the oriented distance function  $h$  and  $h$  is of class  $C^n$ , too. Hence  $\Phi$  has an implicit representation  $h(\mathbf{x}) = 0$  such that  $\|\nabla h\| = 1$  on  $V$ .

On  $V$  function  $h$  has the properties: 1)  $h(\mathbf{x} + \delta \nabla h(\mathbf{x})) = h(\mathbf{x}) + \delta$ , 2)  $\nabla h(\mathbf{x} + \delta \nabla h(\mathbf{x})) = \nabla h(\mathbf{x})$ .

**Proof:** Let  $\Phi : \mathbf{x} = \mathbf{x}(u, v)$  be a curvature line parametrization such that:  $\mathbf{x}_u = \mathbf{t}_1$ ,  $\mathbf{x}_v = \mathbf{t}_2$ . Hence at any surface point we have  $\mathbf{n}_u = -k_1 \mathbf{x}_u$ ,  $\mathbf{n}_v = -k_2 \mathbf{x}_v$ .

The equation  $F(\mathbf{x}, u, v, d) := \mathbf{x}(u, v) + d\mathbf{n}(u, v) - \mathbf{x} = 0$  is solvable for  $u, v, d$  if  $\det(F_u, F_v, F_d) \neq 0$ . From

$$F_u = \mathbf{x}_u + d\mathbf{n}_u = (1 - dk_1)\mathbf{x}_u, \quad F_v = \mathbf{x}_v + d\mathbf{n}_v = (1 - dk_2)\mathbf{x}_v, \quad F_d = \mathbf{n}$$

we get  $\det(F_u, F_v, F_d) = (1 - dk_1)(1 - dk_2)\det(\mathbf{x}_u, \mathbf{x}_v, \mathbf{n}) \neq 0$  in a vicinity  $V : -v \leq d \leq v, v > 0$  of surface  $\Phi$ .

Due to the implicit function theorem there exist functions  $u(x, y, z), v(x, y, z)$  and  $d(x, y, z)$  such that the following equation is fulfilled in the vicinity  $V$  of surface  $\Phi$ :

$\mathbf{x}(u(x, y, z), v(x, y, z)) + d(x, y, z)\mathbf{n}(u(x, y, z), v(x, y, z)) = \mathbf{x}$ . Differentiating this equation for  $x, y, z$  yields  $\nabla u \mathbf{x}_u^\top + \nabla v \mathbf{x}_v^\top + \nabla d \mathbf{n}^\top + d \nabla \mathbf{n}_u^\top + d \nabla \mathbf{n}_v^\top = 3 \times 3$  unit matrix (see curve case!) Multiplying this matrix equation by  $\mathbf{n}$  and respecting  $\mathbf{x}_u^\top \mathbf{n} = \mathbf{x}_v^\top \mathbf{n} = \mathbf{n}_u^\top \mathbf{n} = \mathbf{n}_v^\top \mathbf{n} = 0$ ,  $\mathbf{n}^\top \mathbf{n} = 1$  we get  $\nabla d = \mathbf{n}$ . Hence function  $d(x, y, z)$  is of class  $C^n$ , too.

In order to distinguish function  $d(x, y, z)$  and distance parameter  $d$  we set  $h(x, y, z) := d(x, y, z)$ .

Points  $\mathbf{x}$  with the same foot point  $\mathbf{x}_0$  have the same gradient  $\nabla h(\mathbf{x})$ . Hence we get the essential properties of the distance function  $h$ :

1)  $h(\mathbf{x} + \delta \nabla h(\mathbf{x})) = h(\mathbf{x}) + \delta$ , 2)  $\nabla h(\mathbf{x} + \delta \nabla h(\mathbf{x})) = \nabla h(\mathbf{x})$ . □

The proof of the uniqueness of the normalform function  $h$  is omitted.

Properties 1), 2) of the last theorem show that  $h$  and  $\nabla h$  can be evaluated numerically at point  $\mathbf{x}$  by determining the foot point  $\mathbf{x}_0$  and its unit normal  $\mathbf{n}_0$  of point  $\mathbf{x}$  on surface  $\Phi$ :

$$h(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_0) \cdot \mathbf{n}_0, \quad \nabla h(\mathbf{x}) = \mathbf{n}_0.$$

### 5.4.2 On the first and second derivatives of the normalform of a surface

Let be  $\Phi$  a  $C^2$ -continuous surface with normalform  $h = 0$  and  $\Gamma : \mathbf{x} = \mathbf{c}(s)$  a  $C^2$ -continuous curve on  $\Phi$  with arc length parameter  $s$ . Hence  $h(\mathbf{c}(s)) = 0$ . Differentiating this equation yields

$$(1) \quad \nabla h \cdot \mathbf{c}' = 0, \quad (2) \quad \mathbf{c}'^\top H_h \mathbf{c}' + \nabla h \cdot \mathbf{c}'' = 0.$$

From the second equation we get the following results:

- The normal curvature of surface curve  $\Gamma$  is:  $k_n = \nabla h \cdot \mathbf{c}'' = -\mathbf{c}'^\top H_h \mathbf{c}'$ .  
 $k_n$  depends only on the surface and the tangential direction of  $\Gamma$  at the point of consideration.
- In case of a normal section we get  $k = k_n = \nabla h \cdot \mathbf{c}'' = -\mathbf{c}'^\top H_h \mathbf{c}'$ . Hence the normal curvature of surface  $\Phi$  for tangential direction  $\mathbf{v}$ ,  $\|\mathbf{v}\| = 1$ , is  $k_n = -\mathbf{v}^\top H_h \mathbf{v}$ .  
In following considerations for simplicity we deal with the negative normal curvature  $\kappa := -k_n = \mathbf{v}^\top H_h \mathbf{v}$ .

- Differentiating  $(\nabla h)^2 = 1$  yields  $H_h \nabla h = \mathbf{0}$ .  
The *eigenvalues* of the Hessian matrix  $H_h$  are  
 $\lambda_1 = 0$  with eigenvector  $\nabla h$ ,  
 $\lambda_2 = \kappa_1$ ,  $\lambda_3 = \kappa_2$  with eigenvectors  $\mathbf{v}_1 \perp \mathbf{v}_2$ ,  $\|\mathbf{v}_i\| = 1$ .  
From  $\mathbf{v}_i^\top H_h \mathbf{v}_i = \kappa_i$  we get:  $k_i := -\kappa_i$  is the normal curvature in direction  $\mathbf{v}_i$ . The normal curvature in an arbitrary direction  $\mathbf{v}(\varphi) = \cos(\varphi)\mathbf{v}_1 + \sin(\varphi)\mathbf{v}_2$ ,  $0 \leq \varphi < \pi$ , is  
 $k_n(\varphi) = -\mathbf{v}(\varphi)^\top H_h \mathbf{v}(\varphi) = k_1 \cos(\varphi)^2 + k_2 \sin(\varphi)^2 = k_1 + (k_2 - k_1) \sin(\varphi)^2$ .  
In case of  $k_1 = k_2$  the normal curvature is constant (umbilic case).  
In case of  $k_1 \neq k_2$  the extreme values of  $k_n(\varphi)$ , hence  $k_1 = -\kappa_1$  and  $k_2 = -\kappa_2$  are the main curvatures. The corresponding main directions are the eigen directions  $\mathbf{v}_1, \mathbf{v}_2$  respectively.  
The formula  $k_n(\varphi) = k_1 \cos(\varphi)^2 + k_2 \sin(\varphi)^2$  is called *Euler formula*.
- The *characteristic polynomial* of  $H_h$  is  $-\lambda^3 + (\kappa_1 + \kappa_2)\lambda^2 - \kappa_1\kappa_2\lambda$  with  
*mean curvature*  
 $H = \frac{\kappa_1 + \kappa_2}{2} = -\frac{1}{2}(\kappa_1 + \kappa_2) = -\frac{1}{2}(h_{xx} + h_{yy} + h_{zz})$  and  
*Gaussian curvature*  
 $K = k_1 k_2 = \kappa_1 \kappa_2 = \begin{vmatrix} h_{xx} & h_{xy} \\ h_{yx} & h_{yy} \end{vmatrix} + \begin{vmatrix} h_{xx} & h_{xz} \\ h_{zx} & h_{zz} \end{vmatrix} + \begin{vmatrix} h_{yy} & h_{yz} \\ h_{zy} & h_{zz} \end{vmatrix}$ .  
The *minimal polynomial* is  
the characteristic polynomial if  $\kappa_1 \neq \kappa_2$  (*non umbilic case*) and  $\lambda^2 - \kappa\lambda$  if  $\kappa := \kappa_1 = \kappa_2$  (*umbilic case*).  
(Prove it after diagonalizing  $H_h$  !)
- Let  $\Phi_0$  be a  $C^2$ -continuous surface with main curvatures  $k_1, k_2$  at  $\mathbf{x} \in \Phi_0$ . The distance parameter  $d \in \mathbb{R}$  is chosen in an interval  $[-d_0, d_0]$ ,  $d_0 > 0$ , such that for any  $d$  we have  $1 - dk_1(\mathbf{x}) > 0$ ,  $1 - dk_2(\mathbf{x}) > 0$ . Hence  $h$  is  $C^2$ -continuous at point  $\mathbf{x}_d := \mathbf{x} + d\nabla h(\mathbf{x})$ . From equation  $h(\mathbf{x}_d) = h(\mathbf{x}) + d$  (see Theorem 5.1) we recognize that the level surface  $h = d$  is just the *offset surface*  $\Phi_d$  of  $\Phi_0$  for distance  $d$ . Equation  $\nabla h(\mathbf{x}_d) = \nabla h(\mathbf{x})$  shows that the tangent plane at point  $\mathbf{x}_d$  and at its foot point  $\mathbf{x}$  agree. Differentiating equation  $\nabla h(\mathbf{x}_d) = \nabla h(\mathbf{x})$  yields

$$H_h(\mathbf{x}_d)(I + dH_h(\mathbf{x})) = H_h(\mathbf{x}).$$

Multiplying this matrix equation by vector  $\mathbf{v}_i$  (unit vector in a main direction) we get

$$H_h(\mathbf{x}_d)\mathbf{v}_i = -\frac{k_i(\mathbf{x})}{1 - dk_i(\mathbf{x})}\mathbf{v}_i.$$

Hence the main directions at point  $\mathbf{x}_d$  of the offset surface agree with the main directions at the corresponding foot point on  $\Phi_0$  and the main curvatures are

$$k_i(\mathbf{x}_d) = \frac{k_i(\mathbf{x})}{1 - dk_i(\mathbf{x})}, \quad i = 1, 2.$$

- In case of a *local coordinate system* such that point  $\mathbf{p}$  is the origin  $\mathbf{0}$  and  $\nabla h(\mathbf{0}) = (0, 0, 1)^\top$ . Hence the x-y-plane is the tangent plane at point  $\mathbf{p} = \mathbf{0}$  and for the Hessian we get the simplification (because of  $H_h \nabla h = \mathbf{0}$ )

$$H_h(\mathbf{0}) = \begin{pmatrix} h_{xx} & h_{xy} & 0 \\ h_{yx} & h_{yy} & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

For this specialization we get the following results

- $h_{xx}, h_{yy}$  are the normal curvatures in direction  $(1, 0, 0)^\top, (0, 1, 0)^\top$  respectively.  
From  $-2H = h_{xx} + h_{yy}$  we get the general statement: The mean curvature  $H$  is the arithmetic mean of the normal curvatures of *any* two orthogonal (tangent) directions.  
The Gaussian curvature is  $K = h_{xx}h_{yy} - h_{xy}^2$ .

- $H_h$  is determined uniquely by the normal curvatures of any *three* (tangential) directions (see 3–Tangent–Theorem below).
- From the Taylor expansion of  $h$  at point  $\mathbf{0}$ :

$$h(x, y, z) = z + \frac{1}{2}(h_{xx}x^2 + h_{yy}y^2 + 2h_{xy}xy) + \dots$$

we get the *osculating paraboloid*

$$z = -\frac{1}{2}(h_{xx}x^2 + 2h_{xy}xy + h_{yy}y^2)$$

and the *Dupin indicatrix* of the surface at point  $\mathbf{0}$ :

$$h_{xx}x^2 + 2h_{xy}xy + h_{yy}y^2 = \pm 1.$$

### 5.4.3 $C^n$ –contact, $G^n$ –contact

A  $C^n$ –continuous curve/surface considered as a set of points can be described by several parametric representations. Essentials of a representation are its derivatives which allow to determine geometrical attributes of the curve/surface like tangent line/plane, curvature,... Such geometrical attributes are independent of the representation accidentally chosen. So it is necessary to distinguish between the continuity of representations ( $C^n$ –continuity) and the continuity of geometrical attributes ( $G^n$ –continuity):

- Definition 5.10** 1. Two regular  $C^n$ –continuous parametric curves  $\Gamma_1 : \mathbf{x} = \mathbf{X}_1(t), \Gamma_2 : \mathbf{x} = \mathbf{X}_2(t)$  (surfaces  $\Phi_1 : \mathbf{x} = \mathbf{X}_1(s, t), \Phi_2 : \mathbf{x} = \mathbf{X}_2(s, t)$ ) have  $C^n$ –contact at a common point  $\mathbf{x}_0$  if the (partial) derivatives of  $\mathbf{X}_1$  and  $\mathbf{X}_2$  at point  $\mathbf{x}_0$  up to the  $n$ -th order are the same.
2. Two regular  $C^n$ –continuous implicit curves  $\Gamma_1 : f_1(\mathbf{x}) = 0, \Gamma_2 : f_2(\mathbf{x}) = 0$  (surfaces  $\Phi_1 : f_1(\mathbf{x}) = 0, \Phi_2 : f_2(\mathbf{x}) = 0$ ) have  $C^n$ –contact at a common point  $\mathbf{x}_0$  if the partial derivatives of  $f_1$  and  $f_2$  at point  $\mathbf{x}_0$  up to the  $n$ -th order are the same.
3. Two with respect to the same coordinate explicitly represented plane curves (surfaces) have  $C^n$ –contact at a common point if the corresponding functions are  $C^n$ –continuous at this point.

**Definition 5.11** Two regular parametric or implicit plane curves  $\Gamma_1, \Gamma_2$  (surfaces  $\Phi_1, \Phi_2$ ) with a common point  $\mathbf{x}_0$  have  $G^n$ –contact at  $\mathbf{x}_0$  if they have local explicit representations for one coordinate  $\in \{x, y\}$  ( $\in \{x, y, z\}$ ) with  $C^n$ –contact at point  $\mathbf{x}_0$ .

**Remark:**

In former literature  $G^n$ –contact is called *contact of order  $n$* .

From Lemma 4.1 and Lemma 5.1 we get the following, more useful, characterisation of the  $G^n$ –continuity.

- Theorem 5.2** 1. Two regular  $C^n$ –continuous parametric plane curves  $\Gamma_1, \Gamma_2$  (surfaces  $\Phi_1, \Phi_2$ ) have  $G^n$ –contact at a common point  $\mathbf{x}_0$  if and only if they have  $C^n$ –contact at point  $\mathbf{x}_0$  with respect to some local parametric representations of the two curves (surfaces).
2. Two regular  $C^n$ –continuous implicit plane curves  $\Gamma_1, \Gamma_2$  (surfaces  $\Phi_1, \Phi_2$ ) have  $G^n$ –contact at a common point  $\mathbf{x}_0$  if and only if they have  $C^n$ –contact at point  $\mathbf{x}_0$  with respect to some local implicit representations of the two curves (surfaces).

### 5.4.4 $G^2$ -continuity theorems

There are several theorems on the  $G^2$ -continuity (i.e. continuity of the normal curvatures) of two surface patches. We will give simple proofs of such theorems using the idea of the normal form.

#### Theorem 5.3 3-Tangent-Theorem

Two  $G^2$ -continuous surfaces  $\Phi_1, \Phi_2$  with a common point  $\mathbf{p}$  and a common tangent plane at  $\mathbf{p}$  are  $G^2$ -continuous at point  $\mathbf{p}$  if they have equal normal curvatures along three directions.

**Proof:** We assume, without loss of generality, that  $\mathbf{p} = \mathbf{0}$  and furthermore the three directions are  $\mathbf{v}_1 = (1, 0, 0), \mathbf{v}_2 = (\cos \varphi_2, \sin \varphi_2, 0), \mathbf{v}_3 = (\cos \varphi_3, \sin \varphi_3, 0)$  with corresponding common normal curvatures  $\kappa_1, \kappa_2, \kappa_3$ . From  $\kappa_i = \mathbf{v}_i H_h \mathbf{v}_i^\top$  we get

$$h_{xx} = \kappa_1,$$

$$h_{xx} \cos^2 \varphi_2 + h_{yy} \sin^2 \varphi_2 + 2h_{xy} \cos \varphi_2 \sin \varphi_2 = \kappa_2,$$

$$h_{xx} \cos^2 \varphi_3 + h_{yy} \sin^2 \varphi_3 + 2h_{xy} \cos \varphi_3 \sin \varphi_3 = \kappa_3.$$

The determinant of this linear system for  $h_{xx}, h_{yy}, h_{xy}$  is not 0 if  $0 < \varphi_2 < \varphi_3 < \pi$ . Hence the Hessian for both the surfaces is uniquely determined and the surfaces join  $G^2$ -continuously at the common point  $\mathbf{p}$ .  $\square$

The essential part of the proof of the Linkage Curve Theorem is the following Lemma.

**Lemma 5.3** Let  $\Phi_1, \Phi_2$  be two  $G^2$ -continuous surfaces with common point  $\mathbf{0}$ , common tangent plane  $z = 0$  and common tangent planes along the common smooth curve  $\Gamma : \mathbf{x}(t) = (t, b(t), c(t))^\top, t \in [0, t_0]$  with  $\mathbf{x}(0) = \mathbf{0}$  and  $\dot{\mathbf{x}}(0) = (1, 0, 0)^\top$ . Then we get for the Hessian matrices  $H_{h_1}, H_{h_2}$  of the normalform functions  $h_1, h_2$  of the surfaces  $\Phi_1, \Phi_2$

$$h_{1,xx}(\mathbf{0}) = h_{2,xx}(\mathbf{0}), \quad h_{1,xy}(\mathbf{0}) = h_{2,xy}(\mathbf{0}).$$

**Proof:** For a point  $(t, b(t), c(t))$  on curve  $\Gamma$  we have

$$\nabla h_1(t, b(t), c(t)) = \nabla h_2(t, b(t), c(t)).$$

Differentiating this equation yields

$$H_{h_1}(t, b(t), c(t)) (1, \dot{b}(t), \dot{c}(t))^\top = H_{h_2}(t, b(t), c(t)) (1, \dot{b}(t), \dot{c}(t))^\top.$$

For  $t = 0$  we get  $h_{1,xx}(\mathbf{0}) = h_{2,xx}(\mathbf{0}), h_{1,xy}(\mathbf{0}) = h_{2,xy}(\mathbf{0})$  (Remember the simple Form of the Hessian matrix and  $\dot{b}(0) = 0$  at point  $\mathbf{0}$ !).  $\square$

#### Remark:

Because  $h_{xx}$  is the normal curvature in  $x$ -direction we get from  $h_{1,xx}(\mathbf{0}) = h_{2,xx}(\mathbf{0})$  that the normal curvatures of the surfaces at point  $\mathbf{0}$  in direction of  $\Gamma$  are the same and hence their Dupin indicatrices intersect at a point of the tangent of curve  $\Gamma$ .

Obviously we get for the Hessian matrices of the surfaces  $\Phi_1, \Phi_2$  in the Lemma above the stronger result  $H_{h_1}(\mathbf{0}) = H_{h_2}(\mathbf{0})$  if we choose additionally a suitable geometric restriction which yields  $h_{1,yy}(\mathbf{0}) = h_{2,yy}(\mathbf{0})$ . For example **one** of the following conditions

- The equality of the mean curvatures:  
 $-2H(\mathbf{0}) = h_{1,xx}(\mathbf{0}) + h_{1,yy}(\mathbf{0}) = h_{2,xx}(\mathbf{0}) + h_{2,yy}(\mathbf{0}).$
- The equality of the Gaussian curvatures:  
 $K(\mathbf{0}) = h_{1,xx}(\mathbf{0})h_{1,yy}(\mathbf{0}) - h_{1,xy}(\mathbf{0})^2 = h_{2,xx}(\mathbf{0})h_{2,yy}(\mathbf{0}) - h_{2,xy}(\mathbf{0})^2$   
 if the normal curvature  $h_{1,xx}(\mathbf{0})$  in direction  $(1, 0, 0)$  does not vanish.



- The equality of the normal curvature in a direction  $\mathbf{v} = (a, 1, 0)^\top$ ,  $a \in \mathbb{R}$ , transversal to curve  $\Gamma$  at point  $\mathbf{0}$ :

$$\kappa_a = \frac{a^2 h_{1,xx} + h_{1,yy} + 2ah_{1,xy}}{1 + a^2} = \frac{a^2 h_{2,xx} + h_{2,yy} + 2ah_{2,xy}}{1 + a^2}.$$

These considerations prove the following theorem

**Theorem 5.4 Linkage Curve Theorem:**

Let  $\Phi_1$  and  $\Phi_2$  be two  $G^2$ -continuous surfaces which are tangent plane continuous along a smooth linkage curve  $\Gamma$ .

$\Phi_1$  and  $\Phi_2$  are  $G^2$ -continuous along their linkage curve  $\Gamma$  if **one** of the following conditions is fulfilled:

- $\Phi_1$  and  $\Phi_2$  have the same **mean curvature** along  $\Gamma$   
(Mean Curvature Criterion)
- $\Phi_1$  and  $\Phi_2$  have the same **Gaussian curvature** and non vanishing normal curvature in  $\Gamma$ -direction along  $\Gamma$   
(Gaussian Curvature Criterion)
- There exists a transversal **vector field** (not necessarily continuous) along  $\Gamma$ , such that the normal curvatures of  $\Phi_1$  and  $\Phi_2$  in directions of the vector field are the same.  
(Vector Field Criterion)

### 5.4.5 The curvature of an intersection curve

Let be  $\Phi_A, \Phi_B$  two intersecting  $C^2$ -continuous surfaces and  $h_A = 0, h_B = 0$  their normalforms. The intersection curve  $\Gamma$  is represented by the arc length parameterization  $\mathbf{x} = \mathbf{c}(s)$ . Hence  $\|\nabla h_A\| = 1$ ,  $\|\nabla h_B\| = 1$  and  $\|\mathbf{c}'\| = 1$ ,  $\mathbf{c}' \cdot \mathbf{c}'' = 0$ . Differentiating the following equations

$$h_A(\mathbf{c}(s)) = 0, \quad h_B(\mathbf{c}(s)) = 0$$

yields

$$\nabla h_A \cdot \mathbf{c}' = 0, \quad \nabla h_B \cdot \mathbf{c}' = 0$$

and

$$\nabla h_A \cdot \mathbf{c}'' + \mathbf{c}'^\top H_{h_A} \mathbf{c}' = 0, \quad \nabla h_B \cdot \mathbf{c}'' + \mathbf{c}'^\top H_{h_B} \mathbf{c}' = 0.$$

Because  $\mathbf{c}'$  is orthogonal to  $\nabla h_A$  and  $\nabla h_B$  and  $\|\mathbf{c}'\| = 1$  we can choose the parameterization of  $\Gamma$  such that

$$\mathbf{c}' = \frac{\nabla h_A \times \nabla h_B}{\|\nabla h_A \times \nabla h_B\|}.$$

$\mathbf{c}''$  is a linear combination of  $\nabla h_A$  and  $\nabla h_B$ . Because the calculation becomes simpler we use a different base of the normal plane:

$$\mathbf{c}'' = \alpha(\nabla h_A \times \mathbf{c}') + \beta(\nabla h_B \times \mathbf{c}') = (\alpha \nabla h_A + \beta \nabla h_B) \times \mathbf{c}'.$$

Using  $k_A := -\mathbf{c}'^\top H_{h_A} \mathbf{c}'$ ,  $k_B := -\mathbf{c}'^\top H_{h_B} \mathbf{c}'$  (normal curvature on surface  $\Phi_A$  and  $\Phi_B$ , respectively, in direction  $\mathbf{c}'$ ) we get from the equations  $\nabla h_A \cdot \mathbf{c}'' = k_A$ ,  $\nabla h_B \cdot \mathbf{c}'' = k_B$  the result:

$$\mathbf{c}'' = \frac{k_A \nabla h_B - k_B \nabla h_A}{\|\nabla h_A \times \nabla h_B\|} \times \mathbf{c}'.$$

This formula can be written independently of the representations of the intersecting surfaces:

Let be  $\Phi_A, \Phi_B$  two intersecting  $C^2$ -continuous surfaces with unit normals  $\mathbf{n}_A, \mathbf{n}_B$ ,  $\|\mathbf{n}_A \times \mathbf{n}_B\| \neq 0$ , and normal curvatures  $k_A, k_B$  in direction  $\mathbf{n}_A \times \mathbf{n}_B$  at a point of the intersection curve  $\Gamma$  than

•  $\mathbf{c}' := \frac{\mathbf{n}_A \times \mathbf{n}_B}{\|\mathbf{n}_A \times \mathbf{n}_B\|}$  is the *tangent unit vector* and

•  $\mathbf{c}'' := \frac{(k_A \mathbf{n}_B - k_B \mathbf{n}_A) \times (\mathbf{n}_A \times \mathbf{n}_B)}{(\mathbf{n}_A \times \mathbf{n}_B)^2}$

is the *curvature vector* and from the orthogonality of  $k_A \mathbf{n}_B - k_B \mathbf{n}_A$  and  $\mathbf{n}_A \times \mathbf{n}_B$  we get

$$k = \|\mathbf{c}''\| = \frac{\sqrt{k_A^2 + k_B^2 - 2k_A k_B \cos \varphi}}{\sin \varphi}$$

the *curvature*, where  $\varphi$  is the angle between the surface normals and  $k_A$  and  $k_B$  the normal curvatures of surfaces  $\Phi_A$  and  $\Phi_B$  in direction  $\mathbf{c}'$ .

## 5.5 Normalform of an implicit surface

Let  $f(\mathbf{x}) = 0$  be a regular implicit  $C^2$ -surface and  $h = 0$  its normalform.  
For a surface point  $\mathbf{x}$  we get:

1.  $h = 0$ ,

2. unit normal:  $\nabla h = \frac{\nabla f}{\|\nabla f\|}$ ,

3. Hessian matrix:  $H_h = (I - \nabla h \nabla h^\top) \frac{H_f}{\|\nabla f\|}$ ,

with  $3 \times 3$  unit matrix  $I$  and matrix  $\nabla h \nabla h^\top = (h_x, h_y, h_z)^\top (h_x, h_y, h_z)$ .  
(For proof: differentiate  $\frac{\nabla f}{\|\nabla f\|}$ .)

4. Normal curvature for unit tangent direction  $\mathbf{v}$  is  $k_n = -\frac{\mathbf{v}^\top H_f \mathbf{v}}{\|\nabla f\|}$ .

5. Main curvatures, main directions, mean and Gaussian curvature:

Let be  $\mathbf{e}_1, \mathbf{e}_2$  two orthonormal vectors which are perpendicular to  $\nabla f$ . A main direction can be written as  $\xi \mathbf{e}_1 + \eta \mathbf{e}_2$  and must be an eigenvector of  $H_h$ . Hence an eigenvalue  $\kappa$  of  $H_h$  and its eigendirection have to fulfil the following linear system:

$$\begin{aligned} \left( \frac{\mathbf{e}_1^\top H_f \mathbf{e}_1}{\|\nabla f\|} - \kappa \right) \xi + \frac{\mathbf{e}_1^\top H_f \mathbf{e}_2}{\|\nabla f\|} \eta &= 0 \\ \frac{\mathbf{e}_1^\top H_f \mathbf{e}_2}{\|\nabla f\|} \xi + \left( \frac{\mathbf{e}_2^\top H_f \mathbf{e}_2}{\|\nabla f\|} - \kappa \right) \eta &= 0. \end{aligned}$$

Hence, from matrix

$$\begin{pmatrix} \frac{\mathbf{e}_1^\top H_f \mathbf{e}_1}{\|\nabla f\|} & \frac{\mathbf{e}_1^\top H_f \mathbf{e}_2}{\|\nabla f\|} \\ \frac{\mathbf{e}_1^\top H_f \mathbf{e}_2}{\|\nabla f\|} & \frac{\mathbf{e}_2^\top H_f \mathbf{e}_2}{\|\nabla f\|} \end{pmatrix}$$

we get a) the main curvatures (negative eigenvalues) b) the main directions (eigendirections) c) the mean curvature (half of negative trace) and d) the Gaussian curvature (determinant).

For a point  $\mathbf{x}$  in the vicinity of the surface with foot point  $\mathbf{x}_0 \in \Gamma$  we get:

1.  $\nabla h(\mathbf{x}) = \nabla h(\mathbf{x}_0)$ ,

2.  $h(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_0) \cdot \nabla h(\mathbf{x}_0)$

3. Hessian matrix  $H_h$ :

The 6 coefficients of the matrix  $H_h$  are solutions of the 6 linear equations:

$$1)-3) H_h \nabla h = \mathbf{0} \quad \text{and} \quad 4)-6) \mathbf{v}_i^\top H_h \mathbf{v}_i = -k_i / (1 - hk_i), \quad i = 1, 2, 3,,$$

where  $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$  are 3 linear independent unit tangent vectors at the foot point  $\mathbf{x}_0$  and  $k_1, k_2, k_3$  the corresponding normal curvatures (see above).

## 5.6 Normalform of a parametric surface

Let  $\mathbf{x} = \mathbf{x}(u, v)$  be a regular parametric  $C^2$ -surface and  $h = 0$  its normalform.

For a *surface point*  $\mathbf{x}(u, v)$  we get:

$$1. h = 0 ,$$

$$2. \nabla h = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|} ,$$

3. Hessian matrix  $H_h$ :

The 6 coefficients of the (symmetric) matrix  $H_h$  are solutions of the 6 linear equations:

$$1)-3) H_h \nabla h = \mathbf{0} \quad \text{and} \quad 4) \mathbf{x}_u^\top H_h \mathbf{x}_u = -L, \quad 5) \mathbf{x}_u^\top H_h \mathbf{x}_v = -M, \quad 6) \mathbf{x}_v^\top H_h \mathbf{x}_v = -N.$$

Equations 4)–6) result from the normal curvatures for directions  $\mathbf{x}_u$ ,  $\mathbf{x}_u + \mathbf{x}_v$  and  $\mathbf{x}_v$ .

## 4. Main curvatures, main directions, mean and Gaussian curvature:

A main direction can be written as  $\xi \mathbf{x}_u + \eta \mathbf{x}_v$  and must be an eigenvector of  $H_h$ . Hence an eigenvalue  $\kappa$  of  $H_h$  and its eigendirection have to fulfill the equation

$$H_h(\xi \mathbf{x}_u + \eta \mathbf{x}_v) = \kappa(\xi \mathbf{x}_u + \eta \mathbf{x}_v)$$

Multiplying by  $\mathbf{x}_u$  and  $\mathbf{x}_v$  yields the linear system:

$$\begin{aligned} \xi(\mathbf{x}_u^\top H_h \mathbf{x}_u - \kappa \mathbf{x}_u^2) + \eta(\mathbf{x}_u^\top H_h \mathbf{x}_v - \kappa \mathbf{x}_u \cdot \mathbf{x}_v) &= 0 \\ \xi(\mathbf{x}_u^\top H_h \mathbf{x}_v - \kappa \mathbf{x}_u \cdot \mathbf{x}_v) + \eta(\mathbf{x}_v^\top H_h \mathbf{x}_v - \kappa \mathbf{x}_v^2) &= 0. \end{aligned}$$

With  $\mathbf{x}_u^\top H_h \mathbf{x}_u = -L$ ,  $\mathbf{x}_u^\top H_h \mathbf{x}_v = -M$ ,  $\mathbf{x}_v^\top H_h \mathbf{x}_v = -N$  and  $E = \mathbf{x}_u^2$ ,  $F = \mathbf{x}_u \cdot \mathbf{x}_v$ ,  $G = \mathbf{x}_v^2$  we get

$$\begin{aligned} \xi(L + \kappa E) + \eta(M + \kappa F) &= 0 \\ \xi(M + \kappa F) + \eta(N + \kappa G) &= 0. \end{aligned}$$

(See Section 5.2.2.)

For a point  $\mathbf{x}$  in the vicinity of the surface with foot point  $\mathbf{x}(u, v)$  on the surface we get:

$$1. \nabla h = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|} ,$$

$$2. h = (\mathbf{x} - \mathbf{x}(u, v)) \cdot \nabla h ,$$

3. Hessian matrix  $H_h$ :

The 6 coefficients of the matrix  $H_h$  are solutions of the 6 linear equations:

$$1)-3) H_h \nabla h = \mathbf{0} \quad \text{and} \quad 4)-6) \mathbf{v}_i^\top H_h \mathbf{v}_i = -k_i / (1 - hk_i), \quad i = 1, 2, 3,,$$

with normal curvatures

$$k_1 := L/M, \quad k_2 := (L + 2M + N)/(E + 2F + G), \quad k_3 := N/G$$

and unit tangent vectors:

$$\mathbf{v}_1 := \mathbf{x}_u / \|\cdots\|, \quad \mathbf{v}_2 := (\mathbf{x}_u + \mathbf{x}_v) / \|\cdots\|, \quad \mathbf{v}_3 := \mathbf{x}_v / \|\cdots\|, \quad \text{respectively.}$$

## 5.7 Foot point on a parametric surface, surface inversion

Given:  $\Phi : \mathbf{x} = \mathbf{S}(s, t)$  a regular surface,  $P : \mathbf{p}$  a point and

$L_0, s_0, t_0$  0-th approximation of the foot point  $L$  and its parameters.

Wanted: Foot point  $L : \mathbf{l}$  of the perpendicular line from  $P$  onto  $\Phi$ .

Vector  $\mathbf{p} - \mathbf{l}$  is orthogonal to the tangent plane at point  $L$ . Hence the parameters of  $L$  are solutions of the following system:

$$f(s, t) := (\mathbf{S}(s, t) - \mathbf{p}) \cdot \mathbf{S}_s(s, t) = 0, \quad g(s, t) := (\mathbf{S}(s, t) - \mathbf{p}) \cdot \mathbf{S}_t(s, t) = 0$$

which can be solved

**either** by a Newton-iteration for the starting parameters  $s_0, t_0$  and function  $\mathbf{F} := (f, g)$  with

$$\mathbf{F}' = \begin{pmatrix} \mathbf{S}_s^2 & \mathbf{S}_s \cdot \mathbf{S}_t \\ \mathbf{S}_s \cdot \mathbf{S}_t & \mathbf{S}_t^2 \end{pmatrix} + \begin{pmatrix} \mathbf{S}_{ss} \cdot (\mathbf{S} - \mathbf{p}) & \mathbf{S}_{st} \cdot (\mathbf{S} - \mathbf{p}) \\ \mathbf{S}_{ts} \cdot (\mathbf{S} - \mathbf{p}) & \mathbf{S}_{tt} \cdot (\mathbf{S} - \mathbf{p}) \end{pmatrix}$$

or by

**repeated foot point determination on tangent planes:**

Let  $L_i, s_i, t_i$  be the  $i$ -th approximation for  $L$ .

The linear approximation (tangent plane) of  $\mathbf{S}(s, t)$  at  $L_i$  is

$$\varepsilon_i : \mathbf{x}(\Delta s, \Delta t) = \mathbf{S}(s_i, t_i) + \Delta s \mathbf{S}_s(s_i, t_i) + \Delta t \mathbf{S}_t(s_i, t_i)$$

We determine  $\Delta s$  and  $\Delta t$  such that  $\mathbf{p} - \mathbf{x}(\Delta s, \Delta t)$  is orthogonal to  $\varepsilon_i$ . With the abbreviations  $\mathbf{e}_s := \mathbf{S}_s(s_i, t_i)$ ,  $\mathbf{e}_t := \mathbf{S}_t(s_i, t_i)$  we have to solve the following linear system for  $\Delta s$  and  $\Delta t$

$$\begin{aligned} (\mathbf{p} - \mathbf{S}(s_i, t_i)) \cdot \mathbf{e}_s &= \Delta s \mathbf{e}_s^2 + \Delta t \mathbf{e}_s \cdot \mathbf{e}_t \\ (\mathbf{p} - \mathbf{S}(s_i, t_i)) \cdot \mathbf{e}_t &= \Delta s \mathbf{e}_s \cdot \mathbf{e}_t + \Delta t \mathbf{e}_t^2. \end{aligned}$$

The improved parameters/point:  $s_{i+1} = s_i + \Delta s$ ,  $t_{i+1} = t_i + \Delta t$ ,  $L_{i+1} = \mathbf{S}(s_{i+1}, t_{i+1})$ . Repeat this procedure until the distance of  $L_{i+1}$  and  $L_i$  is small “enough”. Foot point:  $L = L_{i+1}$ .

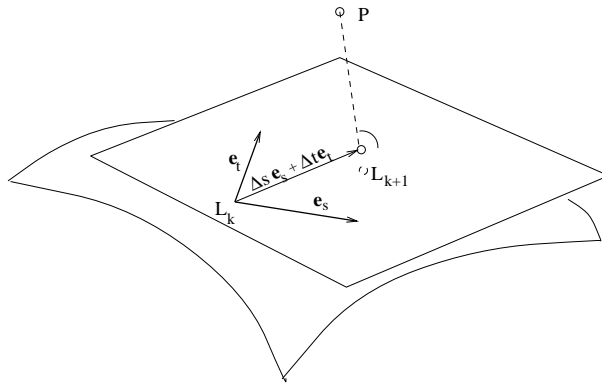


Figure 5.2: Foot point on a tangent plane

**Remark:**

The foot point algorithm can be used for **surface inversion** which means to find for a point  $P_0 : \mathbf{p}_0$  on a surface  $\mathbf{x} = \mathbf{S}(s, t)$  parameters  $s_0, t_0$  such that  $\mathbf{p}_0 = \mathbf{S}(s_0, t_0)$ . ( $s_0, t_0$  may not be unique.)

## 5.8 Stable first order foot point algorithms for surfaces

### 5.8.1 Foot point algorithm for parametric surfaces

Analogously to the stable foot point algorithm for parametric planar curves (see Section 4.5.3.1) that respects the curvature without using second derivatives we give the surface version:

Let  $\Phi : \mathbf{x} = \mathbf{S}(s, t)$  a regular surface,  $\mathbf{p}$  a point in the vicinity of  $\Phi$  and  $s_0, t_0$  the parameter of a starting point for the *foot point algorithm*:

**repeat**

$\mathbf{p}_i = \mathbf{S}(s_i, t_i)$ ,  $\mathbf{e}_s := \mathbf{S}_s(s_i, t_i)$ ,  $\mathbf{e}_t := \mathbf{S}_t(s_i, t_i)$

Solve the linear system for  $\Delta s$  and  $\Delta t$ : 
$$\begin{aligned} (\mathbf{p} - \mathbf{S}(s_i, t_i)) \cdot \mathbf{e}_s &= \Delta s \mathbf{e}_s^2 + \Delta t \mathbf{e}_s \cdot \mathbf{e}_t \\ (\mathbf{p} - \mathbf{S}(s_i, t_i)) \cdot \mathbf{e}_t &= \Delta s \mathbf{e}_s \cdot \mathbf{e}_t + \Delta t \mathbf{e}_t^2. \end{aligned}$$

$\mathbf{q}_i = \mathbf{p}_i + \Delta s \mathbf{S}_s(s_i, t_i) + \Delta t \mathbf{S}_t(s_i, t_i)$  (foot point on tangent plane)

$\mathbf{p}_{i+1} = \mathbf{S}(s_i + \Delta s, t_i + \Delta t)$ ,  $\mathbf{f}_1 := \mathbf{q}_i - \mathbf{p}_i$ ,  $\mathbf{f}_2 := \mathbf{p}_{i+1} - \mathbf{q}_i$ ,

if  $\|\mathbf{q}_i - \mathbf{p}_i\| > \varepsilon$  then (one Newton step for the foot point on the tangent parabola  $\mathbf{x} = \mathbf{p}_i + \alpha \mathbf{f}_1 + \alpha^2 \mathbf{f}_2$ )

$a_0 := (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{f}_1$ ,  $a_1 := 2\mathbf{f}_2 \cdot (\mathbf{p} - \mathbf{p}_i) - \mathbf{f}_1^2$ ,  $a_2 := -3\mathbf{f}_1 \cdot \mathbf{f}_2$ ,  $a_3 := -2\mathbf{f}_2^2$

$\alpha := 1 - \frac{a_0 + a_1 + a_2 + a_3}{a_1 + 2a_2 + 3a_3}$

if  $0 < \alpha < \alpha_{max}$  then (prevent extreme cases)

$s_{i+1} = t_i + \alpha \Delta s$ ,  $t_{i+1} = t_i + \alpha \Delta t$ ,  $\mathbf{p}_{i+1} = \mathbf{S}(s_{i+1}, t_{i+1})$

**until**  $\|\mathbf{p}_i - \mathbf{p}_{i+1}\| < \varepsilon$ .

*foot point*  $\mathbf{f} = \mathbf{p}_{i+1}$ .

(Suitable choices for the boundaries:  $\varepsilon = 10^{-6}$  and  $\alpha_{max} = 20$ .)

### 5.8.2 Foot point algorithm for implicit surfaces

The foot point algorithm for implicit surfaces agrees literally with the version for planar implicit curves (see Section 4.5.2). One has just to replace the words curve and tangent line by surface and tangent plane:

Procedure **surfacepoint**:

(SP1) Choose a starting point  $Q_0 : \mathbf{q}_0 = (x_0, y_0)$  in the vicinity of the surface,

(SP2) **repeat**  $\mathbf{q}_{i+1} = \mathbf{q}_i - \frac{f(\mathbf{q}_i)}{\nabla f(\mathbf{q}_i)^2} \nabla f(\mathbf{q}_i)$  (Newton step for  $f(\mathbf{q}_i + t \nabla f(\mathbf{q}_i))$ )  
 until  $\|\mathbf{q}_{i+1} - \mathbf{q}_i\|$  small enough.  
**surfacepoint** =  $\mathbf{q}_{i+1}$ .

Let  $\mathbf{p}$  be a point in the vicinity of an implicit surface  $\Phi : f(\mathbf{x}) = 0$ . The following algorithm determines the *foot point* of  $\mathbf{p}$  on  $\Phi$ :

(FP0)  $\mathbf{p}_0 = \text{surfacepoint}(\mathbf{p})$

(FP1) **repeat**

$\mathbf{q}_i = \mathbf{p} - \frac{(\mathbf{p} - \mathbf{p}_i) \cdot \nabla f(\mathbf{p}_i)}{\nabla f(\mathbf{p}_i)^2} \nabla f(\mathbf{p}_i)$  (foot point on tangent plane),

$\mathbf{p}_{i+1} = \text{surfacepoint}(\mathbf{q}_i)$ ,  $\mathbf{f}_1 := \mathbf{q}_i - \mathbf{p}_i$ ,  $\mathbf{f}_2 := \mathbf{p}_{i+1} - \mathbf{q}_i$ ,

if  $\|\mathbf{q}_i - \mathbf{p}_i\| > \varepsilon$  then (one Newton step for the foot point on the tangent parabola

$\mathbf{x} = \mathbf{p}_i + \alpha \mathbf{f}_1 + \alpha^2 \mathbf{f}_2$ )

$a_0 := (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{f}_1$ ,  $a_1 := 2\mathbf{f}_2 \cdot (\mathbf{p} - \mathbf{p}_i) - \mathbf{f}_1^2$ ,  $a_2 := -3\mathbf{f}_1 \cdot \mathbf{f}_2$ ,  $a_3 := -2\mathbf{f}_2^2$

$\alpha := 1 - \frac{a_0 + a_1 + a_2 + a_3}{a_1 + 2a_2 + 3a_3}$

if  $0 < \alpha < \alpha_{max}$  then (prevent extreme cases)  
 $\mathbf{q}_i = \mathbf{p}_i + \alpha \mathbf{f}_1 + \alpha^2 \mathbf{f}_2$ ,  $\mathbf{p}_{i+1} = \text{surfacepoint}(\mathbf{q}_i)$   
 until  $\|\mathbf{p}_i - \mathbf{p}_{i+1}\| < \varepsilon$ .  
 foot point  $\mathbf{f} = \mathbf{p}_{i+1}$ .

## 5.9 The normalform of a space curve

### 5.9.1 Definition of the normalform

**Definition 5.12** Let  $g : \mathbf{x} = \mathbf{p}_0 + s\mathbf{r}$ ,  $\|\mathbf{r}\| = 1$ , be a line in  $\mathbb{R}^3$  and  $P : \mathbf{p} = (x, y, z)^\top$  a point. The distance of point  $P$  to line  $g$  is

$$h(\mathbf{p}) = \sqrt{(\mathbf{p} - \mathbf{p}_0)^2 - ((\mathbf{p} - \mathbf{p}_0) \cdot \mathbf{r})^2}.$$

We call equation  $h(\mathbf{p}) = 0$  the normal form of the space line  $g$ .

**Example 5.5**  $h(x, y, z) = \sqrt{x^2 + y^2} = 0$  is the normalform of the  $z$ -axis. The level surfaces  $h = \text{const}$  are cylinders.

Let  $\Gamma : \mathbf{x} = \mathbf{c}(s)$ ,  $s \in [s_1, s_2]$  be a  $C^n$ -continuous,  $n \geq 2$ , space curve parameterized by the arc length  $s$  (i.e.  $\|\mathbf{c}'\| = 1$ ) and  $\mathbf{e}_1(s), \mathbf{e}_2(s)$  two orthogonal unit vectors in the normal plane at point  $\mathbf{c}(s)$  such that the determinant  $|\mathbf{c}', \mathbf{e}_1, \mathbf{e}_2| = 1$ , i.e.  $\mathbf{e}_1$  is a solution of the equation  $\mathbf{c}' \cdot \mathbf{x} = 0$  and  $\mathbf{e}_2 = \mathbf{c}' \times \mathbf{e}_1$ . With unit vector  $\mathbf{e}(s, t) := \mathbf{e}_1(s) \cos t + \mathbf{e}_2(s) \sin t$  we define the family of pipe surfaces with directrix curve  $\Gamma$ :

$$\mathbf{x} = \mathbf{S}(s, t, d) := \mathbf{c}(s) + d\mathbf{e}(s, t), \quad d > 0, \quad t \in [0, 2\pi)$$

From the definition of vector  $\mathbf{e}(s, t)$  we get the following properties:

$$\mathbf{e}_s \cdot \mathbf{e} = \mathbf{e}_t \cdot \mathbf{e} = 0, \quad \mathbf{e}_t \times \mathbf{e} = \mathbf{c}'.$$

For  $d = \text{const}$  function  $\mathbf{S}$  describes the pipe surface with directrix curve  $\Gamma$  and radius  $d$ .

The vector equation  $(x, y, z)^\top = \mathbf{c}(s) + d\mathbf{e}(s, t)$  can be solved for  $s, t, d$  if the determinant  $|\mathbf{S}_s, \mathbf{S}_t, \mathbf{S}_d|$  is not 0. Evaluation of the determinant yields:

$$\begin{aligned} |\mathbf{S}_s, \mathbf{S}_t, \mathbf{S}_d| &= |\mathbf{c}' + d\mathbf{e}_s, d\mathbf{e}_t, \mathbf{e}| = d|\mathbf{c}', \mathbf{e}_t, \mathbf{e}| + d^2|\mathbf{e}_s, \mathbf{e}_t, \mathbf{e}| = d\mathbf{c}' \cdot (\mathbf{e}_t \times \mathbf{e}) + d^2\mathbf{e}_s \cdot (\mathbf{e}_t \times \mathbf{e}) \\ &= d(1 + d\mathbf{c}' \cdot \mathbf{e}_s). \end{aligned}$$

Hence in a vicinity of curve  $\Gamma$  there exists a  $d_0 > 0$  such that equation  $(x, y, z)^\top = \mathbf{c}(s) + d\mathbf{e}(s, t)$  can be solved for  $s, t, d$  with  $0 < d \leq d_0$ . The solutions are functions of  $x, y, z$ :  $s = s(x, y, z), t = t(x, y, z), d = h(x, y, z)$ . Equation  $h(x, y, z) = 0$  is an implicit representation of space curve  $\Gamma$ . Differentiating equation

$$\mathbf{c}(s(\mathbf{x})) + h(\mathbf{x})\mathbf{e}(s(\mathbf{x}), t(\mathbf{x})) = \mathbf{x}$$

yields:

$$\nabla s \mathbf{c}'^\top + \nabla h \mathbf{e}^\top + h \nabla s \mathbf{e}_s^\top + h \nabla t \mathbf{e}_t^\top = 3 \times 3 \text{ unit matrix}$$

Multiplying these equations by vector  $\mathbf{e}$  we get from  $\mathbf{e}_s^\top \mathbf{e} = \mathbf{e}_t^\top \mathbf{e} = \mathbf{c}'^\top \mathbf{e} = 0$  the result

$$\nabla h = \mathbf{e} = \frac{\mathbf{S} - \mathbf{c}}{\|\mathbf{S} - \mathbf{c}\|}.$$

Hence equation  $h = 0$  is an implicit representation of curve  $\Gamma$  with  $\|\nabla h\| = 1$  and in a vicinity of  $\Gamma$  but not on  $\Gamma$  the positive function  $h$  is of class  $C^n$ ,  $n \geq 2$ , too. ( $h$  is not differentiable on  $\Gamma$ !)

The equation  $h = 0$  is called the *normalform* of  $\Gamma$  and  $h$  the *normalform function* or (as of its geometrical meaning) the *distance function* of  $\Gamma$ .

The considerations above prove of the following Theorem:

**Theorem 5.5** For any  $C^n$ -continuous,  $n \geq 2$ , space curve  $\Gamma$  there exists in a vicinity  $V$  of  $\Gamma$  a normalform of class  $C^n$ , i.e. an implicit representation  $h(x, y, z) = 0$  such that  $\|\nabla h\| = 1$  on  $V \setminus \Gamma$ . ( $h$  is not differentiable on  $\Gamma$ .)

**Example 5.6** The normalform of the circle  $x^2 + y^2 = r^2, z = 0$  (as space curve !) is

$$h(x, y, z) = \sqrt{z^2 + (\sqrt{x^2 + y^2} - r)^2} = 0.$$

The level surface  $h = c, 0 < c < r$  is the torus with equation

$$(x^2 + y^2 + z^2 + r^2 - c^2)^2 - 4r^2(x^2 + y^2) = 0.$$

A consequence of the Theorem 5.5 is

**Theorem 5.6** A pipe surface with  $C^n$ -continuous directrix,  $n \geq 2$ , is of class  $C^n$ , too.

**Remark:**

A differentiable implicit representation can be achieved by using  $h^2$  instead of  $h$ :

$$h^2 = (\mathbf{S} - \mathbf{c})^2, \quad \nabla h^2 = 2(\mathbf{S} - \mathbf{c}).$$

The disadvantage  $\nabla h^2 = \mathbf{0}$  on curve  $\Gamma$  remains.

### 5.9.2 Foot point algorithm and evaluation of the normalform of a space curve

Usually the normalform function is not known explicitly. So the evaluation of  $h(\mathbf{p})$  and  $\nabla h(\mathbf{p})$  is done numerically by determining the foot point  $\mathbf{c}(t_{\mathbf{p}})$  of point  $\mathbf{p}$  on curve  $\Gamma$ .

**The foot point algorithm:**

Let  $\Gamma : \mathbf{x}(t) = \mathbf{c}(t)$  be a smooth curve,  $\mathbf{p}$  a point in the vicinity of  $\Gamma$  and  $t_0$  the parameter of a starting point for the *foot point algorithm*:

**repeat**

$\mathbf{p}_i = \mathbf{c}(t_i)$

$\Delta s = \frac{(\mathbf{p} - \mathbf{p}_i) \cdot \dot{\mathbf{c}}(t_i)}{\|\dot{\mathbf{c}}(t_i)\|^2} \quad \mathbf{q}_i = \mathbf{p}_i + \Delta t \dot{\mathbf{c}}(t_i)$  (foot point on tangent)

$\mathbf{p}_{i+1} = \mathbf{c}(t_i + \Delta t), \quad \mathbf{f}_1 := \mathbf{q}_i - \mathbf{p}_i, \quad \mathbf{f}_2 := \mathbf{p}_{i+1} - \mathbf{q}_i,$

if  $\|\mathbf{q}_i - \mathbf{p}_i\| > \varepsilon$  then

(one Newton step for the foot point on the tangent parabola  $\mathbf{x} = \mathbf{p}_i + \alpha \mathbf{f}_1 + \alpha^2 \mathbf{f}_2$ )

$a_0 := (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{f}_1, \quad a_1 := 2\mathbf{f}_2 \cdot (\mathbf{p} - \mathbf{p}_i) - \mathbf{f}_1^2, \quad a_2 := -3\mathbf{f}_1 \cdot \mathbf{f}_2, \quad a_3 := -2\mathbf{f}_2^2$

$\alpha := 1 - \frac{a_0 + a_1 + a_2 + a_3}{a_1 + 2a_2 + 3a_3}$

if  $0 < \alpha < \alpha_{max}$  then (prevent extreme cases)  $t_{i+1} = t_i + \alpha \Delta t, \quad \mathbf{p}_{i+1} = \mathbf{c}(t_{i+1})$

**until**  $\|\mathbf{p}_i - \mathbf{p}_{i+1}\| < \varepsilon.$

*foot point*  $\mathbf{c}(t_{\mathbf{p}}) := \mathbf{p}_{i+1}, \quad t_{\mathbf{p}} := t_{i+1}.$

(Suitable choices:  $\varepsilon = 10^{-6}$  and  $\alpha_{max} = 20$ .)

**The evaluation of  $h$  and  $\nabla h$ :**

$$\begin{aligned} h(\mathbf{p}) &= \|\mathbf{p} - \mathbf{c}(t_{\mathbf{p}})\|, & (\mathbf{c}(t_{\mathbf{p}}) \text{ is the foot point of } \mathbf{p} \text{ on the curve}) \\ \nabla h(\mathbf{p}) &= \frac{\mathbf{p} - \mathbf{c}(t_{\mathbf{p}})}{\|\cdot\|}, & \text{if } \mathbf{p} \text{ is not on curve } \Gamma. \end{aligned}$$

### 5.9.3 Determining foot points on an intersection curve

Because for any  $C^2$ -continuous surface there exists a normal form it suffice to give a foot point algorithm for implicitly defined surfaces.

We start with two implicit surfaces  $\Phi_1 : f_1(\mathbf{x}) = 0$ ,  $\Phi_2 : f_2(\mathbf{x}) = 0$  and assume that they are regular and sufficiently differentiable for the operations described below. An essential step of the foot point algorithm is the following procedure **curvepoint** which determines for a point  $\mathbf{p}$  in the "neighborhood" of the curve a point  $\mathbf{p}_0$  on the intersection curve.

Procedure **curvepoint**:

(C0) Let be  $f_1(\mathbf{x}) = 0, f_2(\mathbf{x}) = 0$  two intersecting implicit surfaces and  $\mathbf{p}$  a point.

(C1)  $\mathbf{q}_0 := \mathbf{p}$ ,  
**repeat**  $\mathbf{q}_{k+1} = \mathbf{q}_k + \Delta_k$ , where  
 $\Delta_k = \alpha_k \nabla f_1(\mathbf{q}_k) + \beta_k \nabla f_2(\mathbf{q}_k)$  and  $\Delta_k \cdot \nabla f_j(\mathbf{q}_k) = -f_j(\mathbf{q}_k)$ ,  $j = 1, 2$ .  
(Newton step for  $g_j(\alpha, \beta) := f_j(\mathbf{q}_k + \alpha \nabla f_1(\mathbf{q}_k) + \beta \nabla f_2(\mathbf{q}_k))$ ,  $j = 1, 2$ .)  
**until**  $\|\mathbf{q}_{k+1} - \mathbf{q}_k\|$  is "sufficiently" small.  
Curve point:  $\mathbf{p}_0 = \mathbf{q}_{k+1}$ .

Procedure **footpoint**:

(F0) Let be  $f_1(\mathbf{x}) = 0, f_2(\mathbf{x}) = 0$  two intersecting implicit surfaces and  $\mathbf{p}$  a point.

(F1)  $\mathbf{c}_0 = \text{curvepoint}(\mathbf{p})$

(F2) **repeat**  $\mathbf{r}_i := \nabla f_1(\mathbf{c}_i) \times \nabla f_2(\mathbf{c}_i)$  (tangent direction),  
 $\mathbf{t}_{i+1} = \mathbf{c}_i + \frac{(\mathbf{p} - \mathbf{c}_i) \cdot \mathbf{r}_i}{\mathbf{r}_i^2} \mathbf{r}_i$  (foot point on tangent),  
 $\mathbf{c}_{i+1} = \text{curvepoint}(\mathbf{t}_{i+1})$ .  
**until**  $\|\mathbf{c}_{i+1} - \mathbf{c}_i\|$  is "sufficiently" small.  
Foot point:  $\mathbf{p}_0 = \mathbf{c}_{i+1}$ .

**Remark:** In case of problems considering convergence of the foot point algorithm one should improve it by a "parabola step" (see Section 5.9.2).

## 5.10 Applications of normalforms

### 5.10.1 Approximation of a set of intersecting surfaces

Given

1. the implicit surface  $(x - 2)^4 + y^4 - r_1^4 = 0$ ,  $r_1 = 2$ ,
2. the parametric surface patch  
 $\mathbf{x} = (10v - 5, 10u - 5, 6(u - u^2 + v - v^2))$ ,  $0 \leq u \leq 1, 0 \leq v \leq 0.8$ ,
3. the parametric surface patch  
 $\mathbf{x} = (6(u - u^2 + v - v^2) - 5, 10u - 5, 10v - 5)$ ,  $0 \leq u \leq 1, 0.5 \leq v \leq 1$ .

Let be  $h_1(\mathbf{x}) = 0, h_2(\mathbf{x}) = 0, h_3(\mathbf{x}) = 0$  the normalforms of the surfaces. The implicit surface  $f(\mathbf{x}) := h_1(\mathbf{x})h_2(\mathbf{x})h_3(\mathbf{x}) = c > 0$  is a smooth approximation of the set of the three given surfaces. Figure 5.3b) shows the triangulation of  $f(\mathbf{x}) = c$  for  $c = 0.2$ .



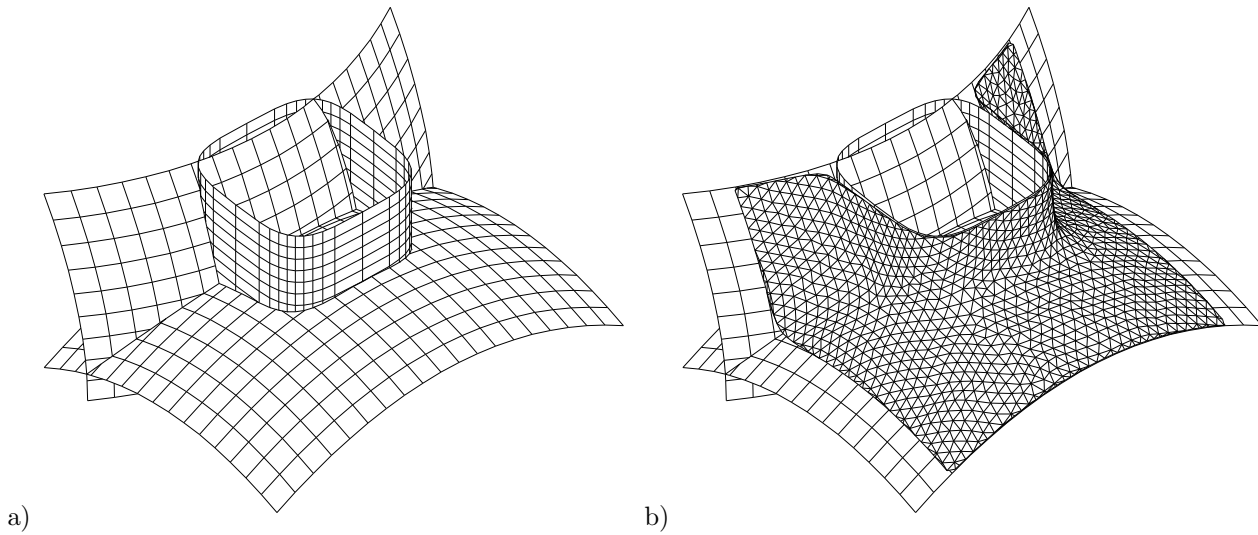


Figure 5.3: Smooth approximation of a set of surfaces

### 5.10.2 Approximation of intersecting pipe surfaces

The level surfaces  $h = \text{const}$  of a normalform function  $h$  of a space curve is a pipe surface. Hence it is an easy task to approximate two (or more) intersecting pipe surfaces  $h_1 = c_1$ ,  $h_2 = c_2$ . The implicit surfaces

$$f := (h_1 - c_1)(h_2 - c_2) = c, \quad c > 0$$

are, for  $c$  small enough, smooth approximations of the two pipe surfaces.

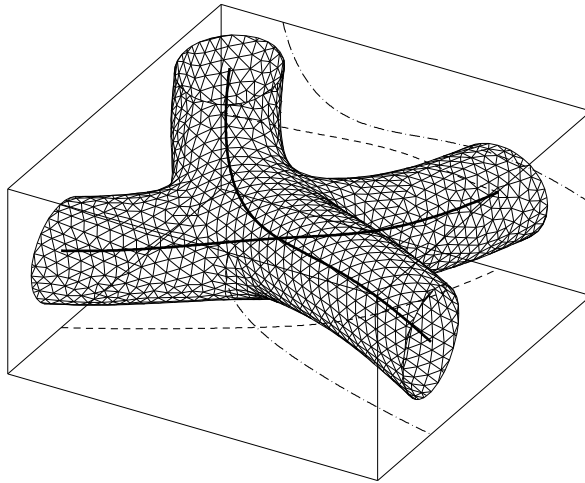


Figure 5.4: Approximation of two intersecting pipe surfaces

### 5.10.3 Numerical parametrization of implicit surfaces

Let  $f(\mathbf{x}) = 0$  be an implicit regular (i.e.  $\nabla f \neq \mathbf{0}$ ) surface with normalform  $h(\mathbf{x}) = 0$  and  $\Phi : \mathbf{x} = \mathbf{S}(u, v), (u, v) \in D \subset \mathbb{R}^2$  a parametric surface in the vicinity of surface  $f = 0$ . (For the application

below  $\Phi$  is chosen as a tensor product Bézier surface which is a rough approximation of a patch of  $f = 0$ .) Both surfaces are supposed to be differentiable enough. Then the parametric surface

$$\Phi_0 : \mathbf{x} = \mathbf{S}_0(u, v) := \mathbf{S}(u, v) - h(\mathbf{S}(u, v))\nabla h(\mathbf{S}(u, v)), \quad (u, v) \in D$$

is a patch of the given surface  $f = 0$ , because of

$$h(\mathbf{S} - h(\mathbf{S})\nabla h(\mathbf{S})) = h(\mathbf{S}) - h(\mathbf{S}) = 0$$

(see basic properties of the normalform function  $h$  in Theorem 5.1).

Differentiating yields:

$$\begin{aligned} \mathbf{S}_{0,u}^T &= (I - \nabla h^T \nabla h - hH_h)\mathbf{S}_u^T, \\ \mathbf{S}_{0,v}^T &= (I - \nabla h^T \nabla h - hH_h)\mathbf{S}_v^T. \end{aligned}$$

Function  $h$ ,  $\nabla h$  and  $H_h$  are evaluated at point  $\mathbf{S}(u, v)$ .

The **algorithm** for the numerical parameterization of an implicit surface  $f(\mathbf{x}) = 0$ :

1. Choose a tensor product Bézier or B-spline surface  $\Phi : \mathbf{x} = \mathbf{S}(u, v)$ ,  $(u, v) \in [0, 1] \times [0, 1]$  which is a rough approximation of the surface patch  $\Phi_0$  to be parameterized.
2. **Point**  $\mathbf{S}_0(u, v)$  is the foot point of point  $\mathbf{S}(u, v)$  on surface  $f = 0$ . (For an appropriate algorithm see Section 5.8.)

**Remark:**

a) Because of the result on normalforms in Theorem 5.1 the introduced parameterization of a surface is of class  $C^{n-1}$  if the auxiliary parametric surface and the given surface are  $C^n$ -continuous.

b) If there is no demand for continuity (for example for texture mapping, one may use the more simple procedure `surfacepoint` out of Section 5.8.2 instead of the foot point procedure. Usually one recognizes no evident differences.

**Example 5.7** Figure 5.5a) shows the auxiliary Bézier surface used for a numerical parameterization of a patch of the implicit surface  $f = 0$  with

$$f(x, y, z) = (1 - \mu)(x^2 + z^2 - 1)(y^2 + z^2 - 1) - \mu((r^2 - x^2 - y^2 - z^2)(c - z^2))^3$$

for  $\mu = 0.002$ ,  $r = 3$ ,  $c = 1.1$  which is a  $G^2$ -blending surface of the cylinders  $x^2 + z^2 = 1$  and  $y^2 + z^2 = 1$ . Figure 5.5b) shows the Bézier surface and the parameterized implicit surface patch.

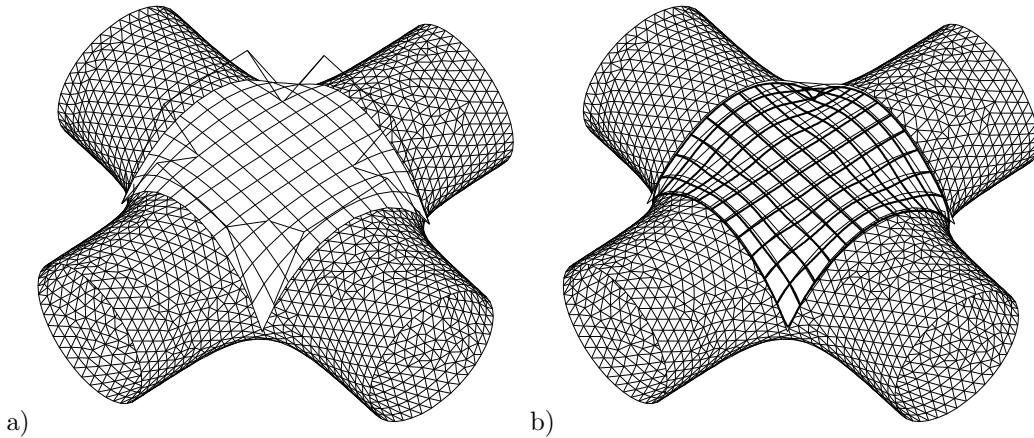


Figure 5.5: a) Bézier surface patch used for parameterization of an implicit surface b) Bézier surface patch and the parameterized implicit surface patch

## Chapter 6

# HIDDENLINE-ALGORITHM FOR NON CONVEX POLYHEDRONS, DISPLAYING PARAMETERIZED SURFACES

### 6.1 The Hiddenline-Algorithm

**Preconditions:**

Given: a) Line segments (edges)  $\{e_1, e_2, e_3, \dots\}$   
b) **planar, convex  $n$ -gons** (faces)  $\{f_1, f_2, f_3, \dots\}$

If one of the faces is not convex one divides it into several convex parts. The necessary additional edges will be ignored during the displaying process. The edges to be tested for visibility need not belong to a face of the given  $n$ -gons (For example the edges of the door and windows, see Example 5.3.).

The algorithm will work correctly for planar faces. But, usually there arise no problems if the faces are “nearly” planar only (normal case for displaying parametric surfaces).

**Idea** of the algorithm:

- (1) If the faces are oriented the invisible faces and edges are detected by the fast *normal-test* and ignored.
- (2) For “obvious cases” the simple *windows-test* omits the expensive hiddenline-test (3)–(5).
- (3) It is tested if the projections of an edge and a face may have points in common.
- (4) If test (3) is positive an investigation using a test point in  $\mathbb{R}^3$  decides whether the edge is (partly) behind a face.
- (5) If the edge is partly hidden by a face the visible segments are determined and stored.
- (6) After testing an edge against all faces the still visible segments will be drawn.

The realization of this idea needs the following procedures:

- a) `is_line_convex_polygon`: determines the intersection between a line segment and a convex polygon.

- b) `intmint`: determines the difference  $[a, b] \setminus [c, d]$  of two real intervals.
- c) `cp_vts3d_vts2d_spez`: is a slight extension of procedure `cp_vts3d_vts2d`. An additional parameter (`pdist: r_array`) contains the distances of the points from the image plane ( $z$ -coordinates of the points in system  $(H; \mathbf{e}_1, \mathbf{e}_2, \mathbf{n}_0)$ ).

```

procedure is_line_convex_polygon(p1,p2 : vt2d; p_pol : vts2d_pol; np : integer;
                               var t1,t2 : real; var ind : integer);
{determines parameters t1,t2 of the line segment p1,p2 intersecting with the
 convex Polygon p_pol[0],...p_pol[np].
 ind=0, 2 : segment within or not,
 ind=1: other cases. vts2d_pol: array[0..npfmax] of vt2d. }
{*****}
procedure intmint(a,b,c,d: real; var e1,f1,e2,f2: real; var ind: integer);
{Determines the difference of intervals [a,b] \ [c,d].
 ind=0: empty, ind=1: 1 interval, ind=2: 2 intervals}
{*****}
procedure cp_vts3d_vts2d_spez(var p: vts3d; n1,n2 : integer; var pp : vts2d;
                             var pdist : r_array);
  {central projection of an array of points.
   pdist[i] : distance of pointp[i] from image plane.}
{*****}

```

On the **data structure**:

For any **face** (polygon) the following informations are stored in *record* `face_dat`:

- 1) `npf, nef`: number of points and edges (of a face).
- 2) `fp[1], ... fp[npf]`: First, second, ... point of a face,  
`fe[1], ... fe[nef]`: First, second, ... edge of a face.  
(In case of oriented faces the points have to be oriented anti clockwise seen from "outside".)
- 3) `box`: contains minimal and maximal  $x$ -values and  $y$ -values of the image points of a face.  
`box.zmax`: maximal distance of the points from the image plane.
- 4) The normal vector `nv` and the real `d` of the equation  $\mathbf{n}_v \cdot \mathbf{x} = d$  of the plane containing the face.
- 5) `discentre`: distance of the centre from the plane containing the face.
- 6) `vis`: `vis=true` or `vis=false` if the oriented face is visible or not.

For any **edge** we use *record* `edge_dat`:

- 1) `ep1, ep2`: First and second point of the edge.
- 2) `color, linewidth`: Color and linewidth of the edge. Used only if parameter `newstyles=true` (see procedure `cp_lines_before_convex_faces`).
- 3) `vis`: `vis=true` or `vis=false` if the edge belongs to a visible face or not.

All records `face_dat` and `edge_dat` are items of the *array* `face` and `edge` respectively.

(For details: see `units/hiddenl.p` and `include/proc_zpo.pas`.)

For any **point** we store the coordinates in array `p` (type: `vts3d`).

The hiddenline-algorithm is contained in the following procedure `cp_lines_before_convex_faces`.

Explanations on `cp_lines_before_convex_faces` (`oriented_faces, is_permitted, newstyles`):

`oriented_faces=true`: The faces are oriented. The fast normal-test will be applied.

`is_permitted=true`: The edges may intersect the faces. (slow !)

`newstyles=true`: Color and linewidth as defined in record `edge[i]`.

- (0) The main program has to provide:
- a)  $np, nf, ne$  : number of points, faces, edges resp.,
  - b) the coordinates of the points  $P_i : \mathbf{p}_i, \quad i = 1, \dots, np$ ,
  - c) for any edge: first and last points  $ep1, ep2$ ,
  - d) for any face:
    - $npf, nef$  : number of points and edges,
    - $fp[1], \dots, fp[npf]$  : points of the face,  $fe[1], \dots, fe[npf]$  : edges of the face, the equation  $\mathbf{n}_v \mathbf{x} = d$  of the corresponding plane.
- (1) there will be
- a) determined the coordinates of the image points and their distance from the centre,
  - b) determined the distance  $discentre$  of the centre from the corresponding plane, stored the box  $(xmin, xmax, ymin, ymax)$  of the window and the maximal distance  $(box.zmax)$  of the points (of the face) to the image plane.
  - b') performed the normal test for oriented faces.  
If the face is visible their edges are visible, too

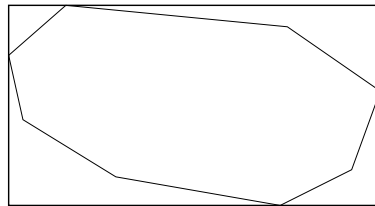


Figure 6.1: Window of a polygon

- (2) testing edge  $e_i$  against face  $f_j$  if both are visible ( $vis=true$ ):
- a) For the edge  $e_i$  the box  $xemin, xmax, yemin, yemax$  of the window and the minimal distance  $zemin$  from the image plane is determined.
  - b) Is the edge an edge of face  $f_j$  ? (yes:  $\rightarrow$  next face.)
  - c) In case of  $box.zmax > zemin$  and if edge widow and face window are overlapping: Intersection of the images of the edge and the face (convex polygon)

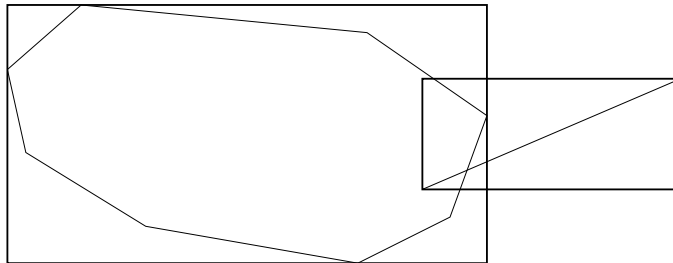


Figure 6.2: Windows test

- d) If in image plane the edge is partly contained in the polygon:
- d1) if `is_permitted=false`:  
 Test using test point  $\mathbf{p}_t$  whether (in space) the edge is in front of the plane of the face or not.

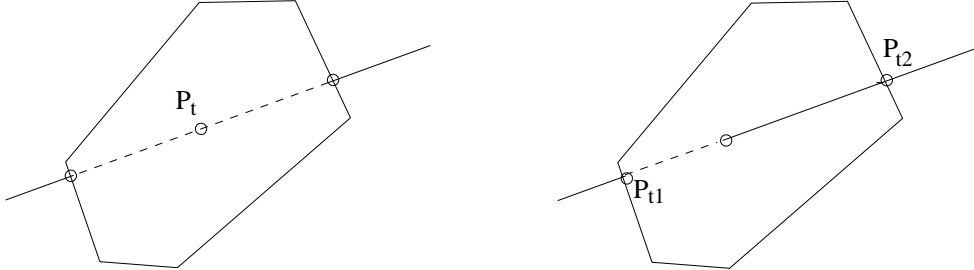


Figure 6.3: Case d1) and d2) resp.

- d2) if `is_permitted=true`:  
 Using test points  $\mathbf{p}_{t1}, \mathbf{p}_{t2}$  the segment of the edge that is in front of the face is determined.
- e) if the edge is hidden partly:  
 The parameter intervals  $[\text{par1}(1), \text{par2}(1)]$  of the visible segments are determined and stored.

(3) after testing edge  $e_i$  against all faces the visible segments are drawn.

```

procedure cp_lines_before_convex_faces(oriented_faces,is_permitted,newstyles : boolean);
{projects and draws segments of an edge in front of planar n-gons.
 oriented_faces=true: faces are oriented,
 is_permitted=true: edges may intersect the faces.
 The main program has to provide:
 np, ne, nf: number of points, edges, faces,
 p[1],...,p[np] : points,
 face[i].fp[k] (face[i].fe[k]): k-th point (k-th edge) of i-th face,
 face[i].npf (face[i].nef): number of points (edges) of th i-th face,
 edge[i].ep1 (edge[i].ep2): first (last) point of i-th edge,
 face[i].nv,face[i].d: coefficients of the equation of the i-th face.}
{*****}

```

**Example 6.1** a) *Polyhedron on a torus (oriented faces)*

b) *Polyhedron on a torus segment (not oriented faces)*

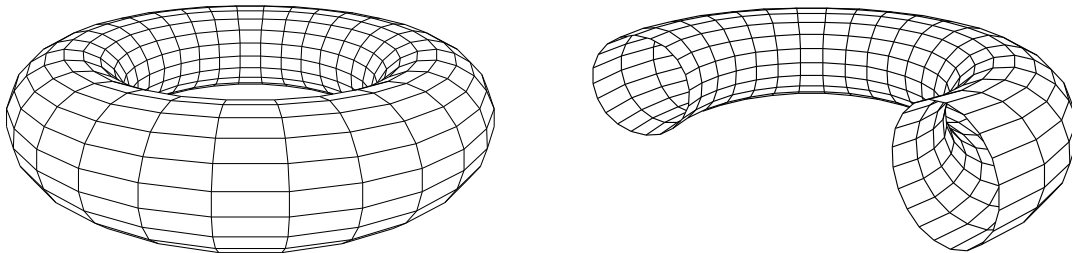


Figure 6.4: Torus (orient. faces) and torus segment (not orient. faces)

## 6.2 Auxiliary procedures for the hiddenline algorithm

In order to reduce preliminary work for the user of the hiddenline algorithm we give 4 auxiliary procedures. The first one is an aid in case of a general polyhedron, the second, third and fourth eases the application in case of parametric surfaces of certain topologies.

### 6.2.1 The procedure `aux_polyhedron`

The main program must provide:

- (1) `np, nf` (number of points and faces resp.),
- (2) the points  $P_i : \mathbf{p}_i$ ,  $i = 1, \dots, np$  of the faces and edges,
- (3) for each face: `npf` (number of points),  
`fp[1], \dots, fp[npf]` (indices of the points defining the face polygon).

The procedure `aux_polyhedron` generates the indices of the edges and determines

- (1) `ne` (number of edges),
- (2) for each edge: `ep1, ep2` (indices of first and last points),
- (3) for each face: `nef, fe[1], \dots, fe[nef]` (number of edges and indices of edges contained in the face).

```

procedure aux_polyhedron;
{np, ne, nf : number of points, edges and faces resp.
 face[i].npf   : number of points (edges) of the i-th face
 edge[k].ep1, ep2 : first and last point the k-th edge
 face[i].fp[k] : k-th point of i-th face (positively oriented !!!)
 face[i].fe[k] : k-th edge of i-th face
 ! This procedure determines from given nf, face[i].fp and face[i].npf:
 ne, face[i].nef, face[i].fe[k], edge[i].ep1 and edge[i].ep2 !.}

```

**Example 6.2** *houses* (see Figure 6.5)

(One should recognize that

- a) the roof faces are not oriented and
- b) edges, which are not edges bounding a face (window, door), must not lie in a face but a little “in front”. Here the set of edges consists of the set of edges bounding faces and additional edges indicating the windows and doors.)

**Example 6.3** a) 3 beams b) kiosk (see Figure 6.6)

### 6.2.2 The procedures `aux_quadrangle`, `aux_cylinder`, `aux_torus` and displaying a parametrized surface

For displaying a parametrized surface  $\Phi : \mathbf{x} = \mathbf{S}(u, v)$ ,  $u \in [a, b]$ ,  $v \in [c, d]$ , one chooses a net of rectangles in the parameter plane ( $u - v$ -plane) and displays its image in  $\mathbb{R}^3$  (see Figure 6.7).

Usually the rectangles of such a net are not planar. But if the rectangles are small enough they will be “nearly planar” and one can apply the introduced hiddenline algorithm for displaying the net. (In case of the torus the rectangles are planar.) The algorithm `aux_polyhedron` of section 6.2 may be applied to the torus example, too. But, as of its special structure (all faces are rectangles!) we use an auxiliary procedure which is more simple to handle.

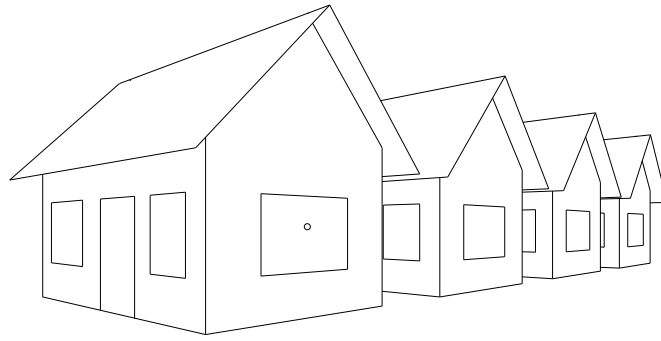


Figure 6.5: houses

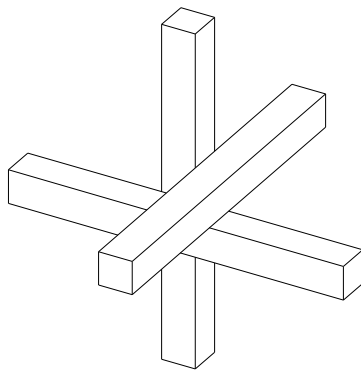
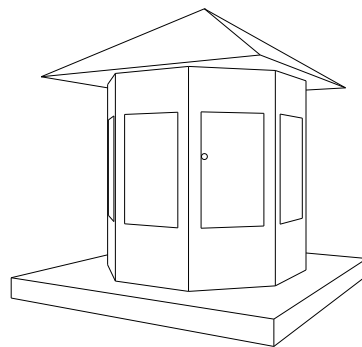
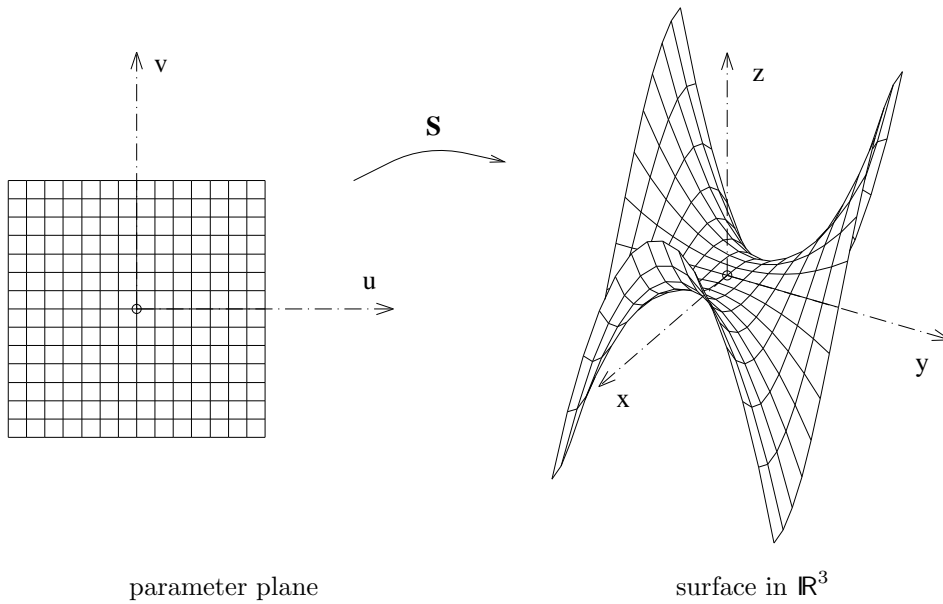


Figure 6.6: 3 beams,



kiosk



parameter plane

surface in  $\mathbb{R}^3$ 

Figure 6.7: a net in parameter plane and its image on the surface



The following procedures should be applied to parametric surface patches which are rectangular in parameter plane and which bear the topology of either an rectangle or a cylinder or a torus. From  $n_1, n_2$  (number of divisions in u- and v-direction) the auxiliary procedures generate the numbering of the faces and edges and the following datas:

- (1) for each face: **fp**[1], ..., **fp**[**npf**] (indices of the points of the polygon),  
**fe**[1], ..., **fe**[**nef**] (indices of the edges of the polygon),
- (2) for each edge: **ep1**, **ep2** (indices of the bounding points);
- (3) **np**, **ne**, **nf** : numbers of the points, edges and faces,
- (4) the equations  $\mathbf{n}_i \mathbf{x} - d_i = 0$  of the planes containing the faces

“quadrangle”

“cylinder”

“torus”

Figure 6.8: net types

The  $n_1 \cdot n_2$  points  $P_1, P_2, \dots$  must be generated in the following way:

$$\begin{aligned} \text{In case of "quadrangle": } & \Delta u = (b - a)/(n_1 - 1) \quad \text{and} \quad \Delta v = (c - d)/(n_2 - 1). \\ \text{In case of "cylinder": } & \Delta u = (b - a)/n_1 \quad \text{and} \quad \Delta v = (c - d)/(n_2 - 1). \\ \text{In case of "torus": } & \Delta u = (b - a)/n_1 \quad \text{and} \quad \Delta v = (c - d)/n_2. \end{aligned}$$

$$P_j : \mathbf{p}_j = \mathbf{S}(a + (i - 1) \cdot \Delta u, c + (k - 1) \cdot \Delta v),$$

$$j = (k - 1) \cdot n_1 + i, \quad 1 \leq i \leq n_1 \quad 1 \leq k \leq n_2.$$

In order to display a collection of surface patches we use the integers **np0**, **ne0**, **nf0** to describe the numbers of points, edges and faces already stored. For the first patch we set **np0=ne0=nf0=0**.

```
procedure aux_quadrangle(n1,n2,np0,ne0,nf0: integer);
{****}
procedure aux_cylinder(n1,n2,np0,ne0,nf0: integer);
{****}
procedure aux_torus(n1,n2,np0,ne0,nf0: integer);
```

The main program has to provide:

- a) **n1, n2**,
- b) The points  $P_j : \mathbf{p}_j$  (for numbering: see above and Figure 6.8),
- c) **npf, nef**, number of points and edges of a face: **npf=4, nef=4**. (When considering intersections of faces (see Section 6.3.2) these numbers may raise.)

**Example 6.4** *Two tori (see sample program `tori.h.p` and Figure 6.9)*

**Example 6.5** *Pipe knot (see Figure 6.10)*

The directrix of a pipe knot lies on a torus and has (in the present case) the following parametric representation:

$$\mathbf{x} = \mathbf{c}(u) := (h \cos(u), h \sin(u), r_2 \sin(cu)),$$

with  $h = r_1 + r_2 \cos(cu)$ ,  $c = 1.5$  and  $0 \leq u \leq 4\pi$ .

( $r_1$  is the “big” radius,  $r_2$  the “small” radius of the torus.) Each point of the directrix is the

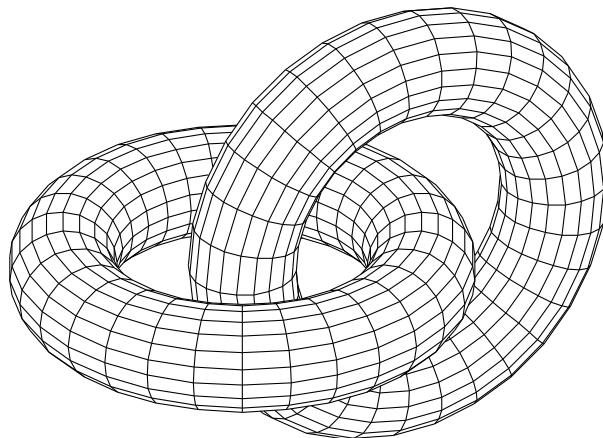


Figure 6.9: Two tori

midpoint of a circle contained in the normal plane of the current directrix point. The normal plane corresponding to  $\mathbf{c}(u)$  will be described by the following base vectors:

$$\mathbf{e}_1(u) := \dot{\mathbf{c}}(u) \times \mathbf{e}_z / \|\dots\|, \quad \mathbf{e}_2(u) := \mathbf{e}_1 \times \dot{\mathbf{c}}(u) / \|\dot{\mathbf{c}}(u)\|,$$

where  $\mathbf{e}_z := (0, 0, 1)$  is.

Hence

$$\mathbf{x} = \mathbf{S}(u, v) := \mathbf{c}(u) + \mathbf{e}_1(u)r_3 \cos v + \mathbf{e}_2(u)r_3 \sin v, 0 \leq r_3 \leq r_2,$$

is a parametric representation of the pipe knot.  $r_3$  is radius of the pipe.

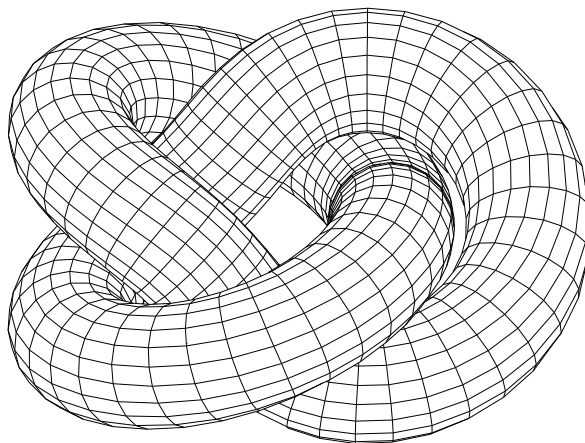


Figure 6.10: pipe knot (type: "torus")

**Example 6.6** *Moebius strip (NOT of type "quadrangle", see Figure 6.11)  
(The boundary curves lie on a torus !)*

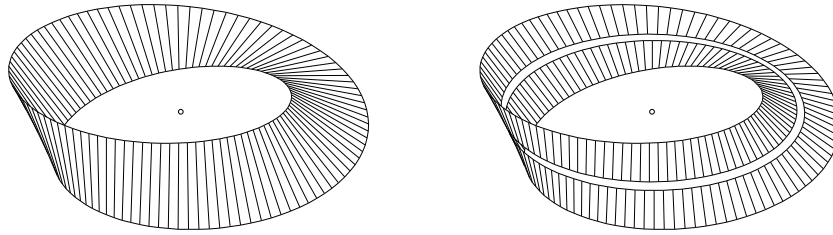


Figure 6.11: Moebius strip, sliced Moebius strip

### 6.3 Intersection of faces bounded by planar polygons in $\mathbb{R}^3$ , intersection of two polyhedrons

#### 6.3.1 Intersection of faces bounded by planar polygons in $\mathbb{R}^3$

Given: Two planar convex polygons  $P_{11}, P_{12}, \dots, P_{1m}, P_{21}, P_{22}, \dots, P_{2n}$  in  $\mathbb{R}^3$ .

The polygons are not contained in a common plane.

Sought: The intersection (line segment) of the faces bounded by the polygons.

**Idea** of the algorithm:

- (1) Determine the intersection line of the planes containing the polygons.
- (2) The intersection of this line with each (convex) polygon consists of two line sections.
- (3) The intersection of these two line sections is the common line section of the faces.

The **algorithm** (procedure `is_n1gon_n2gon3d`):

- (0) For each polygon procedure `box3d_of_pts` determines the corresponding bounding box.
- (1) If the bounding boxes have a non empty intersection, then the equations  $\mathbf{n}_1\mathbf{x} - d_1 = 0, \mathbf{n}_2\mathbf{x} - d_2 = 0$  of the planes containing the polygons are determined by `plane_equ`. `is_plane_plane` returns a point and a direction vector of the intersection line  $g_s$ .
- (2) For intersecting the line  $g_s$  with the first polygon we introduce the following coordinate system:  $P_{11}$  is the origin and the points  $P_{12}, P_{1m}$  are the unit points of the coordinate axes (see Figure 6.12).

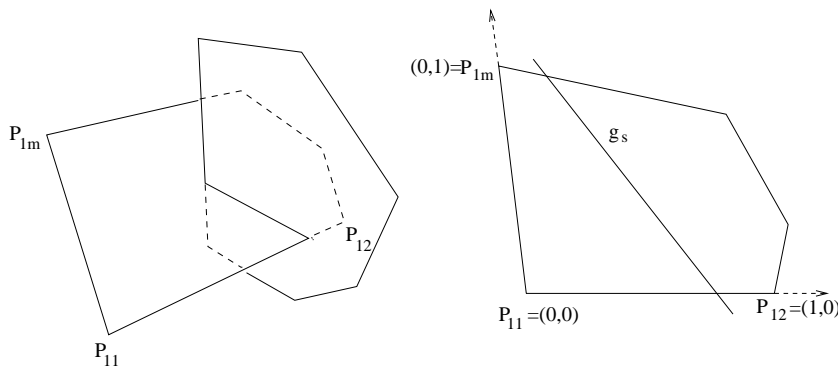


Figure 6.12: to (2) of the algorithm

Because the points of the polygons are not contained exactly in a plane we determine the coordinates of their foot points  $Q_{1i}$  using procedure `ptco_plane3d` (see file `proc_ag.pas`). The

origin and unit points are:  $Q_{11} = (0,0)$ ,  $Q_{12} = (1,0)$  and  $Q_{1m} = (0,1)$ . After determining the foot points of two points of line  $g_s$  procedure `is_line_convex_polygon` calculates the parameters  $t_1, t_2$  of the common line section. Analogously one gets the parameters  $s_1, s_2$  for the second polygon.

- (3) From `is_interv_interv` we get the common interval  $[s, t] := [t_1, t_2] \cap [s_1, s_2]$  and the line segment common to both the faces bounded by the polygons.

```

procedure is_interv_interv(var a,b,c,d,aa,bb : real; var inters: boolean);
{determine the intersection of the intervals [a,b] and [c,d] .}
{*****}
procedure box3d_of_pts(var p : vts3d_pol; np: integer; var box : box3d_dat);
var a,b : real; i: integer;
{determines the bounding box.}
{*****}

function is_two_boxes3d(var box1,box2 : box3d_dat) : boolean;
{intersection of two boxes.}
{*****}
procedure is_line_conv_pol_in_plane3d(var pl,rl: vt3d; var pp : vts3d_pol;
                                     npp : integer;
                                     var t1,t2 : real; var inters : boolean);
{intersection of a convex polygon and a line contained in a common plane.}
{*****}

```

### 6.3.2 Intersection of two polyhedrons

Given: Two intersecting polyhedrons  $\Pi_1, \Pi_2$ .

Sought: Their intersection.

#### Idea:

We use the data structure of the hiddenline algorithm `is_lines_before_convex_faces` (see Section 6.1) This causes some simplifications.

#### The algorithm:

- (1) Determine the bounding boxes using `boxes_of_faces`
- (2) Because the equations of the planes are already determined by the auxiliary procedures (see Section 6.2) we use the more appropriate procedure `is_face_face` for determining the common line section of two faces.
- (3) The common line section of the  $i$ -th and the  $k$ -th face will be appended to
  - array `edge`,
  - arrays `face[i]` and `face[k]`
 as an additional edge.

The heads of procedures `boxes_of_faces`, `is_face_face` and their application in the main program are as follows (see sample program `parsurf.h.p`):

```

procedure boxes_of_faces;
{determines the bounding boxes.}
{*****}

```

```

procedure is_face_face(i,k: integer; var ps1,ps2 : vt3d;
                        var intersection: boolean);
{determines the intersection of two (convex) faces.}
{*****}
.....
{Intersection of two surfaces: (s. parsurf_h.p)}
boxes_of_faces;
for i:= 1 to nf1 do      { 1. loop }
begin
  for k:= nf1+1 to nf do { 2. loop }
  begin
    is_face_face(i,k, ps1,ps2,inters);
    if inters then
      begin
        p[np+1]:= ps1; p[np+2]:= ps2;
        with edge[ne+1] do
          begin ep1:= np+1; ep2:= np+2; vis:= true; end;
        np:= np+2; ne:= ne+1;
        with face[i] do begin fe[nef+1]:= ne; nef:= nef+1; end;
        with face[k] do begin fe[nef+1]:= ne; nef:= nef+1; end;
        end; { if }
      end; { for k }
    end; { for i }
  .....

```

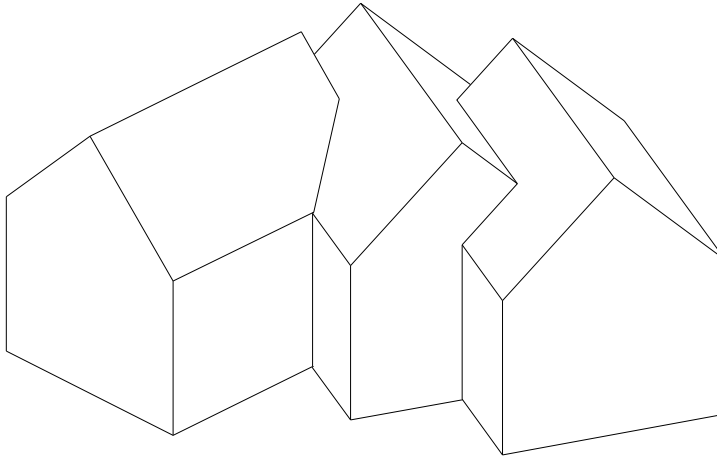


Figure 6.13: Intersection of polyhedrons

**Remark:**

Selfintersections of polyhedrons cause some problems when intersecting neighbouring faces. So, one should detect neighbouring faces first and intersect non neighboured faces only (see Figure 6.14b):  $\mathbf{x}(u, v) = ((-40 + 80u) \cos(v), (-40 + 80u) \sin(v), 40 - 40u + \frac{50}{2\pi}v)^\top$ ,  $0 \leq u \leq 1$ ,  $0 \leq v \leq 3\pi$ .

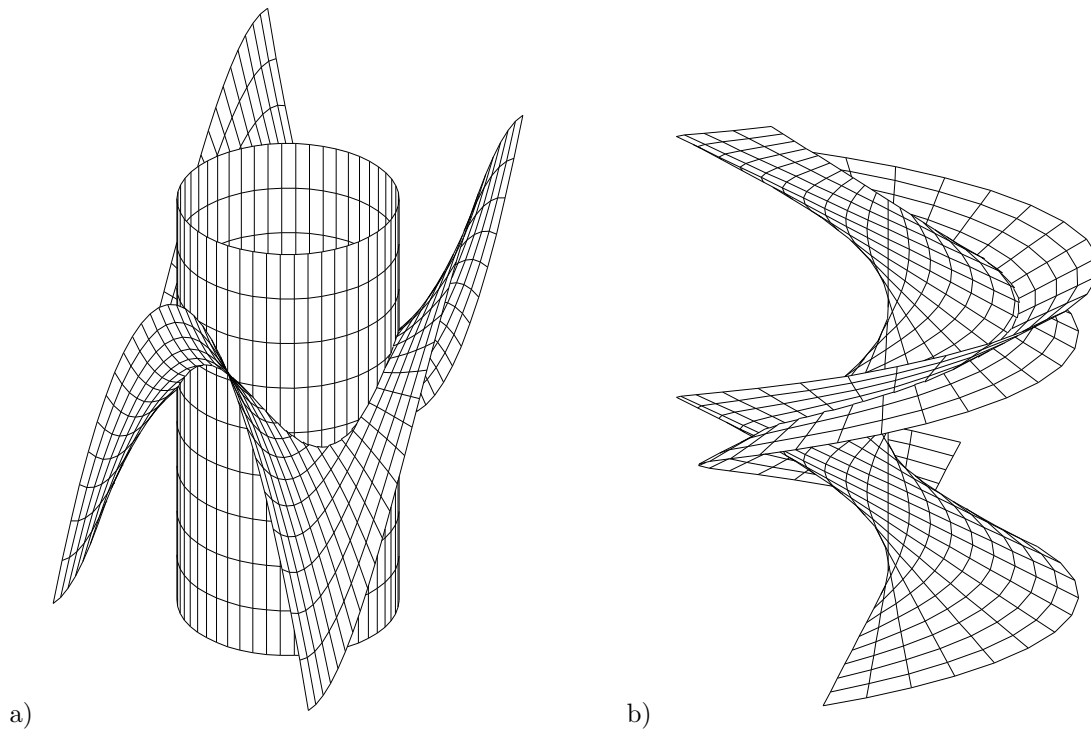


Figure 6.14: a) intersecting surfaces

b) selfintersecting surface

The procedures of this section are contained in file `include/proc_zpo.pas` and are available via unit `hiddenl` (see sample programs `tori.h.p`, `parsurf.h.p` and Figure 6.14).

## 6.4 Intersection of a line segment and a planar polygonal patch, intersection of a polygon and a polyhedron

### 6.4.1 Intersection of a line segment and a planar polygonal patch

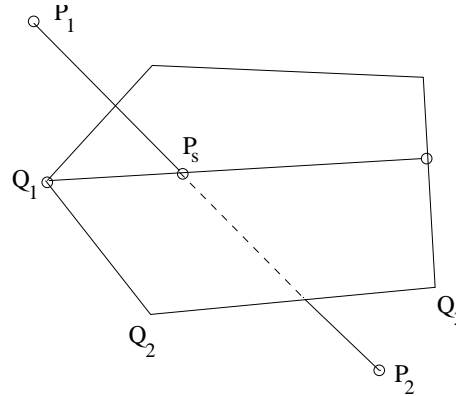
Given: Line segment  $P_1P_2$  and a planar polygonal patch  $Q_1, Q_2, \dots, Q_n$  in  $\mathbb{R}^3$ .

The plane containing the polygon contains not the line segment.

Wanted: The intersection point of the line segment and the patch bounded by the polygon.

The **algorithm**:

- (1) Determining bounding boxes for the line segment and the polygon. Test for intersection of the boxes.
- (2) If the boxes intersect, the equation  $\mathbf{n}\mathbf{x} - d = 0$  of the plane  $\varepsilon$  containing the polygon is determined by procedure `plane_equ`. The signs of  $\mathbf{n} \cdot \mathbf{p}_1 - d, \mathbf{n} \cdot \mathbf{p}_2 - d$ , where  $P_i : \mathbf{p}_i$ , indicate if the line segment intersects the plane  $\varepsilon$ .
- (3) If the line intersects the plane, the intersection point  $P_s$  is determined and using procedure `is_line_conv_pol_in_plane3d` the intersection of the line  $P_s, Q_1$  (contained in  $\varepsilon$ ) and the polygon is investigated. If there are *two* intersection points and  $P_s$  is separating them, point  $P_s$  is contained in the polygonal patch and the wanted intersection point is  $P_s$ .

Figure 6.15: To (3) of the algorithm `is_line_ngon3d`

```

procedure is_line_ngon3d(p1,p2: vt3d; var pp: vts3d_pol; npp: integer;
                        var ps: vt3d; var intersection: boolean);
  {Determines the intersection of a line segment and a planar convex n-gon in space.
   The line segment must not be contained in the plane of the polygon.}
label 5;
var d,t1,t2,xemin,xemax,yemin,yemax,zemin,zemax,dp1,dp2,ts : real;
    j : integer;    box : box3d_dat;    nv,rv,rs : vt3d;    inters : boolean;
begin
  intersection:= false;
  xemax:= max(p1.x,p2.x);    yemax:= max(p1.y,p2.y);    zemax:= max(p1.z,p2.z);
  xemin:= min(p1.x,p2.x);    yemin:= min(p1.y,p2.y);    zemin:= min(p1.z,p2.z);
  box3d_of_pts(pp,npp, box);
  if (xemax<=box.xmin) or (xemin>=box.xmax) or
     (yemax<=box.ymin) or (yemin>=box.ymax) or
     (zemax<=box.zmin) or (zemin>=box.zmax)
  then goto 5; {edge not hidden by j-th face}
  plane_equ(pp[1],pp[2],pp[3] , nv,d,error);
  dp1:= scalarp3d(p1,nv) - d;
  dp2:= scalarp3d(p2,nv) - d;
  if dp1*dp2<=0 then
    begin
      ts:= dp1/(dp1-dp2);          diff3d(p2,p1, rv);
      lcomb2vt3d(1,p1,ts,rv, ps);  diff3d(ps,pp[1], rs);
      is_line_conv_pol_in_plane3d(ps,rs,pp,npp, t1,t2,inters);
      if inters and (t1*t2<=0) then intersection:= true;
    end; {if}
5: end;    { is_line_ngon3d }

```

### 6.4.2 Intersection points of a polygon and a polyhedron

Given: Polygon with edges  $e_1, e_2, \dots, e_n$ , polyhedron with convex faces  $f_1, f_2, \dots, f_m$ .

Wanted: Intersection points of the polygon and the polyhedron.

Solution: Apply procedure `is_line_ngon3d` to any pair  $e_i, f_j, 1 \leq i \leq n, 1 \leq j \leq m$ .

**Example 6.7** *Intersection of a polygon on a circle and a polyhedron on a torus.*

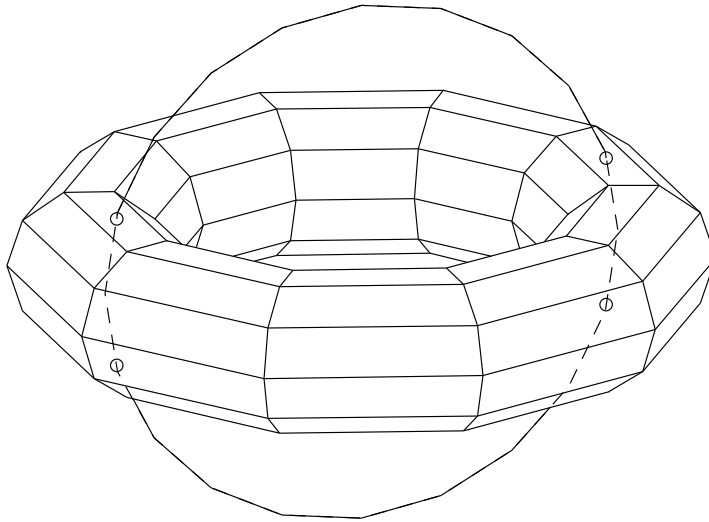


Figure 6.16: Intersection polygon-polyhedron



## Chapter 7

# TRIANGULATION OF IMPLICIT SURFACES

Determining single points of an implicit surface is not trivial. The generation of a net of polygons on an implicit surface (for displaying, finite element methods) is a challenge. Polygonization algorithms for implicit surfaces are available in literature. For example: Algower & Gnutzmann (1991, *Computer Aided Geom. Des.* 8), Schmidt (1993, *The Visual Comp.* 10) and Hartmann (1998, *The Visual Comp.* 14). All these methods except the latter one divide the space into suitable polyhedrons (cubes, tetrahedrons) and determine the section of the given implicit surface with the edges of these polyhedrons.

The method introduced by the author is a *marching* method which means to build successively a mesh of triangles by starting with a point or a prescribed polygon. The triangulation is terminated by several bounding polygons (on the given surface) or a global bounding box. The method was established for implicit surfaces. But using the idea of the *normalform* of a surface the triangulation is applicable to any surface for which *foot points* (i.e. points of minimal distance to the surface) can be determined.

### Remark:

The following triangulation algorithm appeared in *The Visual Computer* 14 (1998), pp 95–108, copyright by Springer-Verlag. The original publication is available on <http://link.springer.de/link/service/journals/00371/tocs/8014003.htm>

## 7.1 The triangulation algorithm (marching method)

The formulation of the triangulation algorithm uses no special representation of the surface to be triangulated. The operations depending on the representation are hidden in the used procedure `surfacepoint` which is defined below.

The next subsection gives first a survey and basic ideas of the algorithm. Afterwards the procedure `surfacepoint` and the data structure used is introduced, and the steps of the algorithm explained in detail.

### 7.1.1 The idea of the algorithm

- S0 Choose a point  $\mathbf{s}$  in the neighborhood of the the surface. Determine the corresponding surface point  $\mathbf{p}_1$ . Surround  $\mathbf{p}_1$  with a regular hexagon  $\mathbf{q}_2, \dots, \mathbf{q}_7$  in the tangent plane. With procedure `surfacepoint` determine the points  $\mathbf{p}_2, \dots, \mathbf{p}_7$  corresponding to the starting points  $\mathbf{q}_2, \dots, \mathbf{q}_7$ .

The triangles of the surface hexagon are the first 6 triangles of the triangulation (Fig. 7.1, Fig. 7.3).

We call the ordered array  $\mathbf{p}_2, \dots, \mathbf{p}_7$  of points the first *actual front polygon*  $\Pi_0$ . If the triangulation should be limited (not necessary for closed surfaces) by closed surface curves  $\Gamma_1, \Gamma_2, \dots$  (c.f. examples below) we determine *bounding* front polygons  $\Pi_1, \Pi_2, \dots$  on these curves.

For special surfaces (cylinder, torus,...) it might be convenient to start with a prescribed actual front polygon first (c.f. section 3).

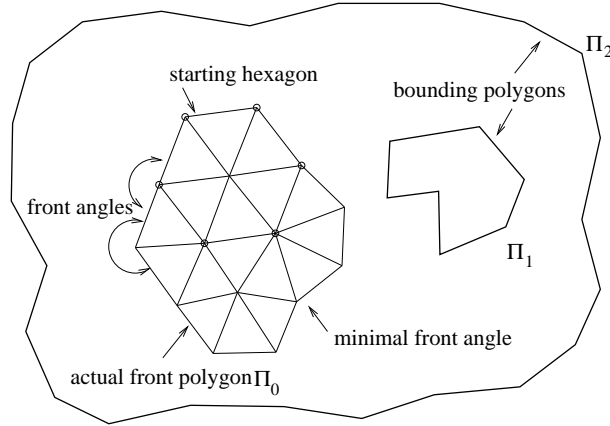


Figure 7.1: Basic notions for the triangulation algorithm

S1 For every point of the actual front polygon  $\Pi_0$  we determine the angle of the area still to be triangulated. We call these angles *front angles* (Fig. 7.1).

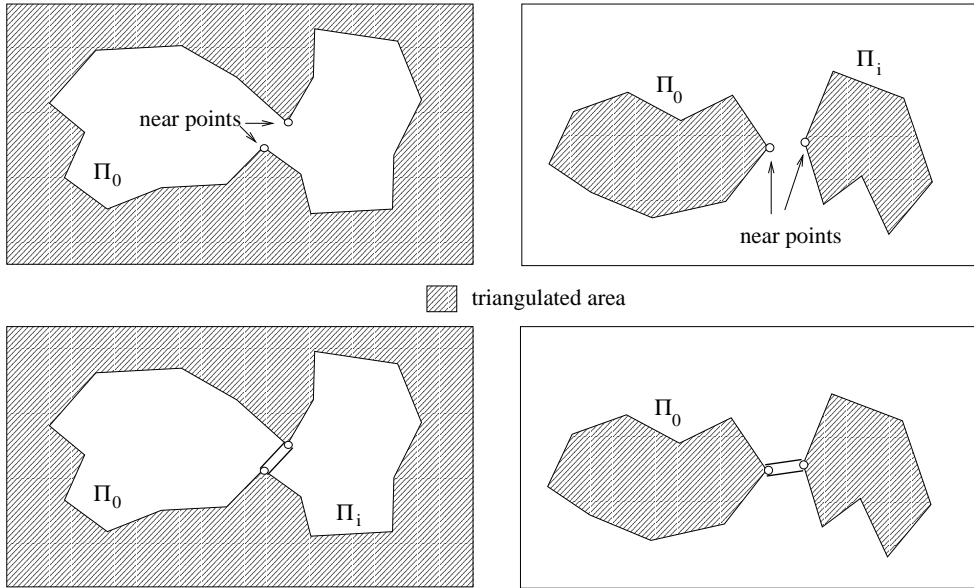


Figure 7.2: Dividing (left) and uniting (right) the actual front polygon

S2 Check if any point  $\mathbf{p}_i$  of the actual front polygon is "near"

- a point of  $\Pi_0$  which is different from  $\mathbf{p}_i$  and its neighbors or
- a point of any other front Polygon  $\Pi_k, k > 0$ .

In the first case: Divide the actual front polygon  $\Pi_0$  into a smaller one and an additional front polygon. (see Fig. 7.2, 7.8a,b)

In the second case: If  $\mathbf{p}_i$  is near a point of the front polygon  $\Pi_m$  then unite the polygons  $\Pi_0$ ,  $\Pi_m$  to a new and larger actual front polygon. Delete  $\Pi_m$ . (see Fig. 7.2, Fig. 7.8d,e)

S3 Determine a front point  $\mathbf{p}_m$  of the actual front polygon  $\Pi_0$  with minimal front angle. Surround  $\mathbf{p}_m$  by triangles with angles  $\approx 60^\circ$ . Delete  $\mathbf{p}_m$  from the polygon  $\Pi_0$  and insert the new points into the actual front polygon  $\Pi_0$ .

S4 Repeat steps S1,S2,S3 until the actual front polygon  $\Pi_0$  consists of only three points which generate a new triangle. If there is still a further (non empty) front polygon left, it becomes the new actual front polygon  $\Pi_0$  and repeat steps S1,S2,S3. If there is no further front polygon then the triangulation is finished.

If the surface is not bounded the triangulation should be limited by bounding polygons on the surface or a global bounding box (c.f. section Examples).

### 7.1.2 The procedure surfacepoint

An essential step of the triangulation algorithm is to determine a surface point  $\mathbf{p}$  that is "near" a given point  $\mathbf{q}$  in the vicinity of a surface.  $\mathbf{q} - \mathbf{p}$  need not be exactly perpendicular to the surface. Because nearly all surfaces can be numerically implicitized we give a solution for implicit surfaces.

We start with an implicitly given surface  $\Phi : f(\mathbf{x}) = 0$  for which the gradient  $\nabla f$  exists and is not zero for any point of consideration and a point  $\mathbf{q}$  in the neighborhood of the surface.

The following procedure `surfacepoint` calculates a surface point  $\mathbf{p}$ , a normal and two tangent vectors at  $\mathbf{p}$ .

1. (a)  $\mathbf{u}_0 = \mathbf{q}$   
 (b) repeat  $\mathbf{u}_{k+1} := \mathbf{u}_k - \frac{f(\mathbf{u}_k)}{\nabla f(\mathbf{u}_k)^2} \nabla f(\mathbf{u}_k)$   
 (Newton-step for the function  $g_k(t) := f(\mathbf{u}_k + t \nabla f(\mathbf{u}_k))$ )  
 until  $\|\mathbf{u}_{k+1} - \mathbf{u}_k\|$  is "sufficiently" small.  
 Surface point  $\mathbf{p} = \mathbf{u}_{k+1}$ .
2. The *surface normal* at surface point  $\mathbf{p}$  is  $\mathbf{n} := \nabla f(\mathbf{p}) / \|\dots\|$ .
3. For *tangent vectors* we choose  
 $\mathbf{t}_1 := (n_y, -n_x, 0) / \|\dots\|$  if  $n_x > 0.5$  or  $n_y > 0.5$   
 else  $\mathbf{t}_1 := (-n_z, 0, n_x) / \|\dots\|$   
 and  $\mathbf{t}_2 := \mathbf{n} \times \mathbf{t}_1$  where  $(n_x, n_y, n_z) := \mathbf{n}$ .

### 7.1.3 The data structure

For the construction of the triangles we need a *step length*  $\delta_t > 0$  which is approximately the length of the edges.

The *points* of the triangulation will get current numbers. For any point  $\mathbf{p}_i$  we keep the following informations:

- the coordinates,
- the surface normal  $\mathbf{n}$  and tangent vectors  $\mathbf{t}_1, \mathbf{t}_2$  such that  $\mathbf{n}, \mathbf{t}_1, \mathbf{t}_2$  are orthonormal,
- the actual front angle if  $\mathbf{p}_i$  is an actual front point,

- the boolean variable `angle_changed` with `angle_changed=true` if the actual front angle was changed and has to be recalculated,  
the boolean variable `border_point`. It is set `true` if point  $\mathbf{p}_i$  is on the border of the triangulation and should be ignored for further considerations (recalculation of front angle, distance check (S2)).

The *triangles* will be numbered consecutively. For any triangle we store the numbers of the vertices.

The *front polygons*  $\Pi_0, \Pi_1, \dots$  are represented by the integer arrays of their point numbers.

### 7.1.4 Step S0

Let be  $\mathbf{s}$  a starting point in the vicinity of the surface. The procedure `surfacepoint` determines the first point  $\mathbf{p}_1$  of the triangulation and the orthonormal system  $\mathbf{n}_1, \mathbf{t}_{11}, \mathbf{t}_{12}$ . The following six points  $\mathbf{p}_2, \dots, \mathbf{p}_7$  are the results of procedure `surfacepoint` applied to

$$\mathbf{q}_{i+2} := \mathbf{p}_1 + \delta_t \cos(i\pi/3)\mathbf{t}_{11} + \delta_t \sin(i\pi/3)\mathbf{t}_{12}, \quad i := 0, \dots, 5,$$

which are points of a regular hexagon in the tangent plane at  $\mathbf{p}_1$ .

We get the first 6 triangles (Fig. 7.3a):

$$(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3), (\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4), (\mathbf{p}_1, \mathbf{p}_4, \mathbf{p}_5), (\mathbf{p}_1, \mathbf{p}_5, \mathbf{p}_6), (\mathbf{p}_1, \mathbf{p}_6, \mathbf{p}_7), (\mathbf{p}_1, \mathbf{p}_7, \mathbf{p}_2).$$

### 7.1.5 Step S1

If a point  $\mathbf{p}_{0i}$  of the actual front polygon  $\Pi_0 = (\mathbf{p}_{01}, \mathbf{p}_{02}, \dots, \mathbf{p}_{0N_0})$  has just been inserted or if a neighbor of  $\mathbf{p}_{0i}$  is a new point then it is necessary to recalculate the actual front angle  $\omega$  at point  $\mathbf{p}_{0i}$ . Let

$$\mathbf{v}_1 := \mathbf{p}_{0,i-1} \text{ if } i > 1 \text{ or } \mathbf{v}_1 := \mathbf{p}_{0N_0} \text{ if } i = 1,$$

$$\mathbf{v}_2 := \mathbf{p}_{0,i+1} \text{ if } i < N_0 \text{ or } \mathbf{v}_2 := \mathbf{p}_{01} \text{ if } i = N_0 \text{ and}$$

$(\xi_1, \eta_1, \zeta_1)$  the coordinates of  $\mathbf{v}_1$ ,  $(\xi_2, \eta_2, \zeta_2)$  the coordinates of  $\mathbf{v}_2$  in the local orthonormal system  $\mathbf{n}, \mathbf{t}_1, \mathbf{t}_2$  at point  $\mathbf{p}_{0i}$ ,

$\omega_1 := \text{polar angle of } (\xi_1, \eta_1), \quad \omega_2 := \text{polar angle of } (\xi_2, \eta_2)$ . Then the front angle at point  $\mathbf{p}_{0i}$  is  $\omega = \omega_2 - \omega_1$  if  $\omega_2 \geq \omega_1$  else  $\omega = \omega_2 - \omega_1 + 2\pi$ . (Fig. 7.3b)

### 7.1.6 Step S2

In order to prevent new triangles from overlapping existing triangles we check

- the distances of pairs of points of the actual front polygon  $\Pi_0$ . If there are points  $\mathbf{p}_{0i}, \mathbf{p}_{0j}, i < j$ , which are neither neighbors nor neighbors of neighbors and  $\|\mathbf{p}_{0i} - \mathbf{p}_{0j}\| < \delta_t$  then the actual front polygon is split into the new actual front polygon  $(\mathbf{p}_{01}, \dots, \mathbf{p}_{0i}, \mathbf{p}_{0j}, \dots, \mathbf{p}_{0N_0})$  with  $N_0 - (j - i - 1)$  points and a further front polygon  $(\mathbf{p}_{0i}, \dots, \mathbf{p}_{0j})$  with  $j - i + 1$  points (Fig. 7.2).  $\mathbf{p}_{0i}$  and  $\mathbf{p}_{0j}$  must not be involved in later distance checks.
- the distance of the points of the actual front polygon  $\Pi_0$  to points of all further front polygons  $\Pi_k, k > 0$ . If there are points  $\mathbf{p}_{0i} \in \Pi_0$  and  $\mathbf{p}_{mj} \in \Pi_m$  with  $\|\mathbf{p}_{0i} - \mathbf{p}_{mj}\| < \delta_t$  then the polygons  $\Pi_0 = (\mathbf{p}_{01}, \dots, \mathbf{p}_{0N_0})$  and  $\Pi_m = (\mathbf{p}_{m1}, \dots, \mathbf{p}_{mN_m})$  are united to the new actual front polygon

$$\Pi_0 = (\mathbf{p}_{01}, \dots, \mathbf{p}_{0i}, \mathbf{p}_{mj}, \dots, \mathbf{p}_{mN_m}, \mathbf{p}_{m1}, \dots, \mathbf{p}_{mj}, \mathbf{p}_{0i}, \dots, \mathbf{p}_{0N_0})$$

with  $N_0 + N_m + 2$  points (Fig. 7.2). The points  $\mathbf{p}_{0i}$  and  $\mathbf{p}_{mj}$  appear twice ! So before any further action is taken one should determine the front angles of these points when they first

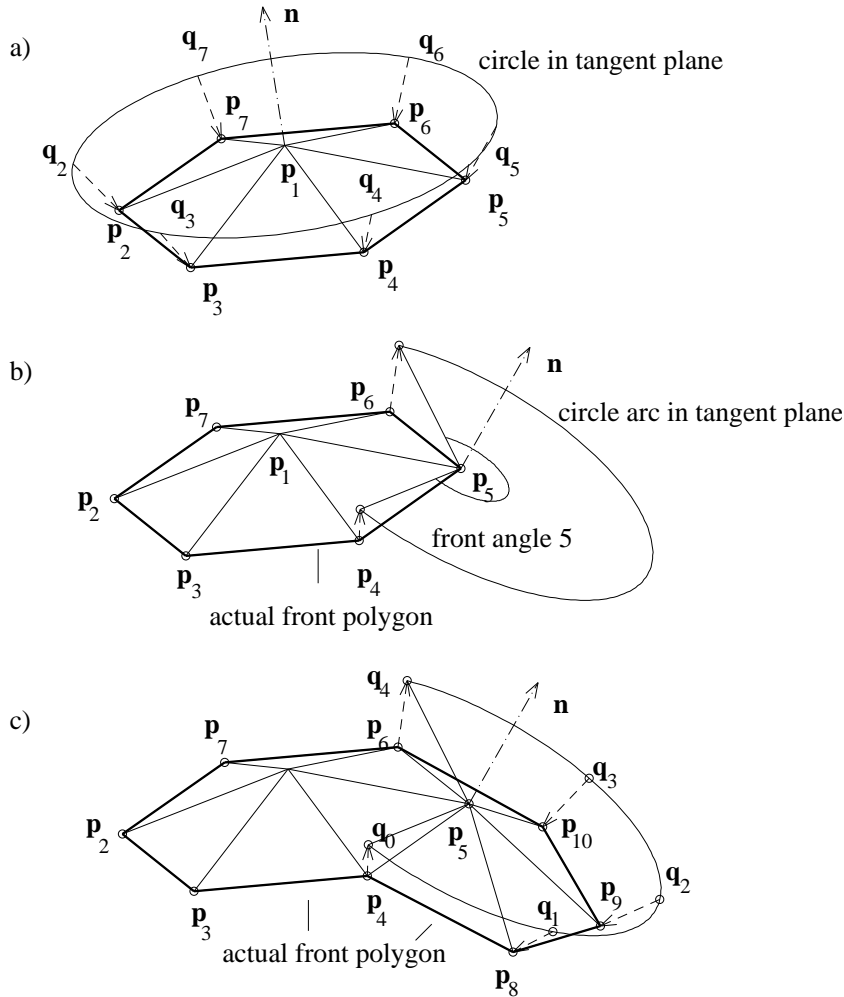


Figure 7.3: The first steps of the algorithm

appear in the polygon  $\Pi_0$  and complete (surround with triangles, see Step S3) first that point with the smallest angle then the second (Fig. 7.8e)). After this operation the first appearance of these two points is deleted from the actual front polygon. The points  $p_{0i}$  and  $p_{mj}$  must not be involved in later distance checks.

**Remarks:**

- a) For "simple" surfaces the distance check can be omitted.
- b) Before applying the distance check one should complete points with front angles smaller than (about)  $75^\circ$ .
- c) The rare case of "bad" near points  $p_{0i}, p_{mj}$  which are connected by an already triangulated area (Fig. 7.4) can be detected by calculating the angle  $\omega$  at point  $p_{0i}$  described in Step 1 using  $p_{mj}$  instead of  $v_2$ .  $p_{0i}, p_{mj}$  are "bad" near points if  $\omega$  is greater than the front angle at point  $p_{0i}$ .
- d) An essential acceleration of the distance check can be achieved by using bounding boxes of the front polygons.

**7.1.7 Step S3**

Let be  $p_{0m}$  a point of the actual front polygon  $\Pi_0$  with minimal front angle  $\omega$ . Complete the triangulation at point  $p_{0m}$  in the following way:

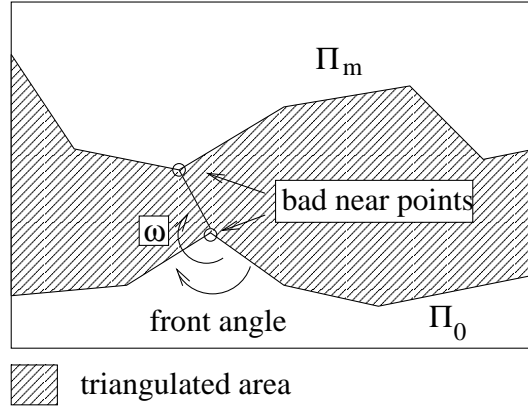


Figure 7.4: Bad near points and their detection

1. Determine the neighbors  $\mathbf{v}_1, \mathbf{v}_2$  of  $\mathbf{p}_{0m}$  (c.f. Step S1).

2. Determine the number  $n_t$  of triangles to be generated:

Let  $n_t := \text{trunc}(3\omega/\pi) + 1$ ,  $\Delta\omega := \omega/n_t$

Correction of  $\Delta\omega$  for extreme cases:

if  $\Delta\omega < 0.8$  and  $n_t > 1$  then  $n_t \rightarrow n_t - 1$  and  $\Delta\omega = \omega/n_t$  (Fig. 7.5a),

if  $n_t = 1$  and  $\Delta\omega > 0.8$  and  $\|\mathbf{v}_1 - \mathbf{v}_2\| > 1.25\delta_t$  then  $n_t = 2$  and  $\Delta\omega \rightarrow \Delta\omega/2$  (Fig. 7.5b),

if  $\omega < 3$  and ( $\|\mathbf{v}_1 - \mathbf{p}_{0m}\| \leq 0.5\delta_t$  or  $\|\mathbf{v}_2 - \mathbf{p}_{0m}\| \leq 0.5\delta_t$ ) then  $n_t = 1$  (Fig. 7.5c).

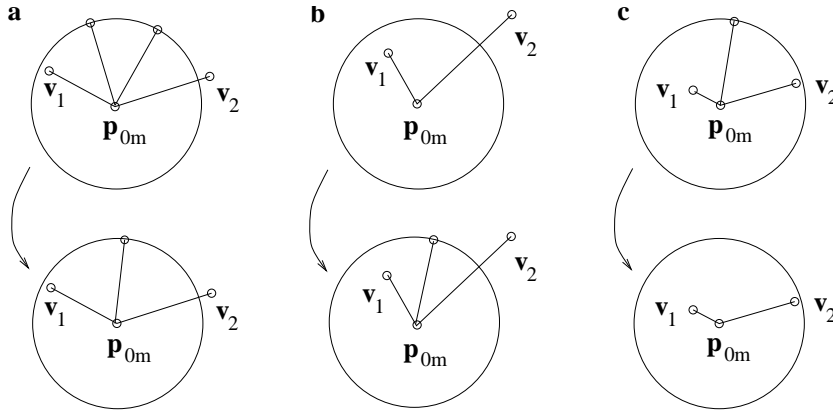


Figure 7.5: Corrections for extreme cases

3. Generate the triangles:

If  $n_t = 1$  then we get one new triangle:  $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{p}_{0m})$

else let be  $\mathbf{q}_0, \mathbf{q}_{n_t}$  the orthogonal projection of  $\mathbf{v}_1, \mathbf{v}_2$  into the tangentplane at point  $\mathbf{p}_{0m}$  and let  $\mathbf{q}_i$  be the result from a rotation of  $\mathbf{p}_{0m} + \delta_t(\mathbf{q}_0 - \mathbf{p}_{0m})/\|\mathbf{q}_0 - \mathbf{p}_{0m}\|$  by the angle  $i\Delta\omega$  around the normal at surface point  $\mathbf{p}_{0m}$  for  $i = 1, \dots, n_t - 1$ . (If a global bounding box is valid the chord  $\mathbf{p}_{0m}\mathbf{q}_i$  is truncated and the variable `border_point` of the corresponding new surface point set true. Boarder points will not be considered any more.) Applying procedure `surfacepoint` on  $\mathbf{q}_i$  we get the new points  $\mathbf{p}_{N+i}$ ,  $i = 1, \dots, n_t - 1$ , where  $N$  is the total number of already existing points, and  $n_t$  new triangles (see Fig. 7.3b,c):

$(\mathbf{v}_1, \mathbf{p}_{N+1}, \mathbf{p}_{0m}), (\mathbf{p}_{N+1}, \mathbf{p}_{N+2}, \mathbf{p}_{0m}), \dots, (\mathbf{p}_{N+n_t-1}, \mathbf{v}_2, \mathbf{p}_{0m})$ .

4. Renew the actual front polygon:  
Delete point  $\mathbf{p}_{0m}$  and if  $n_t > 1$  insert at its position the new points  $\mathbf{p}_{N+1}, \dots, \mathbf{p}_{N+n_t-1}$ .  
All boolean variables `angle_changed` at points  $\mathbf{v}_1, \mathbf{v}_2, \mathbf{p}_{N+1}, \dots, \mathbf{p}_{N+n_t-1}$  are set `true` to ensure recalculation of their front angles.

## 7.2 Examples

### 7.2.1 Sphere

Triangulation of the sphere  $x^2 + y^2 + z^2 - 4 = 0$  with starting point (1,1,1) and step length  $\delta_t = 0.3$ . Figure 7.6 shows the first four actual front polygons and the situation after the generation of 101 and 1531 triangles. The total triangulation of the sphere involves 1544 triangles.

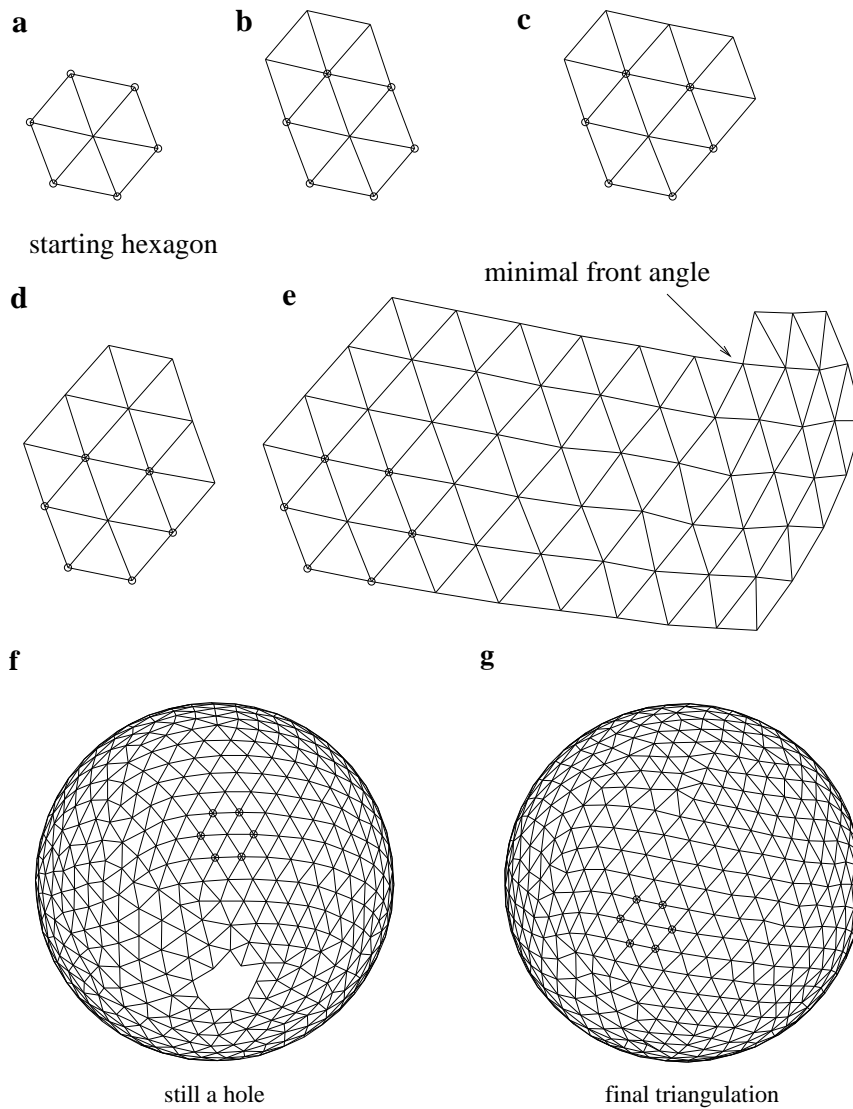


Figure 7.6: Triangulation of a sphere

### 7.2.2 Cylinder

Triangulation of the cylinder  $x^2 + y^2 - 1 = 0$ ,

1. with starting point  $(1, 0, 0)$  and  $\delta_t = 0.2$  before and after the first splitting of the actual front polygon. The cylinder is bounded by a bounding box (Fig. 7.7a,b,c).
2. with points on the top circle as the starting actual front polygon and points on the basic circle as a bounding front polygon (Fig. 7.7d).

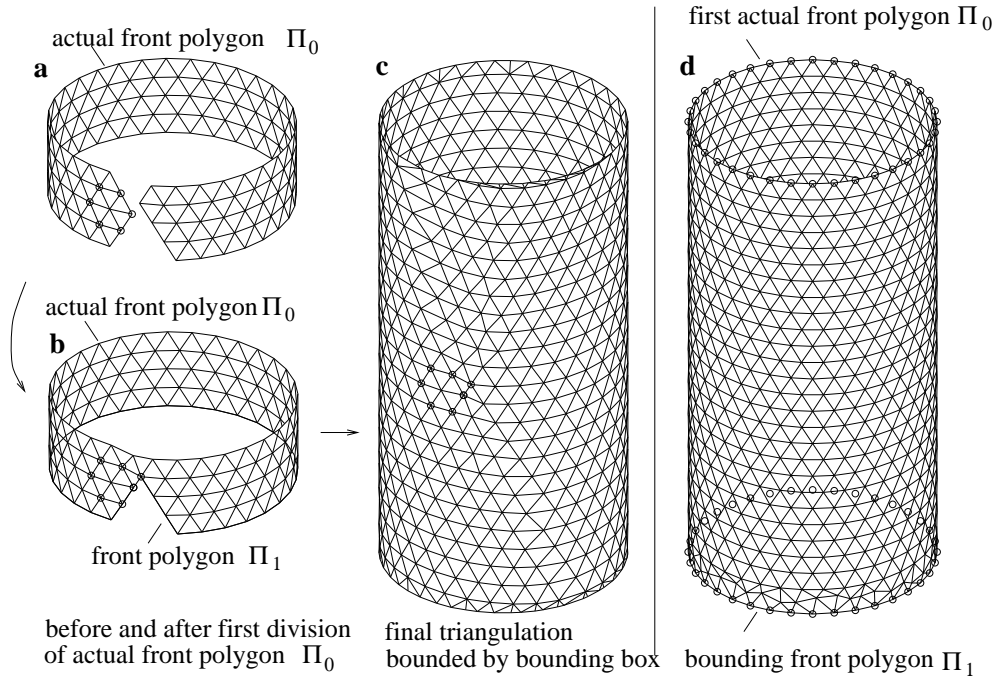


Figure 7.7: Triangulation of a cylinder

### 7.2.3 Torus

Triangulation of the torus

$$(x^2 + y^2 + z^2 + r^2 - a^2)^2 - 4r^2(x^2 + y^2) = 0, \quad r = 1, \quad a = 0.35$$

with  $\delta_t = 0.1$ .

Figure 7.8 shows the triangulation with starting point  $(1, 0, 0.5)$

before and after the first dividing of the front polygon  $\Pi_0$  (Fig. 7.8a,b) and

before and after uniting front polygon  $\Pi_0$  with front polygon  $\Pi_1$  that was generated by the first dividing (Fig. 7.8c,d) and the complete triangulation.

### 7.2.4 Set of two intersecting surfaces

Figure 7.9 shows a triangulation of the two intersecting surfaces  $x^4 + y^4 + z^4 = 16$  and  $x^2 + (z - 1.1)^2 = 1$ . A polygon on the intersection curve is used for the first actual front polygons of both the surfaces. The cylinder is truncated by a bounding box.



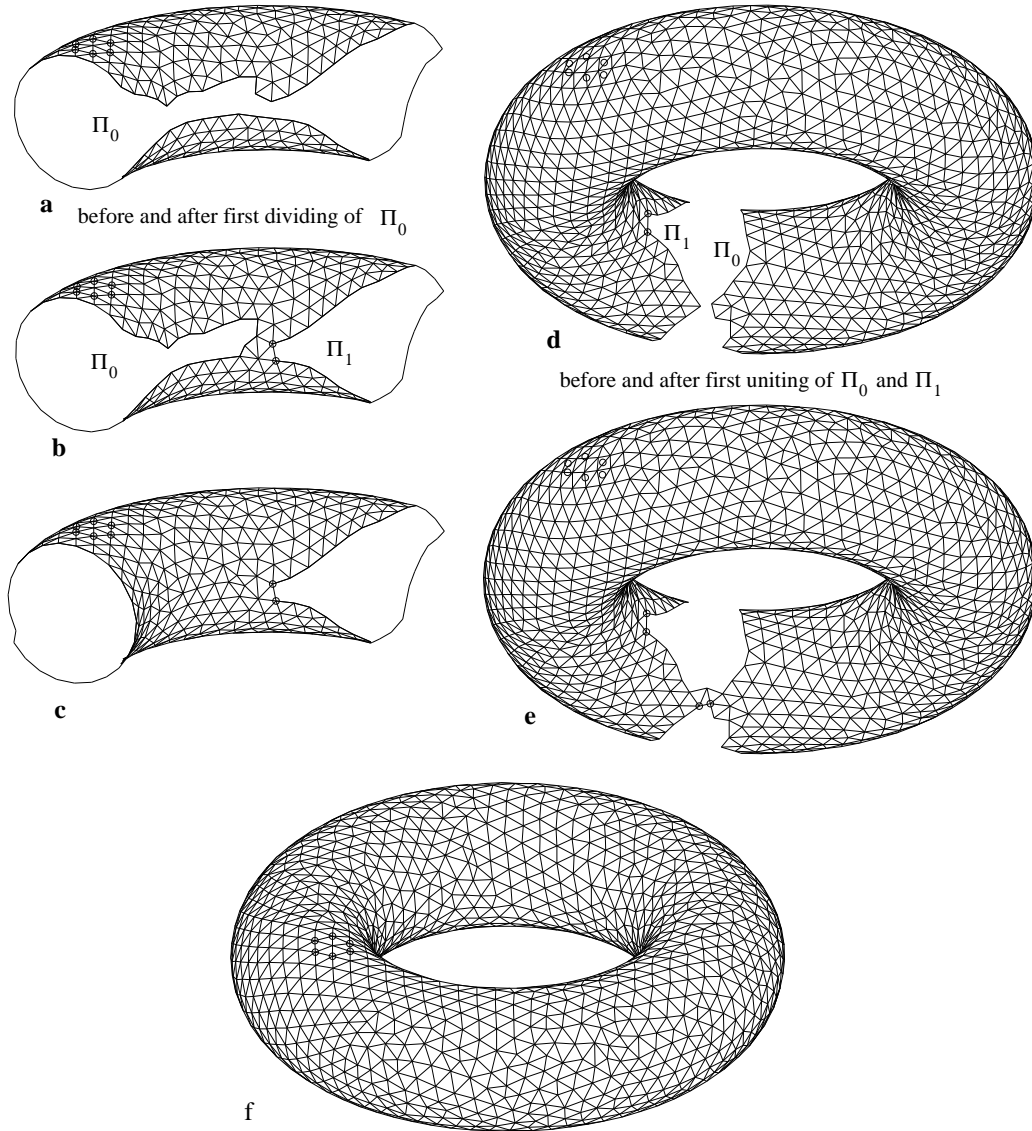


Figure 7.8: Triangulation of a torus

### 7.2.5 $G^2$ -continuous blending of three cylinders

The implicit surface

$$(1 - \mu)(x^2 + y^2 - 1)(x^2 + z^2 - 1)(y^2 + z^2 - 1) - \mu(9 - x^2 - y^2 - z^2)^3 = 0$$

is a  $G^2$ -continuous (i.e. curvature continuous) blending surface of the three cylinders  $x^2 + y^2 - 1 = 0$ ,  $x^2 + z^2 - 1 = 0$ ,  $y^2 + z^2 - 1 = 0$ . The parameter  $\mu$  chosen for Fig. 7.10 is 0.0003.

The triangulation starts with starting point  $(1, 1, 1)$  near the surface (see Step 0 of the algorithm). It is limited by 6 bounding polygons  $\Pi_1, \dots, \Pi_6$  on the curves of contact (circles) with the cylinders using step length  $\delta_t = \pi/20$  and consisting of 8062 triangles.

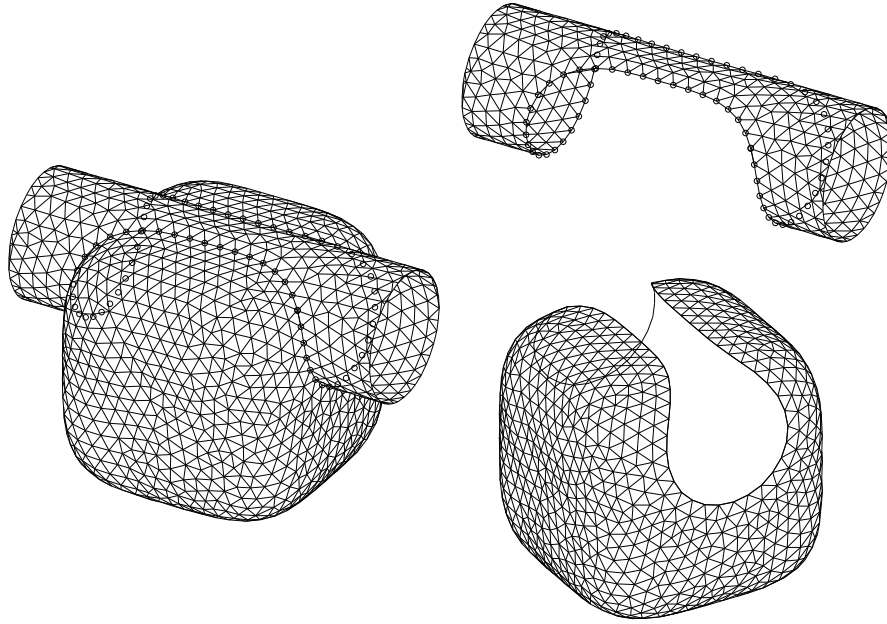


Figure 7.9: Triangulation of a set of two intersecting surfaces

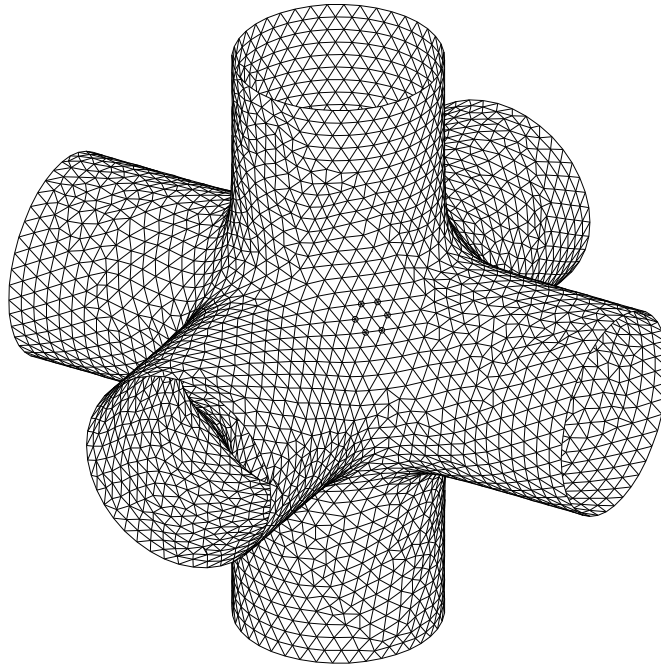


Figure 7.10: Triangulation of a blending surface of three intersecting cylinders

**Remark:**

The triangulation algorithm contained in `triang_proc.p` uses variable `cuttype` which allows the following options:

- a) `cuttype=0`: bounding surface is a **box**,
- b) `cuttype=1`: bounding surface is a **cylinder**,
- c) `cuttype=2`: bounding surface is a **sphere**.

### 7.3 Sample program: trisample.p

File `trisample.p` contains the PASCAL code of a program that

1. triangulates
  - (a) a cylinder using a starthexagon,
  - (b) a cylinder using a startpolygon on the circle at the top and a further polygon at the bottom circle as a bounding polygon,
  - (c) the surface  $x^4 + y^4 + z^4 = 16$  (quartic sphere) and a torus using starthexagons,
2. transfers the data of each triangulation into `datas` for the hiddenline algorithm of Chapter 6 and displays the set of surfaces.

The procedures for the triangulation are contained in (included) file `triang_proc.p`. For the convenience of the reader the necessary data structure and procedures for vectors are as comment included. They are (as in all sample programs) provided by unit `geograph`. As an exercise the reader should replace the given examples by own ones respecting the necessary data structure. Triangulations of surfaces may be combined with other polyhedrons originating from parametrically defined surfaces (see third cylinder in `trisample.p`, Figure 7.11).

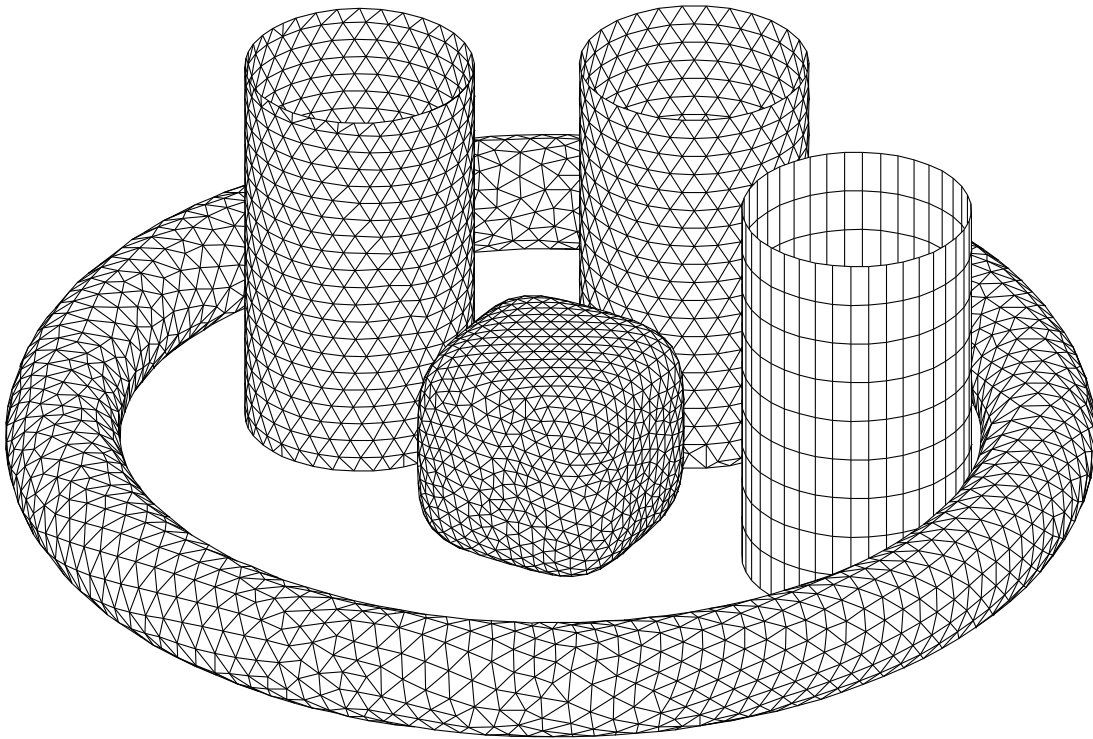


Figure 7.11: Triangulation and display of implicit surfaces

## 7.4 Ray tracing of triangulated surfaces with POV-Ray

It is an easy task to generate a ray tracing picture of a triangulated surface. Example `triblend.pov.p` contains a tool for writing the necessary informations of the triangles (point coordinates and normal vectors) into a pp-file (`triblend.pp`) which will be included while processing file `triblend.pov` by the POV-Ray-device

```
povray +L/usr/lib/povray31/include +Itriblend.pov +Otriblend.tga +W600 +H400 +FT
```

Output is file `triblend.tga` which may be displayed by program `xv` (available for LINUX).

## Chapter 8

# INTERSECTIONS: CURVE – SURFACE, SURFACE – SURFACE

### 8.1 Intersection Curve – Surface

#### 8.1.1 IS parametric curve – implicit surface

Given: regular curve  $\Gamma : \mathbf{x} = \mathbf{c}(t)$ ,  $t \in [a, b]$ , regular surface  $\Phi : f(\mathbf{x}) = 0$ ,  $\mathbf{x} \in D \subset \mathbb{R}^3$ .  
Wanted: Intersection points  $\Gamma \cap \Phi$ .

The **algorithm**:

- (1) Determining start parameters:  
Calculate a (nearly) equidistant polygon on  $\Gamma$  (see Section 4.2). For these points determine  $f(\mathbf{x})$  and locate neighbored pairs of points where the sign of  $f$  changes. The corresponding parameters will be used as starting parameters for the following Newton-iteration.
- (2) For any start parameter  $t_0$  (from (1)): Solve equation  $F(t) := f(\mathbf{c}(t)) = 0$  by a Newton-iteration. Any solution  $t^*$  gives an intersection point  $\mathbf{x}^* := \mathbf{c}(t^*) \in \Gamma \cap \Phi$ .
- (3) Apply (2) on all start parameters from (1).

#### 8.1.2 IS implicit curve – implicit surface

In this case the curve is defined as the intersection curve of two implicit surfaces.

Given: regular curve  $\Gamma : f_1(\mathbf{x}) = 0, f_2(\mathbf{x}) = 0$ ,  $\mathbf{x} \in D_{12} \subset \mathbb{R}^3$ ,  
regular surface  $\Phi : f(\mathbf{x}) = 0$ ,  $\mathbf{x} \in D \subset \mathbb{R}^3$ .  
Wanted: intersection points  $\Gamma \cap \Phi$ .

The **algorithm**:

- (1) Determining start points:  
Calculate a (nearly) equidistant polygon on  $\Gamma$  (see Section 8.2.1). For these points determine  $f(\mathbf{x})$  and locate neighbored pairs of points where the sign of  $f$  changes. One of each pair of points will be used as starting points for the following Newton-iteration.

- (2) For any starting point  $\mathbf{x}_0$  (from (1)): Solve the non linear system  $\mathbf{F}(\mathbf{x}) := (f_1(\mathbf{x}), f_2(\mathbf{x}), f(\mathbf{x})) = 0$ .  $\mathbf{F}'$  is the matrix  $\mathbf{F}' = (\nabla f_1, \nabla f_2, \nabla f)$  by a Newton–iteration. Any solution  $\mathbf{x}^*$  is an intersection point  $\in \Gamma \cap \Phi$ .
- (3) Apply (2) on all starting points from (1).

### 8.1.3 IS implicit curve – parametric surface

Given: regular curve  $\Gamma : f_1(\mathbf{x}) = 0, f_2(\mathbf{x}) = 0, \mathbf{x} \in D_{12} \subset \mathbb{R}^3$ ,  
 regular surface  $\Phi : \mathbf{x} = \mathbf{S}(u, v), (u, v) \in D \subset \mathbb{R}^2$ .

Wanted: intersection points  $\Gamma \cap \Phi$ .

The **algorithm**:

- (1) Intersection of surfaces  $f_1(\mathbf{x}) = 0$  and  $\mathbf{x} = \mathbf{S}(u, v)$ . The intersection curve is implicitly given in parameter plane:  $F_1(u, v) := f_1(\mathbf{S}(u, v)) = 0$ .
- (2) Intersection of surfaces  $f_2(\mathbf{x}) = 0$  and  $\mathbf{x} = \mathbf{S}(u, v)$ . The intersection curve is implicitly given in parameter plane:  $F_2(u, v) := f_2(\mathbf{S}(u, v)) = 0$ .
- (3) The coordinates of the intersection points of the implicit curves  $F_1(u, v) = 0$  and  $F_2(u, v) = 0$  (see 4.4.2) are the parameters of intersection points  $\in \Gamma \cap \Phi$ .

### 8.1.4 IS parametric curve – parametric surface

Given: regular curve  $\Gamma : \mathbf{x} = \mathbf{c}(t), t \in [a, b]$ , regular surface  $\Phi : \mathbf{x} = \mathbf{S}(u, v), (u, v) \in D \subset \mathbb{R}^2$ .

Wanted: intersection points  $\Gamma \cap \Phi$ .

The **Algorithm**:

- (1) Determining starting points:
- (1.1) calculate a (nearly) equidistant polygon on  $\Gamma$  (see 4.2).
  - (1.2) Triangulate the surface.
  - (1.3) Use the algorithm from 6.4.2 to determine pairs  $\sigma_i, \Delta_j$  consisting of a line segment  $\sigma_i$  of the polygon and a triangle  $\Delta_j$  of the surface polyhedron
- (2) For any pair  $\sigma_i, \Delta_j$  from (1) choose the parameter of one of the end points of  $\sigma_i$  as starting parameter  $t_0$ , the parameter of one of the points of the triangle  $\Delta_j$  as starting parameters  $u_0, v_0$  and solve the non linear system  $\mathbf{F}(t, u, v) := \mathbf{c}(t) - \mathbf{S}(u, v), \mathbf{F}(t, u, v) = 0$  by a Newton–iteration. (see 2.4). A solution  $(t^*, u^*, v^*)$  consists of the parameters of an intersection point between curve  $\Gamma$  and surface  $\Phi$ .  
 The necessary matrix is  $\mathbf{F}' = (\dot{\mathbf{c}}, \mathbf{S}_u, \mathbf{S}_v)$ .
- (3) Apply (2) on all pairs determined in (1).

## 8.2 Intersection surface – surface

### 8.2.1 IS of two implicit surfaces

The following *marching algorithm* is based on BA,HO,LY,HO'88.

Given: regular implicit surfaces  $\Phi_1 : f_1(\mathbf{x}) = 0$ ,  $\Phi_2 : f_2(\mathbf{x}) = 0$  which intersect transversally, i.e.  $\nabla f_1 \times \nabla f_2 \neq \mathbf{0}$  on the intersection curve.

Wanted: polygon on the intersection curve.

The **algorithm**:

The heart of the algorithm is procedure **curvepoint** which determines for a starting point  $Q_0 : \mathbf{q}_0$  in the "neighborhood" of the intersection curve a point  $P : \mathbf{p}$  on the intersection curve.

Procedure **curvepoint**:

- (CP1) Choose a starting point  $Q_0 : \mathbf{q}_0$  in the vicinity of the curve,
- (CP2) repeat  $\mathbf{q}_{k+1} = \mathbf{q}_k + \Delta_k$ , where  
 $\Delta_k = \alpha_k \nabla f_1(\mathbf{q}_k) + \beta_k \nabla f_2(\mathbf{q}_k)$  and  $\Delta_k \cdot \nabla f_j(\mathbf{q}_k) = -f_j(\mathbf{q}_k)$ ,  $j = 1, 2$ .  
(Newton-step for  $g_j(\alpha, \beta) := f_j(\mathbf{q}_k + \alpha \nabla f_1(\mathbf{q}_k) + \beta \nabla f_2(\mathbf{q}_k)) = 0$ ,  $j = 1, 2$ .)  
until  $\|\mathbf{q}_{k+1} - \mathbf{q}_k\|$  small enough.  
**curvepoint** =  $\mathbf{q}_{k+1}$ .

The **marching algorithm**:

- (C1) Choose a starting point  $Q_0 : \mathbf{q}_0$  and a step length  $s$  (approximate distance of neighboured points).
- (C2) First curve point  $P_1 : \mathbf{p}_1 = \text{curvepoint}(\mathbf{q}_0)$ .
- (C3) Curve point  $P_{k+1} : \mathbf{p}_{k+1} = \text{curvepoint}(\mathbf{p}_k + s \mathbf{t}_k)$ ,  
with unit tangent  $\mathbf{t}_k = (\nabla f_1(\mathbf{p}_k) \times \nabla f_2(\mathbf{p}_k)) / \|\dots\|$  at point  $P_k$ .
- (C4) Repeat C3) until the desired number of points is reached or  $P_{k+1} \approx P_0$  (closed curve).

**Problem 8.1** Determine the intersection curve of the following surfaces (see Figure 8.1)

$$\begin{aligned} f_1(x, y, z) &= x^4 + y^4 + z^4 - 1 = 0 \\ f_2(x, y, z) &= y^2 + (z - z_0)^2 - z_0^2 = 0 \quad (\text{cylinder}) \end{aligned}$$

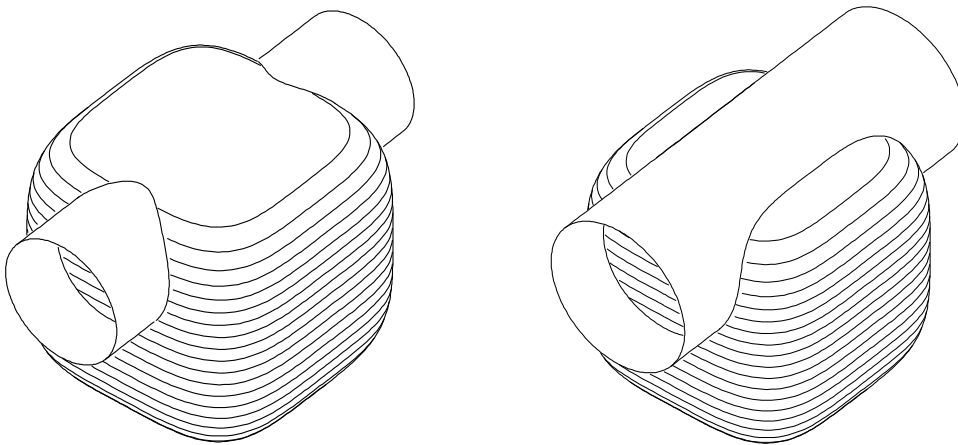


Figure 8.1: To problem 8.1,  $z_0 = 0.47$  (left) and  $z_0 = 0.6$  (right)

**Remark:**

The marching method is rather robust and can even be used for tracing curves with singular points (i.e. points with  $\nabla f_1 \times \nabla f_2 = 0$ , see Figure 8.2). But one should take care of the direction of the unit tangent when passing a singular point (cf. procedure `implicit_curvepts` in program `cassini.p`).

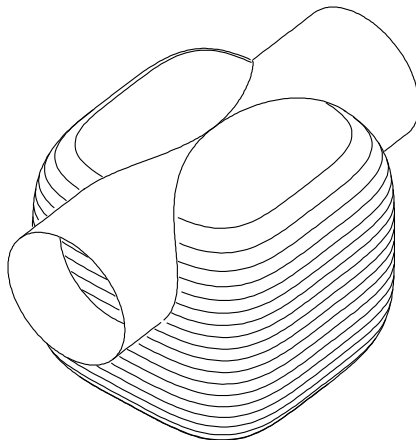


Figure 8.2: Intersection curve containing a singular point,  $z_0 = 0.5$

**Remark:**

Analogously to a parametric curve the determination of  $\mathbf{q}_0$  can be done curvature dependent (cf. Section 4.2). For the curvature vector of an intersection curve see Section 5.4.5.

### 8.2.2 IS of an implicit and a parametric surface

Let  $\Phi_1 : f(\mathbf{x}) = 0$  be a regular implicit surface and  $\Phi_2 : \mathbf{x} = \mathbf{S}(u, v)$  a regular parametric surface. The parameters of intersection points are solutions of the equation  $F(u, v) := f(\mathbf{S}(u, v)) = 0$ , hence points of an implicit curve in the parameter plane. This curve can be traced by the marching algorithm for implicit curves (see Section 4.3.1).

**Remark:**

In order to get a nearly equally spaced polygon in  $\mathbb{R}^3$  (!) with distance  $\delta$ , one should solve the quadratic system

$$\begin{aligned} \mathbf{S}_u^2 \Delta u^2 + 2(\mathbf{S}_u \cdot \mathbf{S}_v) \Delta u \Delta v + \mathbf{S}_v^2 \Delta v^2 &= \delta^2 \\ (\nabla f \cdot \mathbf{S}_u) \Delta u + (\nabla f \cdot \mathbf{S}_v) \Delta v &= 0 \end{aligned}$$

and use  $(u_i + \Delta u, v_i + \Delta v)$  as a new starting point for the `curvepoint` procedure (see 4.3.1).

(From the second equation we find: the solution has the form  $\Delta u = -\alpha(\nabla f \cdot \mathbf{S}_v)$ ,  $\Delta v = \alpha(\nabla f \cdot \mathbf{S}_u)$ . Inserting these expressions into the first equation yields two solutions for  $\alpha$ . The right one can be chosen with help of the marching direction.)

### 8.2.3 IS of two parametric surfaces

Intersecting *two parametric* surfaces is the most difficult case. One simple solution uses the normal-forms  $h_1 = 0$  and  $h_2 = 0$  (see Section 5.4) of the given surfaces and applies the marching method developed for tracing the intersection curve of two implicit surfaces. For the evaluation of  $h_i$  and  $\nabla h_i$  we use the foot point algorithm for parametric surfaces (Section 5.8.1). For the necessary



starting point: see the marching method below.

Given: regular surfaces  $\Phi_1 : \mathbf{x} = \mathbf{S}_1(s, t)$ ,  $\Phi_2 : \mathbf{x} = \mathbf{S}_2(u, v)$  which intersect transversally (i.e. the normals at an intersection point are linearly independent).

Wanted: Polygon on the intersection curve.

The **marching algorithm**: (BA,KE'90)

- (1) Determining of a starting point  $P_0$  and its parameters  $s_0, t_0, u_0, v_0$ :  
Cover the surfaces by rectangular nets, determine bounding boxes for any rectangle and discover intersecting bounding boxes of the two surfaces. By “divide and conquer” we get possible pairs of starting points and their parameters. For any pair  $A_0, B_0$  continue with step (2.3).
- (2)  $P_{i+1}, s_{i+1}, t_{i+1}, u_{i+1}, v_{i+1}$  from  $P_i, s_i, t_i, u_i, v_i$ :
  - (2.1) 0-th approximation  $Q_0$  for  $P_{i+1}$ :  
Determine the the unit normal vectors  $\mathbf{n}_1(s_n, t_n)$ ,  $\mathbf{n}_2(u_n, v_n)$  and the unit tangent vector of the intersection curve  $\mathbf{t} := \mathbf{n}_1 \times \mathbf{n}_2 / \|\dots\|$ . Set  $Q_0 = P_i + \delta \mathbf{t}$  with *step length*  $\delta$ .
  - (2.2) Determine the foot points  $A_0 : \mathbf{a}_0$ ,  $B_0 : \mathbf{b}_0$  for  $Q_0$  on the surfaces. (As starting point for the foot point algorithm one chooses  $P_i$  and its parameters.)

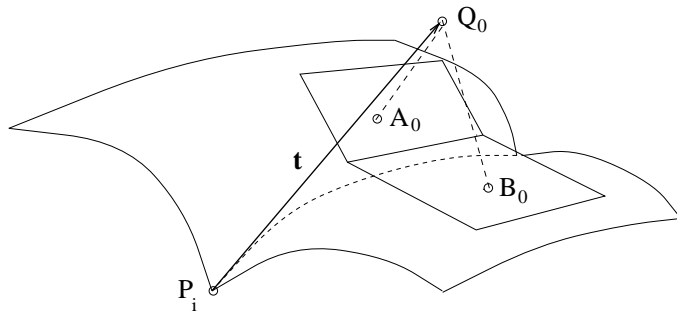


Figure 8.3: The marching algorithm for parametric surfaces

- (2.3)  $Q_{k+1}, A_{k+1}, B_{k+1}$  from  $A_k, B_k$ :  
Let be  $\varepsilon_1, \varepsilon_2$  the tangent planes at points  $A_k, B_k$  and  $\varepsilon_3$  the plane passing the midpoint of the line segment  $A_k B_k$  with normal vector  $\mathbf{n}_3 := \mathbf{n}_1 \times \mathbf{n}_2$  and  $Q_{k+1}$  the common point of the 3 planes  $\varepsilon_1, \varepsilon_2, \varepsilon_3$ :

$$Q_{k+1} : \mathbf{q}_{k+1} = \frac{(\mathbf{n}_1 \cdot \mathbf{a}_k)(\mathbf{n}_2 \times \mathbf{n}_3) + (\mathbf{n}_2 \cdot \mathbf{b}_k)(\mathbf{n}_3 \times \mathbf{n}_1) + (\mathbf{n}_3 \cdot \frac{\mathbf{a}_k + \mathbf{b}_k}{2})(\mathbf{n}_1 \times \mathbf{n}_2)}{\mathbf{n}_1 \cdot (\mathbf{n}_2 \times \mathbf{n}_3)}$$

$A_{k+1}, B_{k+1}$  are the foot points of  $Q_{k+1}$ .

- (2.4) If the distance  $|A_{k+1} B_{k+1}|$  is small enough, we set  $P_{i+1} = Q_{k+1}$ . The parameters of  $P_{i+1}$  as point on both the surfaces we take the parameters of its foot points  $A_{k+1}$  (1. surface) and  $B_{k+1}$  (2. surface).

**Example 8.1** Figure 8.4 shows the intersection curve of the following two parametric surfaces:

$$\mathbf{S}_1(s, t) = (s, t, s^2 + t^2), \quad \mathbf{S}_2(u, v) = (u, v, 2 - u^4 - v^4).$$

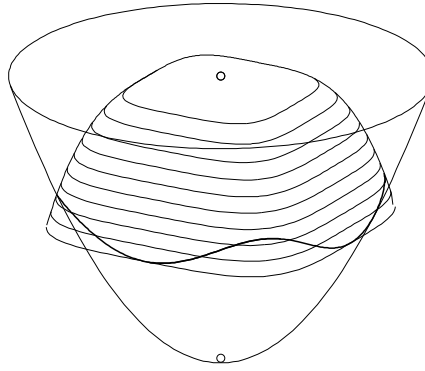


Figure 8.4: Intersection of two parametric surfaces

### 8.3 Contour lines of surfaces

**Definition 8.1** Let  $\Phi$  be a regular implicit or parametric surface,  $\mathbf{p}$  a point  $\in \Phi$  and  $\mathbf{n}(\mathbf{p})$  the surface normal of  $\mathbf{p}$ .

- In case of **parallel projection** the negative projection vector is  $\mathbf{n}_0$ , the normal of the image plane (see Section 3.1).  
Point  $\mathbf{p}$  is a contour point if  $\mathbf{n}(\mathbf{p}) \cdot \mathbf{n}_0 = 0$ .
- In case of **central projection** the negative projection vector of point  $\mathbf{p}$  is  $\mathbf{z} - \mathbf{p}$ , with eyepoint vector  $\mathbf{z}$  (see Section 3.2).  
Point  $\mathbf{p}$  is a contour point if  $\mathbf{n}(\mathbf{p}) \cdot (\mathbf{z} - \mathbf{p}) = 0$ .

The collection of contour points is called contour line.

#### 8.3.1 Contour line of an implicit surface

Let  $\Phi : f(\mathbf{x}) = 0$  be a regular implicit  $C^2$  surface. The contour line of surface  $\Phi$  is the intersection curve of  $\Phi$  with the implicit surface  $g(\mathbf{x}) = 0$ :

- in case of **parallel projection**:  $g(\mathbf{x}) := \nabla f(\mathbf{x}) \cdot \mathbf{n}_0 = 0$ ,  
where  $\mathbf{n}_0$  is the negative projection vector. The necessary derivative of  $g$  is  $\nabla g(\mathbf{x}) = H_f(\mathbf{x})\mathbf{n}_0$ .
- in case of **central projection**:  $g(\mathbf{x}) := \nabla f(\mathbf{x}) \cdot (\mathbf{z} - \mathbf{x}) = 0$ ,  
where  $\mathbf{z}$  is the eye point vector (centre of projection). The necessary derivative of  $g$  is  $\nabla g(\mathbf{x}) = H_f(\mathbf{x})(\mathbf{z} - \mathbf{x}) - \nabla f(\mathbf{x})$ .

and can be traced by the marching algorithm for the intersection curve of two implicit surfaces (Section 8.2.1).

**Example 8.2**  $f(x, y, z) = (1 - x)(1 + x)(1 - y)(1 + y)(1 - z)(1 + z) - c = 0$ ,  $c > 0$ .  
Surface  $f(x, y, z) = 0$  is for fixed  $c$  a smooth approximation of the cube with vertices  $(\pm 1, \pm 1, \pm 1)$ .  
(See Figure 8.5a.)

**Problem 8.2** Establish a smooth approximation of an oktahedron and display it (See Figure 8.5b).)

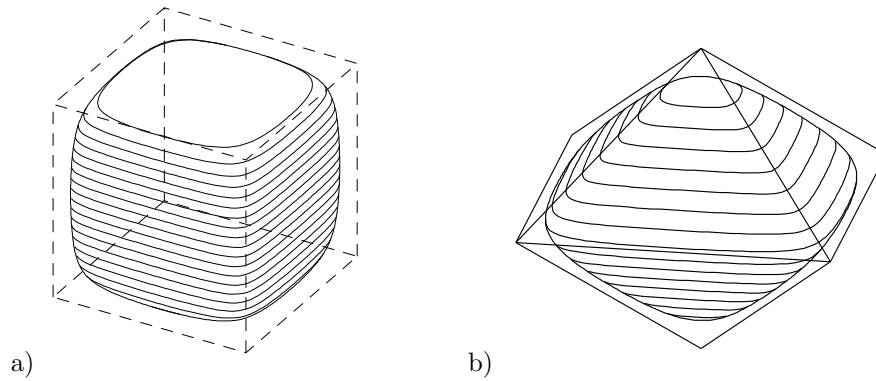


Figure 8.5: Contour line of an approximation of a) a cube b) an octahedron

### 8.3.2 Contour line of a parametric surface

Let  $\Phi : \mathbf{x} = \mathbf{S}(s, t)$  be a regular parametric  $C^2$  surface. The contour line of surface  $\Phi$  consists of points with parameters on the implicit curve  $g(s, t) = 0$  in parameter plane:

- in case of **parallel projection**:  $g(s, t) := (\mathbf{S}_s(s, t) \times \mathbf{S}_t(s, t)) \cdot \mathbf{n}_0 = 0$ , where  $\mathbf{n}_0$  is the negative projection vector.
- in case of **central projection**:  $g(s, t) := (\mathbf{S}_s(s, t) \times \mathbf{S}_t(s, t)) \cdot (\mathbf{z} - \mathbf{S}(s, t)) = 0$ , where  $\mathbf{z}$  is the eye point vector (centre of projection).

and can be traced by the marching algorithm for planar implicit curves (Section 4.3.1).

**Example 8.3** For  $\mathbf{S}(s, t) = (s, t, 1 - s^4 - t^4)$  we get  $\mathbf{S}_s(s, t) = (1, 0, -4s^3)$ ,  $\mathbf{S}_t(s, t) = (0, 1, -4t^3)$  and the normal is  $\mathbf{S}_s(s, t) \times \mathbf{S}_t(s, t) = (4s^3, 4t^3, 1)$ . Hence  $g(s, t) = 4s^3n_{0x} + 4t^3n_{0y} + n_{0z}$  where  $\mathbf{n}_0 = (n_{0x}, n_{0y}, n_{0z})$ . (See 8.6),

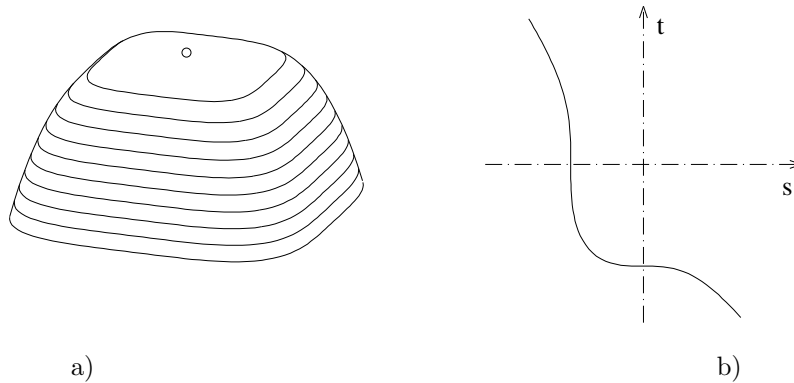


Figure 8.6: Contour line a) on the surface b) in the parameter plane

### 8.3.3 Contour line of a quadric surface

**Definition 8.2** A non planar surface with an implicit representation  $\Phi : f(x, y, z) = a_{11}x^2 + \dots + a_{12}xy + \dots = 0$ , where  $f$  is a polynomial of degree 2, is called quadric.

**Lemma 8.1** *The contour line of a quadric is in any case a **planar** curve (a plane section with the surface). Hence the contour line is a **conic**.*

**Proof:** a) In case of parallel projection the statement is obvious.

b) In case of central projection: Let be  $\Phi : f(x, y, z) = a_{11}x^2 + \dots + a_{12}xy + \dots + l(x, y, z) = 0$ , where  $l(x, y, z)$  is the linear part of  $f$ . From  $\nabla f(x, y, z) \cdot \mathbf{x} = (2a_{11}x + a_{12}y + a_{13}z, \dots, \dots)^\top \cdot (x, y, z)^\top = 2(a_{11}x^2 + a_{12}xy + \dots) = f(x, y, z) - l(x, y, z)$  and  $f(\mathbf{x}) = 0$  we get  $\nabla f(\mathbf{x}) \cdot (\mathbf{z} - \mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{z} - l(\mathbf{x}) = 0$ , which is a linear equation.  $\square$

**Example 8.4** 1. For the sphere  $x^2 + y^2 + z^2 - 1 = 0$  the contour curve is a circle

2. For an ellipsoid  $\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0$  the contour curve is an ellipse.

3. For the hyperboloid of one sheet  $x^2 + y^2 - z^2 - 1 = 0$  the contour curve can be nearly any conic.

(Figure 8.7)

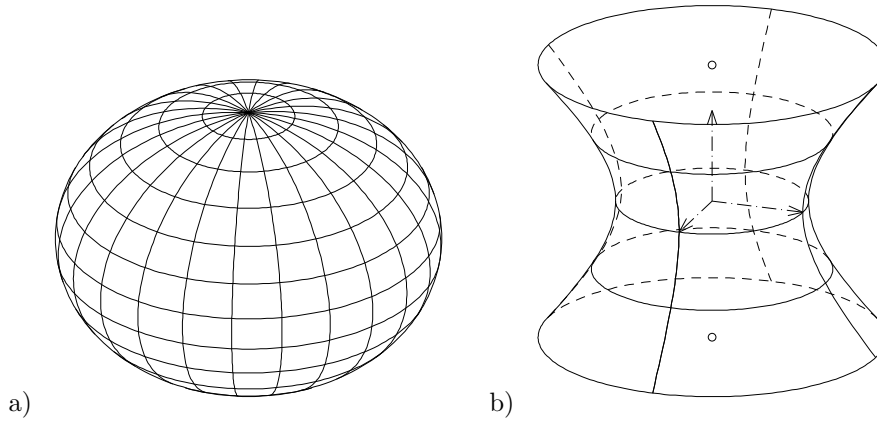


Figure 8.7: a) ellipsoid b) hyperboloid of one sheet

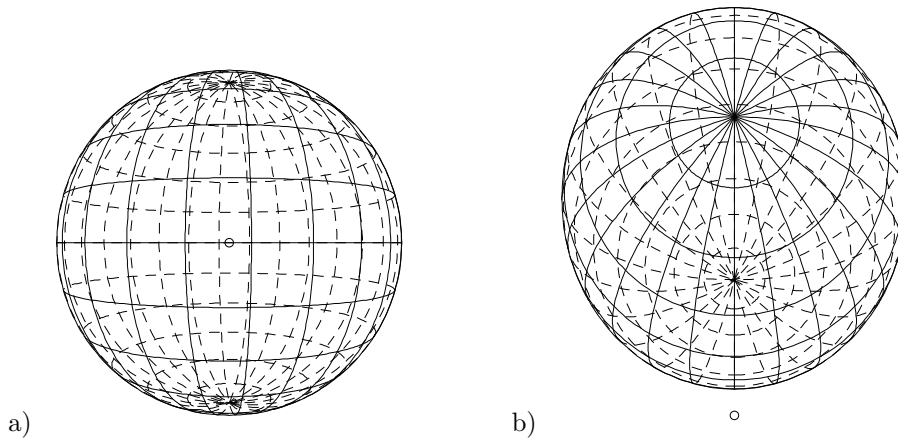


Figure 8.8: central projection of a sphere, contour is a) a circle, b) an ellipse

**Remark:**

In case of central projection the conic may be empty. For example, if the centre  $\mathbf{z}$  is in the inner part of an ellipsoid.

## 8.4 Hiddenline-algorithm for simple cases

Given: Regular oriented surface  $\Phi$  and a polygon  $\Pi : \mathbf{p}_1, \mathbf{p}_2, \dots$  on a curve  $\Gamma \subset \Phi$ .

Wanted: Decision if point  $\mathbf{p}_i$  is **invisible** for  $i = 1, 2, \dots$

Afterwards: The polygon  $\Pi$  is projected and neighbored, still visible points connected by a line segment.

### 8.4.1 Fast normal-test

Let  $\Phi$  be a regular oriented surface,  $\mathbf{p}$  a point  $\in \Phi$  and  $\mathbf{n}(\mathbf{p})$  a normal pointing “outside” at point  $\mathbf{p}$ . Then point  $\mathbf{p}$  is **invisible** if:

- in case of **parallel projection**:  $\mathbf{n}(\mathbf{p}) \cdot \mathbf{n}_0 < 0$ ,  
where  $\mathbf{n}_0$  is the negative projection vector.
- in case of **central projection**:  $\mathbf{n}(\mathbf{p}) \cdot (\mathbf{z} - \mathbf{p}) < 0$ ,  
where  $\mathbf{z}$  is the eye point vector (centre of projection).

The fast normal-test is sufficient for convex surfaces or special situations of non convex surfaces. For example: 1) ellipsoid 2) hyperboloid (non convex, but view not too steep) above (see Figure 8.7). The normal test is not sufficient for example in case of a torus and low projection angle  $v$  (see Figure 8.9).

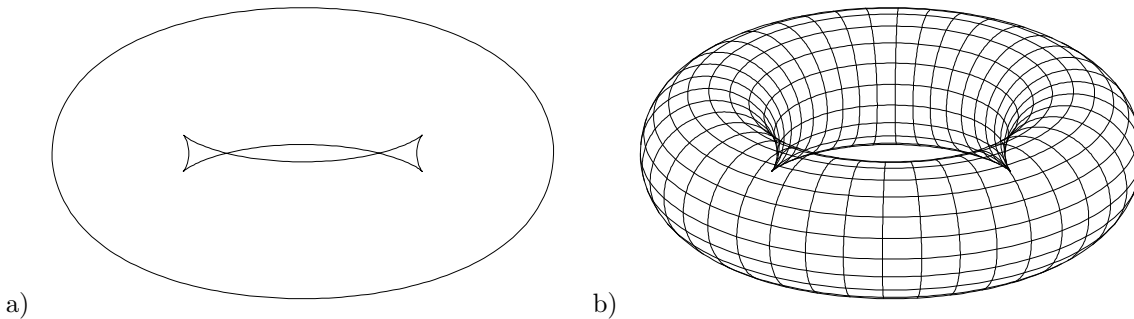


Figure 8.9: Normal test for torus: a) contour lines only b) parametric curves

### 8.4.2 Ray-test for implicit surfaces

An essential improvement of the simple normal-test is the *ray-test*:

Given: Regular implicit surface  $\Phi : f(\mathbf{x}) = 0$  and a polygon  $\Pi : \mathbf{p}_1, \mathbf{p}_2, \dots$  on a curve  $\Gamma \subset \Phi$ .

Wanted: Decision if point  $\mathbf{p}_i$  is **invisible** for  $i = 1, 2, \dots$

**Idea:** We consider the negative projection ray for point  $\mathbf{p}_i$  and look for an intersection of the ray with the surface. In case of an intersection: point  $\mathbf{p}_i$  is invisible.

#### Ray-test algorithm:

1. Choose a simple surface  $\Psi_0$  (sphere, plane, cube or combinations) as **bounding surface** and a minimal step length  $\delta$ .
2. Consider the ray (line)  $g : \mathbf{x} = \mathbf{c}(t) = \mathbf{p}_i + t\mathbf{r}$ ,  $t > 0$ , where in case of **parallel projection**:  $\mathbf{r} = \mathbf{n}_0$  (normal of the image plane),  
**central projection**:  $\mathbf{r} = \mathbf{z} - \mathbf{p}_i$  ( $\mathbf{z}$ : eye point).

3. Intersect  $g$  with the bounding surface  $\Psi_0$ . The corresponding parameter is  $t_0$  and the sign of  $f(\mathbf{c}(t_0))$  is  $\sigma_0$ .
4. **Test** if the sign  $\sigma$  of  $f(\mathbf{c}(t))$  is different from  $\sigma_0$  for
  1. run:  $t = t_0/2$ ,
  2. run:  $t_k := kt_0/4, k = 1, 2, 3$
  - ...
  - n. run:  $t_k = kt_0/2^n, k = 1, 2, \dots, 2^n - 1$ .**until**  $\sigma \neq \sigma_0$  or  $\|\mathbf{r}\|t_0/2^n < \delta$ .  
 (Already tested points are omitted in each run.)
5. In case of  $\sigma \neq \sigma_0$ : point  $\mathbf{p}_i$  is invisible, else:  $\mathbf{p}_i$  is visible.

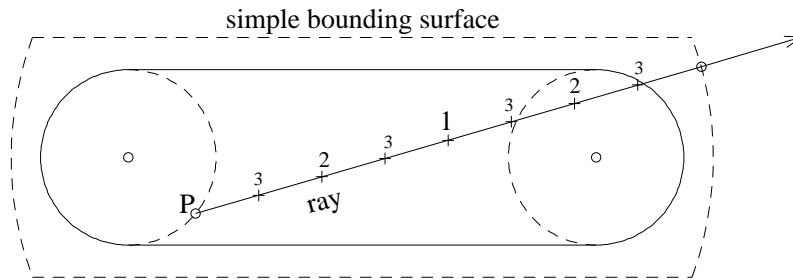


Figure 8.10: Ray-test for a torus

Figure 8.11 shows an application of the ray-test to a torus. The bounding surface is a truncated sphere.

The ray test can be applied to several surfaces, too. (See Figure 8.12.)

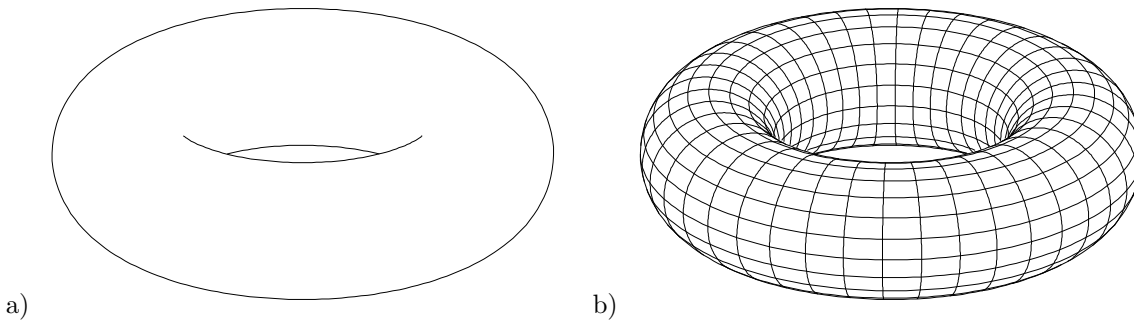


Figure 8.11: Ray-test for a torus: a) contour lines only, b) parametric curves

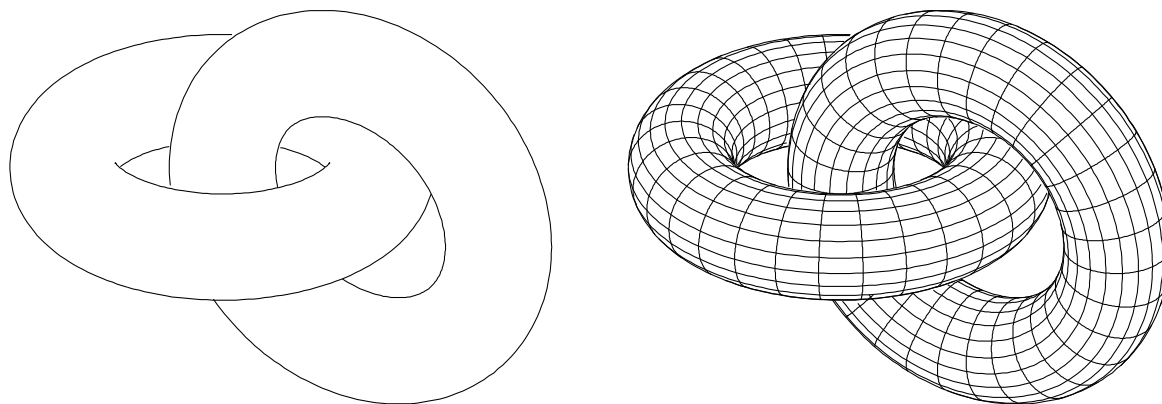


Figure 8.12: two tori, displayed with ray-test

**Remark:**

The ray-test for **quadrics** can be accelerated by determining the intersection points (quadratic equation) exactly instead of the binary search for a change of the sign (see Figure 8.13).

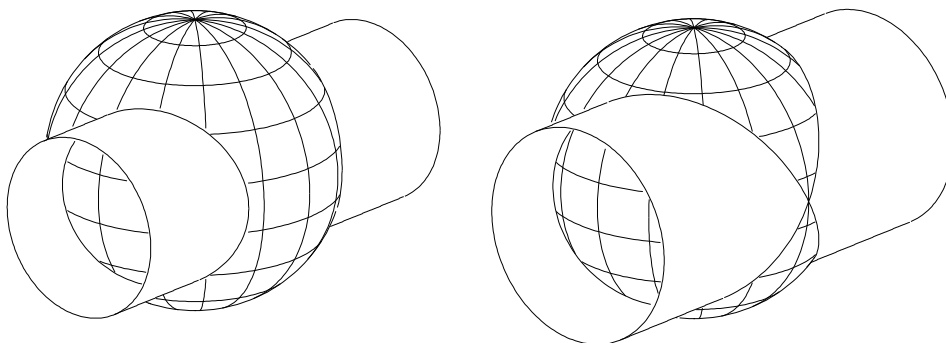


Figure 8.13: quadrics, displayed with ray-test





# Chapter 9

## CONICS AND QUADRICS

**Definition 9.1** Let  $A$  be a regular  $n \times n$  matrix and  $\mathbf{x}_0 \in \mathbb{R}^n$ ,  $n = 1, 2, 3, \dots$  then the function  $\mathbf{x} \rightarrow \mathbf{x}_0 + A\mathbf{x}$  is called affine map.

**Lemma 9.1** Any affine map preserves the parallel relation between lines.

**Proof:** The affine image of a line  $g : \mathbf{x} = \mathbf{p} + t\mathbf{r}$  is determined by the image  $P', Q'$  of the two points  $P : \mathbf{p}, Q : \mathbf{p} + \mathbf{r}$ . From  $P' : \mathbf{p}' = \mathbf{x}_0 + A\mathbf{p}, Q' : \mathbf{x}_0 + A(\mathbf{p} + \mathbf{r}) = \mathbf{p}' + A\mathbf{r}$  we get  $g' : \mathbf{x} = \mathbf{x}_0 + A\mathbf{p} + tA\mathbf{r}$ . Two lines  $g_i : \mathbf{x} = \mathbf{p}_i + t\mathbf{r}_i$  are parallel if and only if there exists a  $\lambda \in \mathbb{R}$  such that  $\mathbf{r}_1 = \lambda\mathbf{r}_2$ . Without loss of generality we assume  $\mathbf{r}_1 = \mathbf{r}_2$ . Hence, the image of two parallel lines are parallel, too.  $\square$

The x,y,z coordinates used in this chapter are always cartesian (refer to a orthonormal base).

### 9.1 Conics

#### 9.1.1 Ellipse

**Definition 9.2**  $C_1 := \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 = 1\} = \{(\cos t, \sin t) | 0 \leq t \leq 2\pi\}$  is called unit circle.

**Definition 9.3** Any affine image of the unit circle is called ellipse. Hence for any  $\mathbf{p}_0 \in \mathbb{R}^2$  and linear independent vectors  $\mathbf{f}_1, \mathbf{f}_2 \in \mathbb{R}^2$  the point set

$$E = \{\mathbf{p}_0 + \mathbf{f}_1 \cos t + \mathbf{f}_2 \sin t \mid 0 \leq t \leq 2\pi\}.$$

is an ellipse and any ellipse can be represented in such a way.

Let be  $E$  an ellipse  $\mathbf{x} = \mathbf{f}_1 \cos t + \mathbf{f}_2 \sin t$ . From  $\mathbf{f}_1 \cos(t + \pi) + \mathbf{f}_2 \sin(t + \pi) = -\mathbf{f}_1 \cos t - \mathbf{f}_2 \sin t$  we get:

**Lemma 9.2** An ellipse  $E : \mathbf{x} = \mathbf{p}_0 + \mathbf{f}_1 \cos t + \mathbf{f}_2 \sin t$  is symmetric to point  $\mathbf{p}_0$ .

$P_0 : \mathbf{p}_0$  is called midpoint of ellipse  $E$  and the line segment  $\mathbf{x}(t)\mathbf{x}(t + \pi)$  a diameter of  $E$ .

**Definition 9.4** Two diameters  $X_1X'_1$  and  $X_2X'_2$  of an ellipse are called conjugated diameters, if the tangents at  $X_1, X'_1$  are parallel to  $X_2X'_2$  and vice versa.

**Lemma 9.3** Conjugated diameters are the images of two orthogonal diameters of  $K_1$  with respect of the accompanying affine map. Hence  $\mathbf{f}_1$  and  $\mathbf{f}_2$  are the directions of two conjugated diameters.

**Proof:** Let be  $E$  an ellipse  $\mathbf{x} = \mathbf{f}_1 \cos t + \mathbf{f}_2 \sin t$ . From  $\dot{\mathbf{x}}(t) = -\mathbf{f}_1 \sin t + \mathbf{f}_2 \cos t = \mathbf{f}_1 \cos(t + \frac{\pi}{2}) + \mathbf{f}_2 \sin(t + \frac{\pi}{2}) = \mathbf{x}(t + \frac{\pi}{2}) = -\mathbf{x}(t - \frac{\pi}{2})$  we get the statement.  $\square$

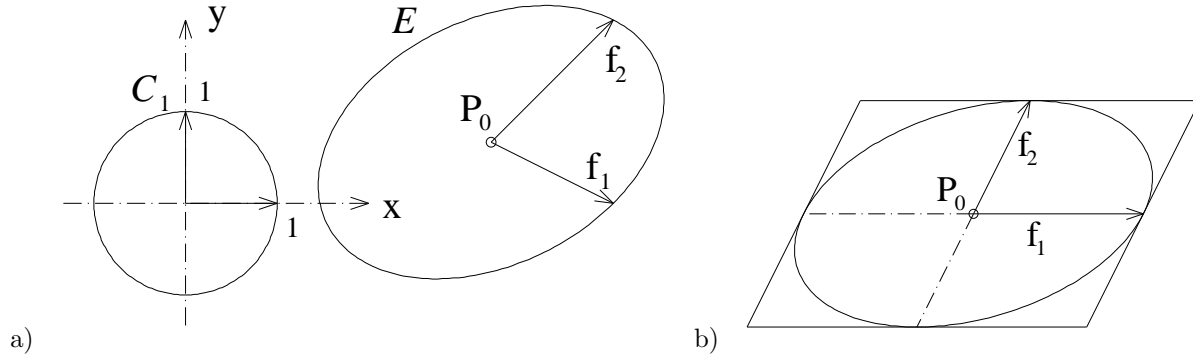


Figure 9.1: a) Ellipse as affine map of the unit circle b) Conjugated diameters of an ellipse

**Lemma 9.4** *Let \$E\$ be an ellipse that is no circle. There exists exactly one orthogonal pair of conjugated diameters, the main axis of \$E\$. The corresponding ellipse points are called the 4 vertices of \$E\$.*

**Proof:** Let be \$E\$ the ellipse \$\mathbf{x} = \mathbf{f}\_1 \cos t + \mathbf{f}\_2 \sin t\$. \$E\$ is a circle if and only if \$\mathbf{f}\_1^2 = \mathbf{f}\_2^2\$ and \$\mathbf{f}\_1 \cdot \mathbf{f}\_2 = 0\$. Let be \$E\$ no circle: \$\mathbf{x}(t)\$ and \$\dot{\mathbf{x}}(t)\$ are the directions of conjugated diameters. The equation \$\mathbf{x}(t) \cdot \dot{\mathbf{x}}(t) = \mathbf{f}\_1 \cdot \mathbf{f}\_2 (\cos^2 t - \sin^2 t) + (\mathbf{f}\_2^2 - \mathbf{f}\_1^2) \sin t \cos t = \mathbf{f}\_1 \cdot \mathbf{f}\_2 \cos(2t) + \frac{(\mathbf{f}\_2^2 - \mathbf{f}\_1^2)}{2} \sin(2t) = 0\$. has exactly 4 solutions in \$[0, 2\pi)\$:

In case of \$\mathbf{f}\_1^2 = \mathbf{f}\_2^2: \pm \frac{\pi}{4}, \pi \pm \frac{\pi}{4}\$. Other cases: \$\alpha := \frac{1}{2} \arctan(\frac{2\mathbf{f}\_1 \cdot \mathbf{f}\_2}{\mathbf{f}\_1^2 - \mathbf{f}\_2^2})\$, \$\alpha \pm \frac{\pi}{2}, \alpha + \pi\$. \$\square\$

### 9.1.2 Hyperbola

**Definition 9.5** \$H\_1 := \{(x, y) \in \mathbb{R}^2 | x^2 - y^2 = 1\} = \{(\pm \cosh t, \sinh t) | t \in \mathbb{R}\}\$ is called unit hyperbola. The lines \$y = \pm x\$ are called asymptotes of \$H\_1\$.

**Definition 9.6** Any affine image of the unit hyperbola is called hyperbola. Hence for any \$\mathbf{p}\_0 \in \mathbb{R}^2\$ and linear independent vectors \$\mathbf{f}\_1, \mathbf{f}\_2 \in \mathbb{R}^2\$ the point set

$$H = \{\mathbf{p}_0 + \pm \mathbf{f}_1 \cosh t + \mathbf{f}_2 \sinh t \mid t \in \mathbb{R}\}.$$

is a hyperbola and any hyperbola can be represented in such a way.

**Lemma 9.5** A hyperbola \$H : \mathbf{x} = \mathbf{p}\_0 + \pm \mathbf{f}\_1 \cosh t + \mathbf{f}\_2 \sinh t\$ is symmetric to point \$\mathbf{p}\_0\$.

\$P\_0 : \mathbf{p}\_0\$ is called midpoint of hyperbola \$H\$ and the line segment \$\mathbf{x}(t)\mathbf{x}(t + \pi)\$ a diameter of \$H\$.

**Proof:** The points \$\mathbf{p}\_0 + \mathbf{f}\_1 \cosh t + \mathbf{f}\_2 \sinh t\$, \$\mathbf{p}\_0 - \mathbf{f}\_1 \cosh(-t) + \mathbf{f}\_2 \sinh(-t)\$ are points of \$H\$ and \$\mathbf{p}\_0\$ their midpoint. \$\square\$

**Definition 9.7** Let be \$H : \mathbf{x} = \mathbf{x}(t) := \mathbf{p}\_0 + \pm \mathbf{f}\_1 \cosh t + \mathbf{f}\_2 \sinh t\$. A point \$\mathbf{x}(t)\$ is called vertex of \$H\$ if \$\dot{\mathbf{x}}(t) \cdot (\mathbf{x}(t) - \mathbf{p}\_0) = 0\$.

**Lemma 9.6** Let be \$H : \mathbf{x} = \mathbf{x}(t) := \mathbf{p}\_0 + \pm \mathbf{f}\_1 \cosh t + \mathbf{f}\_2 \sinh t\$. The tangent line at \$P\_1 : \mathbf{p}\_0 + \mathbf{f}\_1\$ is parallel to \$\mathbf{f}\_2\$ and \$P\_{12} : \mathbf{p}\_0 + \mathbf{f}\_1 + \mathbf{f}\_2\$ is on an asymptote. In case of \$\mathbf{f}\_1 \perp \mathbf{f}\_2\$ the points \$\mathbf{p}\_0 \pm \mathbf{f}\_1\$ are the two vertices of the hyperbola.

Point \$P\_2 : \mathbf{p}\_0 + \mathbf{f}\_2\$ is not a point of the hyperbola \$H\$ ! But is contained in the conjugated hyperbola

$$H' := \{\mathbf{p}_0 + \mathbf{f}_1 \sinh t \pm \mathbf{f}_2 \cosh t \mid \dots\}$$

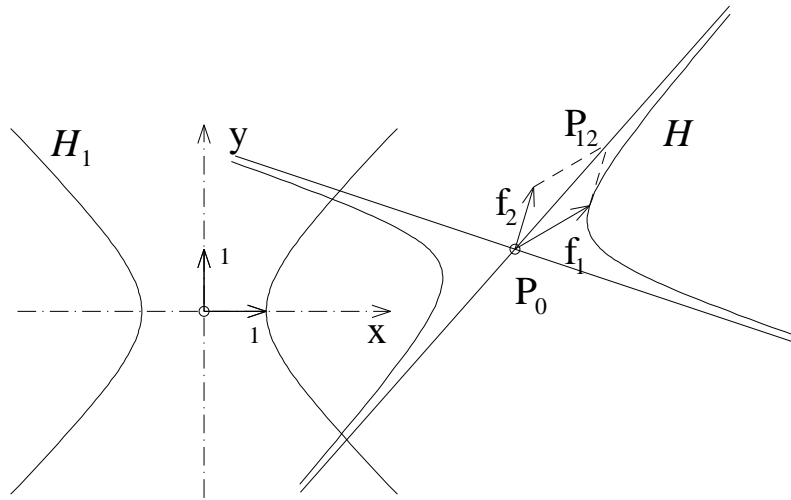


Figure 9.2: Hyperbola as affine map of the unit hyperbola

### 9.1.3 Parabola

**Definition 9.8**  $P_1 := \{(x, y) \in \mathbb{R}^2 | y = x^2\}$  is called unit parabola. Its vertex is  $(0, 0)$  and its axis is the  $y$ -axis.

**Definition 9.9** Any affine image of the unit parabola is called parabola. Hence for any  $\mathbf{p}_0 \in \mathbb{R}^2$  and linear independent vectors  $\mathbf{f}_1, \mathbf{f}_2 \in \mathbb{R}^2$  the point set

$$P = \{\mathbf{p}_0 + \mathbf{f}_1 t + \mathbf{f}_2 t^2 \mid t \in \mathbb{R}\}.$$

is a parabola and any parabola can be represented in such a way.

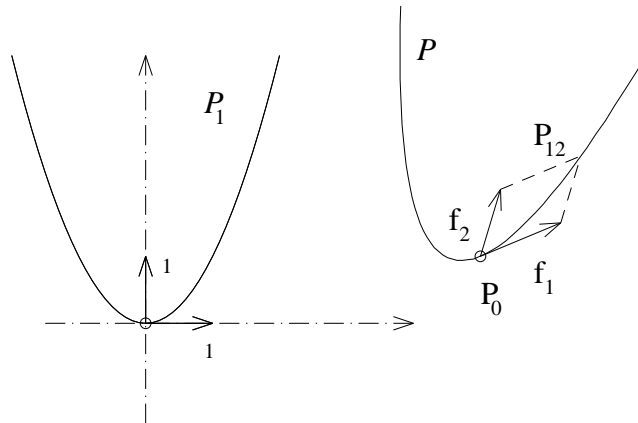


Figure 9.3: Parabola as affine map of the unit parabola

**Definition 9.10** Let be  $P : \mathbf{x} = \mathbf{x}(t) := \mathbf{p}_0 + \mathbf{f}_1 t + \mathbf{f}_2 t^2$ . A point  $\mathbf{x}(t)$  is called vertex of  $P$  if  $\dot{\mathbf{x}}(t) \cdot \mathbf{f}_2 = 0$ .

$P_0 : \mathbf{p}_0$  is a point of  $P$ ,  $\mathbf{f}_1$  tangent vector at point  $P_0$  and  $\mathbf{f}_2$  direction of the *parabola axis* and  $P_{12} : \mathbf{p}_0 + \mathbf{f}_1 + \mathbf{f}_2$  an additional point of the parabola.

If  $\mathbf{f}_1, \mathbf{f}_2$  are perpendicular, then  $P_0$  is the vertex of the parabola  $P$ .

### 9.1.4 Equation of a conic

Given: **ellipse**  $E : \mathbf{x} = \mathbf{f}_1 \cos t + \mathbf{f}_2 \sin t$

Wanted: Equation  $f(x, y) = 0$  of  $E$  (implicit representation of  $E$ ).

Solving the linear system

$$\begin{aligned} x &= f_{1x} \cos t + f_{2x} \sin t & (\text{with } \mathbf{x} = (x, y), \mathbf{f}_i = (f_{ix}, f_{iy}).) \\ y &= f_{1y} \cos t + f_{2y} \sin t \end{aligned}$$

for  $\cos t$  and  $\sin t$  yields from  $(\cos t)^2 + (\sin t)^2 - 1 = 0$ :

$$f(x, y) := (xf_{2y} - yf_{2x})^2 + (yf_{1x} - xf_{1y})^2 - (f_{1x}f_{2y} - f_{1y}f_{2x})^2 = 0.$$

This is the *equation* of the ellipse  $E$ .

Analogously one gets the equation of the **hyperbola**  $\mathbf{x} = \pm \mathbf{f}_1 \cosh t + \mathbf{f}_2 \sinh t$  from  $(\cosh t)^2 - (\sinh t)^2 - 1 = 0$ :

$$f(x, y) := (xf_{2y} - yf_{2x})^2 - (yf_{1x} - xf_{1y})^2 - (f_{1x}f_{2y} - f_{1y}f_{2x})^2 = 0,$$

Equation of the **parabola**  $\mathbf{x} = \mathbf{f}_1 t + \mathbf{f}_2 t^2$  from  $t \cdot t - t^2 = 0$ :

$$f(x, y) := (xf_{2y} - yf_{2x})^2 - (yf_{1x} - xf_{1y})(f_{1x}f_{2y} - f_{1y}f_{2x}) = 0.$$

### 9.1.5 Conics in space

The description of the conics given in the previous sections by vectors  $\mathbf{p}_0, \mathbf{f}_1, \mathbf{f}_2$  allows the simple extension into  $\mathbb{R}^3$ : Just choose the vectors  $\mathbf{p}_0, \mathbf{f}_1, \mathbf{f}_2$  from  $\mathbb{R}^3$ .

## 9.2 Quadrics: ellipsoid, cylinder

The definition of quadrics in  $\mathbb{R}^3$  can be done analogously to the definition of the conics. We give just two examples: ellipsoid and cylinder.

### 9.2.1 Ellipsoid

**Definition 9.11**  $S_1 := \{(x, y, z) \in \mathbb{R}^3 \mid x^2 + y^2 + z^2 = 1\} = \{(\cos \beta \cos \alpha, \cos \beta \sin \alpha, \sin \beta) \mid 0 \leq \alpha \leq 2\pi, -\pi/2 \leq \beta \leq \pi/2\}$ , is called unit sphere.

**Definition 9.12** Any affine image of the unit sphere is called ellipsoid. Hence for any  $\mathbf{p}_0 \in \mathbb{R}^3$  and linear independent vectors  $\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3 \in \mathbb{R}^3$  the point set

$$E = \{\mathbf{p}_0 + \mathbf{f}_1 \cos \beta \cos \alpha + \mathbf{f}_2 \cos \beta \sin \alpha + \mathbf{f}_3 \sin \beta \mid \dots\}$$

is an ellipsoid and any ellipsoid can be represented in such a way.

The coordinates  $\xi, \eta, \zeta$  of a point of  $E$  with respect of the coordinate system with origin  $P_0: \mathbf{p}_0$  and base vectors  $\{\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3\}$  fulfil the following equation

$$\xi^2 + \eta^2 + \zeta^2 = (\cos \beta \cos \alpha)^2 + (\cos \beta \sin \alpha)^2 + (\sin \beta)^2 = \dots = 1,$$

i.e.  $E$  is the unit sphere with respect of the system  $(P_0; \mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3)$ .

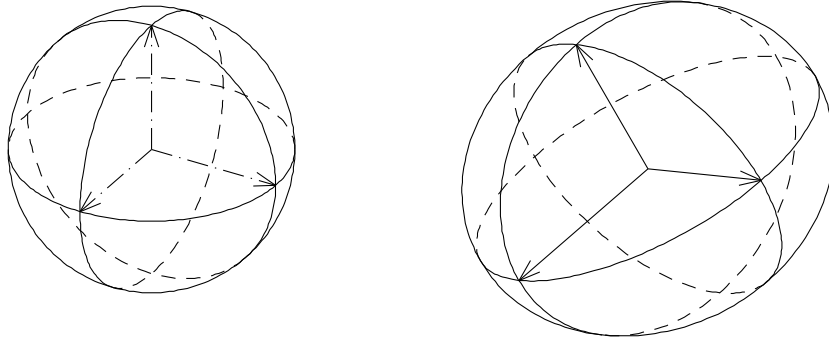


Figure 9.4: Ellipsoid as affine image of the unit sphere

**Definition 9.13** Let be  $E$  an ellipsoid with accompanying affine map  $\alpha$ . Hence,  $E = \alpha(K_1)$ . The images of three mutually perpendicular diameters of the unit sphere  $K_1$  are called conjugated diameters of the ellipsoid  $E$ .

$P_0 : \mathbf{p}_0$  is the *midpoint* of the ellipsoid  $E$ .  
 $\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3$  *conjugated half axes* of  $E$  and  
 $P_1 : \mathbf{p}_0 + \mathbf{f}_1, P_2 : \mathbf{p}_0 + \mathbf{f}_2, P_3 : \mathbf{p}_0 + \mathbf{f}_3$  **conjugated points** of  $E$ .  
 $\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3$  are i.g. not perpendicular. But the following is true:

**Lemma 9.7** The tangent plane at  $P_1$  is parallel  $\mathbf{f}_2, \mathbf{f}_3$ , a.s.o..

**Proof:** straight forward calculation. □

If  $\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3$  are mutually perpendicular then  $\mathbf{p}_0 \pm \mathbf{f}_1, \mathbf{p}_0 \pm \mathbf{f}_2, \mathbf{p}_0 \pm \mathbf{f}_3$  are the *vertices* of  $E$ .

The equation  $f(x, y, z) = 0$  of the ellipsoid  $\mathbf{x} = \mathbf{f}_1\xi + \mathbf{f}_2\eta + \mathbf{f}_3\zeta, \xi^2 + \eta^2 + \zeta^2 = 1$  yields from the system

$$\begin{aligned} x &= f_{1x}\xi + f_{2x}\eta + f_{3x}\zeta & (\text{with } \mathbf{x} &:= (x, y, z), \mathbf{f}_i = (f_{ix}, f_{iy}, f_{iz})) \\ y &= f_{1y}\xi + f_{2y}\eta + f_{3y}\zeta \\ z &= f_{1z}\xi + f_{2z}\eta + f_{3z}\zeta. \end{aligned}$$

Solving for  $\xi, \eta$  and  $\zeta$  one gets from  $\xi^2 + \eta^2 + \zeta^2 = 1$  the equation

$$f(x, y, z) := \det(\mathbf{x}, \mathbf{f}_2, \mathbf{f}_3)^2 + \det(\mathbf{f}_1, \mathbf{x}, \mathbf{f}_3)^2 + \det(\mathbf{f}_1, \mathbf{f}_2, \mathbf{x})^2 - \det(\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3)^2 = 0,$$

where  $\det(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$  is the  $3 \times 3$ -determinant with columns  $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{R}^3$ .

### 9.2.2 Cylinder

**Definition 9.14**  $C_1 := \{(x, y, z) \mid x^2 + y^2 = 1\} = \{(\cos \varphi, \sin \varphi, \alpha) \mid 0 \leq \varphi \leq 2\pi, \alpha \in \mathbb{R}\}$ , is called unit cylinder

**Definition 9.15** Any affine image of the unit cylinder is called cylinder. Hence for any  $\mathbf{p}_0 \in \mathbb{R}^3$  and linear independent vectors  $\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3 \in \mathbb{R}^3$  the point set

$$C = \{\mathbf{p}_0 + \mathbf{f}_1 \cos \varphi + \mathbf{f}_2 \sin \varphi + \mathbf{f}_3 \alpha \mid \dots\}$$

is a cylinder and any cylinder can be represented in such a way.

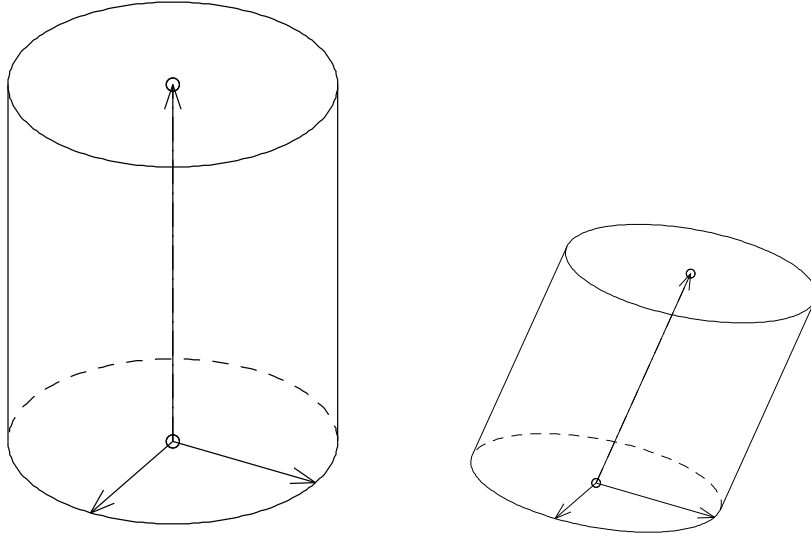


Figure 9.5: Cylinder as affine image of the unit cylinder

The coordinates  $\xi, \eta, \zeta$  of a point of  $C$  with respect of the coordinate system with origin  $P_0$ :  $\mathbf{p}_0$  and base vectors  $\{\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3\}$  fulfill the following equation

$$\xi^2 + \eta^2 = \cos^2 \varphi + \sin^2 \varphi = 1,$$

i.e.  $C$  is the unit cylinder with respect of the system  $(P_0; \mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3)$ .

$E_0 := \{\mathbf{p}_0 + \mathbf{f}_1 \cos \varphi + \mathbf{f}_2 \sin \varphi \mid \dots\}$  is called *base ellipse* of  $C$ .

$\mathbf{f}_3$  the *axis vector* of  $C$ .

$\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3$  are i.g. not perpendicular.

The **equation**  $f(x, y, z) = 0$  of the cylinder  $C : \mathbf{x} = \mathbf{f}_1 \cos \varphi + \mathbf{f}_2 \sin \varphi + \mathbf{f}_3 \alpha$  follows from the system

$$\begin{aligned} x &= f_{1x} \cos \varphi + f_{2x} \sin \varphi + f_{3x} \alpha & (\text{with } \mathbf{x} := (x, y, z), \mathbf{f}_i := (f_{ix}, f_{iy}, f_{iz})) \\ y &= f_{1y} \cos \varphi + f_{2y} \sin \varphi + f_{3y} \alpha \\ z &= f_{1z} \cos \varphi + f_{2z} \sin \varphi + f_{3z} \alpha. \end{aligned}$$

Solving for  $\cos \varphi$  and  $\sin \varphi$  and using  $(\cos \varphi)^2 + (\sin \varphi)^2 = 1$  we get

$$f(x, y, z) := (\det(\mathbf{x}, \mathbf{f}_2, \mathbf{f}_3))^2 + (\det(\mathbf{f}_1, \mathbf{x}, \mathbf{f}_3))^2 - (\det(\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3))^2 = 0.$$

## Chapter 10

# SURFACES OF REVOLUTION, RULED SURFACES, COONS PATCHES, CANAL SURFACES

This chapter deals with surfaces which are generated by moving a curve/surface in space. A surface of revolution is generated by rotating a planar curve. Moving a line generates a ruled surface. Moving spheres and cones yield canal surfaces and embankment surfaces, respectively. A Coons patch is an interpolation of a curved quadrangle. For its generation ruled surfaces are used.

### 10.1 Surfaces of revolution

**Definition 10.1** *Let be*

- $\Gamma : (r(t), z(t))^T, t \in [a, b]$  a parametric curve in  $r$ - $z$ -plane with  $r > 0$ .  
Surface  $\Phi : \mathbf{x} = (r(t) \cos \varphi, r(t) \sin \varphi, z(t))^T, t \in [a, b], \varphi \in [0, 2\pi]$  is called parametric surface of revolution,
- $\Gamma : f(r, z) = 0$  an implicit curve in  $r$ - $z$ -plane with  $r > 0$ .  
Surface  $\Phi : f(\sqrt{x^2 + y^2}, z) = 0$  is called implicit surface of revolution,

$\Gamma$  its profile curve, the  $z$ -axis its axis and a section with a plane containing the axis a meridian curve.

**Lemma 10.1** *If the profile curve of a surface of revolution (parametric or implicit) is a regular  $C^n$ -curve then the surface is regular of class  $C^n$ , too.*

**Proof:** Parametric case: From  $\mathbf{x}_t = (r'(t) \cos \varphi, r'(t) \sin \varphi, z'(t))^T, \mathbf{x}_\varphi = (-r(t) \sin \varphi, r(t) \cos \varphi, 0)^T$  we get  $\|\mathbf{x}_t \times \mathbf{x}_\varphi\| = \|(-rz' \cos \varphi, -rz' \sin \varphi, rr')^T\| = r \sqrt{(r')^2 + (z')^2} \neq 0$ .

Implicit case: From  $F(x, y, z) := f(\sqrt{x^2 + y^2}, z) = 0$  we get

$$\|\nabla F\| = \|(f_r x / \sqrt{x^2 + y^2}, f_r y / \sqrt{x^2 + y^2}, f_z)^T\| = \sqrt{f_r^2 + f_z^2} \neq 0. \quad \square$$

**Example 10.1** *Figure 10.1 shows surfaces of revolution with a) a Bézier curve b) the implicit curve  $(r - 2.5)^4 + z^4 - 1 = 0$  as profile curves.*

**Lemma 10.2** *Let be  $\Phi$  a regular  $C^2$  continuous surface of revolution (parametric or implicit). The meridians and the circles which are sections of  $\Phi$  with planes perpendicular to its axis are the lines of curvature.*

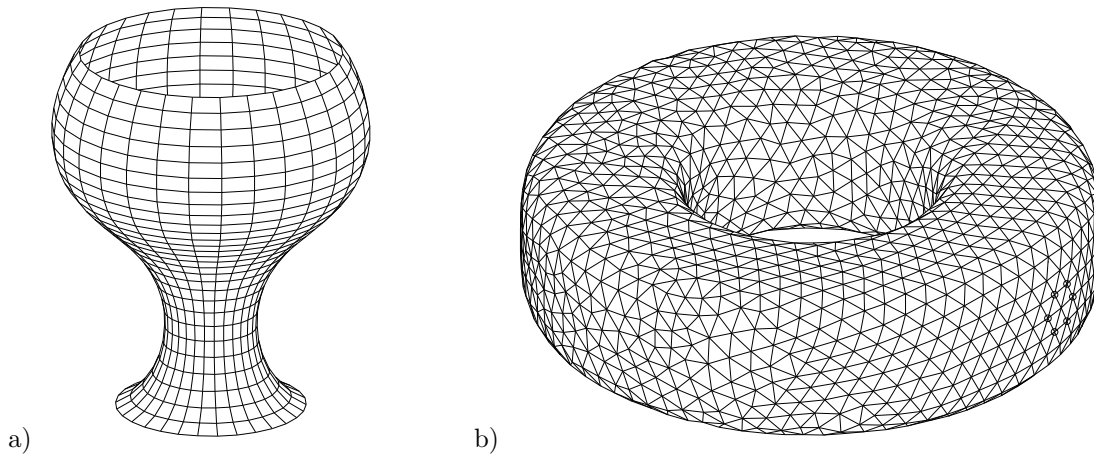


Figure 10.1: a) parametric b) implicit surface of revolution

**Proof:** The change of the unit normal while marching in tangent direction of a meridian is parallel to this tangent direction. Hence, the tangent of a meridian is a eigen direction of the Hessian  $H_h$  of the normalform function  $h$  (see Section 5.4). Any circle (intersection of  $\Phi$  with a plane orthogonal to the axis) intersects the meridian orthogonally. That means, they are the second family of lines of curvatures.  $\square$

## 10.2 Ruled surfaces

**Definition 10.2** Let be  $\Gamma : \mathbf{x} = \mathbf{c}(t), t \in [a, b]$  a regular  $C^n, n \geq 1$  curve in  $\mathbb{R}^3$  and  $\mathbf{r}(t), t \in [a, b]$  nonzero vectors of class  $C^n$ . The surface  $\Phi : \mathbf{x} = \mathbf{x}(s, t) := \mathbf{c}(t) + s\mathbf{r}(t), (s, t) \in [a, b] \times [c, d]$  is called ruled surface. Any line  $\mathbf{x} = \mathbf{x}(s, t = \text{const})$  is called ruling and curve  $\Gamma$  the base curve of the parameterization.

Simple examples of ruled surfaces are cylinders and cones. Figure 10.2 shows non trivial ruled surfaces.

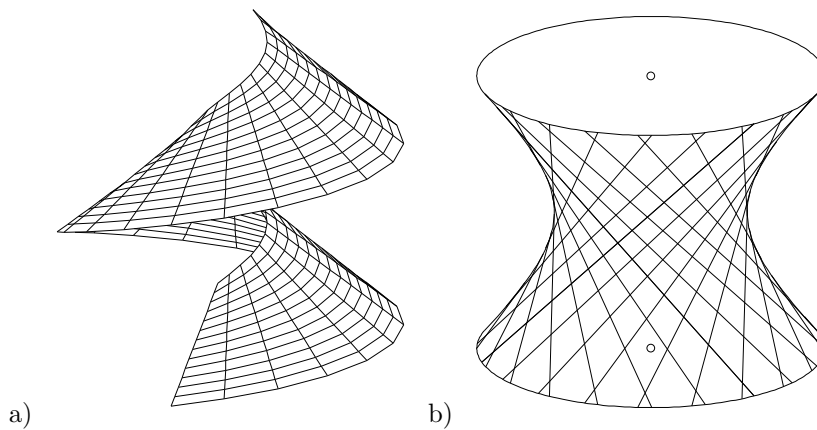


Figure 10.2: Ruled surfaces

Obviously the following statement on the regularity is true:



**Lemma 10.3** *The above representation of a ruled surface is regular and of class  $C^n$ , if  $(\dot{\mathbf{c}} + s\dot{\mathbf{r}}) \times \mathbf{r} \neq \mathbf{0}$ .*

Ruled surfaces of important practical use are the developable ones.

**Definition 10.3** *A ruled surface is called developable if the tangent planes along any ruling are the same.*

Considering the curvature of a ruled surface we proof

**Lemma 10.4** *Let be  $\Phi$  a regular ruled surface.*

- *At any point the Gauss-curvature  $K \leq 0$ .*
- *If  $\Phi$  is developable the Gauss-curvature is  $K = 0$ .*

**Proof:** Obviously, at any point the normal curvature in at least one direction is 0. From the Euler-formula (see Section 5.4.2)  $k_n(\varphi) = -\kappa_1 + (\kappa_1 - \kappa_2) \sin(\varphi)^2$  we recognize: The normal curvature is either a) constant 0 or b) one of the main curvatures is 0 or c)  $k_1$  and  $k_2$  have opposite signs in order to reach the value 0. In any case we get  $K \leq 0$ .

If  $\Phi$  is developable the normals along a ruling do not change. Hence one eigenvalue (negative main curvature) of  $H_h$  is 0 and so  $K$ , too.  $\square$

**Lemma 10.5** *Let be  $\Phi : \mathbf{x} = \mathbf{x}(s, t) := \mathbf{c}(t) + s\mathbf{r}(t)$ , a regular ruled surface.  $\Phi$  is developable if  $\det(\mathbf{r}, \dot{\mathbf{c}}, \dot{\mathbf{r}}) = 0$ .*

**Proof:**  $\mathbf{x}_s \times \mathbf{x}_t = \mathbf{r} \times (\dot{\mathbf{c}} + s\dot{\mathbf{r}})$  are parallel for fixed parameter  $t$  if the vectors  $\mathbf{r}, \dot{\mathbf{c}}, \dot{\mathbf{r}}$  are linearly dependent, hence  $\det(\mathbf{r}, \dot{\mathbf{c}}, \dot{\mathbf{r}}) = 0$ .  $\square$

### 10.3 Developable connections between curves

Let be  $\Gamma_1 : \mathbf{p}(u)$  and  $\Gamma_2 : \mathbf{q}(u)$ ,  $u \in [a, b]$  A surface that interpolates both curves is the simple ruled surface

$$\Phi : \mathbf{x}(u, v) := (1 - v)\mathbf{p}(u) + v\mathbf{q}(u), \quad (u, v) \in [a, b] \times [0, 1].$$

$\Phi$  depends essentially on the parameterization of the given curves, because points with equal parameter  $u$  are connected by a line.

In praxis there often is need of **developable** connections between two curves. In this case we are not allowed to connect points with the same parameter  $u$  because the surface should have the same tangent planes along any ruling.

Let be  $\Gamma_1 : \mathbf{p}(u)$  and  $\Gamma_2 : \mathbf{q}(w)$  two regular curves which are not contained in a plane. In order to get a developable surface that connects  $\Gamma_1, \Gamma_2$  a point  $P : \mathbf{p}(u)$  of  $\Gamma_1$  has to be connected to a point  $Q(u) : \mathbf{q}(w(u))$  on  $\Gamma_2$  such that the tangent planes for surface points on the line  $PQ$  agree. This is the case if the tangent vector  $\dot{\mathbf{p}}(u)$  at  $P$ , the tangent vector  $\dot{\mathbf{q}}(w(u))$  at  $Q(u)$  and the direction (of the ruling)  $\mathbf{p}(u) - \mathbf{q}(w(u))$  are linearly dependent, hence if

$$(DV) \quad \det(\dot{\mathbf{p}}(u), \dot{\mathbf{q}}(w(u)), \mathbf{p}(u) - \mathbf{q}(w(u))) = 0.$$

The parametrization of the surface is

$$\Phi : \mathbf{x}(u, v) := (1 - v)\mathbf{p}(u) + v\mathbf{q}(w(u)).$$

For any  $u$  equation (DV) has to be solved. One has to provide suitable starting parameters.

**Developable polyhedral approximation:**

The equation above is the key for the construction of a developable polyhedral approximation:

The **algorithm:**

1. Generate a Polygon  $\Pi : \mathbf{p}(u_1), \mathbf{p}(u_2), \dots$  on curve  $\Gamma_1$ .
2. For any point  $\mathbf{p}(u_i)$  of  $\Pi$ : Determine a point  $\mathbf{q}(w_i) \in \Gamma_2$  with  $\det(\dot{\mathbf{p}}(u_i), \dot{\mathbf{q}}(w_i), \mathbf{p}(u_i) - \mathbf{q}(w_i)) = 0$  by a Newton-iteration for a suitable starting parameter. "Suitable" means, that the surface to be generated should be similar to a pipe to meet the technical demand.

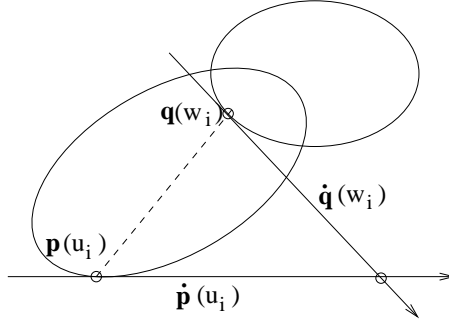


Figure 10.3: Condition for "developable"

3. **Developing** the generated polyhedral surface:  
 Divide any quadrangle between two rulings into two triangles and develop the strip of triangles. For developing a triangle  $P, Q, R$  let be  $\bar{P}, \bar{Q}$  the developing of the points  $P, Q$ .

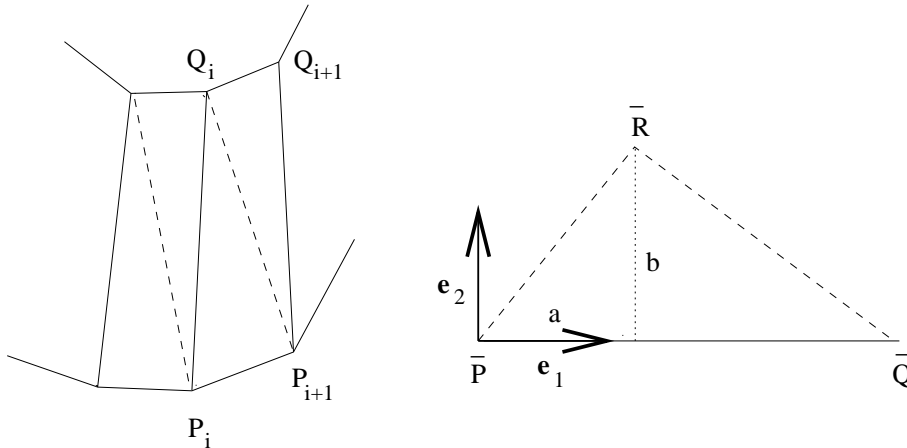


Figure 10.4: Developing a triangle

Determine

- (1) the lengths of line segments  $PQ(= \bar{P}\bar{Q}), PR, QR$ .
- (2) the unit vector  $\mathbf{e}_1$  in direction  $\bar{P}\bar{Q}$ ,  $\mathbf{e}_2 \perp \mathbf{e}_1$ .
- (3)  $a, b$  are the coordinates of point  $\bar{R}$  in system  $(\bar{P}; \mathbf{e}_1, \mathbf{e}_2)$ . Hence, point  $(a, b)$  is one of the two intersection points of the circles  $x^2 + y^2 = |PR|^2$ ,  $(x - |PQ|)^2 + y^2 = |RQ|^2$ . Solving the system yields:  

$$a = \frac{|PQ|^2 + |RP|^2 - |RQ|^2}{2|PQ|}, \quad b = |RP|^2 - a^2.$$
- (4)  $\bar{R} = \bar{P} + a\mathbf{e}_1 + b\mathbf{e}_2$  (developing of  $R$ ).

**Example 10.2** Figure 10.5 shows the development of a connection surface of two ellipses:

1. Ellipse in  $x$ - $y$ -plane:  $\frac{(x-50)^2}{30^2} + \frac{y^2}{50^2} = 1$ , 2. Ellipse in  $y$ - $z$ -plane:  $\frac{y^2}{20^2} + \frac{(z-50)^2}{40^2} = 1$ .

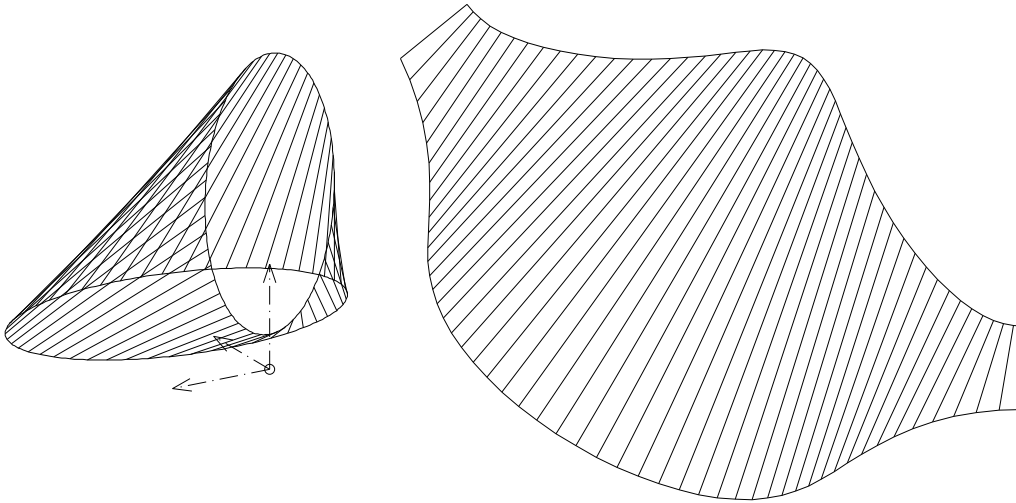


Figure 10.5: Developing of a connection surface between two ellipses

## 10.4 Coons patches

Given: The boundaries of a curved quadrangular patch:

$$\mathbf{x} = \mathbf{x}(u, 0), \mathbf{x} = \mathbf{x}(u, 1), \mathbf{x} = \mathbf{x}(0, v), \mathbf{x} = \mathbf{x}(1, v), \quad u, v \in [0, 1].$$

Wanted: A surface  $\mathbf{x} = \mathbf{x}(u, v)$  that interpolates the four curves.

**Idea:** consider the two ruled surfaces  $\mathbf{r}_1, \mathbf{r}_2$  which interpolate  $\mathbf{x}(u, 0), \mathbf{x}(u, 1)$  and  $\mathbf{x}(0, v), \mathbf{x}(1, v)$  respectively. Combine these surfaces together with a suitable correction  $\mathbf{r}_{cd}$ , in order to meet the demand:

$$\begin{aligned} \mathbf{r}_1(u, v) &:= (1-v)\mathbf{x}(u, 0) + v\mathbf{x}(u, 1), & \mathbf{r}_2(u, v) &:= (1-u)\mathbf{x}(0, v) + u\mathbf{x}(1, v), \\ \mathbf{r}_{12} &:= (1-u, u) \begin{pmatrix} \mathbf{x}(0, 0) & \mathbf{x}(0, 1) \\ \mathbf{x}(1, 0) & \mathbf{x}(1, 1) \end{pmatrix} \begin{pmatrix} 1-v \\ v \end{pmatrix} \\ &= (1-u)((1-v)\mathbf{x}(0, 0) + v\mathbf{x}(0, 1)) + u((1-v)\mathbf{x}(1, 0) + v\mathbf{x}(1, 1)) \end{aligned}$$

The surface

$$\Phi : \mathbf{x} = \mathbf{r}_1(u, v) + \mathbf{r}_2(u, v) - \mathbf{r}_{12}(u, v)$$

is the solution of the problem, because

$$\mathbf{r}_1(0, v) + \mathbf{r}_2(0, v) - \mathbf{r}_{12}(0, v) = (1-v)\mathbf{x}(0, 0) + v\mathbf{x}(0, 1) + \mathbf{x}(0, v) - ((1-v)\mathbf{x}(0, 0) + v\mathbf{x}(0, 1)) = \mathbf{x}(0, v),$$

...

(The surface  $\mathbf{x} = \mathbf{r}_{12}(u, v)$  is a *bilinear* interpolation of the quadrangle  $\mathbf{x}(0, 0), \mathbf{x}(1, 0), \mathbf{x}(0, 1), \mathbf{x}(1, 1)$ .)  $\Phi$  is called *bilinear blended Coons patch* (see Farin: *Curves and Surfaces ...*).

Figure 10.6 shows an example of a Coons patch together with the ruled surfaces needed for its generation.

## 10.5 Canal surface, embankment surface

Canal surfaces and embankment surfaces rely on the same method of generation:

Given a one parameter family of regular implicit surfaces  $\Phi_c : f(\mathbf{x}, c) = 0, c \in [c_1, c_2]$ . The intersection curve of two neighbored surfaces  $\Phi_c$  and  $\Phi_{c+\Delta c}$  fulfills the two equations  $f(\mathbf{x}, c) = 0$  and

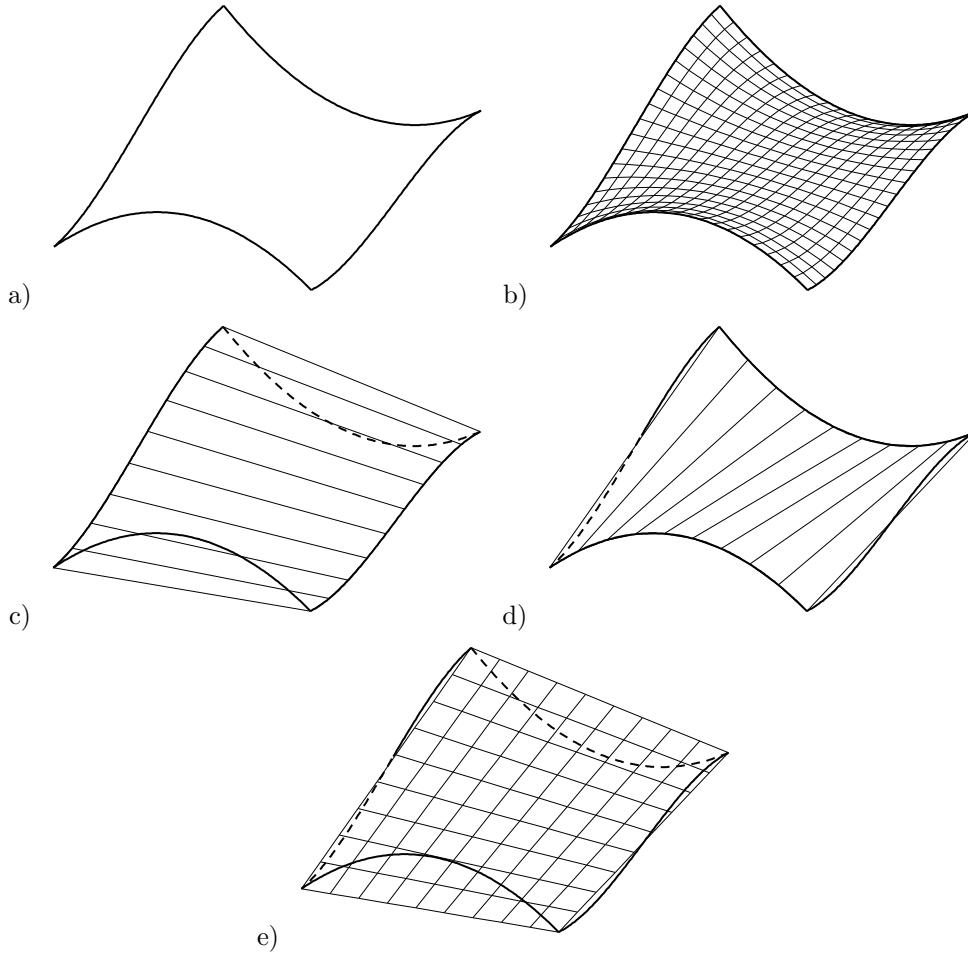


Figure 10.6: Generation of a Coons patch: a) boundary curves, b) Coons patch, c),d) ruled surfaces  $\mathbf{r}_1$  and  $\mathbf{r}_2$  e) bilinear surface  $\mathbf{r}_{12}$

$f(\mathbf{x}, c + \Delta c) = 0$ . We consider the limit for  $\Delta c \rightarrow 0$  and get  $f_c(\mathbf{x}, c) = \lim_{\Delta c \rightarrow 0} \frac{f(\mathbf{x}, c) - f(\mathbf{x}, c + \Delta c)}{\Delta c} = 0$ . The last equation motivates the following definition.

**Definition 10.4** Let be  $\Phi_c f(\mathbf{x}, c) = 0, c \in [c_1, c_2]$  a one parameter family of regular implicit  $C^2$ -surfaces. The surface which is defined the two equations

$$f(\mathbf{x}, c) = 0, \quad f_c(\mathbf{x}, c) = 0$$

is called envelope of the given family of surfaces.

The condition for the regularity of the envelope is

**Lemma 10.6** Let be  $\Phi_c f(\mathbf{x}, c) = 0, c \in [c_1, c_2]$  a one parameter family of regular implicit  $C^n$ -surfaces,  $n \geq 2$ . The envelope of the family  $\Phi_c$  is a regular  $C^{n-1}$ -surface if  $f_{cc}(\mathbf{x}, c) \neq 0$ . The envelope has  $G^1$ -contact to the surfaces  $\Phi_c$ .

**Proof:** Because of  $f_{cc}(\mathbf{x}, c) \neq 0$  the equation  $f_c(\mathbf{x}, c) = 0$  can be solved locally for  $c$ . If  $c(\mathbf{x})$  is the solution we get the (theoretical) implicit representation of the envelope:  $F(\mathbf{x}) = f(\mathbf{x}, c(\mathbf{x})) = 0$ . Differentiating yields:  $\nabla F(\mathbf{x}) = \nabla f(\mathbf{x}, c(\mathbf{x})) - \frac{f_c(\mathbf{x}, c(\mathbf{x}))}{f_{cc}(\mathbf{x}, c(\mathbf{x}))} \nabla f_c(\mathbf{x}, c(\mathbf{x}))$ . Hence, for points on the envelope we have  $\nabla F(\mathbf{x}) = \nabla f(\mathbf{x}, c(\mathbf{x})) \neq \mathbf{0}$ .  $\square$

### 10.5.1 Canal surfaces

**Definition 10.5** Let  $\Gamma : \mathbf{x} = \mathbf{c}(u) = (a(u), b(u), c(u))^T$  be a regular space curve and  $r(u)$  a  $C^1$ -function with  $r > 0$  and  $|\dot{r}| < \|\dot{\mathbf{c}}\|$ . The envelope of the one parameter family of spheres

$$f(\mathbf{x}; u) := (\mathbf{x} - \mathbf{c}(u))^2 - r(u)^2 = 0$$

is called a canal surface and  $\Gamma$  its directrix.

In case of a constant radius function the envelope is called pipe surface.

In order to get a parametric representation we consider the envelope condition:

$$f_u(\mathbf{x}, u) := 2\left((\mathbf{x} - \mathbf{c}(u))\dot{\mathbf{c}}(u) - r(u)\dot{r}(u)\right) = 0$$

which is for any parameter  $t$  the equation of a plane orthogonal to  $\dot{\mathbf{c}}(u)$ . Hence, the envelope is the collection of circles. This statement is the key for the following parametric representation:

The midpoint of the circle has distance  $d := \frac{r\dot{r}}{\|\dot{\mathbf{c}}\|} < r$  (see condition above) from the centre of the sphere and radius  $\sqrt{r^2 - d^2}$ .

$$\mathbf{x} = \mathbf{x}(u, v) := \mathbf{c}(u) - \frac{r(u)\dot{r}(u)}{\|\dot{\mathbf{c}}(u)\|^2}\dot{\mathbf{c}}(u) + \frac{r(u)\sqrt{\|\dot{\mathbf{c}}(u)\|^2 - \dot{r}^2}}{\|\dot{\mathbf{c}}(u)\|}(\mathbf{e}_1(u)\cos(v) + \mathbf{e}_2(u)\sin(v)),$$

with  $\mathbf{e}_1, \mathbf{e}_2$  an orthonormal base orthogonal to tangent vector  $\dot{\mathbf{c}}$ , is a parametric representation of the canal surface.

For  $\dot{r} = 0$  we get the parametric representation of a pipe surface:

$$\mathbf{x} = \mathbf{x}(u, v) := \mathbf{c}(u) + r(\mathbf{e}_1(u)\cos(v) + \mathbf{e}_2(u)\sin(v)).$$

Figure 10.7 shows a canal surface with the helix  $(\cos(u), \sin(u), 0.25u)$ ,  $u \in [0, 4]$  as directrix and  $r(u) := 0.2 + 0.8u/2\pi$ . The choice for  $\mathbf{e}_1, \mathbf{e}_2$  is:  $\mathbf{e}_1 := (b, -a, 0)/\|\cdots\|$ ,  $\mathbf{e}_2 := (\mathbf{e}_1 \times \dot{\mathbf{c}})/\|\cdots\|$ .

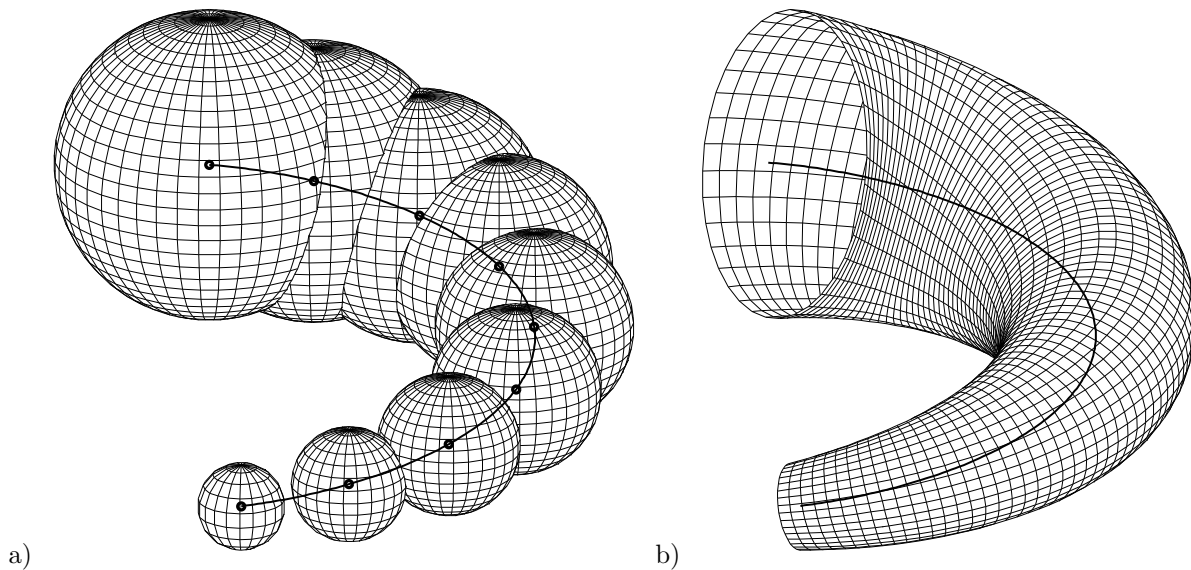


Figure 10.7: Generation of a canal surface a) spheres b) envelope of the spheres

### 10.5.2 Embankment surfaces

**Definition 10.6** Let  $\Gamma : \mathbf{x} = \mathbf{c}(u) = (a(u), b(u), c(u))^\top$  be a regular space curve and  $m \in \mathbb{R}, m > 0$  with  $|m\dot{c}| < \sqrt{\dot{a}^2 + \dot{b}^2}$ . The envelope of the one parameter family of cones

$$f(\mathbf{x}; u) := (x - a(u))^2 + (y - b(u))^2 - m^2(z - c(u))^2 = 0$$

is called an embankment surface and  $\Gamma$  its directrix.

In order to get a parametric representation we consider the envelope condition:

$$f_u(\mathbf{x}, u) := (x - a(u))\dot{a}(u) + (y - b(u))\dot{b}(u) - m^2(z - c(u))\dot{c}(u) = 0$$

which is the equation of a plane passing the vertex of the cone with parameter  $u$ . The condition  $|m\dot{c}| < \sqrt{\dot{a}^2 + \dot{b}^2}$  guarantees an intersection with the cone. (For proof: consider the intersection within the plane  $z = c + 1$ .)

The envelope consists of two ruled surfaces which can be represented parametrically as follows:

$$\mathbf{x} = \mathbf{x}(u, v) := \mathbf{c}(u) + v\mathbf{r}_{1/2}(u)$$

with  $\mathbf{r}_1(u) := (\xi_1(u), \eta_1(u), 1)$ ,  $\mathbf{r}_2(u) := (\xi_2(u), \eta_2(u), 1)$  and  $(\xi_1, \eta_1)$ ,  $(\xi_2, \eta_2)$  intersection points of the circle  $\xi^2 + \eta^2 - m^2 = 0$  and the line  $\xi\dot{a} + \eta\dot{b} - m^2\dot{c} = 0$ .

Figure 10.8 curve  $\Gamma$  is the helix  $(\cos(t), \sin(t), 0.1t + 0.2)^\top$ ,  $u \in [0, 4]$  and  $m = 1$ . For determining the vectors  $\mathbf{r}_1, \mathbf{r}_2$  the intersection with plane  $z = 0$  was considered and  $v \in [0, 1]$ .

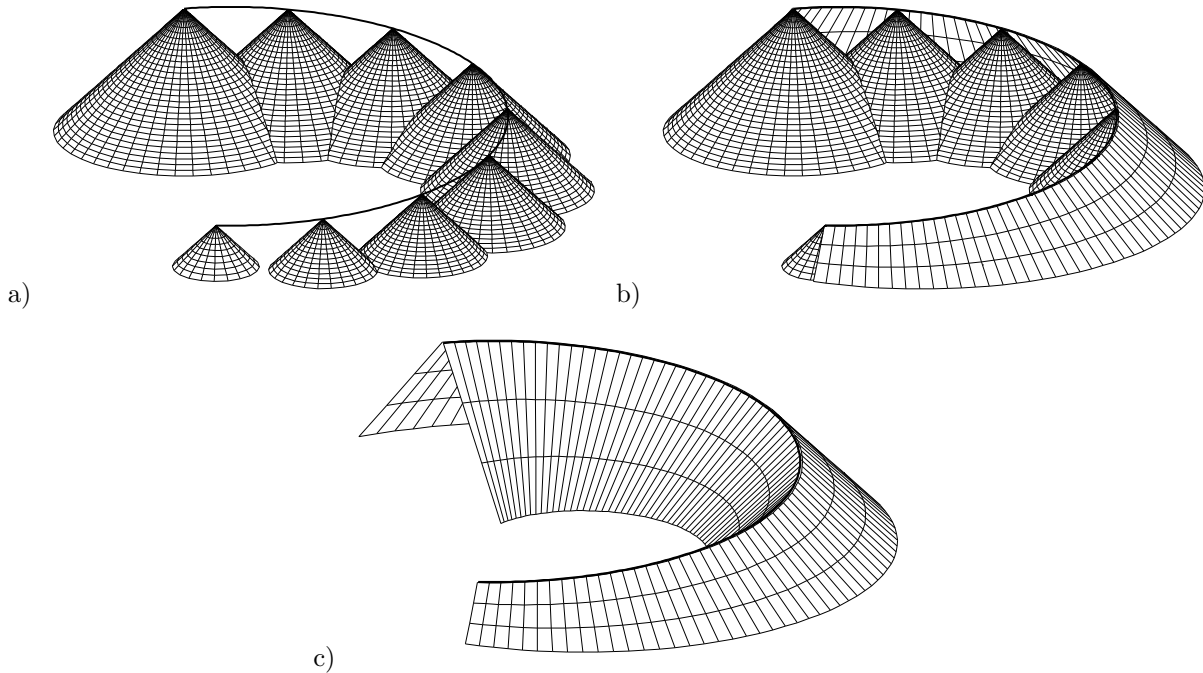


Figure 10.8: Generation of an embankment surface a) cones b) c) envelope of the cones

# Chapter 11

## BLENDING CURVES AND SURFACES

Smoothing edges and corners of objects defined by intersecting surfaces is a basic task in CAD. The operation is called *blending*. A measure of the smoothness of the blending surface is the degree  $n$  of the so called *geometric continuity*,  $G^n$ -continuity.  $G^1$ -continuity means the tangent planes at the curves of contact are the same.  $G^2$ -continuity guarantees the continuity of the normal curvatures. There are several methods for blending *two* surfaces. They depend essentially on the representations of the surfaces to be blended.

### 11.1 $G^n$ -blending

**Definition 11.1** a) A regular  $C^n$ -continuous curve  $\Gamma$  which has  $G^n$ -contact (see Section 5.4.3) to regular  $C^n$ -continuous curves  $\Gamma_1, \Gamma_2, \dots$  at points  $\mathbf{p}_1 \in \Gamma_1, \mathbf{p}_2 \in \Gamma_2, \dots$  respectively, is called a  $G^n$ -blending curve of  $\Gamma_1, \Gamma_2, \dots$  with contact points  $\mathbf{p}_1, \mathbf{p}_2, \dots$ .

b) A regular  $C^n$ -continuous surface  $\Phi$  which has  $G^n$ -contact to regular  $C^n$ -continuous surfaces  $\Phi_1, \Phi_2, \dots$  at curves  $\Gamma_1 \subset \Phi_1, \Gamma_2 \subset \Phi_2, \dots$  respectively, is called a  $G^n$ -blending surface of  $\Phi_1, \Phi_2, \dots$  with contact curves  $\Gamma_1, \Gamma_2, \dots$ .

### 11.2 Implicit $G^n$ -blending curves and surfaces

For functional splines we need the following statement.

**Lemma 11.1** Let  $\alpha : \mathbb{R} \rightarrow \mathbb{R}$  be a  $C^n$ -function with  $\alpha^{(i)}(0) = 0$  for  $i = 0, 1, \dots, n$  and  $f, g, v : \mathbb{R}^k \rightarrow \mathbb{R}$  be  $C^n$ -functions.

Then Function  $F : \mathbb{R}^k \rightarrow \mathbb{R}$  with

$$F(\mathbf{x}) := f(\mathbf{x}) - \alpha(g(\mathbf{x}))v(\mathbf{x})$$

and function  $f$  have at points  $\mathbf{x} \in \mathbb{R}^k$  with  $f(\mathbf{x}) = g(\mathbf{x}) = 0$  the same partial derivatives up to the  $n$ -th order.

Hence: If  $f = 0$  and  $g = 0$  are implicit plane curves (surfaces) then the curve (surface)  $F = 0$  has  $C^n$ -contact to  $f = 0$  at intersection points  $f = 0, g = 0$ .

**Proof:** straight forward calculation.

The known implicit blending methods can be considered as generalizations of simple plane cases:

- The parabola  $y - x^2 = 0$  is a  $G^1$ -blending curve of the line  $y = 0$  with the intersection point of the coordinate axes, i.e.  $(0, 0)$ , as contact point.  
The generalization leads to the *parabolic functional splines*.
- The circle  $(x - 1)^2 + (y - 1)^2 - 1 = 0$  is a  $G^1$ -blending curve of the lines  $x = 0$  and  $y = 0$  (coordinate axes) with the intersection points of line  $x = 0$  with line  $y = 1$ , i.e. point  $(0, 1)$ , of line  $y = 0$  with line  $x = 1$ , i.e. point  $(1, 0)$ , as points of contact.  
The generalization leads to the *elliptic functional splines*.

### 11.2.1 Parabolic functional splines

**Theorem 11.1** *Let  $f = 0, f_0 = 0$  be two intersecting  $C^n$ -continuous regular implicit plane curves (surfaces) and let the explicit plane curve  $d = \alpha(c)$ , the correlation curve, have  $C^n$ -contact to the  $c$ -coordinate axis at  $(0, 0)$ , i.e.  $\alpha^{(k)}(0) = 0$  for  $k = 0, \dots, n$ .  
Then the implicit **blending curve (surface)***

$$F := f - \alpha(f_0) = 0$$

has  $C^n$ -contact to  $f = 0$  at points (curves) of contact  $f = 0, f_0 = 0$ .

$F$  is regular at points with  $\nabla f \times \nabla f_0 \neq \mathbf{0}$ .

$F = 0$  is called **parabolic functional spline**,  $f = 0$  the base curve (surface) and  $f_0 = 0$  the transversal curve (surface).

**Proof:** obvious.

For blending two or more base curves (surfaces) one may choose  $f := f_1 f_2 \cdots f_k$  and a suitable transversal  $f_0$  (see examples below).

An extension of Theorem 11.1 is the following statement:

**Theorem 11.2** *Let  $f_1 = 0, f_{10} = 0$  and  $f_2 = 0, f_{20} = 0$  be two pairs of intersecting  $C^n$ -continuous regular implicit plane curves (surfaces) and let the explicit plane curves  $d = \alpha_1(c)$  and  $d = \alpha_2(c)$ , the correlation curves, have  $C^n$ -contact to the  $c$ -coordinate axis at  $(0, 0)$ , i.e.  $\alpha_1^{(k)}(0) = 0, \alpha_2^{(k)}(0) = 0$  for  $k = 0, \dots, n$ .*

*Intersection point (curve)  $f_i = 0, f_{i0} = 0$  must not be contained in curves (surfaces)  $f_k = 0$  and  $f_{k0} = 0$  for  $i \neq k$ .*

*Then the implicit **blending curve (surface)***

$$F(\mathbf{x}) := f_1(\mathbf{x}) \alpha_2(f_{20}(\mathbf{x})) - f_2(\mathbf{x}) \alpha_1(f_{10}(\mathbf{x})) = 0$$

has  $G^n$ -contact to  $f_i = 0$  at regular points (curves) of contact  $f_i = 0, f_{i0} = 0, \alpha_k(f_{k0}) \neq 0$  for  $i, k = 1, 2, i \neq k$ .

$F = 0$  is called **symmetric parabolic functional spline**,  $f_i = 0, i = 1, 2$ , the base curves (surfaces) and  $f_{i0} = 0, i = 1, 2$  the transversal curves (surfaces).

**Proof:** In the vicinity of regular intersection points (curves)  $f_1 = 0, f_{10} = 0, \alpha_2(f_{20}) \neq 0$  equation  $F = 0$  is equivalent to  $\bar{F} := f_1 - \alpha_1(f_{10}) \frac{f_2}{\alpha_2(f_{20})} = 0$ . From Lemma 11.1 we get the  $G^n$ -continuity. Analogously one treats the second case.  $\square$

Simple examples for **explicit  $C^n$ -correlation curves** are

$$d = \frac{\mu}{1-\mu} c^{n+1}, \quad 0 < \mu < 1, \quad n \geq 1$$

Hence the  $G^n$ -blending curves (surfaces)

a) of Theorem 11.1 can be expressed by



$$\tilde{F} := (1 - \mu)f - \mu f_0^{n+1} = 0, \quad 0 < \mu < 1, \quad n \geq 1$$

Varying  $\mu$  effects a variation of the closeness of  $\tilde{F} = 0$  to the base curve (surface). Hence  $\mu$  plays the role of a *thumbweight*.

b) of Theorem 11.2 using  $\alpha_1(c) = (1 - \mu)c^{n+1}$ ,  $\alpha_2(c) = \mu c^{n+1}$  by

$$\tilde{F} := (1 - \mu)f_1 f_{20}^{n+1} - \mu f_2 f_{10}^{n+1} = 0 \quad 0 < \mu < 1, \quad n \geq 1.$$

Varying  $\mu$  effects an asymmetric variation of the closeness of  $\tilde{F} = 0$  to the base curves (surfaces). We call  $\mu$  *balance*.

**Example 11.1** (*parabolic functional splines*)

a)  $f(x, y, z) = 1 - x^2 - y^2 - z^2 = 0$ , (sphere),  
 $f_0(x, y, z) = z = 0$  (plane).

The parabolic functional spline  $F := (1 - \mu)f - \mu f_0^{n+1} = 0$ , has  $G^n$ -contact to the sphere along the "equator".

Figure 11.1a) shows the functional spline for  $z \geq 0$  together with the lower part of the sphere pasted at the equator. The parameters are  $n = 2, \mu = 0.056$ .

b)  $f_1(x, y, z) = r^2 - x^2 - z^2 = 0$  (cylinder),  $f_2(x, y, z) = r^2 - y^2 - z^2 = 0$  (cylinder),  
 $f_0(x, y, z) = z = 0$  (plane).

The parabolic functional spline  $F := (1 - \mu)f_1 f_2 - \mu f_0^{n+1} = 0$ , has  $G^n$ -contact to the cylinders. Figure 11.1b) shows the functional spline for  $z \geq 0$ .

c)  $f_i = 0$ ,  $i = 1, \dots, 4$ ,  $f_0 = 0$  are planes. The parabolic functional spline  $F := (1 - \mu)f_1 f_2 f_3 f_4 - \mu f_0^{n+1} = 0$  has  $G^n$ -contact to the "roof".  
 (see Figure 11.1c), parameters:  $n = 2, \mu = 0.5$ ).

d) For  $f_1(x, y, z) := x^2 + y^2 - 1 = 0$ , (cylinder),  $f_2(x, y, z) := x^2 + z^2 - 1 = 0$ , (cylinder),  
 $f_3(x, y, z) := y^2 + z^2 - 1 = 0$ , (cylinder),  $f_0(x, y, z) := 9 - x^2 - y^2 - z^2 = 0$  (sphere).  
 the parabolic functional spline

$$(1 - \mu)f_1 f_2 f_3 - \mu f_0^3 = 0$$

has  $G^2$ -contact to the 3 cylinders at the 3 circles  $f_i = 0$ ,  $f_0 = 0$ ,  $i = 1, 2, 3$ .  
 (see Figure 11.2, parameters:  $n = 2, \mu = 0.0003$ ).

**Example 11.2** (*symmetric parabolic functional splines*)

a)  $f_1(x, y, z) = 4 - x^2 - y^2 = 0$  (cylinder),  $f_{10}(x, y, z) = z + 1 = 0$  (plane)  
 $f_2(x, y, z) = x^2 - y^2 - 1 = 0$  (cylinder),  $f_{20}(x, y, z) = 1 - z = 0$  (plane).

The symmetric parabolic functional spline  $(1 - \mu)f_1 f_{20}^{n+1} - \mu f_2 f_{10}^{n+1} = 0$  has  $G^n$ -contact to the cylinders. (Figure 11.3, left, parameters  $n = 2, \mu = 0.5$ .)

b)  $f_1(x, y, z) = x^2 + z^2 - 1 = 0$  (cylinder),  $f_{10}(x, y, z) = (2x - y)(2x + y) = 0$  (2 planes).  
 $f_2(x, y, z) = y^2 + z^2 - 1 = 0$  (cylinder),  $f_{20}(x, y, z) = (2y - x)(2y + x) = 0$ , (2 planes)  
 (Figure 11.3, right, parameters:  $n = 2, \mu = 0.5$ .)

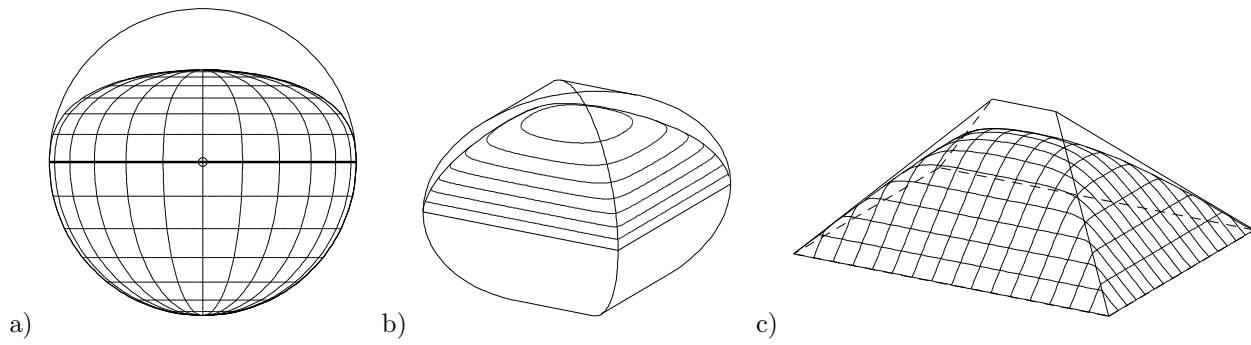


Figure 11.1:  $G^2$ -blending of a) sphere, b) two cylinders and c) 4 planes (parabol. funct. splines)

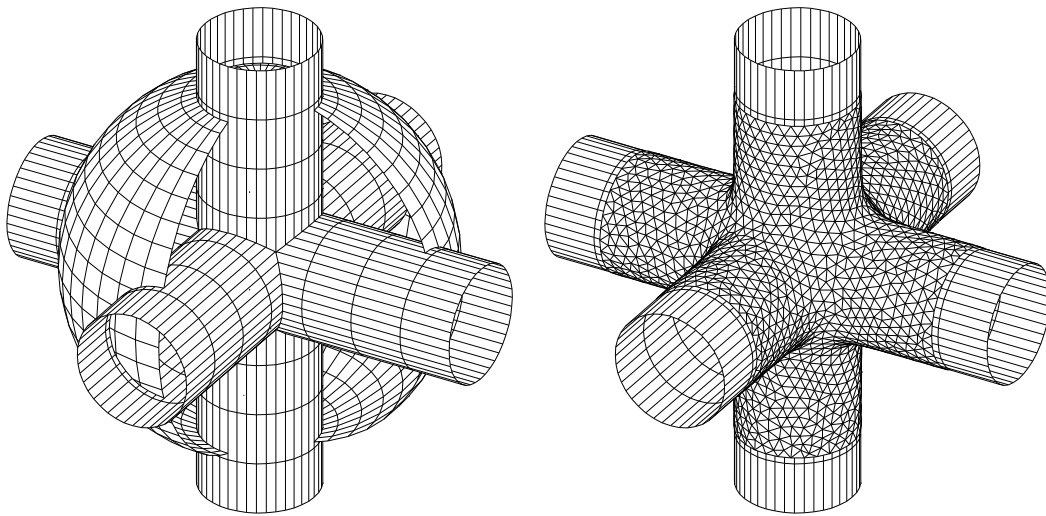


Figure 11.2:  $G^2$ -blending of 3 cylinders (parabolic functional spline)

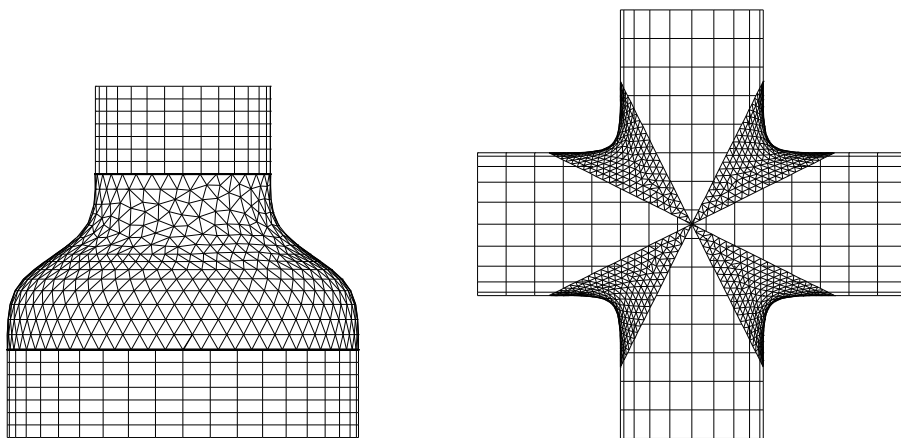


Figure 11.3:  $G^2$  symmetric parabolic functional splines

The following example shows repeated application of parabolic functional splines.

**Example 11.3** (blending “half cylinders”) Let be

$$f_1(x, y, z) := r_1^2 - y^2 - z^2 = 0 \quad (\text{cylinder}),$$

$$f_2(x, y, z) := r_2^2 - x^2 - z^2 = 0 \quad (\text{cylinder}),$$

$$f_3(x, y, z) := r_3^2 - x^2 - y^2 = 0 \quad (\text{cylinder}),$$

$$F_0(x, y, z) := r_0^2 - x^2 - y^2 - z^2 = 0 \quad (\text{sphere}).$$

In the first step we generate “ $G^n$ -closings” of the cylinders (see Figure 11.4):

$$F_1(x, y, z) := (1 - \lambda_1)f_1(x, y, z) - \lambda_1(\sqrt{r_0^2 - r_1^2} - x)^{n+1} = 0, \quad 0 < \lambda_1 < 1,$$

$$F_2(x, y, z) := (1 - \lambda_2)f_2(x, y, z) - \lambda_2(\sqrt{r_0^2 - r_2^2} - y)^{n+1} = 0, \quad 0 < \lambda_2 < 1,$$

$$F_3(x, y, z) := (1 - \lambda_3)f_3(x, y, z) - \lambda_3(\sqrt{r_0^2 - r_3^2} - z)^{n+1} = 0, \quad 0 < \lambda_3 < 1,$$

The following parabolic functional spline:

$$F := -(1 - \mu)F_1F_2F_3 - \mu F_0^{n+1} = 0$$

is a  $G^n$ -connection surface for the closings  $F_1 = 0, F_2 = 0$  and  $F_3 = 0$  and hence for the cylinders  $f_1 = 0, f_2 = 0$  and  $f_3 = 0$ .

The parameters for Figure 11.4 are:  $r_1 = r_2 = r_3 = 0.45, r_0 = 1, n = 2, \lambda_1 = \lambda_2 = \lambda_3 = 0.2$  and  $\mu = 0.01$ .

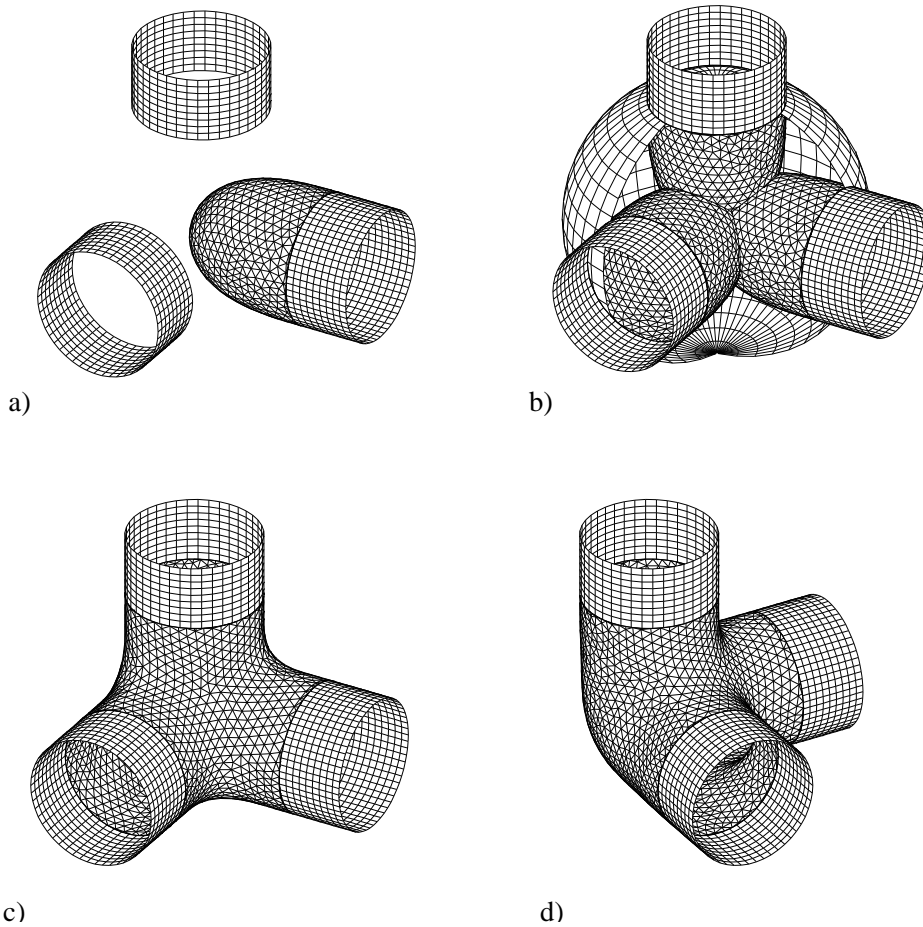


Figure 11.4: a)  $G^2$ -closing of one cylinder b)  $G^2$ -closings of the cylinders c) and d)  $G^2$ -connection

### 11.2.2 Elliptic functional splines

**Theorem 11.3** Let be  $c_0, d_0 \in \mathbb{R}$  with  $c_0 > 0, d_0 > 0$  and  $f_1 = c, f_2 = d, 0 \leq c \leq c_0, 0 \leq d \leq d_0$  be two pencils of regular implicit surfaces and  $k(c, d) = 0$  a regular implicit plane curve, the correlation curve, that has  $G^n$ -contact to the  $c$ -axis at point  $(c_0, 0)$  and to the  $d$ -axis at point  $(0, d_0)$ , i.e.

$$\frac{\partial^i k(c_0, 0)}{\partial c^i} = 0, \quad \frac{\partial^i k(0, d_0)}{\partial d^i} = 0, \quad i = 0, 1, \dots, n.$$

Then

$$F(\mathbf{x}) := k(f_1(\mathbf{x}), f_2(\mathbf{x})) = 0$$

has  $G^n$ -contact to the curve (surface)

$f_1 = 0$  at the point (curve) of contact  $f_1 = 0, f_2 = d_0$  and to

$f_2 = 0$  at the point (curve) of contact  $f_1 = c_0, f_2 = 0$ .

$F = 0$  is called **elliptic functional spline** and  $f_1 = 0$  and  $f_2 = 0$  the base curve (surface).

**Proof:** At point  $(c_0, 0)$  there exists a local explicit representation  $d = \alpha(c - c_0)$  with  $\alpha^{(i)}(0) = 0$  for  $i = 0, 1, \dots, n$  (cf. Lemma 2.1 c)). Hence equation  $F = k(f_1, f_2) = 0$  is locally equivalent to  $\bar{F} := f_2 - \alpha(f_1 - c_0) = 0$ . From Lemma 3.2 we get:  $\bar{F} = 0$  is a locally regular implicit curve (surface) which has  $G^n$ -contact to curve (surface)  $f_2 = 0$  at point (curve)  $f_2 = 0, f_1 = c_0$ . Analogously the  $G^n$ -contact between  $F = 0$  and  $f_1 = 0$  at point (curve)  $f_1 = 0, f_2 = d_0$  is proved.  $\square$

Examples for **implicit  $G^n$ -correlation curves** are

$$k(c, d) = (1 - \mu) \frac{cd}{c_0 d_0} - \mu \left( 1 - \frac{c}{c_0} - \frac{d}{d_0} \right)^{n+1} = 0, \quad 0 < \mu < 1, \quad n \geq 1.$$

( $k(c, d) = 0$  are parabolic functional spline curves !)

For  $n = 1$  we get a pencil of conics which contains the ellipses  $(1 - \frac{c}{c_0})^2 + (1 - \frac{d}{d_0})^2 = 1$ . Hoffmann and Hopcroft used this pencil of conics for their *potential method*.

Rockwood and Owen used the *super ellipses*

$$k(c, d) = 1 - \left( 1 - \frac{c}{c_0} \right)^{n+1} - \left( 1 - \frac{d}{d_0} \right)^{n+1} = 0, \quad n \geq 1.$$

for blending surfaces.

**Example 11.4** (*elliptic functional splines*)

a)  $f_1(x, y, z) = x^2 + z^2 - 1.5^2 = 0$  (cylinder)

$f_2(x, y, z) = y^2 + z^2 - 2.3^2 = 0$  (cylinder).

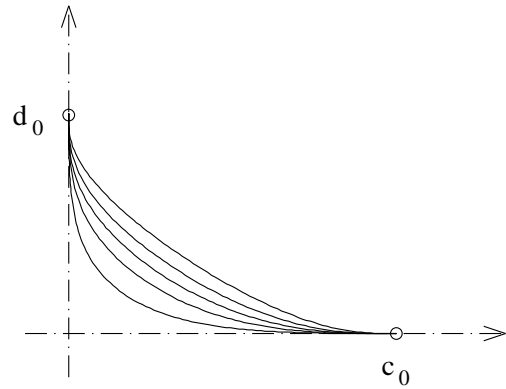
The elliptic functional spline  $(1 - \mu) \frac{f_1 f_2}{c_0 d_0} - \mu \left( 1 - \frac{f_1}{c_0} - \frac{f_2}{d_0} \right)^{n+1} = 0$  has  $G^n$ -contact to the cylinders.

For Figure 5 (left) the parameters are:  $n = 2, \mu = 0.2, c_0 = 2.4, d_0 = 6$ .

b)  $f_1(x, y, z) = (x - 2)^2 + y^2 + z^2 - 4 = 0$  (sphere)

$f_2(x, y, z) = (x^2 + y^2 + z^2 + 2^2 - 0.8^2)^2 - 16(x^2 + y^2) = 0$  (torus).

For Figure 5 (right) the parameters are:  $n = 2, \mu = 0.2, c_0 = 2, d_0 = 8$ .



$$(1 - \mu) \frac{cd}{c_0 d_0} - \mu \left(1 - \frac{c}{c_0} - \frac{d}{d_0}\right)^3 = 0, \quad \mu = 0.1, \dots, 0.82$$

Figure 11.5: Parabolic functional splines as correlationcurves for elliptic functional splines

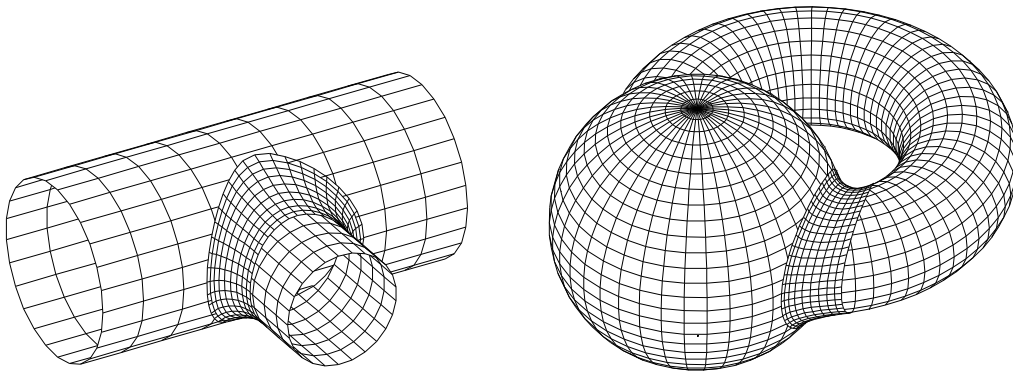


Figure 11.6:  $G^2$  elliptic functional spline

For **displaying** an elliptic functional spline we generate a net of quadrangles:

- Step 1: Determine  $n_1$  points  $(c_1, d_1), \dots$  of the correlation curve  $\Gamma$ .
- Step 2: Calculate  $n_2$  points  $P_{11}, P_{12}, \dots$  of the intersection curve  $f_1 = c_1, f_2 = d_1$  using the tracing algorithm for intersection curves (see Section 8.2.1).
- Step 3: Determine point  $P_{2i}$  to starting point  $P_{1i}$  (of the first curve) on the (second) curve  $f_1 = c_2, f_2 = d_2$  by procedure **surfacepoint** of the tracing algorithm.

a.s.o.

**Enhanced elliptic functional splines:**

Blending two surfaces  $f_1 = 0$  and  $f_2 = 0$  by elliptic functional splines yields solutions with curves of contact that are intersection curves  $f_1 = 0, f_2 = d_0$  and  $f_1 = c_0, f_2 = 0$ . Hence the pencils  $f_1 = c$  and  $f_2 = d$  have essential influence on the curves of contact. If one decides to have the intersection curves  $f_1 = 0, f_{22} = 0$  and  $f_{11} = 0, f_2 = 0$  as curves of contact for some additional surfaces  $f_{11} = 0$  and  $f_{22} = 0$ , one should use the following pencils of surfaces

$$(1 - c)f_1 + cf_{11} = 0, \quad 0 \leq c \leq 1, \quad (1 - d)f_2 + df_{22} = 0, \quad 0 \leq d \leq 1.$$

**Example 11.5** (enhanced elliptic functional splines)

a)  $f_1(x, y, z) = x^2 + z^2 - r^2 = 0$ ,  $r = 2$  (cylinder)

$f_2(x, y, z) = y^2 + z^2 - r^2 = 0$  (cylinder).

The cylinders are embedded into the following pencils of cylinders:

$(1 - c)f_1 + cf_{11} = 0$  which are the levelsets of  $\tilde{f}_1 := \frac{f_1}{f_1 - f_{11}} = c$

$(1 - d)f_2 + df_{22} = 0$  which are levelsets of  $\tilde{f}_2 = \frac{f_2}{f_2 - f_{22}} = d$ .

With  $f_{11}(x, y, z) = \frac{x^2}{(2r)^2} + \frac{z^2}{r^2} - 1 = 0$ ,  $f_{22}(x, y, z) = \frac{y^2}{(2r)^2} + \frac{z^2}{r^2} - 1 = 0$

the elliptic functional spline  $(1 - \mu)\frac{\tilde{f}_1\tilde{f}_2}{c_0d_0} - \mu(1 - \frac{\tilde{f}_1}{c_0} - \frac{\tilde{f}_2}{d_0})^{n+1} = 0$  has  $G^n$ -contact to the cylinders.

For Figure 11.7a) the parameters are:  $n = 2, \mu = 0.2, c_0 = 1, d_0 = 1$ .

b)  $f_1(x, y, z) = (x - 2)^2 + y^2 + z^2 - 4 = 0$  (sphere)

$f_{11}(x, y, z) = y - \tan(75^\circ)x = 0$  (plane)

$\tilde{f}_1(x, y, z) = (1 - c)f_1 + cf_{11} = 0$ ,

$f_2(x, y, z) = (x^2 + y^2 + z^2 + 2^2 - 0.8^2)^2 - 16(x^2 + y^2) = 0$  (torus),

$\tilde{f}_2(x, y, z) = f_2$ .

For Figure 11.7b) the parameters are:  $n = 2, \mu = 0.2, c_0 = 1, d_0 = 10$ .

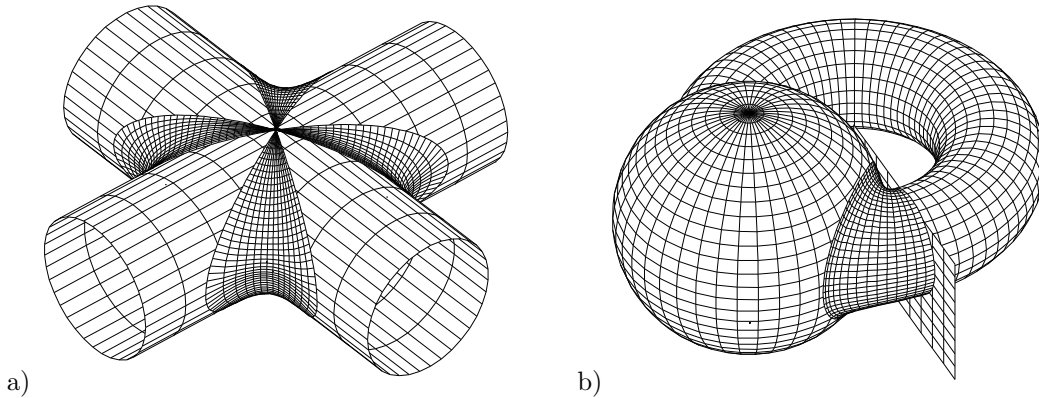


Figure 11.7: Enhanced  $G^2$  elliptic functional splines

**Remark:**

Using symmetric parabolic functional splines for blending two intersecting cylinders (see example above) we get a *balance* as design parameter. Elliptic functional splines provide the more interesting design parameter  $\mu$  which is a *thumbweight*.

**Remark:**

An elliptic functional spline surface can be considered as the collection of the intersection curves  $f_1 = c, f_2 = d$  for points  $(c, d)$  of its correlation curve  $k(c, d) = 0, 0 \leq c \leq c_0, 0 \leq d \leq d_0$ . This formulation allows a simple generalization to pairs of parametric or implicit surfaces with parametric or implicit correlation curves.

**Remark:**

All implicit blending methods can be applied to parametric curves and surfaces too, if one uses the *normalform* introduced in Section 5.4.

Figure 11.8a) shows elliptic functional spline blending surfaces between two Bézier patches and between one Bézier patch and an implicit surface (cylinder).

Figure 11.8b) shows enhanced elliptic functional splines. Three curves of contact are plane sections the fourth is the intersection of one Bézier patch with a quartic cylinder.

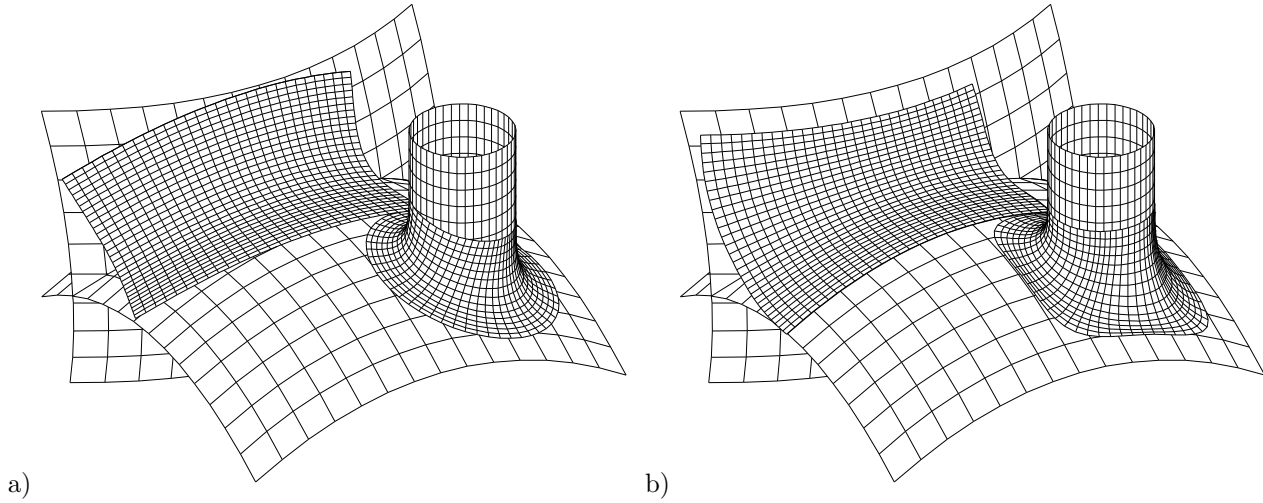


Figure 11.8:  $G^2$  elliptic functional splines between a Bézier patch and a cylinder/Bézier patch (right: enhanced)

## 11.3 Parametric $G^n$ -blending curves and surfaces

### 11.3.1 $G^m$ -blending of parametric curves

A simple  $G^0$ -blending method for curves which suggests itself works as follows:

Let  $\Gamma_1 : \mathbf{x} = \mathbf{c}_1(t)$ ,  $t \in [0, 1]$  and  $\Gamma_2 : \mathbf{x} = \mathbf{c}_2(t)$ ,  $t \in [0, 1]$  be two  $C^n$ -continuous planar or spatial curves, then  $\Gamma : \mathbf{x} = (1-t)\mathbf{c}_1(t) + t\mathbf{c}_2(t)$ ,  $t \in [0, 1]$  is a  $G^0$ -blending curve of  $\Gamma_1$  and  $\Gamma_2$  with  $\mathbf{c}_1(0)$  and  $\mathbf{c}_2(1)$  as points of contact. In order to smooth the contact of  $\Gamma$  to the curves  $\Gamma_1, \Gamma_2$  we generalize this simple blending method by replacing the linear function  $t$  by a suitable function  $f(t)$ .

**Theorem 11.4** Let  $\Gamma_1 : \mathbf{x} = \mathbf{c}_1(t)$ ,  $t \in [0, 1]$  and  $\Gamma_2 : \mathbf{x} = \mathbf{c}_2(t)$ ,  $t \in [0, 1]$  be two regular  $C^n$ -continuous planar or spatial curves (base curves) and  $f(t), t \in [0, 1]$  a  $C^n$ -continuous real function (blending function) with the following properties:

$$f(0) = 0, f(1) = 1, f^{(k)}(0) = f^{(k)}(1) = 0 \text{ for } k = 1, \dots, n.$$

Then the curve (blending curve)

$$\Gamma : \mathbf{x} = \mathbf{c}(t) := (1 - f(t))\mathbf{c}_1(t) + f(t)\mathbf{c}_2(t), \quad t \in [0, 1]$$

has  $C^n$ -contact to the curves  $\Gamma_1, \Gamma_2$  at the points  $\mathbf{c}_1(0)$  and  $\mathbf{c}_2(1)$ , .i.e.  $\mathbf{c}^{(k)}(0) = \mathbf{c}_1^{(k)}(0)$ ,  $\mathbf{c}^{(k)}(1) = \mathbf{c}_2^{(k)}(1)$  for  $k = 0, 1, \dots, n$ .

**Proof:** The derivatives of  $\mathbf{c}$  are:

$$\begin{aligned} \dot{\mathbf{c}} &= -f' \mathbf{c}_1 + (1 - f) \dot{\mathbf{c}}_1 + f' \mathbf{c}_2 + f \dot{\mathbf{c}}_2, \\ \ddot{\mathbf{c}} &= -f'' \mathbf{c}_1 - 2f' \dot{\mathbf{c}}_1 + (1 - f) \ddot{\mathbf{c}}_1 + f'' \mathbf{c}_2 + 2f' \dot{\mathbf{c}}_2 + f \ddot{\mathbf{c}}_2, \\ &\dots \end{aligned}$$

From the properties of function  $f$  we get  $\mathbf{c}^{(k)}(0) = \mathbf{c}_1^{(k)}(0)$ ,  $\mathbf{c}^{(k)}(1) = \mathbf{c}_2^{(k)}(1)$  for  $k = 0, 1, \dots, n$ .  $\square$

**Remark:**

- a) For the application of Theorem 11.4 to the examples below there is need of parameter transformations to meet the precondition " $t \in [0, 1]^n$ ". So the blend curve  $\Gamma$  will have only  $G^n$ - (instead of  $C^n$ -) continuous contact to the base curves.
- b) Due to the definition (see Theorem 1) a blending curve is contained within the *convex hull* of the arcs to be blended.

Though there exist polynomial blending functions we choose the following *rational*  $G^n$ -blending functions:

$$f_n(t) = \frac{t^{n+1}}{t^{n+1} + (1-t)^{n+1}}, \quad t \in [0, 1], \quad n \geq 0.$$

Figure 11.9 a) shows graphs of the blending functions  $f_0, f_1$  and  $f_2$ .

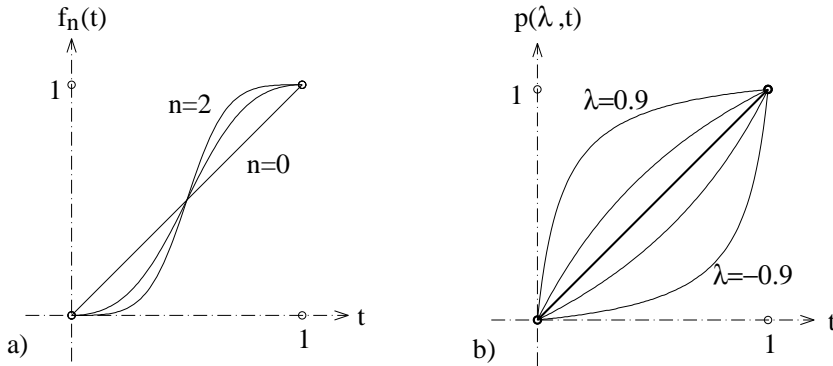


Figure 11.9: Left: graphs of blending function  $f_n(t)$  for  $n = 0, 1, 2$  and right: graphs of the parameter transformations  $p(\lambda, t)$  for  $\lambda = -0.9, -0.5, 0, 0.5, 0.9$

In order to get design parameters we use the following family of rational parameter transformations:

$$p(\lambda, t) := \begin{cases} \frac{t}{1 - \lambda + \lambda t} & , 0 \leq \lambda < 1, \\ \frac{t(1 + \lambda)}{1 + \lambda t} & , -1 < \lambda \leq 0. \end{cases}$$

Figure 11.9b) shows the graphs of functions  $p(\lambda, \cdot)$  for  $\lambda = -0.9, -0.5, 0, 0.5, 0.9$ .

(Recognize:  $p(0, t) = t$  !) We apply these parameter transformations as follows:

$$\Gamma : \mathbf{x} = \mathbf{c}(t) := (1 - f_n(t)) \mathbf{c}_1(p(\lambda_1, t)) + f_n(t) \mathbf{c}_2(p(-\lambda_2, t)), \quad t \in [0, 1].$$

The negative sign for  $\lambda_2$  ensures that for raising  $\lambda_i$  the blending curve comes closer to curve  $\Gamma_i$  near the point of contact. The following simple examples show the effect of the design parameters  $\lambda_1, \lambda_2$  and the effect of reversing the orientation of one arc.

**Example 11.6** *parallel line segments*

Let  $\Gamma_1$  and  $\Gamma_2$  be the line segments



$\Gamma_1 : \mathbf{c}_1(t) := (t, 0)^\top, t \in [0, 1], \quad \Gamma_2 : \mathbf{c}_2(t) := (t, 1)^\top, t \in [0, 1].$  Any curve

$$\Gamma : \mathbf{x} = \mathbf{c}(t) := (1 - f_n(t)) \mathbf{c}_1(p(\lambda_1, t)) + f_n(t) \mathbf{c}_2(p(-\lambda_2, t)), \quad t \in [0, 1],$$

for arbitrary pairs  $\lambda_1, \lambda_2$  with  $-1 < \lambda_1, \lambda_2 < 1$ , is a  $G^n$ -blending curve of  $\Gamma_1$  and  $\Gamma_2$ .

Figure 11.10 shows  $G^2$ -blending curves for different pairs of  $\lambda_1, \lambda_2$ .

Figure 11.11 shows the case in which the orientation of the upper segment is reversed, i.e.  $\mathbf{c}_2(t) := (1 - t, 1)^\top$ .

In order to visualize the used parameter transformations points on  $\Gamma_1$  and  $\Gamma_2$  with the same parameter are connected.

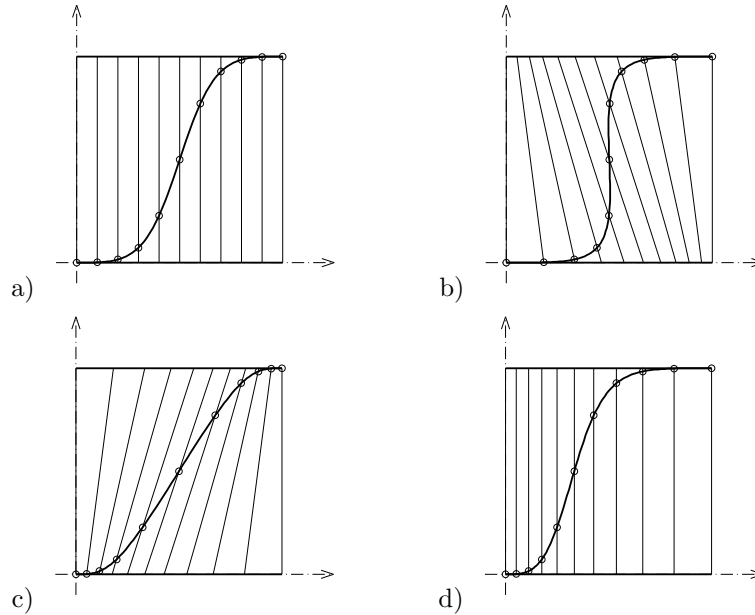


Figure 11.10:  $G^2$ -blending curves of two parallel line segments for parameters a)  $\lambda_1 = \lambda_2 = 0$ , b)  $\lambda_1 = \lambda_2 = 0.5$  c)  $\lambda_1 = \lambda_2 = -0.5$  and d)  $\lambda_1 = -0.5, \lambda_2 = 0.5$

Figure 11.12 shows  $G^2$ -blending curves of two arcs on two Bézier curves with different design parameters  $\lambda_1, \lambda_2$ . The arcs actually used for blending are bounded by points.

**Example 11.7** space curves

Figure 11.13 shows a  $G^3$ -blending curve between an arc on a helix and an arc on a horizontal line together with the corresponding pipe surfaces. Because the blending curve has  $G^3$ -contact to the helix and the line, the whole pipe surface is  $G^2$ -continuous.

Helix:  $\mathbf{x}(t) = (r_0 \cos(t), r_0 \sin(t), 5t), r_0 = 50, t_1 = \pi + 0.2, t_2 = \pi + 0.6.$

Line:  $\mathbf{x}(t) = (1 - t)\mathbf{p}_1 + t\mathbf{p}_2, \quad \mathbf{p}_1 = (r_0, 0, 40), \mathbf{p}_2 = (r_0, 100, 40), t_1 = 0.2, t_2 = 0.5.$

Radius of the pipe:  $r = 10.$  Design parameter  $\mu = 0.5.$

The pipe surface is defined as ringed surface  $\mathbf{x} = \mathbf{c}(t) + r \cos(s)\mathbf{a}(t) + r \sin(s)\mathbf{b}(t),$  where  $\mathbf{x} = \mathbf{c}(t)$  is the directrix curve and  $\{\dot{\mathbf{c}}(t), \mathbf{a}(t), \mathbf{b}(t)\}$  is a right-handed orthonormal system. One can prove that the pipe surface of this example is even  $G^3,$  too (see Section 5.6).

**11.3.2  $G^n$ -blending of parametric surfaces**

The  $G^n$ -blending method introduced for curves can be extended to surfaces. As for curves the flexibility of the method depends essentially on suitable parameter transformations.

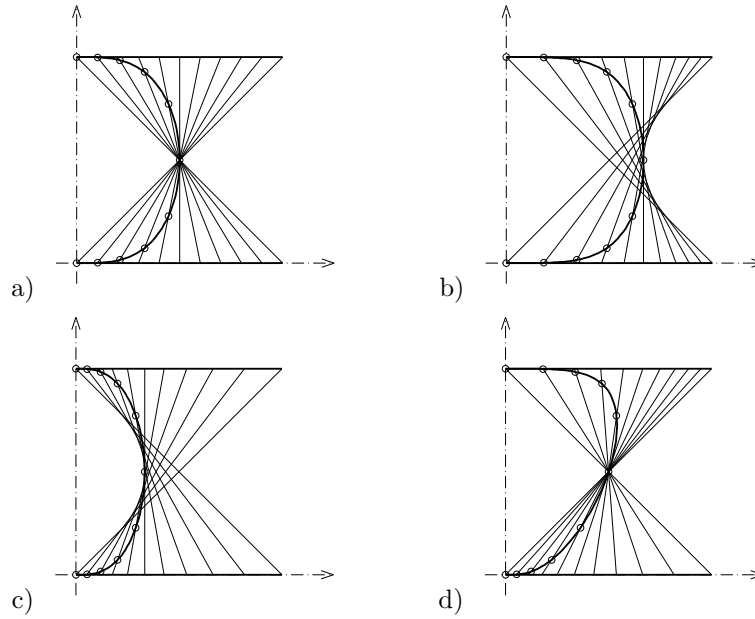


Figure 11.11: Examples of the previous figure with reversed orientation of the upper line

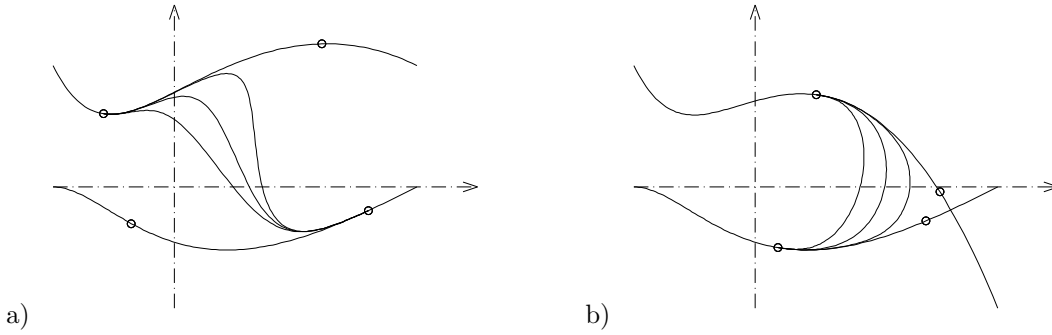


Figure 11.12:  $G^2$ -bleeding of Bézier curves with parameters a)  $\lambda_1 = 0, 0.5, 0.8$  and  $\lambda_2 = 0$ , b)  $\lambda_1 = \lambda_2 = 0, 0.5, 0.8$

**Theorem 11.5** Let  $\Phi_1 : \mathbf{x} = \mathbf{S}_1(s, t)$ ,  $s \in [s_1, s_2]$ ,  $t \in [0, 1]$  and  $\Phi_2 : \mathbf{x} = \mathbf{S}_2(s, t)$ ,  $s \in [s_1, s_2]$ ,  $t \in [0, 1]$  be two regular  $C^n$ -continuous surface patches (base surfaces) and  $f(t)$ ,  $t \in [0, 1]$  a  $C^n$ -continuous real function (blending function) with the following properties:

$$f(0) = 0, \quad f(1) = 1, \quad f^{(k)}(0) = f^{(k)}(1) = 0 \quad \text{for } k = 1, \dots, n.$$

Then the surface patch (blending surface)

$$\Phi : \mathbf{x} = \mathbf{S}(s, t) := (1 - f(t)) \mathbf{S}_1(s, t) + f(t) \mathbf{S}_2(s, t), \quad s \in [s_1, s_2], \quad t \in [0, 1]$$

has  $C^n$ -contact to the surface patch  $\Phi_1$  along the curve  $\mathbf{S}_1(s, 0)$ ,  $s \in [s_1, s_2]$  and to  $\Phi_2$  along the curve  $\mathbf{S}_2(s, 1)$ ,  $s \in [s_1, s_2]$ .

The proof is straight forward differentiating analogously to the curve case.

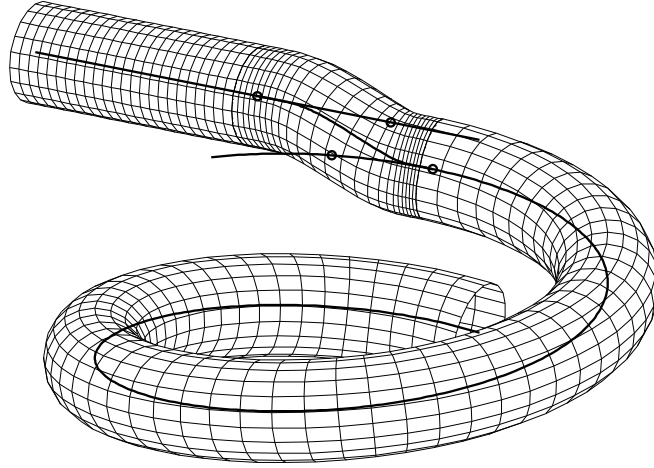


Figure 11.13:  $G^3$ -blending curve between a helix and a line and a corresponding  $G^2$ -continuous pipe surface

As blending functions for the examples below we use the functions  $f_n$  introduced above.

**Remarks:**

a) The curves  $\mathbf{S}_1(s, 0)$  and  $\mathbf{S}_2(s, 1)$   $s \in [s_1, s_2]$  are the *contact curves* between the base surfaces and the blend surface.

The curves  $\mathbf{S}_1(s, 1)$  and  $\mathbf{S}_2(s, 0)$   $s \in [s_1, s_2]$  are called *auxiliary curves*.

Contact curves and auxiliary curves are boundaries of the surface patches to be blended.

b) Curves  $\mathbf{x} = \mathbf{S}(s = \text{const}, t)$  are *profile curves* which locally define the shape of the blend surface.

**Remark:**

For the application of Theorem 11.5 to the examples below there is need of parameter transformations to meet the precondition " $s \in [s_1, s_2], t \in [0, 1]$ ". So the blend surface  $\mathbf{S}$  will have only  $G^n$ - (instead of  $C^n$ -) contact to the base surfaces.

**Example 11.8 Bézier patches**

Let  $\Phi_1$  and  $\Phi_2$  be two tensor product Bézier patches with parametric representations

$$\Phi_1: \quad \mathbf{x} = \mathbf{B}_1(u, v) = (10v - 5, 10u - 5, 6(u - u^2 + v - v^2)),$$

$$\Phi_2: \quad \mathbf{x} = \mathbf{B}_2(u, v) = (6(u - u^2 + v - v^2), 10u - 5, 10v - 5).$$

The boundary curves on  $\Phi_1$  are:

curve of contact:  $v = v_{11} = 0.5$ , auxiliary curve:  $v = v_{12} = 0.3$ ,

boundary curves:  $u = 0$  and  $u = 0.75$ ,

The boundary curves on  $\Phi_2$  are:

curve of contact:  $v = v_{21} = 0.9$ , auxiliary curve:  $v = v_{22}(u) = 0.8(1 - (0.5 - u)^2)$ ,

boundary curves:  $u = 0$  and  $u = 0.75$ .

In order to meet the preconditions of Theorem 2 we use the following parameter transformations:

$$\mathbf{S}_1(s, t) := \mathbf{B}_1(s, v_{11} + (v_{12} - v_{11})t), \quad t \in [0, 1]$$

$$\mathbf{S}_2(s, t) := \mathbf{B}_2(s, v_{21} + (v_{22}(s) - v_{21})t), \quad t \in [0, 1]$$

$$s \in [0, 0.75].$$

Hence

$$\Phi: \mathbf{x} = \mathbf{S}(s, t) := (1 - f_n(t)) \mathbf{S}_1(s, t) + f_n(t) \mathbf{S}_2(s, t), \quad s \in [0, 0.75], \quad t \in [0, 1]$$

is a  $G^n$ -blending surface.

( $f_n$  is a  $G^n$ -blending function defined above.)

Figure 11.14 (left) shows the base surfaces, the curves of contact (thick) and the auxiliary curves (dashed). The right hand picture shows a  $G^2$ -blending surface.

In order to get thumbweight parameters we apply parameter transformation  $p(\lambda, \cdot)$  on parameter  $t$  before blending:

$$\Phi : \mathbf{x} = \mathbf{S}(s, t) := (1 - f_n(t)) \mathbf{S}_1(s, p(\lambda_1, t)) + f_n(t) \mathbf{S}_2(s, p(-\lambda_2, t)), \quad s \in [0, 0.75], \quad t \in [0, 1],$$

with the thumbweights  $\lambda_1, \lambda_2$ .

Figure 11.15 shows blending surfaces of the given Bézier patches with thumbweights  $\lambda_1 = \lambda_2 = 0.6$  (left), and  $\lambda_1 = 0, \lambda_2 = 0.8$  (right). Increasing  $\lambda_i \in [0, 1]$  brings the blend surface closer to the surfaces  $\Phi_i$ .

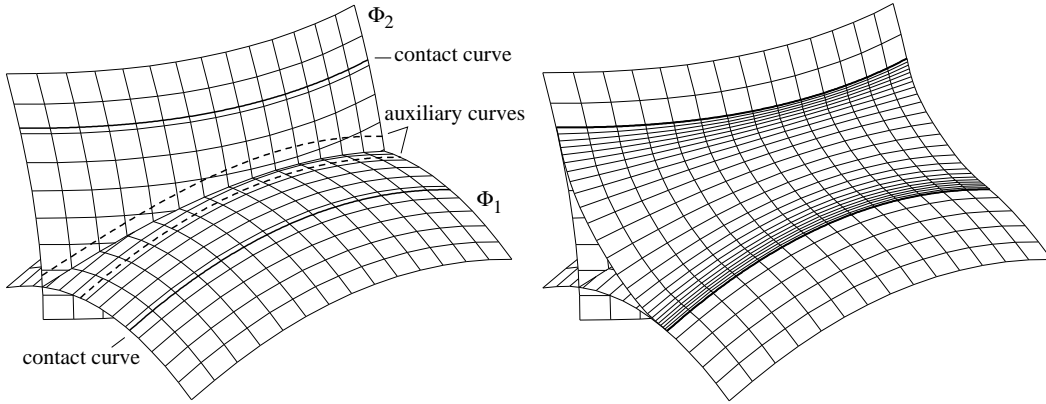


Figure 11.14:  $G^2$ -blending surface between two tensor product Bézier surfaces thumbweights  $\lambda_1 = \lambda_2 = 0$  (right) and the boundaries of the patches used for blending (left)

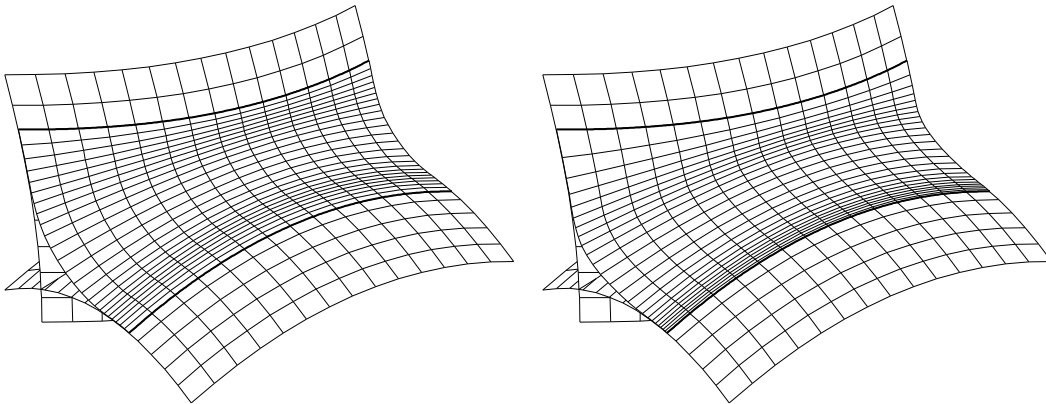


Figure 11.15:  $G^2$ -blending surface between two tensor product Bézier surfaces with thumbweights  $\lambda_1 = \lambda_2 = 0.6$  (left) and  $\lambda_1 = 0, \lambda_2 = 0.8$  (right)

**Example 11.9** *cylinder and plane*

Let be  $\Phi_1$  a cylinder and  $\Phi_2$  an inclined plane.

$$\Phi_1 : \quad \mathbf{x} = (\cos(u), \sin(u), v),$$

$$\Phi_2 : \quad \mathbf{x} = (v \cos(u), v \sin(u), cv \cos(u)), \quad c = 0.4.$$

The boundary curves on  $\Phi_1$  are:

curve of contact:  $v = v_{11} = 1.5$ , auxiliary curve:  $v = v_{12}(u) = 0.2 + c \cos u$ ,  
 The boundary curves on  $\Phi_2$  are:

curve of contact:  $v = v_{21} = 1.2$ , auxiliary curve:  $v = v_{22} = 2$ .

Figure 11.16 (left) shows the curves of contact and auxiliary curves for a  $G^2$ -blending surface (right). The design parameters are balance  $\mu = 0.5$  and thumbweight  $\lambda := 0.5$ .

Figure 11.17 shows a  $G^2$ -blending surface with a changed curve of contact:  $v = v_{11}(u) = 1.5 + 0.2 \cos(4u)$ .

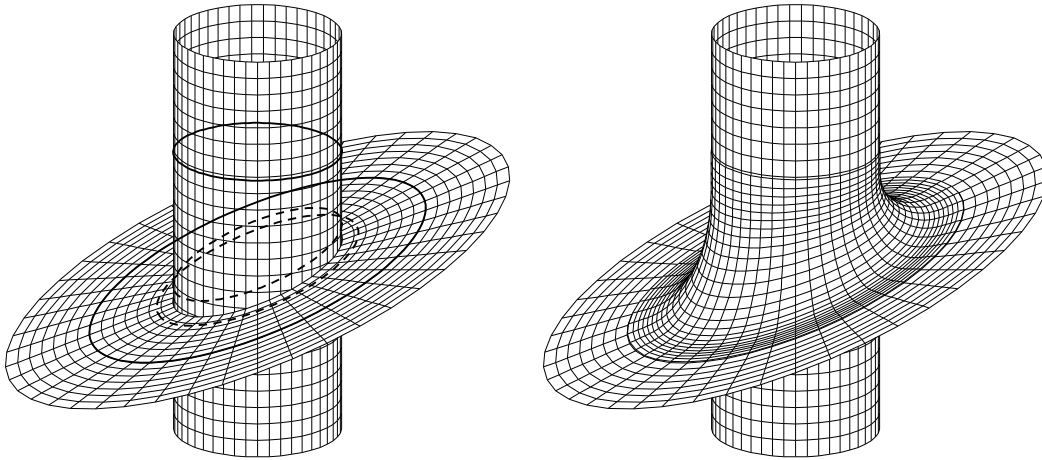


Figure 11.16:  $G^2$ -blending surface between a cylinder and a plane (right) and the boundaries of the patches used for blending (left)

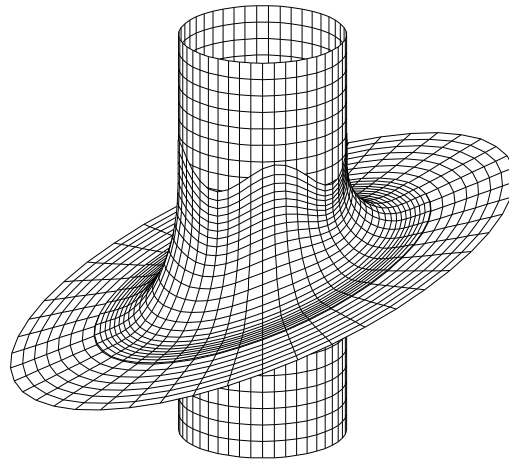


Figure 11.17: Further  $G^2$ -blending surface between a cylinder and a plane

## 11.4 Blending with rolling ball contact curves

The blending method:

1. Let be  $\Phi_1, \Phi_2$  two parametric (or implicit) surfaces. We assume that the offset surfaces for distances 0..1 exist and the intersection curve of the offsets are  $C^2$ -continuous for any distance  $t \in [0, 1]$ .
2. Any intersection curve of pairs of offsets (corresponding to the same distance  $t$ ) can be numerically parametrized by a suitable common Bézier or B-spline curve  $\Gamma : \mathbf{c}(s), s \in [0, 1]$  (cf. Section 4.7.3, see Figure 11.18). Hence we get a (numerical) parameterization  $\mathbf{V}(s, t), s, t \in [0, 1]$  of the Voronoi surface (points which have equal distance to both the surfaces, see Figure 11.18) which is  $C^{n-1}$  if  $\Phi_1, \Phi_2, \Gamma$  are  $C^n$ .
3. The parameterization  $\mathbf{V}(s, t)$  is used for a (numerical) parameterization (see Section 5.10.3) of the given surfaces  $\Phi_1, \Phi_2$  which we assign by  $\mathbf{S}_1(s, t), \mathbf{S}_2(s, t), s, t \in [0, 1]$ .
4. We apply the  $C^n$ -blending method out of Section 11.3.2 and get the blending surface

$$\Psi : \mathbf{x} = \mathbf{S}(s, t) := (1 - f(t)) \mathbf{S}_1(s, 1 - t) + f(t) \mathbf{S}_2(s, t), \quad s, t \in [0, 1]$$

with blending function  $f(t)$  given in Section 11.3.2. The curves of contact  $\mathbf{S}_1(s, 1), \mathbf{S}_2(s, 1), s \in [0, 1]$  can be considered as curves of contact of a rolling ball of *fixed* radius  $r = 1$  (see Figure 11.19). The curve containing the midpoints of the rolling balls, the *spine curve*, is the curve  $\mathbf{V}(s, 1)$ . The auxiliary curves coincide with the intersection curve of given surfaces  $\Phi_1, \Phi_2$ .

The **algorithm** for determining point  $\mathbf{S}(s, t)$  of blending surface  $\Psi$ :

1. Calculate  $\mathbf{c}(s)$  (point of the auxiliary curve)
2. Determine foot point  $\mathbf{V}(s, t)$  of  $\mathbf{c}(s)$  on the intersection curve of the offset surfaces belonging to distance  $t$ .  
Determine foot point  $\mathbf{V}(s, 1-t)$  of  $\mathbf{c}(s)$  on the intersection curve of the offset surfaces belonging to distance  $1 - t$ .
3. Determine foot point  $\mathbf{S}_1(s, 1 - t)$  of  $\mathbf{V}(s, 1 - t)$  on surface  $\Phi_1$ .  
Determine foot point  $\mathbf{S}_2(s, t)$  of  $\mathbf{V}(s, t)$  on surface  $\Phi_2$ .
4. **Point** on blending surface:

$$\mathbf{S}(s, t) = (1 - f(t)) \mathbf{S}_1(s, 1 - t) + f(t) \mathbf{S}_2(s, t),$$

where  $f(t) = \frac{t^{n+1}}{t^{n+1} + (1 - t)^{n+1}}$ .

5. For the examples shown below the parameter transformations  $p_1(t), p_2(t)$  are used:

$$\begin{aligned} \mathbf{S}(s, t) &= f(t) \mathbf{S}_1(s, 1 - p_1(t)) + (1 - f(t)) \mathbf{S}_2(s, p_2(t)), \\ &= f(t) \mathbf{S}_1(s, p_2(1 - t)) + (1 - f(t)) \mathbf{S}_2(s, p_2(t)), \end{aligned}$$

where  $p_1(t) := \frac{t}{1 - \lambda + \lambda t}$ , and  $p_2(t) := \frac{t(1 - \lambda)}{1 - \lambda t}$   $0 \leq \lambda < 1$ .

**Example 11.10** Figure 11.18 shows the numerical parameterization of the Voronoi surface of the two Bézier patches given in Example 11.8. The auxiliary curve is the Bézier curve of degree 4 with control points:

$$(-3.3, -4.9, 2), (-2.5, -3.5, 2.7), (-2, -2, 3.4), (-2, 0, 3.6), (-2.5, 2.4, 3).$$

The rolling ball has radius  $r = 2$ .

Figure 11.19 shows the (numerical) re-parameterization of the two Bézier patches by the Voronoi surface and a resulting  $G^2$ -blending surface.

Figure 11.20 shows two further examples. The right one is an application of the method on an example involving the **implicit** surface  $x^4 + y^4 + z^4 = 1$ .

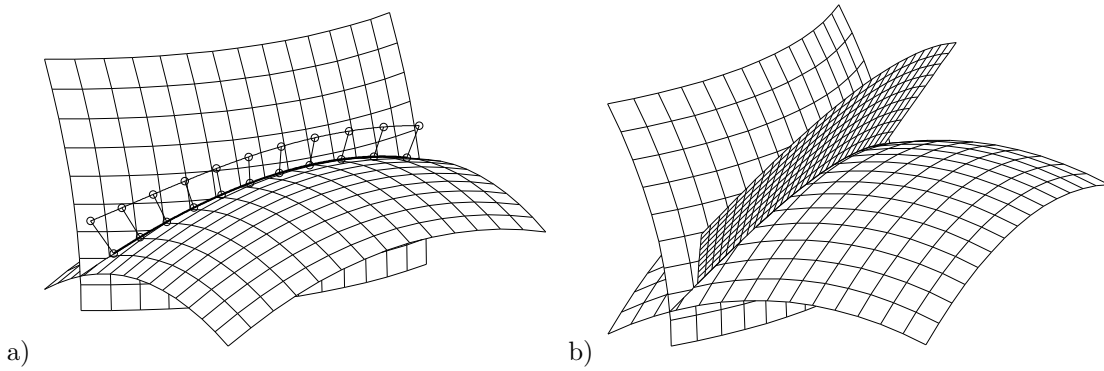


Figure 11.18: Parameterization of the intersection curve (left) and the Voronoi surface (right) of two tensor product Bézier surfaces

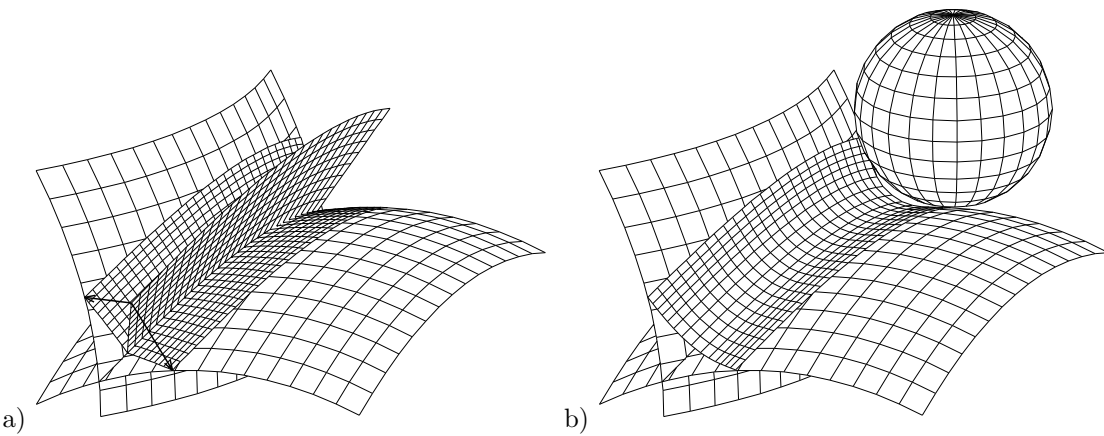


Figure 11.19: Re-parameterization (left) and  $G^2$ -blending surface (right) of two tensor product Bézier surfaces with constant radius rolling ball contact curves

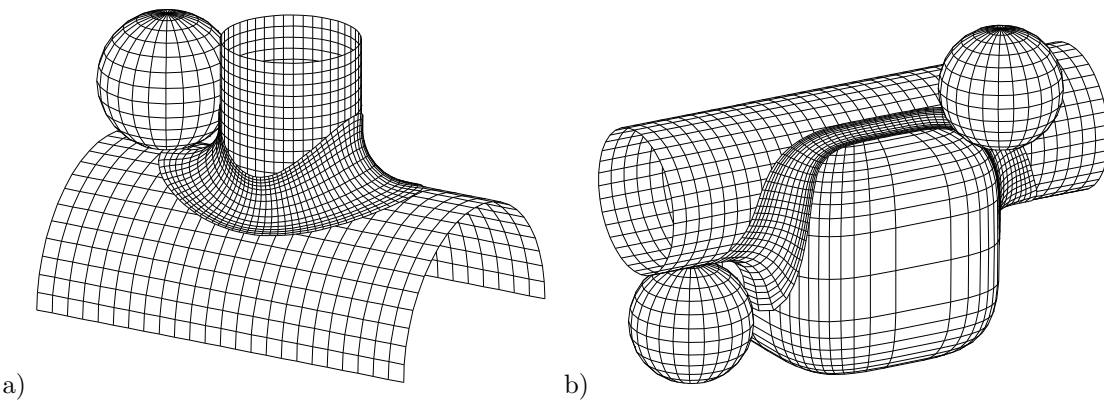


Figure 11.20:  $G^2$ -blending of two cylinders (left) and a further example (right) with constant radius rolling ball contact curves





## Chapter 12

# PROPERTIES OF BÉZIER CURVES

### 12.1 Properties of the Bernstein polynomes

We extend the definition

- a) of the Bernstein polynomes  $B_i^n(t) := \binom{n}{i} t^i (1-t)^{n-i}$ ,  $0 \leq i \leq n$ :  $B_i^n(t) = 0$ ,  $i \notin \{0, 1, \dots, n\}$ .  
b) of the binomial coefficients:  $\binom{n}{i} = 0$ ,  $i \notin \{0, 1, \dots, n\}$ .

(1) Relation Bernstein- and monom-basis:

$$t^i = \sum_{j=i}^n \frac{\binom{j}{i}}{\binom{n}{i}} B_j^n(t), \quad B_i^n(t) = \sum_{j=i}^n (-1)^{j-i} \binom{n}{j} \binom{j}{i} t^j,$$

(2) Recursion:  $B_i^n(t) = (1-t)B_{i-1}^{n-1}(t) + tB_{i+1}^{n-1}(t)$ ,

(3) Scaling:  $B_i^n(ct) = \sum_{j=0}^n B_i^j(c) B_j^n(t)$ ,

(4) Derivative:  $\frac{d}{dt} B_i^n(t) = n(B_{i-1}^{n-1}(t) - B_i^{n-1}(t))$ ,

(5) Product:  $B_i^n(t) B_j^n(t) = \frac{\binom{m}{i} \binom{n}{j}}{\binom{m+n}{i+j}} B_{i+j}^{m+n}(t)$ .

### 12.2 The Casteljau-algorithm

For polygon  $\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_n \in \mathbb{R}^2$  (or  $\mathbb{R}^3$ ) and  $t \in \mathbb{R}$  we define recursively for any  $r = 1, \dots, n$  the following polygon

$$\mathbf{b}_i^r(t) = (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t), \quad i = 0, \dots, n-r.$$

with  $\mathbf{b}_i^0(t) = \mathbf{b}_i$ .

The polygon for  $r = 0$  agrees with the starting polygon, for  $r = n$  we get a point.

From the recursion property of the Bernstein polynomes (s. 12.1) we get

$$\mathbf{b}_i^r(t) = \sum_{j=0}^r \mathbf{b}_{i+j} B_j^r(t), \quad r = 0, \dots, n, \quad i = 0, \dots, n-r$$

(Prove it by induction on  $r$ .)  
Hence

$$\mathbf{b}_0^n(t) = \sum_{j=0}^n \mathbf{b}_j B_j^n(t)$$

is the Bézier curve with control polygon  $\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_n$ . This method for determining a point by repeated linear interpolation is called **Casteljau–algorithm**.

Figure 12.1 shows how to generate a point for a  $t \in \mathbb{R}$  from a control polygon by repeated linear interpolation for  $n = 3$ .

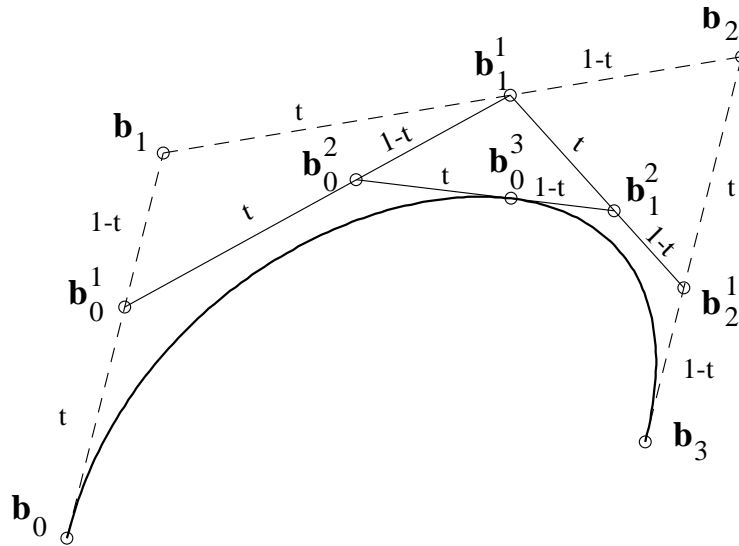


Figure 12.1: Casteljau–algorithm for  $n=3$

schematically:

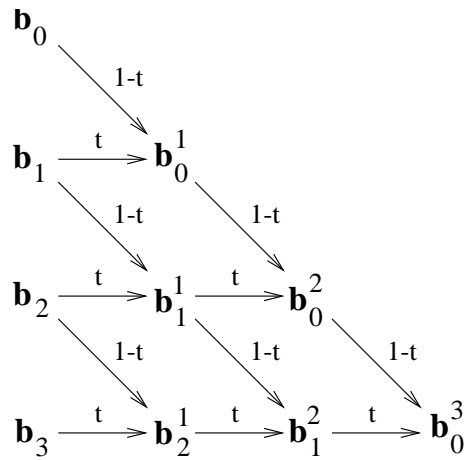


Figure 12.2: Scheme of the Casteljau–algorithm

## 12.3 Derivative of a Bézier curve

Using the derivations of the Bernstein polynomials (s.12.1) we get the first derivative of a Bézier curve  $\mathbf{b}(t) = \sum_{i=0}^n \mathbf{b}_i B_i^n(t)$ :

$$\frac{d}{dt} \mathbf{b}(t) = n \sum_{i=0}^{n-1} (\mathbf{b}_{i+1} - \mathbf{b}_i) B_i^{n-1}(t).$$

Hence the tangent vectors of a Bézier curve form a Bézier curve with control points  $\Delta \mathbf{b}_i := \mathbf{b}_{i+1} - \mathbf{b}_i$ . Especially:

$$\frac{d}{dt} \mathbf{b}(0) = n(\mathbf{b}_1 - \mathbf{b}_0) \quad \text{and} \quad \frac{d}{dt} \mathbf{b}(1) = n(\mathbf{b}_n - \mathbf{b}_{n-1}).$$

In order to rewrite the derivatives in a more impressive way we introduce the following difference operators

$$\Delta^r \mathbf{b}_i := \Delta^{r-1} \mathbf{b}_{i+1} - \Delta^{r-1} \mathbf{b}_i,$$

and get

$$\begin{aligned} \Delta^0 \mathbf{b}_i &= \mathbf{b}_i \\ \Delta^1 \mathbf{b}_i &= \mathbf{b}_{i+1} - \mathbf{b}_i \\ \Delta^2 \mathbf{b}_i &= \mathbf{b}_{i+2} - 2\mathbf{b}_{i+1} + \mathbf{b}_i \\ \Delta^3 \mathbf{b}_i &= \mathbf{b}_{i+3} - 3\mathbf{b}_{i+2} + 3\mathbf{b}_{i+1} - \mathbf{b}_i \\ &\dots = \dots \\ \Delta^r \mathbf{b}_i &= \sum_{j=0}^r \binom{r}{j} (-1)^{r-j} \mathbf{b}_{i+j} \end{aligned}$$

The  $r$ -th derivative of the Bézier curve  $\mathbf{b}(t) = \sum_{i=0}^n \mathbf{b}_i B_i^n(t)$  can be written as:

$$\frac{d^r}{dt^r} \mathbf{b}(t) = \frac{n!}{(n-r)!} \sum_{i=0}^{n-r} \Delta^r \mathbf{b}_i B_i^{n-r}(t)$$

Especially for  $t = 0$  and  $t = 1$  we get

$$\frac{d^r}{dt^r} \mathbf{b}(0) = \frac{n!}{(n-r)!} \Delta^r \mathbf{b}_0 \quad \text{and} \quad \frac{d^r}{dt^r} \mathbf{b}(1) = \frac{n!}{(n-r)!} \Delta^r \mathbf{b}_{n-r}$$

## 12.4 Degree elevation of a Bézier curve

An essential manipulation of a Bézier curve is the so called *degree elevation*, comparable with the addition of terms  $0t^{n+1}, 0t^{n+2}, \dots$  to a polynomial  $a_0 + a_1t + \dots + a_nt^n$ . The polynomial is not changed really but the formal degree is elevated. Analogously one rewrites a given Bézier curve  $\mathbf{b}(t) = \sum_{i=0}^n \mathbf{b}_i B_i^n(t)$  in the following way:  $\mathbf{b}(t) = \sum_{i=0}^{n+1} \mathbf{b}_i^{(1)} B_i^{n+1}(t)$  with suitable new control points  $\mathbf{b}_0^{(1)}, \dots, \mathbf{b}_{n+1}^{(1)}$ . In order to determine the new control points we multiply the original representation

by  $t + (1 - t)$ :

$$\begin{aligned}
 \mathbf{b}(t) &= \sum_{i=0}^n \mathbf{b}_i B_i^n(t)(t + (1 - t)) \\
 &= \sum_{i=0}^n \mathbf{b}_i \binom{n}{i} t^i (1 - t)^{n-i} (t + (1 - t)) \\
 &= \sum_{i=0}^n \mathbf{b}_i \binom{n}{i} t^{i+1} (1 - t)^{n-i} + \sum_{i=0}^n \mathbf{b}_i \binom{n}{i} t^i (1 - t)^{n-i+1} \\
 &= \sum_{i=1}^{n+1} \mathbf{b}_{i-1} \binom{n}{i-1} t^i (1 - t)^{n+1-i} + \sum_{i=0}^n \mathbf{b}_i \binom{n}{i} t^i (1 - t)^{n-i+1} \\
 &= \sum_{i=0}^{n+1} \left( \mathbf{b}_{i-1} \binom{n}{i-1} + \mathbf{b}_i \binom{n}{i} \right) t^i (1 - t)^{n+1-i} \\
 &= \sum_{i=0}^{n+1} \mathbf{b}_i^{(1)} \binom{n+1}{i} t^i (1 - t)^{n+1-i} = \sum_{i=0}^{n+1} \mathbf{b}_i^{(1)} B_i^{n+1}(t) \quad , \text{ with} \\
 \mathbf{b}_i^{(1)} &: = \frac{i}{n+1} \mathbf{b}_{i-1} + \left(1 - \frac{i}{n+1}\right) \mathbf{b}_i, \quad i = 0, \dots, n+1 \\
 &= \mathbf{b}_{i-1}^1 \binom{n+1-i}{n+1} \quad (\text{s. Casteljau-Algor.}) \quad i = 1, \dots, n
 \end{aligned}$$

The new control points are:  $\mathbf{b}_0, \mathbf{b}_0^1 \binom{n}{n+1}, \mathbf{b}_1^1 \binom{n-1}{n+1}, \dots, \mathbf{b}_{n-1}^1 \binom{1}{n+1}, \mathbf{b}_n$ .

**Essential properties** of the degree elevation:

- Repeated elevation yields an approximation of the Bézier curve by its control polygon.
- The growing number of control points gives more freedom for changing the curve.
- Several Bézier curves of different degrees can be represented by Bézier curves of the same degree which is important for tensor-product Bézier surfaces (see Chapter 13).

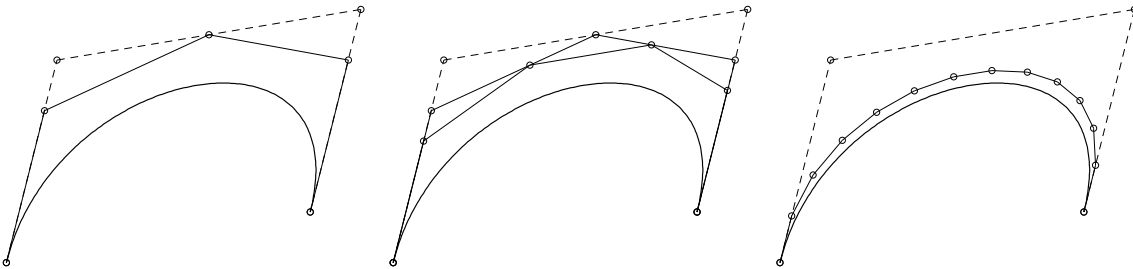


Figure 12.3: control polygons for 1-, 2- and 10-times degree elevation

## 12.5 Bézier spline curves

### 12.5.1 Division of a Bézier curve

Normally a Bézier curve  $\Gamma_0 : \mathbf{b}(t)$  is defined for  $0 \leq t \leq 1$ . Let be  $c \in (0, 1)$ . Then  $\Gamma_1 : \mathbf{b}(t)$  with  $0 \leq t \leq c$  is a segment of the given Bézier curve. We now want to represent this segment  $\Gamma_1$  as

Bézier curve  $\mathbf{c}(t) = \sum_{i=0}^n \mathbf{c}_i B_i^n(s)$  mit  $s \in [0, 1]$  of the same degree  $n$  and a suitable control polygon  $\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_n$ . For  $s := \frac{t}{c}$  we demand:

$$\sum_{r=0}^n \mathbf{c}_r B_r^n\left(\frac{t}{c}\right) = \sum_{i=0}^n \mathbf{b}_i B_i^n(t), \quad \text{for } t \in [0, c]$$

This is fulfilled for

$$\mathbf{c}_r = \mathbf{b}_0^r(c) \quad (\text{s. Casteljau-alg. and Figures 12.1 and 12.2})$$

because of

$$\begin{aligned} \sum_{r=0}^n \mathbf{c}_r B_r^n\left(\frac{t}{c}\right) &= \sum_{r=0}^n \sum_{i=0}^r \mathbf{b}_i B_i^r(c) B_r^n\left(\frac{t}{c}\right) \\ &= \sum_{r=0}^n \sum_{i=0}^n \mathbf{b}_i B_i^r(c) B_r^n\left(\frac{t}{c}\right) \quad \text{from } B_i^r(\dots) = 0 \quad \text{for } i > r \\ &= \sum_{i=0}^n \mathbf{b}_i \sum_{r=0}^n B_i^r(c) B_r^n\left(\frac{t}{c}\right) \\ &= \sum_{i=0}^n \mathbf{b}_i B_i^n(t) \quad \text{because of property (3) of the Bernstein polynomes (see Section 12.1)} \end{aligned}$$

## 12.5.2 Smoothness conditions

### 12.5.2.1 $C^1$ - and $C^2$ -connections

For connecting two Bézier curves to a new at least  $C^1$  curve we have to define a Bézier curve on a non standard interval.

We transform the Bézier curve

$$\Gamma_1 : \mathbf{b}(t) = \sum_{i=0}^n \mathbf{b}_i B_i^n(t), \quad t \in [0, 1]$$

by the parameter transformation  $t = \frac{s-u_0}{u_1-u_0}$  and get a representation on interval  $[u_0, u_1]$ :

$$\mathbf{c}(s) := \mathbf{b}\left(\frac{s-u_0}{u_1-u_0}\right), \quad s \in [u_0, u_1].$$

The derivation at  $u_1$  is:

$$\dot{\mathbf{c}}(u_1) = \frac{\dot{\mathbf{b}}(1)}{u_1-u_0} = n \frac{\mathbf{b}_n - \mathbf{b}_{n-1}}{u_1-u_0}.$$

The second Bézier curve

$$\Gamma_2 : \mathbf{b}'(t) = \sum_{i=0}^n \mathbf{b}_{n+i} B_i^n(t), \quad t \in [0, 1]$$

has the same degree as  $\Gamma_1$  and has to be connected  $C^1$ -continuously to  $\Gamma_1$ . We apply the parameter transformation  $t = \frac{s-u_1}{u_2-u_1}$ :

$$\mathbf{c}(s) := \mathbf{b}'\left(\frac{s-u_1}{u_2-u_1}\right), \quad s \in [u_1, u_2].$$

$\mathbf{b}_n$  is the last control point of  $\Gamma_1$  and the first of  $\Gamma_2$ . Hence  $\Gamma : \mathbf{c}(s), s \in [u_0, u_2]$  is at least continuous curve. In order to achieve a  $C^1$ -connection at  $u_1$  the following condition has to be fulfilled:

$$(C1) \quad \frac{\mathbf{b}_n - \mathbf{b}_{n-1}}{u_1 - u_0} = \frac{\mathbf{b}_{n+1} - \mathbf{b}_n}{u_2 - u_1},$$

i.e. point  $\mathbf{b}_n$  divides the line segment  $\mathbf{b}_{n-1}, \mathbf{b}_{n+1}$  with ratio  $(u_1 - u_0) : (u_2 - u_1)$  (see Figure 12.4).

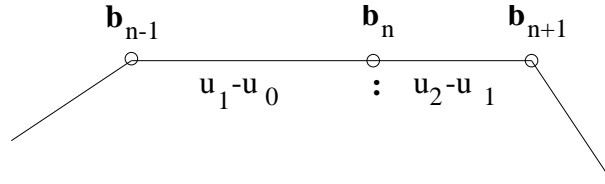


Figure 12.4:  $C^1$ -continuity condition

We get a  $C^2$ -condition by considering the second derivative of  $\mathbf{c}(s)$  at  $s = u_1$ :

$$\ddot{\mathbf{c}}(u_1) = \frac{\ddot{\mathbf{b}}(1)}{(u_1 - u_0)^2} = \frac{\ddot{\mathbf{b}}(0)}{(u_2 - u_1)^2}$$

The second derivative of a Bézier curve is a Bézier curve with control points consisting of differences of the given control points (see Section 12.3). Hence the additional condition that guarantees a  $C^2$ -connection is:

$$(C2) \quad \frac{\mathbf{b}_n - 2\mathbf{b}_{n-1} + \mathbf{b}_{n-2}}{(u_1 - u_0)^2} = \frac{\mathbf{b}_{n+2} - 2\mathbf{b}_{n+1} + \mathbf{b}_n}{(u_2 - u_1)^2}.$$

A geometric interpretation of this condition is as follows: Let  $\mathbf{d}$  be that point on the line through  $\mathbf{b}_{n-2}, \mathbf{b}_{n-1}$  such that  $\mathbf{b}_{n-1}$  divides the segment  $\mathbf{b}_{n-2}, \mathbf{d}$  with ratio  $(u_1 - u_0) : (u_2 - u_1)$ . With the abbreviations  $\Delta_0 := u_1 - u_0$ ,  $\Delta_1 := u_2 - u_1$  we get:

$$\begin{aligned} \Delta_1(\mathbf{b}_{n-1} - \mathbf{b}_{n-2}) &= \Delta_0(\mathbf{d} - \mathbf{b}_{n-1}) \\ \Leftrightarrow -\Delta_1(\mathbf{b}_{n-2} - 2\mathbf{b}_{n-1} + \mathbf{b}_n) + \Delta_1(\mathbf{b}_n - \mathbf{b}_{n-1}) &= \Delta_0(\mathbf{d} - \mathbf{b}_{n-1}) \\ \Leftrightarrow -\Delta_1(\mathbf{b}_{n-2} - 2\mathbf{b}_{n-1} + \mathbf{b}_n) &= \Delta_0(\mathbf{d} - \mathbf{b}_{n-1} - \mathbf{b}_{n+1} + \mathbf{b}_n) \quad \text{because of (C1)} \end{aligned}$$

Let  $\mathbf{d}'$  be the point of the line through  $\mathbf{b}_{n+1}, \mathbf{b}_{n+2}$  such that  $\mathbf{b}_{n+1}$  divides the segment  $\mathbf{d}', \mathbf{b}_{n+2}$  with ratio  $(u_1 - u_0) : (u_2 - u_1)$ . Similar considerations for  $\mathbf{d}'$  done for  $\mathbf{d}$  above yield:

$$-\Delta_0(\mathbf{b}_{n+2} - 2\mathbf{b}_{n+1} + \mathbf{b}_n) = \Delta_1(\mathbf{d}' - \mathbf{b}_{n-1} - \mathbf{b}_{n+1} + \mathbf{b}_n)$$

Hence: (C2) is true iff  $\mathbf{d} = \mathbf{d}'$  (see Figure 12.5).

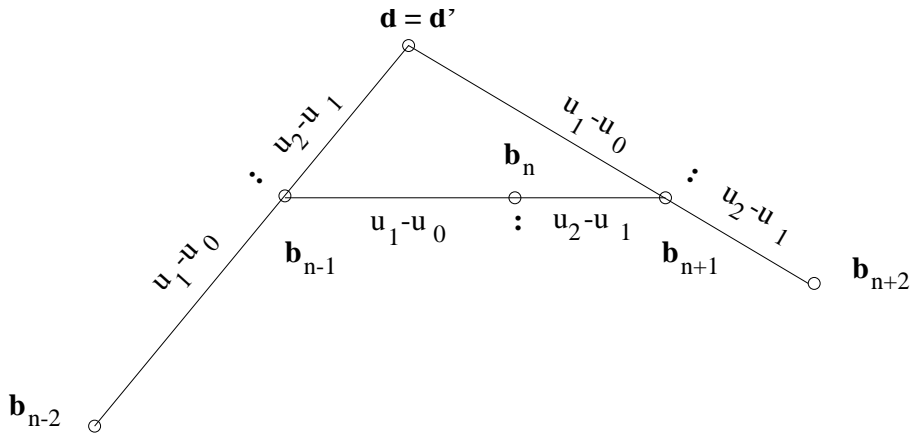


Figure 12.5:  $C^2$ -condition

12.5.2.2  $G^1$ - and  $G^2$ -Connections

Connecting the Bézier curves

$$\Gamma_1 : \mathbf{b}(t) = \sum_{i=0}^n \mathbf{b}_i B_i^n(t), \quad t \in [0, 1]$$

$$\Gamma_2 : \mathbf{c}(t) = \sum_{i=0}^m \mathbf{c}_i B_i^m(t), \quad t \in [0, 1]$$

at point  $\mathbf{b}_n$  with continuous tangent line, i.e.  $G^1$ -continuously, besides condition  $\mathbf{b}_n = \mathbf{c}_0$  (i.e.  $G^0$ -continuity) the following weaker condition

$$(G1) \quad \mathbf{b}_n - \mathbf{b}_{n-1} \text{ parallel } \mathbf{c}_1 - \mathbf{c}_0$$

has to be fulfilled.

For the additional continuity of the curvature at  $\mathbf{b}_n$  i.e.  $G^2$ -continuity, the condition

$$\frac{\|\dot{\mathbf{b}}(1) \times \ddot{\mathbf{b}}(1)\|}{\|\dot{\mathbf{b}}(1)\|^3} = \frac{\|\dot{\mathbf{c}}(0) \times \ddot{\mathbf{c}}(0)\|}{\|\dot{\mathbf{c}}(0)\|^3}$$

has to be fulfilled. With

$$\dot{\mathbf{b}}(1) = n(\mathbf{b}_n - \mathbf{b}_{n-1}) \quad \text{and} \quad \dot{\mathbf{c}}(0) = m(\mathbf{c}_1 - \mathbf{c}_0),$$

$$\ddot{\mathbf{b}}(1) = n(n-1)((\mathbf{b}_n - \mathbf{b}_{n-1}) - (\mathbf{b}_{n-1} - \mathbf{b}_{n-2})) \quad \text{and} \quad \ddot{\mathbf{c}}(0) = m(m-1)((\mathbf{c}_2 - \mathbf{c}_1) - (\mathbf{c}_1 - \mathbf{c}_0))$$

we get the condition

$$(G2) \quad \frac{(n-1)\|(\mathbf{b}_n - \mathbf{b}_{n-1}) \times (\mathbf{b}_{n-1} - \mathbf{b}_{n-2})\|}{n\|\mathbf{b}_n - \mathbf{b}_{n-1}\|^3} = \frac{(m-1)\|(\mathbf{c}_1 - \mathbf{c}_0) \times (\mathbf{c}_2 - \mathbf{c}_1)\|}{m\|\mathbf{c}_1 - \mathbf{c}_0\|^3}.$$

(In case of planar curves the vector product has to be replaced by a determinant.) An essential consequence of the representation of the curvature is: The control point  $\mathbf{b}_{n-2}$  and  $\mathbf{c}_2$  respectively can be moved parallel to the tangent at point  $\mathbf{b}_n$  and  $\mathbf{c}_0$  respectively without changing the curvature of  $\Gamma_1$  and  $\Gamma_2$  respectively at this point. Once a curvature continuous solution is found, the control points  $\mathbf{b}_{n-2}$  and  $\mathbf{c}_2$  can be moved independently parallel to the tangent line at the common point  $\mathbf{b}_n$ .

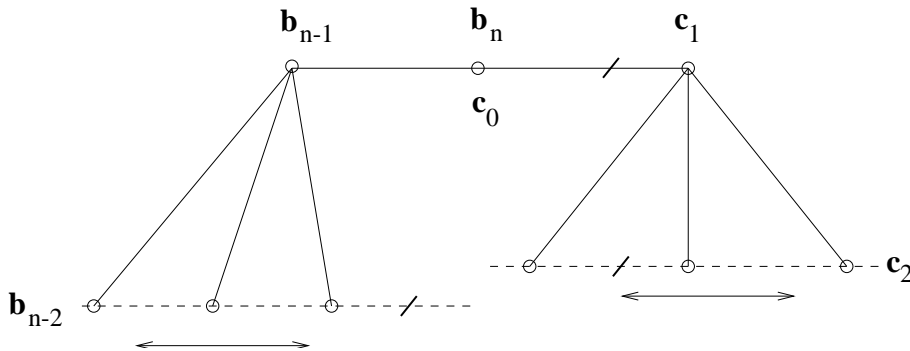


Figure 12.6:  $G^2$ -continuity

Condition  $G2$  can be seen as a relation between the areas of the triangles  $\mathbf{b}_{n-2}, \mathbf{b}_{n-1}, \mathbf{b}_n$  and  $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2$ , too.

The criteria derived in this subsection are applicable to space curves, too.

## 12.6 Rational Bézier curves

### 12.6.1 Rational curves and projective curves

Parametrizations of Bézier curves use polynomials, only. But technical important and geometrically simple curves like circles have no polynomial representation. This lack motivates an extension of the base functions to rational ones. Any conic has a rational representation. A curve  $\Gamma$  with a rational representation

$$\left( \frac{p_1(t)}{q_1(t)}, \frac{p_2(t)}{q_2(t)} \right)$$

where  $p_i$  and  $q_i$  are polynomials, has with respect to homogeneous coordinates the polynomial representation

$$\langle (p_1(t)q_2(t), p_2(t)q_1(t), q_1(t)q_2(t)) \rangle .$$

Hence a planar rational curve can be considered as a central projection of a Bézier curve in  $\mathbb{R}^3$  onto the plane  $x_3 = 1$ .

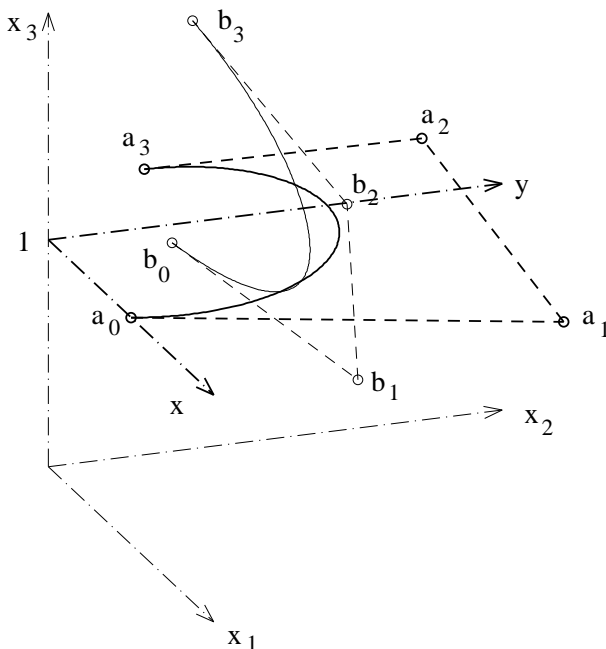


Figure 12.7: Central projection of a Bézier curve into the plane  $x_3 = 1$

The analogous statement is true for curves in  $\mathbb{R}^3$ . They can be considered as a central projection of a Bézier curve in  $\mathbb{R}^4$  into the hyperplane  $x_4 = 1$ . And any advantage of the Bézier-representation of a polynomial curve is available for rational curves, too.

## 12.7 Rational Bézier curves

Let be  $n > 0$  a fixed integer and  $\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_n, \mathbf{b}_i \neq \mathbf{0}$ , a polygon in  $\mathbb{R}^3$ . Then  $\mathbf{x}(t) := \mathbf{b}_0 B_0^n(t) + \mathbf{b}_1 B_1^n(t) + \dots + \mathbf{b}_n B_n^n(t), 0 \leq t \leq 1$ , is a Bézier-curve of (maximal) degree  $n$ . The points  $\mathbf{b}_0, \dots, \mathbf{b}_n$  are the *control points* of the Bézier curve. (see Section 4.6) We consider the 1-dimensional subspaces

$$\langle \mathbf{b}_0 B_0^n(t) + \mathbf{b}_1 B_1^n(t) + \dots + \mathbf{b}_n B_n^n(t) \rangle, 0,$$



as points of the real projective plane with  $x_3 = 0$  as plane of infinity. The affine part of this *projective Bézier curve* is called *rational Bézier curve*.

The control points of the original Bézier curve in  $\mathbb{R}^3$  can be described by:

$\mathbf{b}_i = w_i(x_i, y_i, 1)$ ,  $w_i > 0$  in case of  $\mathbf{b}_i$  not on the line of infinity  $x_3 = 0$  and

$\mathbf{b}_i = w_i(x_i, y_i, 0)$ ,  $w_i > 0$  in case of  $\mathbf{b}_i$  on the line of infinity.

Switching to inhomogeneous coordinates maps a control point either onto the affine point  $\mathbf{a}_i := (x_i, y_i)$  or onto the point of infinity in direction  $\mathbf{a}_i := (x_i, y_i)^\top$ . Point  $\mathbf{a}_i$  is called *finite control point* or *infinite control point* respectively of the rational Bézier curve and the  $w_i$  the *weight* of control point  $\mathbf{a}_i$ .

Hence, a Bézier curve has the representation:

$$\frac{\sum_{i=0}^n w_i \mathbf{a}_i B_i^n(t)}{\sum_{i=0}^n \delta_i w_i B_i^n(t)},$$

where  $\delta_i = 1$  in case of finite and  $\delta_i = 0$  in case of infinite control points.

The rational Bézier curves have the following properties:

If  $\mathbf{a}_i, w_i, i = 0, \dots, n$ , are the finite control points and weights of a rational Bézier curve  $\Gamma$ , then

1. curve  $\Gamma$  contains the control points  $\mathbf{a}_0, \mathbf{a}_n$  (first and last point of the control polygon),
2. the tangent line at point  $\mathbf{a}_0$  and  $\mathbf{a}_n$  has the direction  $\mathbf{a}_1 - \mathbf{a}_0$  and  $\mathbf{a}_n - \mathbf{a}_{n-1}$ , respectively.
3. rising the weight  $w_i$  brings the curve closer to the control point  $\mathbf{a}_i$ . (see Figure 12.7). The weights are not unique, because curve  $\Gamma$  can be represented by the same control points, but different weights.

## 12.8 Conics as rational Bézier curves

A Bézier curve of degree two with non collinear control points  $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$  in  $\mathbb{R}^3$  is a parabola and can be represented by

$$\mathbf{x}(t) = \mathbf{p}_0 + t\mathbf{f}_1 + t^2\mathbf{f}_2$$

(cf. Section 9.1.3). If a parabola in  $\mathbb{R}^3$  is used for the generation of a (planar) projective curve and its plane does not contain the origin, then its projection onto the plane  $x_3 = 1$  is an ellipse or a hyperbola or a parabola. For example

$$\langle t^2(1, 0, 0) + 2t(1-t)(0, 0, 1/2) + (1-t)^2(0, 1, 0) \rangle, \quad t \in \mathbb{R},$$

are points of the projective conic with equation  $x_1x_2 = x_3^2$  (see Section A.5. As a curve of the affine plane  $x_3 = 1$  this curve is a hyperbola (see Figure 12.8). Changing to inhomogeneous coordinates yields the rational parametric representation  $x(t) := t/(1-t), y(t) = (1-t)/t$  of the hyperbola  $y = 1/x$ .

For the following example the control point  $b_1$  is an infinite one:

$$\langle t^2(1, 0, 1) + 2t(1-t)(0, 1, 0) + (1-t)^2(-1, 0, 1) \rangle, \quad 0 \leq t \leq 1,$$

is the homogeneous representation of one half of a circle. The projective points are on the conic with equation  $x_1^2 + x_2^2 = x_3^2$  (see Figure 12.8).

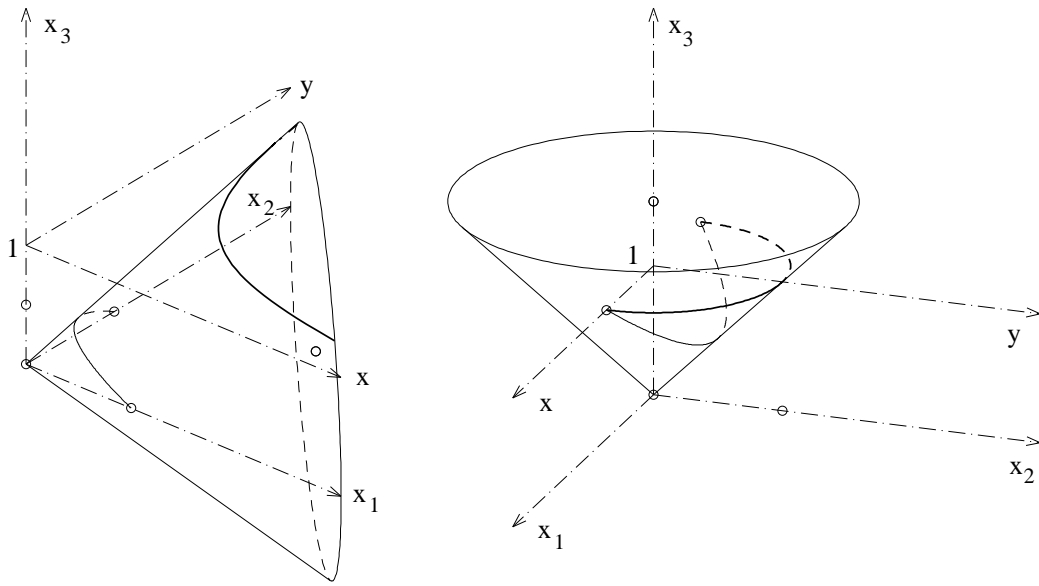


Figure 12.8: a) hyperbola as rational Bézier curve with two infinite control points, b) half of a circle as rational Bézier curve with an infinite control point

# Chapter 13

## BÉZIER–SURFACES

There are two possibilities extending the Bézier technique to surfaces which suggest them-self: tensor product Bézier surfaces and Bézier triangular surfaces. The latter is the more natural extension.

### 13.1 Tensor product Bézier surfaces

Let be  $\mathbf{b}^m(u) = \sum_{i=0}^m \mathbf{b}_i B_i^m(u)$  a Bézier curve in  $\mathbb{R}^3$ . We assume that the control points depend on an additional parameter  $v$  in such away that they lie on Bézier curves:  $\mathbf{b}_i(v) = \sum_{j=0}^n \mathbf{b}_{ij} B_j^n(v)$ . Hence

$$\mathbf{b}^{m,n}(u,v) = \sum_{i=0}^m \sum_{j=0}^n \mathbf{b}_{ij} B_i^m(u) B_j^n(v), \quad u, v \in [0, 1]$$

is a surface, the **(m,n)–tensor product Bézier surface** with *control points* or *control net*  $\{\mathbf{b}_{ij} | i = 0, \dots, m, j = 0, \dots, n\}$ .

The surface contains the points  $\mathbf{b}_{00}, \mathbf{b}_{m0}, \mathbf{b}_{0n}, \mathbf{b}_{mn}$  and the bounding curves are Bézier curves.

A (1, 1)–tensor–product Bézier surface contains lines but usually it is not planar . For example: for  $\mathbf{b}_{00} = (0, 0, 0), \mathbf{b}_{10} = (1, 0, 0), \mathbf{b}_{01} = (0, 1, 0), \mathbf{b}_{11} = (1, 1, 1)$  we get a patch of the quadric  $z = xy$ .

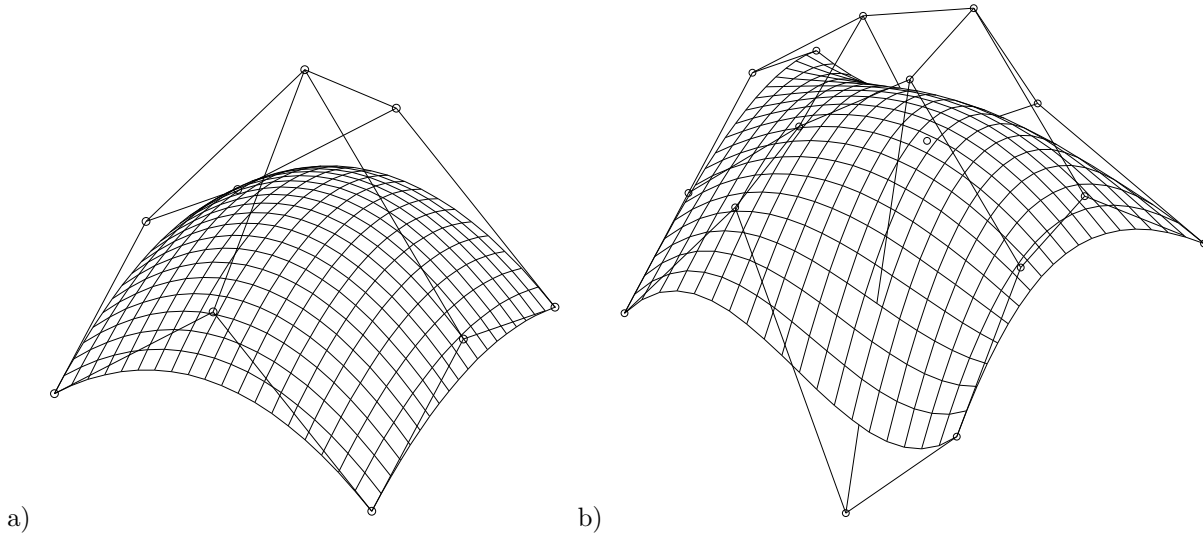


Figure 13.1: a) (2,2)– b) (3,3)–tensor–product Bézier surface and its control net

## 13.2 The Casteljau algorithm

Linear interpolation of pairs of points is the basic idea of the Casteljau algorithm for curves. The analogous technique for tensor product surfaces is the *bilinear interpolation* (see Section 10.4 on Coons patches) of four points which appears in the simplest example of a tensor product Bézier surface: A (1,1)-tensor-product Bézier surface for the four points  $\mathbf{b}_{00}, \mathbf{b}_{10}, \mathbf{b}_{01}, \mathbf{b}_{11}$  is:

$$\mathbf{b}^{1,1}(u, v) = (1-u)(1-v)\mathbf{b}_{00} + u(1-v)\mathbf{b}_{10} + (1-u)v\mathbf{b}_{01} + uv\mathbf{b}_{11}$$

Or in matrix form:

$$\mathbf{b}^{1,1}(u, v) = (1-u, u) \begin{pmatrix} \mathbf{b}_{00} & \mathbf{b}_{01} \\ \mathbf{b}_{10} & \mathbf{b}_{11} \end{pmatrix} \begin{pmatrix} 1-v \\ v \end{pmatrix}$$

We start with a  $(n \times n)$ -control-net and determine for  $r = 1, 2, \dots, n$  and a pair of parameters  $(u, v)$  intermediate vectors by bilinear interpolation:

$$\mathbf{b}_{i,j}^r = (1-u, u) \begin{pmatrix} \mathbf{b}_{i,j}^{r-1} & \mathbf{b}_{i,j+1}^{r-1} \\ \mathbf{b}_{i+1,j}^{r-1} & \mathbf{b}_{i+1,j+1}^{r-1} \end{pmatrix} \begin{pmatrix} 1-v \\ v \end{pmatrix},$$

where  $\mathbf{b}_{i,j}^0 := \mathbf{b}_{i,j}$ . Hence point  $\mathbf{b}_{0,0}^n$  is the surface point belonging to  $(u, v)$ .

In case of  $m > n$  and  $r \geq n$  the second index is constant  $j = 0$  and the interpolation is just linear (instead of *bilinear*) as it is for Bézier curves. Point  $\mathbf{b}_{0,0}^m$  is the surface point.

In case of  $m < n$  we deal analogously.

## 13.3 Degree elevation

It is an advantage if for a  $(m, n)$ -tensor-product Bézier surface we have  $m = n$ . If this is not the case it can be achieved by suitable degree elevations.

The degree elevation from  $(m, n)$  to  $(m+1, n)$  of the tensor product Bézier surface

$$\mathbf{b}^{m,n}(u, v) = \sum_{j=0}^n \left[ \sum_{i=0}^m \mathbf{b}_{ij} B_i^m(u) \right] B_j^n(v)$$

leads to the  $n+1$  degree elevations for the Bézier curves  $[\dots]$ :

$$\sum_{i=0}^m \mathbf{b}_{ij} B_i^m(u) = \sum_{i=0}^{m+1} \mathbf{b}_{ij}^{(1,0)} B_i^{m+1}(u), \quad j = 0, \dots, n$$

with

$$\mathbf{b}_{ij}^{(1,0)} := \left(1 - \frac{i}{m+1}\right) \mathbf{b}_{i,j} + \frac{i}{m+1} \mathbf{b}_{i-1,j}, \quad i = 0, \dots, m+1.$$

## 13.4 Derivatives of a Bézier surface

The partial derivative of the tensor product Bézier surface

$$\mathbf{b}^{m,n}(u, v) = \sum_{j=0}^n \sum_{i=0}^m \mathbf{b}_{ij} B_i^m(u) B_j^n(v)$$

for  $u$  is

$$\frac{\partial}{\partial u} \mathbf{b}^{m,n}(u, v) = \sum_{j=0}^n \left[ \frac{\partial}{\partial u} \sum_{i=0}^m \mathbf{b}_{ij} B_i^m(u) \right] B_j^n(v).$$

From the result for the derivative of a Bézier curve (12.3) we get:

$$\frac{\partial}{\partial u} \mathbf{b}^{m,n}(u, v) = m \sum_{j=0}^n \left[ \sum_{i=0}^{m-1} \Delta^{1,0} \mathbf{b}_{ij} B_i^{m-1}(u) \right] B_j^n(v),$$

with  $\Delta^{1,0} \mathbf{b}_{i,j} := \mathbf{b}_{i+1,j} - \mathbf{b}_{i,j}$ . Analogously we get the partial derivative for  $v$  and higher derivatives.

The vectors  $\Delta^{1,0} \mathbf{b}_{0,0}, \Delta^{0,1} \mathbf{b}_{0,0}$  are tangent to the surface at point  $\mathbf{b}_{0,0}$ . Hence  $\Delta^{1,0} \mathbf{b}_{0,0} \times \Delta^{0,1} \mathbf{b}_{0,0}$  is a normal vector to the surface at this point and the tangent plane at a vertex of the control net is determined by the vertex and its two closest neighbours.

### 13.5 Bézier triangular surfaces

A natural extension of the Bernstein polynomials to functions with two variables would start with the relation  $1 = (u + v + (1 - u - v))^n = \dots$ . For the positivity of all terms the  $(u, v)$  must be contained by the triangle  $(0, 0), (1, 0), (0, 1)$ . Two of the three line segments correspond to the intervals of the coordinate axes in the parameter plane. In order to omit this different meanings we introduce *homogeneous* coordinates  $u, v, w$  with the constraint  $u + v + w = 1, u, v, w \geq 0$ .  $u, v, w$  are called *barycentric coordinates*. the generalized Bernstein polynomials are items of the development of  $(u + v + w)^n$ :

$$B_{ijk}^n(u, v, w) := \frac{n!}{i!j!k!} u^i v^j w^k$$

with  $i + j + k = n, i, j, k \geq 0$  and  $u + v + w = 1, u, v, w \geq 0$ .

With the notations  $\mathbf{I} := (i, j, k), \mathbf{u} := (u, v, w)$  and  $|\mathbf{I}| := i + j + k, |\mathbf{u}| := u + v + w$  we get

$$B_{\mathbf{I}}^n(\mathbf{u}) := \frac{n!}{i!j!k!} u^i v^j w^k, \quad |\mathbf{u}| = 1 \quad \text{and} \quad \sum_{|\mathbf{I}|=n} B_{\mathbf{I}}^n = 1.$$

Let be  $\mathbf{b}_{00n}, \mathbf{b}_{10,n-1}, \dots, \mathbf{b}_{n00}, \mathbf{b}_{01,n-1}, \mathbf{b}_{11,n-2}, \dots, \mathbf{b}_{n-1,10}, \dots, \mathbf{b}_{0n0}$  a triangular net of points in  $\mathbb{R}^3$ , the **control points**.

$$\mathbf{b}^n(\mathbf{u}) := \sum_{|\mathbf{I}|=n} \mathbf{b}_{\mathbf{I}} B_{\mathbf{I}}^n(\mathbf{u})$$

is the corresponding **Bézier triangular surface**.

Figure 13.2 shows the net for the case  $n = 4$ .

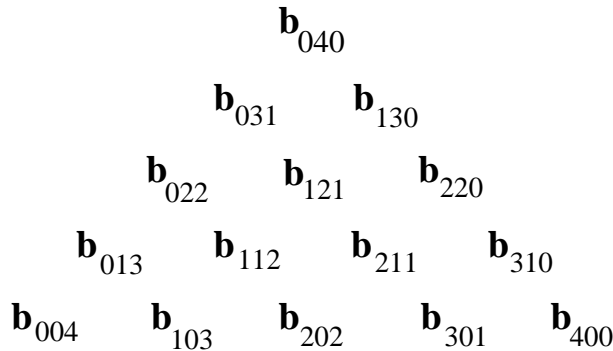


Figure 13.2: control points of a Bézier triangular surface for  $n = 4$

In order to describe the **Casteljau algorithm** for Bézier triangular surfaces we introduce the following notations:  $\mathbf{e}_1 := (1, 0, 0)$ ,  $\mathbf{e}_2 := (0, 1, 0)$ ,  $\mathbf{e}_3 := (0, 0, 1)$  and  $\mathbf{o} := (0, 0, 0)$ .

Let be  $\{\mathbf{b}_{\mathbf{I}} | \mathbf{I} = (i, j, k), i + j + k = n\}$  a triangular net of points in  $\mathbb{R}^3$  and  $\mathbf{u}$  a parameter vector in barycentric coordinates. We define for  $r = 1, \dots, n$  and  $\mathbf{I} := (i, j, k)$  with  $i + j + k = n - r$

$$\mathbf{b}_{\mathbf{I}}^r := u\mathbf{b}_{\mathbf{I}+\mathbf{e}_1}^{r-1}(\mathbf{u}) + v\mathbf{b}_{\mathbf{I}+\mathbf{e}_2}^{r-1}(\mathbf{u}) + w\mathbf{b}_{\mathbf{I}+\mathbf{e}_3}^{r-1}(\mathbf{u})$$

with  $\mathbf{b}_{\mathbf{I}}^0(\mathbf{u}) := \mathbf{b}_{\mathbf{I}}$ . Point  $\mathbf{b}_{\mathbf{o}}^n(\mathbf{u})$  is a point of the Bézier triangular surface. The proof of this statement uses (analogously to the curve case) the following formulae of recursions for the Bernstein polynomes:

$$B_{\mathbf{I}}^n(\mathbf{u}) = uB_{\mathbf{I}-\mathbf{e}_1}^{n-1}(\mathbf{u}) + vB_{\mathbf{I}-\mathbf{e}_2}^{n-1}(\mathbf{u}) + wB_{\mathbf{I}-\mathbf{e}_3}^{n-1}(\mathbf{u}), \quad \mathbf{I} = (i, j, k) \text{ with } i + j + k = n.$$

For more details see Farin *Curves and Surfaces for Computer Aided Geometric Design* or Hoschek & Lasser *Fundamentals of Computer Aided Geometric Design*.

# Chapter 14

## B-SPLINE CURVES

There are two essential disadvantages using Bézier curves:

- A Bézier curve determined by many control points is of high degree.
- Changing one of the control points affects the whole curve.

This disadvantages can be omitted by using base functions with *local* effect.

### 14.1 The B-Spline base functions

The B-spline base functions are piecewise polynomial and non vanishing within a prescribed interval only (local support). They are defined recursively:

A vector  $(t_0, t_1, \dots, t_m) \in \mathbb{R}^{m+1}$  with  $t_0 < t_1 < \dots < t_m$  is called *knot vector*.

Given  $n, k \in \mathbb{N}, n, k \geq 1$  and the knot vector  $T := (t_0, t_1, \dots, t_{n+k})$ . With functions

$$N_{i1} := \begin{cases} 1 & , \quad t_i \leq t \leq t_{i+1} \\ 0 & , \quad \text{sonst} \end{cases}$$

for  $i = 1, 2, \dots, n+k$  we define recursively for  $k > 1$ :

$$N_{ik} := \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t).$$

The functions  $N_{ik}$  are called **B-Spline-base functions of order k**. They are *piecewise polynomial* of degree (k-1) and have the following properties

(B1)  $N_{ik}(t) > 0$  for  $t_i < t < t_{i+k}$ , the interval  $[t_i, t_{i+k}]$  is called *support* of  $N_{ik}$ ,

(B2)  $N_{ik}(t) = 0$  for  $t_0 \leq t \leq t_i$ ,  $t_{i+k} \leq t \leq t_{n+k}$ ,

(B3)  $\sum_{i=0}^n N_{ik}(t) = 1$  for  $t \in [t_{k-1}, t_{n+1}]$ ,

(B4) at knots  $t_l$  with  $t_i \leq t_l \leq t_{i+k}$  functions  $N_{ik}$  are  $C^{k-2}$ -continuous.

In case of special knot vectors  $(0, 1, 2, \dots, n+k)$  the base functions are called *uniform*.

Figures 14.1 show graphs of some B-spline base functions of order 1,2 and 3, respectively. They are piecewise polynomial of degree 0,1 bzw. 2.

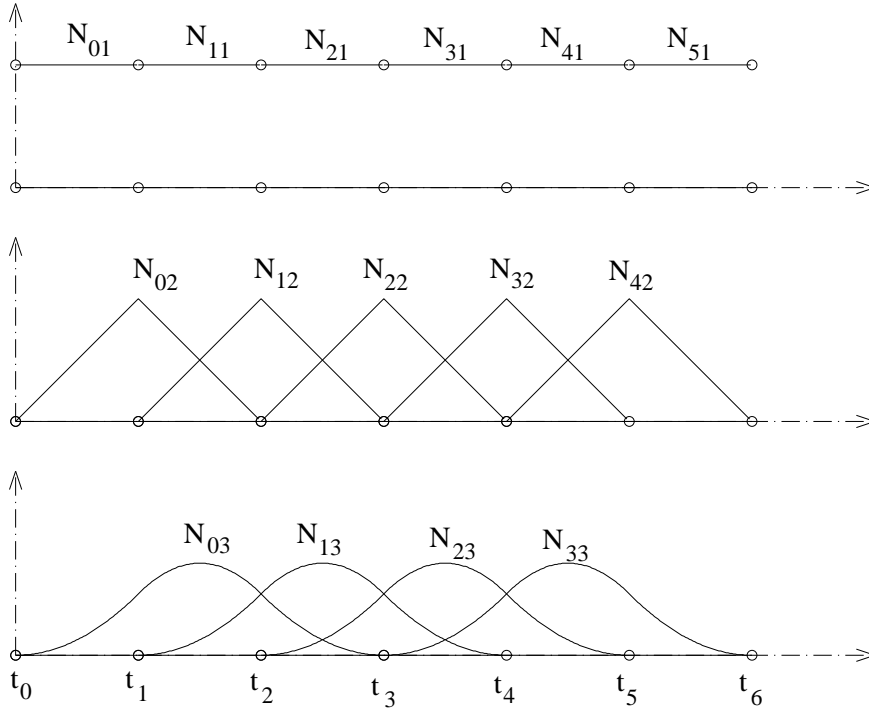


Figure 14.1: B-spline base functions with exclusively simple knots

An essential extension of the definition of the base functions is the possibility of *multiple knots*. For example in case of  $t_1 = t_0$  we get  $N_{01} = 0$  and one omits  $\frac{t-t_0}{t_1-t_0}N_{01}(t)$  for the evaluation of  $N_{02}$ . Especially for  $t_0 = t_1 = 0, t_2 = 1$  we get  $N_{02} = 1 - t$ , the Bernstein polynome  $B_1^1(t)$ . In general:

$$T = (\underbrace{0, 0, \dots, 0}_{k\text{-mal}}, \underbrace{1, 1, \dots, 1}_{k\text{-mal}}) \implies N_{ik}(t) = B_i^{k-1}(t).$$

For a multiple knot with multiplicity  $l$  function  $N_{ik}$  is  $C^{k-l-1}$ -continuous only.

Figure 14.2 shows base functions with multiple knots for  $k = 3$ .  $N_{23}$  and  $N_{33}$  are discontinuous at the 3-fold knot  $t_3 = t_4 = t_5$ .

Analogously to the Bézier curve the **B-Spline curve** is defined using the B-spline base functions: Let be  $n, k \in \mathbb{N}$ ,  $T = (t_0, t_1, \dots, t_{n+k})$  a knot vector and  $\mathbf{d}_i \in \mathbb{R}^2, i = 0, 1, \dots, n$ . Hence

$$\mathbf{x}(t) := \sum_{i=0}^n \mathbf{d}_i N_{ik}(t), \quad t_{k-1} \leq t \leq t_{n+1}$$

is a **B-spline curve** with **control points** or **de-Boor points**  $\mathbf{d}_0, \dots, \mathbf{d}_n$ .

If the curve should contain the points  $\mathbf{d}_0, \mathbf{d}_n$ , the first and the last knot must have multiplicity  $k$ .



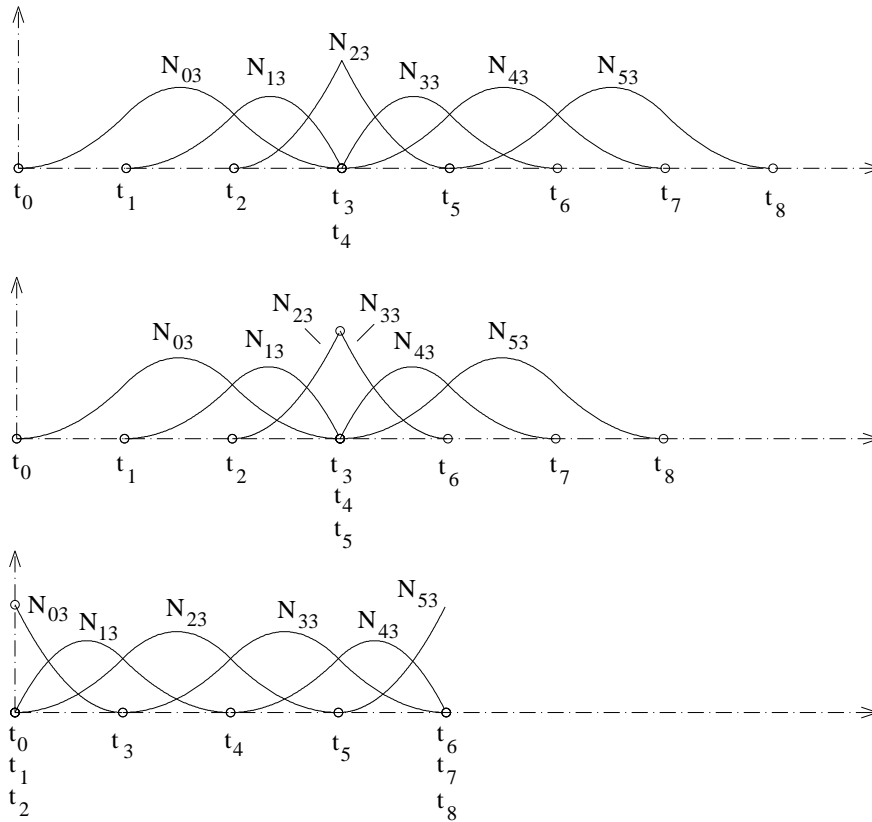


Figure 14.2: B-Spline base functions with multiple knots

## 14.2 The de-Boor algorithm

Let be  $n, k \in \mathbb{N}$ ,  $T = (t_0, t_1, \dots, t_{n+k})$  the knot vector and  $\mathbf{d}_i \in \mathbb{R}^2, i = 0, 1, \dots, n$  the control points of a B-spline curve

$$\mathbf{x}(t) := \sum_{i=0}^n \mathbf{d}_i N_{ik}(t), \quad t_{k-1} \leq t \leq t_{n+1}.$$

The *de-Boor algorithm* defines (analogously to the Casteljau algorithm for Bézier curves) intermediate points by linear interpolations:

$$\mathbf{D}_i^j := (1 - \alpha_i^j) \mathbf{D}_{i-1}^{j-1} + \alpha_i^j \mathbf{D}_i^{j-1}, \quad j > 0.$$

with

$$\alpha_i^j := \frac{t - t_i}{t_{i+k-j} - t_i} \quad \text{and} \quad \mathbf{D}_i^0 := \mathbf{d}_i.$$

With the recursion formulae for the base functions one proves:

$$\mathbf{x}(t) = \mathbf{D}_r^{k-1} \quad \text{for} \quad t \in [t_r, t_{r+1}].$$

For more details: see Hoschek & Lasser: *Fundamentals of Computer Aided Geometric Design*.



# Appendix A

## THE REAL PROJECTIVE PLANE

### A.1 The real affine plane

**Definition A.1** *Let be*

$$\begin{aligned} \mathbf{P} &:= \mathbb{R}^2, \text{ the points} \\ \mathbf{G} &:= \{ \{(x, y) \in \mathbb{R}^2 \mid ax + by + c = 0\} \mid (0, 0) \neq (a, b) \in \mathbb{R}^2 \} \\ &= \{ \{(x, y) \in \mathbb{R}^2 \mid y = mx + d\} \mid m, d \in \mathbb{R} \} \cup \{ \{(x, y) \in \mathbb{R}^2 \mid x = c\} \mid c \in \mathbb{R} \} \end{aligned}$$

the lines.

$$\mathbf{A}(\mathbb{R}) := (\mathbf{P}, \mathbf{G}, \in) \text{ is called real affine plane.}$$

A permutation of  $\mathbf{P}$  which maps lines onto lines is called collineation of  $\mathbf{A}(\mathbb{R})$ .

**Theorem A.1** *Any collineation  $\kappa$  of  $\mathbf{A}(\mathbb{R})$  can be represented by*

$$\kappa : (x, y) \rightarrow (ax + by + s, cx + dy + t), \quad ad - bc \neq 0,$$

with suitable  $a, b, c, d, s, t \in \mathbb{R}$ .  $\kappa$  is called affinity.

**proof:** omitted.

**Example A.1**

- a)  $(x, y) \rightarrow (x + s, y + t), \quad s, t \in \mathbb{R}$  translation.
- b)  $(x, y) \rightarrow (x, dy), \quad 0 \neq d \in \mathbb{R}$  axial affinity with axis  $y = 0$  and direction  $(0, 1)^\top$ .
- c)  $(x, y) \rightarrow (ax, y), \quad 0 \neq a \in \mathbb{R}$  axial affinity with axis  $x = 0$  and direction  $(1, 0)^\top$ .
- d)  $(x, y) \rightarrow (ax, ay), \quad 0 \neq a \in \mathbb{R}$  dilatation at point  $(0, 0)$ .
- e)  $(x, y) \rightarrow (x + by, y), \quad b \in \mathbb{R}$  shearing with axis  $y = 0$ .
- f)  $(x, y) \rightarrow (x, y + cx), \quad c \in \mathbb{R}$  shearing with axis  $x = 0$ .

### A.2 The real projective plane

**Definition A.2** *Let be*

$$\begin{aligned} \overline{\mathbf{P}}_1 &= \mathbb{R}^2 \cup \mathbb{R} \cup \{\infty\}, \quad \infty \notin \mathbb{R} \\ \overline{\mathbf{G}}_1 &= \{ \{(x, y) \in \mathbb{R}^2 \mid y = mx + d\} \cup \{(m)\} \mid m, d \in \mathbb{R} \} \\ &\cup \{ \{(x, y) \in \mathbb{R}^2 \mid x = c\} \cup \{(\infty)\} \mid c \in \mathbb{R} \} \\ &\cup \underbrace{ \{ \{(m)\} \mid m \in \mathbb{R} \} \cup \{(\infty)\} \} }_{g_\infty} \end{aligned}$$

$P_1(\mathbb{R}) := (\overline{\mathbf{P}}_1, \overline{\mathbf{G}}_1, \epsilon)$  is called inhomogeneous model of the real projective plane

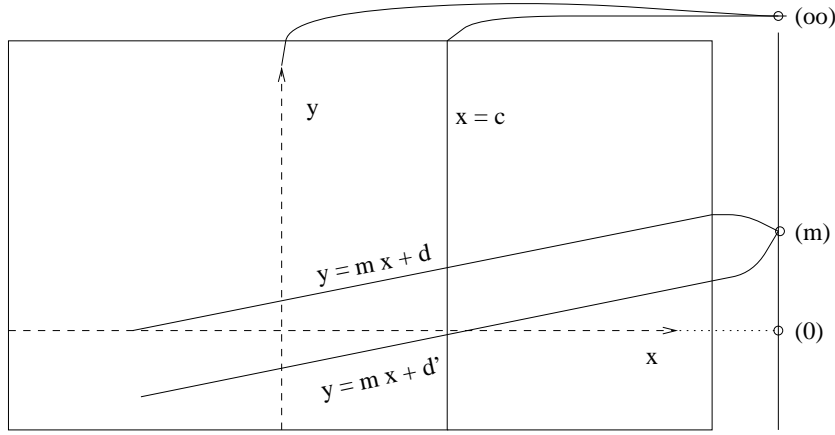


Figure A.1: Inhomogeneous model of the real projective plane

**Remark:**

Essential properties of the projective plane are:

- Any two points are connected by exactly one line (same in an affine plane).
- Any two lines intersect in exactly one point (new !).

**Definition A.3** Let be  $O := (0, 0, 0)$  and  $\langle \mathbf{x} \rangle$  the 1-dimensional subspace generated by  $\mathbf{x} \neq \mathbf{o}$ .

$$\begin{aligned} \overline{\mathbf{P}}_2 &= \{\text{lines of } \mathbb{R}^3 \text{ containing point } O\} \\ &= \{\langle (x_1, x_2, x_3)^\top \rangle \mid (0, 0, 0)^\top \neq (x_1, x_2, x_3) \in \mathbb{R}^3\} \\ \overline{\mathbf{G}}_2 &= \{\text{planes of } \mathbb{R}^3 \text{ containing point } O\} \\ &= \{\{\langle (x_1, x_2, x_3)^\top \rangle \in \overline{\mathbf{P}}_2 \mid ax_1 + bx_2 + cx_3 = 0\} \mid (0, 0, 0)^\top \neq (a, b, c)^\top \in \mathbb{R}^3\} \end{aligned}$$

$P_2(\mathbb{R}) = (\overline{\mathbf{P}}_2, \overline{\mathbf{G}}_2, \epsilon)$  is called homogeneous model of the real projective plane

### A.3 Isomorphism between the two models

The following isomorphism maps the inhomogeneous model onto the homogeneous one in such a way that the (affine) points  $\mathbb{R}^2$  are mapped onto the 1-dimensional subspaces of  $\mathbb{R}^3$  which have a point with plane  $x_3 = 1$  in common:

$$\begin{aligned} (x, y) &\rightarrow \langle (x, y, 1) \rangle = \langle (xx_3, yx_3, x_3) \rangle, & x_3 &\neq 0. \\ (m) &\rightarrow \langle (1, m, 0) \rangle = \langle (x_1, mx_1, 0) \rangle, & x_1 &\neq 0. \\ (\infty) &\rightarrow \langle (0, 1, 0) \rangle = \langle (0, x_2, 0) \rangle, & x_2 &\neq 0. \end{aligned}$$

The inverse mapping:

$$\begin{aligned} \langle (x_1, x_2, x_3) \rangle &\rightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, & \text{falls } x_3 &\neq 0 \\ \langle (x_1, x_2, 0) \rangle &\rightarrow \begin{pmatrix} x_2 \\ x_1 \end{pmatrix}, & \text{falls } x_1 &\neq 0 \\ \langle (0, x_2, 0) \rangle &\rightarrow (\infty), & \text{falls } x_2 &\neq 0. \end{aligned}$$

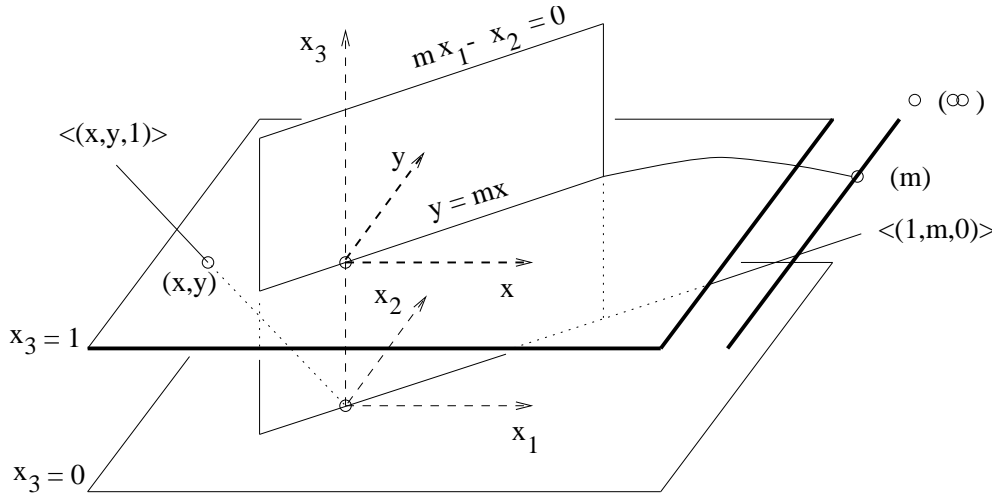


Figure A.2: Isomorphism between the homogeneous and the inhomogeneous model

For special points we have the corresponding:

$$\begin{aligned} (0, 0) &\longleftrightarrow \langle (0, 0, 1)^\top \rangle, & (0) &\longleftrightarrow \langle (1, 0, 0) \rangle \\ (\infty) &\longleftrightarrow \langle (0, 1, 0)^\top \rangle, & (1, 1) &\longleftrightarrow \langle (1, 1, 1)^\top \rangle \end{aligned}$$

The correspondence between the lines is:

$$\begin{aligned} y = mx + d &\longleftrightarrow \frac{x_2}{x_3} = m \frac{x_2}{x_3} + d \longleftrightarrow mx_1 - x_2 + dx_3 = 0 \\ x = c &\longleftrightarrow \frac{x_1}{x_3} = c \longleftrightarrow x_1 - cx_3 = 0 \\ g_\infty &\longleftrightarrow \text{line containing } \langle (1, 0, 0)^\top \rangle, \langle (0, 1, 0)^\top \rangle \longleftrightarrow x_3 = 0 \end{aligned}$$

**Remark:**

Of course there are different isomorphisms possible.

For example:  $(x, y) \longrightarrow \langle (1, x, y)^\top \rangle$ , i.e.  $g_\infty$  is the line  $x_1 = 0$  (plane in  $\mathbb{R}^3$ ).

## A.4 Collineations of the real projective plane

**Definition A.4** A permutation of the set of points which maps lines onto lines is called collineation.

**Installment of the affinities:**

| in $\mathbb{P}_1(\mathbb{R})$                                                                           | in $\mathbb{P}_2(\mathbb{R})$                                                                                                                                                                                                                      |
|---------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| a) $(x, y) \rightarrow (x + s, y + t)$<br>$(m) \rightarrow (m)$<br><br>$(\infty) \rightarrow (\infty)$  | $\langle (x_1, x_2, x_3) \rangle \rightarrow \langle (x_1 + sx_3, x_2 + tx_3, x_3) \rangle$<br>(in $\mathbb{R}^3$ : shearing with axis $x_3 = 0$ )<br>$\begin{pmatrix} 1 & 0 & s \\ 0 & 1 & t \\ 0 & 0 & 1 \end{pmatrix}$                          |
| b) $(x, y) \rightarrow (x, dy)$<br>$(m) \rightarrow (md)$<br><br>$(\infty) \rightarrow (\infty)$        | $\langle (x_1, x_2, x_3) \rangle \rightarrow \langle (x_1, dx_2, x_3) \rangle$<br>(in $\mathbb{R}^3$ : axial affinity with axis $x_2 = 0$ and direction $(0, 1, 0)^\top$ .)<br>$\begin{pmatrix} 1 & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & 1 \end{pmatrix}$ |
| c) $(x, y) \rightarrow (ax, ay)$<br>$(m) \rightarrow (m)$<br><br>$(\infty) \rightarrow (\infty)$        | $\langle (x_1, x_2, x_3) \rangle \rightarrow \langle (ax_1, ax_2, x_3) \rangle$<br>(in $\mathbb{R}^3$ : dilatation at line $x_1 = 0, x_2 = 0$ )<br>$\begin{pmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 1 \end{pmatrix}$                             |
| d) $(x, y) \rightarrow (x, y + cx)$<br>$(m) \rightarrow (m + d)$<br><br>$(\infty) \rightarrow (\infty)$ | $\langle (x_1, x_2, x_3) \rangle \rightarrow \langle (x_1, x_2 + cx_1, x_3) \rangle$<br>(in $\mathbb{R}^3$ : shearing with axis $x_1 = 0$ )<br>$\begin{pmatrix} 1 & 0 & 0 \\ c & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$                                 |
| e) general case:                                                                                        |                                                                                                                                                                                                                                                    |

$$\begin{aligned} (x, y) &\rightarrow (ax + by + s, cx + dy + t) \\ (m) &\rightarrow \left( \frac{c+dm}{a+bm} \right) \end{aligned}$$

(“Calculate” with  $\infty$  in “usual” manner:

$$m \rightarrow \begin{cases} \frac{c+dm}{a+bm}, & \text{falls } a + bm \neq 0 \\ \infty, & \text{falls } a + bm = 0 \end{cases}, \quad \infty \rightarrow \begin{cases} \frac{d}{b}, & \text{falls } b \neq 0 \\ \infty, & \text{falls } b = 0 \end{cases} .)$$

$$\langle (x_1, x_2, x_3) \rangle \rightarrow \langle (ax_1 + bx_2 + sx_3, cx_1 + dx_2 + tx_3, x_3) \rangle$$

Hence, the installment of an affinity can be represented in the homogeneous model by a linear map with matrix

$$\begin{pmatrix} a & b & s \\ c & d & t \\ 0 & 0 & 1 \end{pmatrix}$$

## A.5 Projective equivalency of the conics

In this section we shall show that all *non degenerated* conics (ellipse, hyperbola, parabola) are projectively equivalent, i.e. any pair of non degenerate conics are related by a projective collineation.

### The hyperbola

The transformation of the equation  $y = \frac{1}{x}$  of the unit hyperbola into homogeneous coordinates by  $x = x_1/x_2, y = x_2/x_3$  shows that the points of the hyperbola fulfil the equation

$$x_1x_2 = x_3^2 \quad (\text{C1}).$$

The set of all (projective) points which fulfil this equation is called a *non degenerate conic* (One should recognize that the (projective) points  $\langle (1, 0, 0) \rangle, \langle (0, 1, 0) \rangle$  of the line of infinity  $g_\infty$  fulfil equation (C1), too !)

### The parabola

Following the same procedure we get from the equation  $y = x^2$  of the unit parabola the equation in homogeneous coordinates

$$x_2x_3 = x_1^2 \quad (\text{C2}).$$

(Recognize that the (projective) point  $\langle (0, 1, 0) \rangle$  of the line of infinity  $g_\infty$  fulfils equation (C2), too.)

Obviously conic (C1) can be transformed into conic (C2) by a simple coordinate transformation, i.e. by a projective collineation.

### The circle

Starting with the unit circle  $x^2 + y^2 = 1$  we get the equation

$$x_1^2 + x_2^2 = x_3^2 \quad (\text{C3}).$$

No point of the line of infinity  $g_\infty$  fulfils this equation.

In  $\mathbb{R}^3$  equation (C3) describes a cone with vertex  $(0, 0, 0)$  and  $x_3$ -axis as cone axis. In order to recognize the projective equivalence of (C3) to (C1) and (C2) we apply a linear coordinate transformation which induces a projective collineation:

$$x'_1 = x_1, \quad x'_2 = x_3 - x_2, \quad x'_3 = x_3 + x_2$$

maps

$$(\text{C3}): x_1^2 = (x_3 - x_2)(x_3 + x_2).$$

onto conic (C2) which proves the projective equivalence of the conics.

Figure A.3 shows the hyperbola, parabola and circle as affine curves in plane  $x_3 = 1$  and the corresponding cones which contain the conics as plane sections.

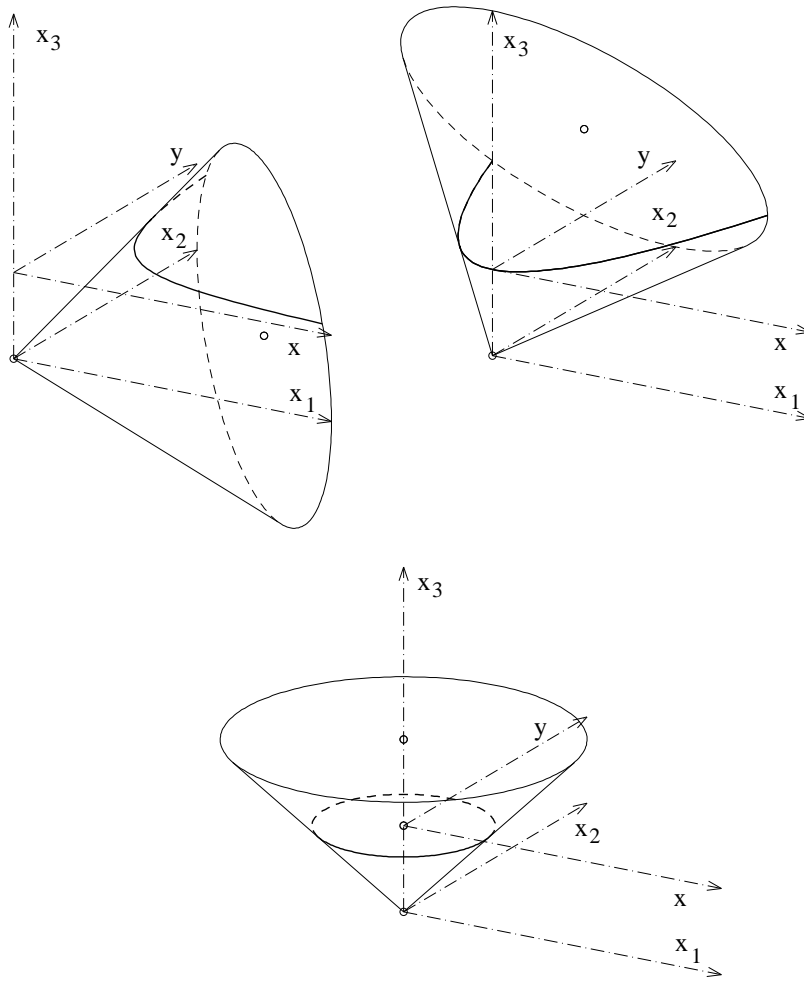


Figure A.3: Projective conics and the corresponding cones for (C1), (C2) (C3) respectively,