PIXEL FUNCTIONS IN COMPUTER GRAPHICS

J. PEREDY

Department of Descriptive Geometry II

Technical University, H-1521, Budapest

Received: June 1989

Abstract

The paper presents a set of curve drawing algorithms with the corresponding theoretical background based on a "discreet analysis". The proposed algorithms for drawing plane curves use integer addition onely. The set of algorithms constitutes a homogeneous system comprising straight lines, different kinds of polinomials (including all conic sections), exponentials, hyperbolic and trigonometric functions, etc. and represents a new viewpoint for classification of curves. The paper points out the possibilities of generalization for 3D curves and surfaces as well as the prospects of practical application in the different fields of computational geometry, CAD, robotics, etc.

Fundamental concepts

The finite plane rectangle of finite subdivision

In the present paper some problems of discrete representation of functions on a raster type computer display will be discussed. The discussion starts with the simplest case, that of single variable functions (i.e. plane curves). Let us consider a rectangle of finite dimensions, divided to a finite number of strips parallel to its sides. Let the two directions be x and y, and the corresponding number of subdivision (n_x+1) and (n_y+1), respectively. Now, divisions in directions x and y can be numbered using natural numbers completed with zero as $0 \leq i_x \leq n_x$ and $0 \leq i_y \leq n_y$, resp. Let i_x and i_y two integers satisfying the quoted inequalities, then the pair of numbers (i_x, i_y) may indicate the small elementary rectangle common between division stripes i_x and i_y in directions x and y, respectively. This is the way, how the screen of raster type displays is divided into elementary units. The elementary rectangles are called in the following plane elements or in short "pixels".

Relative position of two pixels

Let (i_x, i_y) and (j_x, j_y) be two different pixels. They are said to be adjacent if either

- $i_x = j_x$ and $i_y = j_y \pm 1$ (adjacent in direction y); or
- $i_y = j_y$ and $i_x = j_x \pm 1$ (adjacent in direction x).

Both visually and by definition, it is clear that in general a pixel has four neighbours, onely on the edges or in the corners of the rectangle representing the screen appear pixels with three or two neighbours, respectively.

In compliance with the definition the relative position of two pixels may be as follows:

— Two pixels are aligned (are in the same row) if $i_y = j_y$.

— The first pixel is higher or lower than the other if $i_y > j_y$ or $i_y < j_y$, respectively.

— Two pixels are in the same column if $i_x = j_x$.

— The first pixel is to the right or to the left of the other if $i_x > j_x$ or $i_x < j_x$, respectively.

Refinement of the subdivision

The subdivision can be refined by further dividing elementary rectangles obtained in the first division in the same way. Theoretically, every pixel could be further divided by different finite numbers in any direction. A uniform refinement is that where every pixel has the same number of finite subdivisions in both directions. For a uniform refinement e.g. by 10 both the original and the refined pixels can be expressed conveniently by applying a "decimal point". (For instance, refined pixels obtained by refining the original pixel (3, 6) are (3.0, 6.0), (3.1, 6.0), ... (3.9, 6.9), i.e. the original pixel contains 100 refined pixels.)

Obviously, the denotation using decimal point does not affect the essence of the stipulation to assign integer numbers to the finite subdivisions. Namely, omitting the decimal point leads to such numbers, (e.g. to 62 from 6.2), and the decimal point helps onely to distinguish between the original and refined pixels and express their mutual correspondence, namely, the "fine" pixel with integer serial numbers won by omitting the "decimal points" is a subdivision of the original pixel with serial numbers expressed by the integers left from the "decimal point", e.g. in case of (3.1, 6.2), the "fine" pixel (31, 62) belongs to the original pixel (3, 6).

If the relative position of a pixel with respect to an other has been found to be higher, lower, before (to the left) or after (to the right), the same holds true for any pair of their refined pixels. If two pixels are in the same row (or column), then there are among their refined pixels absolutely sure such pairs for which the same holds true, but not for any pair. If two pixels are adjacent in some sense, then among the pairs of their refined pixels exist adjacent in the same sense, adjacent in an other sense surely do not, while at all not adjacents do. These self-evident facts have been quoted to illustrate that such important fundamental relations between pixels as the relative position or neighbourhood preserve in some sense their validity after refinement, too. In this paper pixel

properties being "invariant" in some sense, preserving their "coherency" after refinement and being not bound to any special number of division will be considered onely.

The difference of two pixels

Let (i_x, i_y) and (j_x, j_y) be two different pixels and assume that $i_x > j_x$ and $i_y > j_y$ i.e. the first one is higher than and to the right of the second one. The difference

$$(i_x, i_y) - (j_x, j_y)$$

of the two pixels is defined as a part of the plane composed of four adjacent pixels

$$(i_x - j_x - 1,\ i_y - j_y)\ \ (i_x - j_x,\ i_y - j_y)$$
$$(i_x - j_x - 1,\ i_y - j_y - 1)\ \ (i_x - j_x,\ i_y - j_y - 1).$$

This formula for computing differences is well-known in the arith metics of intervals. It can be shown, that the four adjacent pixels given by t his rule include all the difference pixels in case of any refinement, too, and the differences of all possible refined pixel-pairs perfectly fill out the original quad ruplet. Thus the just mentioned rule for computing pixel-differences is in full accordance with any refinement.

The initial stipulation in discussing the difference of two pixels h as been that the "minuend" is to the right of, and higher than the "subtrahend". This stipulation is necessary to ensure that the difference quadruplets consist out of the pixels of the original plane rectangle onely. For being able to neglect this restriction the original plane quadrant has to be extended to its double in both directions. To number the new pixels resulting from this extension it is convenient to use negativ integers. After this extension of the original finite plane rectangle. the difference of any two original pixel may be computed without any restriction.

Introduction of negative integers to denote the pixels of the extended rectangle is in no essential contradiction to the former stipulation of assigning natural numbers to the finite subdivisions of finite plane rectangles, namely the proposed serial numbers using negativ integers $-(n_x + 1) \ldots n_x$ may be replaced by equivalent numbering $0 \ldots (2n_x + 1)$ and $-(n_y + 1) \ldots \ldots n_y$ by $0 \ldots (2n_x + 1)$. The negative signs simply refer to the fact that for expressing the differences of any two pixels of the original finite rectangle an extended rectangle is needed, and the pixels with negative numbering are pixels of the extended parts.

Pixel functions

Starting from an arbitrary pixel considered as element № 0 of the pixel function let element № 1 be any of its neighbours. The element № 2 of the pixel function may be any neighbour of element № 1, and so on. Repeating m times this stepping from a pixel to one of its four neighbours leads to a pixel function affecting $m + 1$ pixels (with the starting pixel included).

The pixel function is somewhat more than simply the set of the involved pixels, the sequence is of importance. For instance one and the same pixel may be "stepped on" more than once while traversing all the pixels of a pixel function.

The fundamental description of pixel functions

The most convenient writing of a pixel function in conformity with its definition is:

— to indicate the starting pixel (i.e. the pixel № 0) by its "coordinates" (i_{x0}, i_{y0}), and

— to indicate the sense of neighbourhood between all pixels of the function.

Four kinds of neighbourhood are possible: right- or left-hand adjacency in direction x, and upper or lower adjacency in direction y. Thereby, description of a neighbourhood relation requires two bits of information, i.e. a pair of bits. In written form the four tokens $+x$, $-x$, $+y$, $-y$ will be used to indicate the neighbourhood relations and in general such a pair of bits will be denoted by p_i. (Thus p_i may assume any of the "values" $+x$, $-x$, $+y$, $-y$.)

Using the introduced notations a pixel function may be written in the form

$$(i_{x0}, i_{y0}), p_1, p_2, \cdots p_i, \cdots p_m.$$

This form is called the "basic writing" or "basic form" of a pixel function containing $m + 1$ pixels. The basic writing of a pixel function consists of the starting pixel "coordinates" and of the set of "neighbourhood bit pairs".

Monotonic pixel functions

A pixel function is monotonic if its basic form contains onely one of the neighbourhood tokens $+x$, $-x$ and one of $+y$, $-y$. This means that if a pixel of a monotonic pixel function is e.g. right-hand neighbour of the former one, nowhere in the function follows a left-hand neighbour, or if there is somewhere an upper neighbour, nowhere in the function follows a lower one (of course in the sense of advancement from the starting pixel to the last one). In the following investigations—unless explicitely stated otherwise—monotonic pixel

functions (or monotonic sections of general pixel functions considered separately) will be discussed, sometimes—for the sake of conciseness—without any direct reference to this restriction.

As a further ease of formulation in case of monotonic pixel functions the tokens $+x$, $+y$ will be used onely. Since by rotating the function or by interchanging the expressions "left—right" and "higher—lower" any monotonic pixel function can be transformed into such a position, this simplification means no essential restriction beyond the requirement of monotonic character.

Concurrent monotonic sequences and series

Let us consider two monotonically non-decreasing sequence of natural numbers

$$S_{x1}, S_{x2}, \ldots S_{xm}, (S_{xi+1} \geq S_{xi}, i = 1, 2, \ldots m-1),$$

$$S_{y1}, S_{y2}, \ldots S_{yn}, (S_{yj+1} \geq S_{yj}, j = 1, 2, \ldots n-1).$$

Let us order this two sets of numerical values (one with subscript x, other with subscript y) into a common sequence corresponding to their absolute values, agreeing that if the two sequences happen to contain identical values, in the common sequence the term with subscript x comes first. The two sequences together with their joint sequence ordered by magnitude are termed a pair of concurrent sequences.

There is a obvious interrelation between monotonic pixel functions and pairs of concurrent monotonic sequences. Starting from an initial pixel (pixel № 0) and stepping in direction x if in the common order the first term has the subscript x and in direction y if it has the subscript y, and continuing this process allways selecting adjacent pixels corresponding the indices of terms in the common sequence a pixel function of $n+m+1$ pixels can be constructed. Thereby a pair of concurrent sequences has a pixel function as counterpart. A starting pixel and a pair of concurrent sequences unambigously determine a pixel function.

It is easy to demonstrate that for any pixel function can be constructed a corresponding pair of concurrent sequences describing the given function. The basic form of the pixel function is

$$(i_{x0}, i_{y0}), p_1, p_2, \ldots p_i, \ldots p_m.$$

Let us consider the subscripts of the p_i bitpairs i.e. the natural numbers $i = 1, 2, \ldots m$ as the joint order of the concurrent sequences to be constructed and classify this subscript value i as term of the x or y sequence depending on wether the respective neighbourhood token p_i refers to an adjacency in direction x or y. Thereby, obviously, a pair of concurrent sequences describing the given function has been constructed.

The same pixel function can be described by several different pairs of concurrent sequences. Let us consider an arbitrary term S_{xk} of the sequence with subscript x for which $S_{yj} < S_{xk} < S_{yj+1}$. Add the same number D to each of the terms S_{xk}, S_{xk+1}, ... S_{xm} and S_{yj+1}, S_{yj+2}, ... S_{jn}. In consequence of this modification the joint order of magnitude of the terms is not changed, that is, the modified pair of concurrent sequences describes the original pixel function. Thus, every pixel function can be described by its initial pixel and by a pair of concurrent sequences, and this can be done—at least what concerns the pair of concurrent sequences—not in a single form but in several different forms.

Terms of both sequences S_{x1}, S_{x2}, ... S_{xm} and S_{y1}, S_{y2}, ... S_{yn} can be considered as the partial sums of series

$$A_{x0}, A_{x1}, \ldots A_{xm-1} \text{ and } A_{y0}, A_{y1}, \ldots A_{yn-1}.$$

In case of monotonic S_{xi} and S_{yj} sequences $A_{xi} \geq 0$ and $A_{yj} \geq 0$ for all i and j. The sequences S_{x1}, S_{x2}, ... S_{xm} and S_{y1}, S_{y2}, ... S_{yn} uniquely define the series A_{x0}, A_{x1}, ... A_{xm-1} and A_{y0}, A_{y1}, ... A_{yn-1}. Thereby pairs of concurrent, series can be assigned to the pairs of concurrent sequences. To describe pixel functions pairs of concurrent series are of use, too.

The resulting equivalent pairs of concurrent sequences and series are connected by the equations

$$A_{x0} = S_{x1}, \; A_{xi} = S_{xi+1} - S_{xi}, \; S_{xi} \sum_{k=0}^{i-1} A_{xk},$$

$$A_{y0} = S_{y1}, \; A_{yj} = S_{yj+1} - S_{yj}, \; S_{yj} = \sum_{k=0}^{j-1} A_{yk}.$$

Since in formulating pixel functions in terms of pairs of concurrent sequences and series the properties of pixel functions are expressed by arithmetics, equations and inequalities, this type of description is called concisely arithmetic representation of pixel functions.

Pairs of series with a uniform base

An arithmetic representation of a pixel function is said to be of a uniform base if at least one of the member series (say that with subscript y) is "equidistant", i.e.

$$A_{y0} = A_{y1} = \ldots A_{yn-1} = A_y = \text{const.}$$

and consequently

$$S_{Uy1} = A_y, \; S_{Uy2} = 2A_y, \ldots S_{Uyk} = kA_y, \ldots \; S_{Uyn} = nA_y.$$

Obviously, any pixel function described by an arithmetic representation of general type can be represented by an arithmetic representation with an

arbitrary uniform base A_y, too. For doing this, it is sufficient to chose the S_{Uxi} values in the sequences

$$S_{Ux1}, S_{Ux2}, \ldots S_{Uxi}, \ldots$$

$$A_y, 2A_y, \ldots kA_y, \ldots$$

so that if

$$S_{yk} < S_{xi} \leq S_{yk+1}$$

in the original (general, non-uniform) representation, S_{Uxi} should meet the inequalities

$$kA_y < S_{Uxi} \leq (k+1)A_y.$$

(Of course, the requirement of monotonic character has to be obeyed, too.) It can be seen from the last inequality that if $A_y > 1$, the pixel function can still be constructed in many different ways even with the prescribed uniform base.

An important special case is that where the uniform base is the unit i.e. $A_y = 1$. The former considerations are also valid for this special case, thus, every pixel function has a "unit based" uniform description, too. In this case the defining inequalities for S_{Uxi} have the form

$$k < S_{Uxi} \leq (k+1),$$

or considering that S_{Uxi} is a natural number

$$S_{Uxi} = (k+1).$$

Thus, between the pixel functions and their unit-based representations there is a one-to-one correspondence: a pixel function has a single unit-based representation, and a unit-based arithmetic description defines a single pixel function (not considering the initial pixels).

In the paragraphs above the uniform base was $A_y = const.$ i.e. the base was uniform with respect to y. Similarily there are uniform descriptions with respect to x (and as a special case unit based descriptions with respect to x), too. Their properties need no separate discussion, simply follow from the above investigations considering the perfect symmetry of the two member series with subscripts x and y, respectively.

Fields of differences

Full difference fields

The difference of any two pixel of a pixel function can be computed. Let us consider those differences onely where the minuend does not precede the substrahend in the order of advancing from the starting pixel of the function

to last one. This restriction obviously causes no loss of information, the differences in the revers order have the opposite sign but "don't contain any new information". The set of differences of all possible pixel-pairs taken in the specified order is termed the full difference field of the function.

Obviously, the field of differences does not depend on the starting pixel but on the neighbourhood relations of the subsequent pixels onely. Thus, for constructing the field of differences the starting pixel (i_{x0}, i_{y0}) is not needed only the set of neighbourhood tokens $p_1, p_2, \ldots$ of the basic form or the equivalent pair of concurrent sequences (series) in the arithmetic description has to be known.

Field of differences in direction x

The investigation of a difference field is facilitated by grouping the great many differences according to certain aspects of convenience. Let us consider a column i_x of the finite rectangle containing at least one pixel of the function to be examined. In this column there is either a single pixel belonging to the function or there are more than one such pixels but (due to the assumed monotonic character of the pixel functions) forming an uninterrupted continuous sequence in the order of advancing along the function. Let us pick out of this column the first (or the onely) pixel belonging to the pixel function under investigation and denote it by (i_x, i_y). Let dx assume such an integer value that the column $i_x + dx$ should contain at least one pixel of the function, too. Let us pick out the onely or the first pixel out of this column, too and denote it by $(i_x + dx, j_y)$. The difference of this two pixel "characteristic" to the corresponding columns of the function is the quadruplet

$$(i_x + dx, j_y) - (i_x, i_y) = (dx - 1, j_y - i_y) \quad (dx, j_y - i_y)$$
$$(dx - 1, j_y - i_y - 1) \quad (dx, j_y - i_y - 1).$$

Such difference can be found to any dx and i_x meaningful for the given function. These differences are termed "differences in direction x", and the whole set of all such differences possible for the given function is the field of differences in direction x. Let us classify this pixel differences by considering those with different i_x but identical dx as "coherent", as belonging to the same class. The difference quadruplets of such a class are all in the same columns dx and $dx-1$, so the explicit indication of the columns may be omitted and replaced by a reference to the class. Each difference quadruplet consists of pixels of two adjacent rows, thus it is sufficient to indicate the higher one onely. It is, however, of great importance to know (and to express by the notation) to what initial column the difference belongs, this information is namely lost in the substarction. On the basis of all this considerations the original difference quadruplet may be replaced by a pair of pixels in the column i_x and in

the rows $(j_y - i_y)$ and $(j_y - i_y - 1)$, and for this pair of pixels as a concise expression of the difference in direction x and class dx belonging to column i_x the notation

$$(i_x, (j_y - i_y))dx$$

will be used. The doubling of the right-hand paranthesis emphasizes. that it is a pair of pixels in the rows $(j_y - i_y)$ and $(j_y - i_y - 1)$.

If for a given pixel function the field of differences in direction x is considered, this does. of course, not contain directly all the differences of the full field. Namely, for computing the differences in direction x out of each column onely one pixel has been used. though, a column may contain several pixels of the function. On the basis of the differences in direction x the full field of differences can be all the same "reconstructed" (at least in case of monotonic pixel functions). To demonstrate this it is sufficient to consider that the $(j_y - i_y)$ difference of any two pixels of columns i_x and $i_x + dx$ is either identical to one of the differences in direction x of class $(dx - 1)$ belonging to the column $(i_x + 1)$ and of class $(dx + 1)$ belonging to the column i_x or is between this two, and among the differences of the full field all possibilities matching this condition really occur. Thereby the field of differences in direction x is a complete and unambigous description of the full field.

Ordering pixel differences with respect to x or y

In the previous paragraph pixel differences in direction x have been written in the form $(i_x, j_y - i_y)\ dx$. Thereby a kind of ordering of the pixel differences has been realised: the pixel-pairs standing for the difference-quadruplets of a class have been represented in the column i_x of the finite rectangle. This is the field of differences in direction x ordered with respect to x.

Pixel differences in direction x can also be ordered with respect to y. In this case the values $(j_y - i_y)$, being of primary importance for differences in direction x, are represented in the row i_y rather than in the column i_x of the subtrahend characteristic pixel. Thus the pair of pixels representing the original difference quadruplet of class dx in concise form consists of pixels $((j_y - i_y - 1), i_y)$ and $((j_y - i_y), i_y)$. The corresponding notation of the differences in direction x ordered with respect to y has the form

$$((j_y - i_y), i_y)dx.$$

Investigating pixel functions, fields of differences in direction x ordered with respect to both x and y are of importance. Fields of differences in direction y can be defined similarily, and they can be ordered again with respect to both x and y. They need no special investigation, their properties can be deduced from the results won for the difference fields in direction x by systematically

interchanging the variables x and y. Also in the following the text will refer in general to fields of differences in direction x, but the results will be valid for differences in direction y, too.

Derived pixel functions fitting on fields of differences

As explained in the previous paragraphs the full field of differences of a pixel function can be perfectly accounted for by the field of differences in direction x ordered with respect to x or y. In constructing them the pixel pairs $(i_x, j_y - i_y)\ dx$ or $(j_y - i_y, i_y)\ dx$ have been classified into classes corresponding to dx.

The difference pixel pairs belonging to the same class dx in general do not form a pixel function. In many cases the continuity, i.e. the possibility of stepping form neighbour to neighbour is not available (e.g. in difference fields in direction x ordered with respect to y sometimes there is at all no pixel in some rows). In other cases in contrary there are "too many" pixels (e.g. pixel pairs in neighbouring columns are in the same rows), thus some pixels have more than two neighbours and this is what prevents the simple construction of a pixel function out of the pixels of a field of differences in direction x.

A pixel function containing at least one pixel out of each pair of pixels belonging to class dx of a difference field is called a fitting function of class dx. This definition makes clear that there are many fitting pixel functions of class dx for any difference field.

In the following the mutual relation of a pixel function and the functions fitting on its difference fields will be discussed. The original function will be called the primitive function. The derivatives are a set of pixel functions out of which the first one is a fitting function of class $dx = 1$ on the field of differences of the primitive function, the second one a fitting function of class $dx = 2$, etc. A set of derivatives has to contain $dx_{\max}$ different fitting functions corresponding to the clases $dx = 1, 2, \ldots dx_{\max}$. (The value $dx_{\max}$ is the difference of the x "coordinates" of the last and the first pixel of the primitive function: $dx_{\max} = i_{x\,\mathrm{last}} - i_{x0}$.) Of course the same primitive function has many different sets of derivatives. When working with derivatives one has to define the field of differences on which the derivatives fit. There are derivatives fitting on a field of differences in direction x ordered with respect to x or y, and derivatives fitting on a field of differences in direction y ordered with respect to x or y.

The restriction still holds that monotonic pixel functions (or monotonic sections of general pixel functions) are considered onely. Of course functions fitting on difference fields of monotonic functions themselves are not always monotonic. Thus maintaining the condition of monotonic character for the fitting functions too means a further restriction on the primitive functions.

Correlation between arithmetic description of primitive and derived functions

Let us consider a general arithmetic description of a primitive function in form of a pair of concurrent sequences

$$(i_{x0}, i_{y0})\ S_{x1}, S_{x2}, \ldots, S_{xi}, \ldots$$
$$S_{y1}, S_{y2}, \ldots, S_{yj}, \ldots$$

or in the form of an equivalent pair of concurrent series

$$(i_{x0}, i_{y0})\ A_{x0}, A_{x1}, \ldots, A_{xi-1}, \ldots$$
$$A_{y0}, A_{y1}, \ldots, A_{yj-1}, \ldots .$$

The first pixel of this function in column $(i_{x0} + i)$ is in the row $(i_{y0} + j)$, where j is the highest integer value meeting the inequality

$$S_{yj} < S_{xi}.$$

Knowing this, the differences in direction x can be computed. Let us consider first the field of differences in direction x ordered with respect to x. The pixel pairs of this field are written in the form $(i_{x0}+i,\ k-j)\ dx$ with j defined as above and k being the highest integer meeting inequality

$$S_{yk} < S_{xi+dx}.$$

Let us denote two pixel pairs of the same class dx being in adjacent columns by $(i_{x0}+i,\ k-j)\,dx$ and $(i_{x0}+i+1,\ k'-j')\,dx$. The pixel function fitting on the differences of class dx is sought for in the form

$$(i_{x0}, 0)\ S^{dx}_{x1}, S^{dx}_{x2}, \ldots$$
$$S^{dx}_{y1}, S^{dx}_{y2}, \ldots .$$

The con dition of fitting is the inequality

$$k - j - 1 \leq l \leq k' - j',$$

where l is the highest integer value satisfying the inequality

$$S^{dx}_{yl} < S^{dx}_{xi+1}.$$

A similar system of conditions can be formulated for functions fitting on the differences in direction x ordered with respect to y. The difference pixel-pair of class dx is $(k-j,\ j)dx$. The fitting function of class dx is sought for in this case in the form

$$(0, i_{y0})\ S^{dx}_{x1}, S^{dx}_{x2}, \ldots$$
$$S^{dx}_{y1}, S^{dx}_{y2}, \ldots .$$

Let us determine the highest integer l meeting the inequality

$$S_{xl}^{dx} \leq S_{yj}^{dx},$$

and the highest integer l' meeting the inequality

$$S_{xl'}^{dx} \leq S_{yj+1}^{dx}.$$

Using this l and l' the fitting condition is expressed by the inequality

$$l \leq (k-j) \leq l' + 1.$$

In the case of monotonic primitive and derived functions this is a necessary and sufficient condition simply expressing that in the row № j at least one of the pixels $(k-j, j)$ and $(k-j-1, j)$ belongs to the fitting function.

Beside the general form of the fitting conditions presented just now, there are simpler and more illustrative forms too for special cases. For example if both the primitive function and its derivatives are given by their arithmetic representations of the same uniform base A_y, i.e.

$$S_{yi} = S_{yi}^{dx} = i\,A_y,$$

then

$$j = \operatorname{int}\left(\frac{S_{xi}-1}{A_y}\right),\ k = \operatorname{int}\left(\frac{S_{xi+dx}-1}{A_y}\right),\ l = \operatorname{int}\left(\frac{S_{xi+1}^{dx}-1}{A_y}\right),$$

and the difference

$$k-j = \operatorname{int}\left(\frac{S_{xi+dx}-1}{A_y}\right) - \operatorname{int}\left(\frac{S_{xi}-1}{A_y}\right) = \operatorname{int}\left(\frac{S_{xi+dx}-S_{xi}}{A_y}\right) + b_i^{dx},$$

where b_i^{dx} is either 0 or 1 (being 1 if and only if the residue of the division S_{xi}/A_y exceeds the residue of the division S_{xi+dx}/A_y). Thus the general condition of fitting on a field of differences in direction x ordered with respect to x as deduced in the previous paragraphs

$$k-j-1 \leq l \leq k'-j'$$

takes the form

$$\operatorname{int}\left(\frac{S_{xi+dx}-S_{xi}}{A_y}\right) + b_i^{dx} - 1 \leq \operatorname{int}\left(\frac{S_{xi+1}^{dx}-1}{A_y}\right) \leq \operatorname{int}\left(\frac{S_{xi+1+dx}-S_{xi+1}}{A_y}\right) + b_{i+1}^{dx}.$$

Considering that in this formulae all variables are integers the following transformation is possible:

$$A_y\left[\operatorname{int}\left(\frac{S_{xi+dx}-S_{xi}}{A_y}\right) + b_i^{dx} - 1\right] \leq$$

$$\leq S_{xi+1}^{dx} \leq A_y\left[\operatorname{int}\left(\frac{S_{xi+1+dx}-S_{xi+1}}{A_y}\right) + b_{i+1}^{dx} + 1\right].$$

This inequalities give the rule of constructing the derivatives fitting on the field of differences in direction x ordered with respect to x for the case of arithmetic description with a uniform base common to the primitive function and its derivatives. (Of course, the lower limit on the left-hand side has to be really less than the upper limit on the right-hand side. If not, there is no monotonic derivative. This case has been excluded from the examinations for the time being.) Note that pixel functions with

$$S^{dx}_{xi'} = S_{xi'+dx} - S_{xi'} = \sum_{j=i'}^{i'+dx-1} A_{xj}$$

are derived functions fitting on the field of differences in direction x ordered with respect to x. This can be demonstrated by substituting

$$S^{dx}_{xi'} = S^{dx}_{xi+1} = \sum_{j=i+1}^{i+dx} A_{xj}$$

into the inequalities expressing the "condition of fitting":

$$A_y\left[\operatorname{int}\left(\frac{\sum_{j=1}^{i+dx-1} A_{xj}}{A_y}\right) + b_i^{dx} - 1\right] \leq \sum_{j=i+1}^{i+dx} A_{xj} \leq A_y\left[\operatorname{int}\left(\frac{\sum_{j=i+1}^{i+dx} A_{xj}}{A_y}\right) + b_{i+1}^{dx} + 1\right].$$

The upper limit condition is obviously met, since for any C

$$A_y\left[\operatorname{int}\left(\frac{C}{A_y}\right) + 1\right] > C.$$

In testing the lower limit two circumstances have to be taken into consideration. On one side

$$\sum_{j=i}^{i+dx-1} A_{xj} \leq \sum_{j=i+1}^{i+dx} A_{xj}$$

as consequence of the monotonic character, on the other side b_i^{dx} may equal 1 onely if

$$A_y \operatorname{int}\left(\frac{\sum_{j=i}^{i+dx-1} A_{xj}}{A_y}\right) \leq \sum_{j=i}^{i+dx-1} A_{xj}.$$

Thus the lower limit condition is always met too, and with this the stated form of derivative functions is fully proven.

The conditions for fitting functions become even simpler if a unit base is chosen. With $A_y = 1$ all the denominators vanish, appear integers instead of fractions and all the b values will be identically zeros. Thus the condition of fitting (or the rule of derivation) is simply

$$S_{xi+dx} - S_{xi} \leq S^{dx}_{xi+1} \leq S_{xi+dx-1} - S_{xi+1} + 1$$

in case of unit based arithmetic representation of the primitive function and its derivatives fitting on the field of differences in direction x ordered with respect to x.

Some types of pixel functions

Constants

A pixel function is a constant one, if all of its pixels are neighbours either in direction x or in direction y exclusively. Hence the basic writing of a constant pixel function is either $(i_{x0}, i_{y0}) +x, +x, \ldots$ or $(i_{x0}, i_{y0}) +y, +y, \ldots$ This two possible different kinds of constant pixel functions are called constants with respect to x, or constants with respect to y.

The arithmetic description of a constant pixel function is simple, too. Let us consider a pixel function constant with respect to x, consisting of $m + 1$ pixels. Such a pixel function is described by any pair of concurrent sequences

$$S_{x1}, S_{x2}, \ldots, S_{xm}$$

$$S_{y1},$$

for which $S_{y1} \geq S_{xm}$ (the actual values are not important). The equivalent condition in case of series:

$$\sum_{i=0}^{m-1} A_{xi} \leq A_{y0},$$

or for a uniform base with respect to x

$$A_x \leq A_{y0}/m.$$

This later form is rather illustrative concerning the size dependent character of "constancy" of pixel functions. Practically all pixel functions have —shorter or longer—constant sections. If a pixel function is a constant on a finite rectangle it may be a constant or a non-constant pixel function on an extended one. If in the uniform based description shown above the base is chosen to be equal to zero i.e. $A_x = 0$, there is no need to make any restriction concerning the length of the function i.e. concerning the maximum number of its pixels: in this special case the pixel function is constant with respect to x independently of its size.

Defining types of pixel functions on the basis of properties of their derivatives

It was simple to define the constant pixel functions relying on their basic form. In the following new types of pixel functions will be defined by prescribing some properties of their derivatives. Since for every pixel function there

are many sets of derivatives having different properties, the way of making such prescriptions has to be been specified exactly.

It will be stated that the derivatives of a pixel function have a given property. Thereby it is ment that in each class meaningful for the given pixel function there is at least one fitting function which has the given property (or at least does not contradict to it). Thus not every fitting function needs to have the given property, but must not exist such a class of differences for which not a single fitting function could be produced with the given property. Any subsequent statement or condition concerning the properties of derivatives of a pixel function has to be understood in this sense.

The straight line or linear pixel function

The linear pixel function has constant derivatives. This is the definition of the linear pixel functions i.e. "stright lines" on a finite rectangle consisting of pixels.

Let us consider the following arithmetic description:

$$\begin{array}{llll} S_{x1}, & S_{x2} = S_{x1} + A_x, & \ldots, & S_{xi} = S_{x1} + (i-1)A_x, \quad \ldots \\ S_{y1} = A_y, & S_{y2} = 2A_y, & \ldots & S_{yj} = jA_y, \quad \ldots \end{array}$$

This is termed a doubly uniform based description. It is easy to demonstrate that any pixel function with such a doubly uniform arithmetic description is linear i.e. has constant derivatives. Namely, it has been proven in the previous chapters that the function

$$S_{xi}^{dx} = S_{xi+dx} - S_{xi}$$

is a fitting function in case of a uniform description with the same A_y. But for double uniform description

$$S_{xi+dx} - S_{xi} = S_{x1} + (i + dx - 1)A_x - [S_{x1} + (i-1)A_x],$$

and in consequence

$$S_{xi}^{dx} = dx\, A_x$$

demonstrating that S_{xi}^{dx} does not depend on i, i.e. the fitting function is a constant. It can be demonstrated too that any straight line i.e. any linear pixel function has at least one arithmetic description with doubly uniform base. Before doing that, however, some preliminary remarks are needed.

Let us start with a unit-based description. It has been proven that any pixel function has such a description, thus the linear ones have, too. So it is sufficient to demonstrate that for any unit-based pair of concurrent sequences describing a linear pixel function a corresponding equivalent doubly uniform

description can be constructed. In case of unit base the condition of fitting has been deduced in the form

$$S_{xi+dx} - S_{xi} \leq S^{dx}_{xi+1} \leq S_{xi+dx+1} - S_{xi+1} + 1.$$

Considering that for linear functions the derivatives are constants independent of i, i.e.

$$S^{dx}_{xi+1} = S^{dx}_{x},$$

and all the numbers appearing in the inequalities are integers, the condition of fitting takes the form of the following equation

$$S_{xi+dx} - S_{xi} = S^{dx}_{x} - b^{dx}_{i},$$

where b may take again onely one of the values *0* or *1* (actually which of them, depeds on dx and i). Introducing an "intermediate" dx' value $(1 \leq dx' < dx)$

$$S_{xi+dx} - S_{xi} = S_{xi+dx} - S_{xi+dx'} + S_{xi+dx'} - S_{xi} = S^{dx-dx'}_{x} - b^{dx-dx'}_{i+dx'} + S^{dx'}_{x} - b^{dx'}_{i}$$

Comparing the last two equations havig the same left-hand sides a relation can be found among the S^{dx}_{x} values of the different classes:

$$S^{dx}_{x} = S^{dx-dx'}_{x} + S^{dx'}_{x} - b^{dx'}_{dx}.$$

Let us try to construct now an equivalent doubly uniform-based description in the form

$$\begin{array}{llll} S'_{x1}, & S'_{x2} = S'_{x1} + A_x, & \ldots & S'_{xi} = S'_{x1} + (i-1)A_x, \quad \ldots \\ S'_{y1} = A_y, & S'_{y1} = 2A_y, & \ldots & S'_{yj} = jA_y, \quad \ldots \end{array}$$

The condition of the equivalence of the two forms with unit base and with doubly uniform base is the following pair of inequalities:

$$(S_{xi} - 1)A_y \leq S'_{x1} + (i-1)A_x \leq S_{xi}A_y,$$

or after some transformation

$$(S_{xi} - 1)A_y - (i-1)A_x < S'_{x1} \leq S_{xi}A_y - (i-1)A_x$$

giving the necessary and sufficient condition for the existence of a doubly uniform based description:

$$(S_{xi} - 1)A_y - (i-1)A_x < S_{xj}A_y - (j-1)A_x,$$

which has to be valid for any possible pair of i and j. If $i = j$, the condition is trivial. If $i > j$, then let $i = j + dx$, and the condition takes the form

$$(S^{dx}_{x} - b^{dx}_{j} - 1)A_y < dx\,A_x.$$

If $i < j$, then let $j = i + dx$, and the condition transforms to

$$dx\,A_x < (S^{dx}_{x} - b^{dx}_{i} + 1)A_y.$$

These conditions have to be met in the case of the less favourable values of b_j^{dx} and b_i^{dx}. Thereby to prove the existence of a doubly uniform description such values have be found for A_x and A_y in case of which the inequalities

$$(S_x^{dx} - 1)A_y < dx\, A_x \leq S_x^{dx} A_y$$

are satisfied independently of the actual value of dx. In fact, however, such values can be found always. Namely, let us find the least one among the ratios (S_x^{dx}/dx) and let A_x and A_y equal the corresponding S_x^{dx} and dx. With this choice the upper limit is obviously met. Investigating the lower limit let us consider first the case $dx < A_y$. Usig the formula connecting the S_x^{dx} values of the different classes and considering that $A_x = S_x^1$, in the less favourable case

$$A_x = S_x^{dx} + S_x^{A_y - dx} - 1,$$

from which expressing $(S_x^{dx} - 1)$ and substituting into the lower limit condition, we have

$$(A_x - S_x^{A_y - dx})A_y < dx\, A_x = [A_y - (A_y - dx)]A_x.$$

After some transformation the lower limit condition has the form:

$$\frac{S_x^{A_y - dx}}{A_y - dx} > \frac{A_x}{A_y},$$

which considering the minimum property of A_x/A_y always holds. Of course, with this deduction the proof is valid for dx values not exceeding A_y onely. But always can be found a k value so that

$$1 \leq dx - kA_y \leq A_y,$$

and using the equation

$$S_x^{xd} = kA_y + S_x^{dA_y - x},$$

these cases can be reduced to the already proven ones.

With this conclusion the statement concerning the possibility of describing any straight line in a doubly uniform based arithmetic description has been fully demonstrated. But, in addition, the above (somewhat lenghty) investigation points to some other interesting facts, too. First of all it shows that the stipulation of constant derivatives is a "meaningful" one, there are really pixel functions meeting this condition. It can be observed too that if a finite rectangle is considered with highest serial numbers of subdivisions n_x and n_y, all linear pixel functions of this rectangle (i.e. all straight lines of such a screen) have a doubly uniform arithmetic description with $A_x \leq n_y$ and $A_x \leq n_y$, namely both A_x and A_y have been chosen out of the (on the screen really existing) coordinate differences of two specified pixels of the linear pixel function to be described. (Note that such a description always exists but, of course, is not the onely possible doubly uniform description of the given pixel func-

tion.) The fact that A_x and A_y have such upper limits, has its practical importance: the computer arithmetic may work with rather "short" integers.

Rethinking the above investigations on linear pixel functions may attract notice that it has not been stated explicitely, which derivatives are constants, but tacitly the derivatives fitting on the field of differences in direction x and ordered with respect to x have been used. Since in consequence of the symmetry of the variables x and y in the doubly uniform-based descriptions, any derivatives of a linear pixel functions are constats, there is no need to specify the type of the constant derivatives. The straight lines are "anyway straights".

Concluding the remarks on linear pixel functions a further important property of this type of functions has to be demonstrated. Starting from the well-known doubly unifor description

$$\begin{array}{llll} S_{x1}, & S_{x2} = S_{x1} + A_x, & \ldots & S_{xi} = S_{x1} + (i-1)A_x, \ldots \\ S_{y1} = A_y, & S_{y1} = 2A_y, & \ldots & S_{yj} = jA_y, \ldots, \end{array}$$

let us consider the pixel function described by the following pair of concurrent sequences:

$$\begin{array}{llll} S'_{x1} = \sum_{k=1}^{1+dx-1} S_{xk}, & S'_{x2} = \sum_{k=2}^{2+dx-1} S_{xk}, & \ldots & S'_{xi} = \sum_{k=i}^{i+dx-1} S_{xk}, \quad \ldots. \\ S'_{y1} = A_y, & S'_{y1} = 2A_y, & \ldots & S'_{yj} = jA_y, \quad \ldots. \end{array}$$

This later pair of concurrent sequences marked by' is constructed from the original one by summing some values of the x sequence. The original one is a doubly uniform based arithmetic description of a linear pixel function. It is easy to see that the second one (constructed by summation) is doubly uniform based, too. Namely,

$$S'_{xi} = \sum_{k=i}^{i+dx-1} [S_{x1} + (k-1)A_x] = dx\, S_{x1} + \frac{dx^2}{2} A_x + dx(i-1)A_x,$$

and choosing here

$$S'_{x1} = dx\, S_{x1} + \frac{dx^2}{2} A_x \text{ and } A'_x = dx\, A_x$$

the "normal form" of a doubly uniform based description is arrived at. This property of some pixel functions is told "summation homogeneity", and—as demonstrated just now—the linear pixel functions are homogeneous in this sense.

Pixel functions of order q

Let us assume that functions of order $(q-1)$ exist and they are "summation homogeneous" in the just defined sense. Let the pair of concurrent sequences

$$S'_{x1},\ S'_{x2},\ \ldots\ S'_{xi},\ \ldots$$
$$A_y,\ 2A_y,\ \ldots\ jA_y,\ \ldots$$

describe such a function of order *(q—1)*.

A pixel function is of order q if its derivatives fitting on the field of differences in direction x and ordered with respect to x are of order *(q—1)*.

Let us consider the pixel function

$$A_{x0},\ A_{x1} = S'_{x1},\ A_{x2} = S'_{x2},\ \ldots\ A_{xi} = S'_{xi},\ \ldots$$
$$A_y,\ 2A_y,\ \ldots jA_y,\ \ldots.$$

This is sure a pixel function of order q, its derivatives

$$S^{dx}_{xi} = \sum_{k=i}^{i+dx-1} A_{xk} = \sum_{k=1}^{i+dx-1} S'_{xk},$$

being of order *(q—1)* due to the assumed summation homogeneity. The pixel function of order q itself is summation homogeneous, because its summative is of power q, too. Namely, if

$$S''_{xi} = \sum_{k=1}^{i+dx-1} S_{xk},$$

then its derivatives

$$S''^{dx}_{xi} = S''_{xi+dx} - S''_{xi} = \sum_{i=j}^{j+dx} \sum_{k=1}^{i+dx-1} S_{xk}$$

are of order *(q—1)* due to the assumed summation homogeneity. Now it has been demonstrated that if summation homogeneous pixel functions of order *(q—1)* exist, then pixel functions of order q exist, too, and they are homogeneous in the same sense. This gives us the possibility of constructing pixel functions of any arbitrary (of course positive integer) order. Namely, starting from linear pixel functions the quadratic ones can be constructed, on the basis of the quadratic pixel functions of order two the cubic pixel functions of order 3 can be built up, and so on until any required order q.

The exponential pixel function

A pixel function is told to be exponential if its derivatives fitting on the field of differences in direction x ordered with respect to y are linear.

Let us consider the pixel function with the y uniform based arithmetic description

$$S_{x1},\ S_{x2},\ \ldots\ S_{xi},\ \ldots$$
$$A_y,\ 2A_y,\ \ldots\ jA_y,\ \ldots.$$

This form describes an exponential pixel function if

$$S_{xi+1} = c\,S_{xi}.$$

for any i with a constant c. (Restrictions concerning the value of c will be considered later.) As a verification of this statement it will be demonstrated that the

$$S_{x1}^{dx}, \quad A_x^{dx}, \quad A_y^{dx}$$

values in the doubly uniform-based description

$$S_{x1}^{dx}, S_{x2}^{dx} = S_{x1}^{dx} + A_x^{dx}, \ \ldots \ S_{xi}^{dx} = S_{x1}^{dx} + (i-1)\,A_x^{dx}$$

$$S_{y1}^{dx} = A_y^{dx}, S_{y2}^{dx} = 2A_y^{dx}, \ \ldots \ S_{ym}^{dx} = mA_y^{dx}$$

can be chosen such a way that the resulting linear pixel function fits on the field of differences of class dx (in direction x and ordered with respect to y) of the exponential pixel function written in the above specified form.

First, take notice that if $S_{xi+1} = c\,S_{xi}$, as assumed, then

$$S_{xi+dx} = c^{dx}\,S_{xi}.$$

Thus, the first pixels of the columns i and $(i+dx)$ are in the rows

$$j = \operatorname{int}\left(\frac{S_{xi}-1}{A_y}\right) \text{ and } k = \operatorname{int}\left(\frac{c^{dx}S_{xi}-1}{A_y}\right)$$

respectively. Consequently the difference pixel pair in direction x, ordered with respect to y and being of class dx is

$$((k-j), j)dx,$$

where

$$k - j = \operatorname{int}\left(\frac{c^{dx}S_{xi}-1}{A_y}\right) - \operatorname{int}\left(\frac{S_{xi}-1}{A_y}\right) = \operatorname{int}\left(\frac{(c^{dx}-1)S_{xi}}{A_y}\right) + b_i^{dx}.$$

On the other hand, the first and last pixels of the assumed fitting function of class dx are in the columns

$$l = \operatorname{int}\left(\frac{S_{x1}^{dx} + A_y^{dx}\operatorname{int}\left(\dfrac{S_{xi}-1}{A_y} - 1\right)}{A_x^{dx}}\right), \quad \text{and}$$

$$l' = \operatorname{int}\left(\frac{S_{x1}^{dx} + A_y^{dx}\operatorname{int}\left(\dfrac{S_{xi}-1}{A_y}\right)}{A_x^{dx}}\right),$$

respectively. The condition of fitting, as demonstrated previously, is

$$l \leq (k-j) \leq l' + 1.$$

Using the just deduced formulae for $(k-j)$, l and l_3, making the following choice for the till now undetermined quantities

$$S_{x1}^{dx} = A_y^{dx} = (c^{dx} - 1)\,A_y \quad \text{and} \quad A_x^{dx} = A_y,$$

and taking the less favourable value of b_i^{dx}, the condition of fitting is equivalent with the following pair of inequalities

$$\operatorname{int}\left[(c^{dx} - 1)\operatorname{int}\left(\frac{S_{xi} - 1}{A_y}\right)\right] \leq \operatorname{int}\left(\frac{(c^{dx} - 1)S_{xi}}{A_y}\right),$$

$$\operatorname{int}\left(\frac{(c^{dx} - 1)S_{xi}}{A_y}\right) + 1 \leq \operatorname{int}\left[(c^{dx} - 1)\left(1 + \operatorname{int}\left(\frac{S_{xi} - 1}{A_y}\right)\right)\right] + 1.$$

Obviously, the first condition holds for any $(c^{dx} - 1) \geq 1$. The second one is equivalent to the inequality

$$S_{xi} \leq A_y + A_y \operatorname{int}\left(\frac{S_{xi} - 1}{A_y}\right),$$

where S_{xi} always can be written in the form $S_{xi} = pA_y + q$ with integer p and q, the latter satisfying the condition $0 \leq q \leq A_y$. Using this substitution

$$pA_y + q \leq A_y + A_y \operatorname{int}\left(\frac{pA_y + q - 1}{A_y}\right),$$

and this inequality holds in any case, because if $q = 0$, then

$$pA_y = A_y + A_y(q - 1),$$

and if $0 < q \leq A_y$, then

$$pA_y + q < A_y + A_y p.$$

With this the given form of the exponential pixel functions has been established.

According to the definition of concurrent sequences describing pixel functions their terms have to be (non-negative) integers. In case of exponential functions

$$S_{xi} = c^{i-1} S_{x1},$$

and therefore S_{xi} is integer, if c is integer. But the factor c may have the form

$$c = \frac{e}{f}$$

(where both e and f are integers) too, in condition that S_{x1} is an integer multiple of $f^{(i-1)}$ for any i meaningful in the case of the function under consideration. Theoretically it is always possible to multiply both S_{x1} and A_y by $f^{(i_{\max}-1)}$ and produce an equivalent pair of concurrent sequences describing the same

pixel function and meeting the divisibility condition for S_{x1}. Whether such sequences consisting of (possibly) very large integer terms can be practically useful or not (and if yes, how), will be investigated soon.

Pixel functions and analytic functions

In the previous chapters different types of pixel functions have been defined. The names, which have been given to these types of pixel functions, are used in the traditional mathematical analysis of plane curves, too. The types of pixel functions and the continuous curves of the traditional analysis are, in fact, closely related.

Let us consider a finite rectangle of the (x, y) coordinate plane defined by the inequalities

$$0 \leq x \leq n_x d_x \text{ and } 0 \leq y \leq n_y d_y.$$

The pixel (i_x, i_y) is the elementary rectangle

$$(i_x - 1)\, d_x \leq x < i_x d_x \text{ and } (i_y - 1)\, d_y < y \leq i_y d_y.$$

Obviously, a pixel function can be considered as a counterpart of a continuous analytic function if it consists of those and onely those pixels whose corresponding elementary rectangles are "passed through" by the continuous analytic function. From this viewpoint the values of the analytic function on the boundaries of the elementary rectangles corresponding to the pixels are of importance, behaviour of the continuous functions inside the elementary rectangles can not be expressed by the pixel function.

Let us investigate now the "legitimacy" of some names given to the different types of pixel functions. The correspondence of the constant pixel functions and the $x = C$ or $y = C$ analytic functions is trivial. The analytic counterpart of the linear pixel functions, among others, may have the form $y(x) = a\,x + b$. The difference of the y values on the pixel boundaries x and $(x + dx\ d_x)$ is

$$y(x + dx\, d_x) - y(x) = dy_{dx} = a\, dx\, d_x$$

and does not depend on x, is a constant. The analytic counterpart of the quadratic pixel function (i.e. of the pixel function of order 2) is the second order polynomial $y = a\,x^2 + b\,x + c$ (parabola), because

$$dy_{dx} = (2a\, dx\, d_x)\, x + (a\, dx^2\, d_x^2 + b\, dx\, d_x)$$

is a straight line. Similarily, for $y = e^x$

$$dy_{dx} = e^{x + dx\, d_x} - e^x = e^x (e^{dx\, d_x} - 1) = y\,(e^{dx\, d_x} - 1)$$

being a linear function of y, the analytic function $y = e^x$ is a counterpart of the exponential pixel function.

The different types of the analytic functions are characterised by the corresponding differential equations e.g. the quadratic ones by the equation $y' = \frac{a}{2}x + b$, the exponentials by $y' = c\,y$, etc. The "pixel counterparts" of this differential equations are the prescribed properties of pixel derivatives fitting on the field of differences in one or in the other direction and ordered with respect to one or to the other variable. From the aspect of analytic representation these prescribed properties refer to "finite differences" taken on the pixel boundaries. The traditional usage of the equations of finite differences considers the differences taken in distances d_x onely, therefore being an approximation of the differential equation, while the formulation of the pixel derivatives considers the finite differences of all possible classes simultaneously, therefore is not an approximation but an exact full description of the pixel counterpart of the required analytic functions (at least for monotonic sections and inside the finite rectangle under investigation).

The pixel functions as defined in the previous chapters are perfectly exact in the sense, that they contain all those pixels and onely those pixels which are passed through by their "theoretical" analytic counterparts. If, say, the exponential pixel function had to be displayed on a raster screen by computing the e^x values on the boundaries of the columns, this, possibly, wouldn't be exact, namely for computing the e^x values some approximative methods had to be used, the arithmetic unit of the computers (at least in case of multiplications or floating point arithmetics) worked with a restricted accuracy, etc. Thus, in case of such a traditional approach the appearance of "illegal" pixels or the absence of needed ones is not absolutely excluded, while the arithmetic descriptions working with integers and operating with additions and substractions onely give always the exact result.

Algorithms for plotting pixel functions

Plotting functions given in their basic form

Plotting a function is understood as the process of finding all the pixels making up the function from the starting pixel to the last one by stepping from neighbour to neighbour, in order to mark out them on the screen by giving them a shade or colour different from other pixels not belonging to the function, and thereby to display the considered function. The way, how the pixels can be set to a given colour or shade, is different in the case of the different computers and graphic softwares and is irrevelant from the point of view of the present investigations. Thus, a polotting algorithm is understood in this paper simply as a process consisting of as many steps as many pixels

constitute the considered function, the integer variables i_x and i_y defining in each step a pixel (i_x, i_y) of the function and one (and onely one) of them varying from step to step by ± 1 corresponding to the way of adjacency of the consecutive neighbouring pixels. In case of actual computer algorithms the instructions to set the pixels have to be added.

Let us consider the basic form

$$(i_{x0}, i_{y0})\, p_1, p_2 \ldots p_j, \ldots$$

of the fuction to be plotted. Remember, that a "neighbourhood bit pair" p_j represents two bit of information (b_{j1}, b_{j2}), where, say, the bit b_{j2} expresses wether the next pixel is adjacent in the direction x or y and the bit b_{j1} is the "sign" defining wether the next pixel lies above (to the right of) or below (to the left of) the previous one. Thus the basic form of a pixel function is equivalent to the following plotting algorithm:

— "*Starting step*":

$$i_x = i_{x0},\ i_y = i_{y0},\ j = 1.$$

— "*Going on step*":

$$\textit{if } b_{j2} = 0, \textit{ then } i_x = i_x + sgn(p_j),\ j = j + 1;$$

$$\textit{if } b_{j2} = 1, \textit{ then } i_y = i_y + sgn(p_j),\ j = j + 1.$$

$$(\textit{Note, if } b_{j1} = 0, \textit{ then } sgn(p_j) = +1,$$

$$\textit{if } b_{j1} = 1, \textit{ then } sgn(p_j) = -1.)$$

The starting step specifies the initial values of the variables used in the algorithm and is executed once while plotting a function. The going on step is repeatedly executed as many times as the number of pixels in the function (not counted the starting one). In this step the values of the variables are modified in dependence of some conditions. A going on step has essentially two branches corresponding to the x or y adjacency of the pixel just stepped on. For describing plotting algoritms the same structure will be used in the further investigations, too. (The plotting algorithms to be investigated are all very simple, therefore there is no need to take use of the strict syntactics of a specific programming language to avoid misunderstanding. It is to be mentioned, however, that the equal sign "=" has two different roles in the descriptions of plotting algorithms. It is used mainly as the sign of an "assigment statement" meaning that the variable specified on its left hand side takes the value prescribed on its right hand side. An exceptional case for using this "=" sign is between the words "if" ... "then" in the conditions defining the x or y branches, where it means a "relational operator", but this special meaning is clear out of the wording.)

Plotting functions given by arithmetic description

Let us consider a pixel function given by the pair of concurrent series

$$(i_{x0}, i_{y0})\ A_{x0}, A_{x1}, \ldots A_{xj}, \ldots$$
$$A_{y0}, A_{y1}, \ldots A_{yk}, \ldots .$$

The corresponding plotting algorithm is as follows:

— "*Starting step*":

$$R = A_{x0} - A_{y0},\ i_x = i_{x0},\ i_y = i_{y0},\ j = 1,\ k = 1.$$

— "*Going on step*":

if $R \leq 0$, *then* $i_x = i_x + 1, \quad R = R + A_{xj}, \quad j = j + 1$;

if $R > 0$, *then* $i_y = i_y + 1, \quad R = R - A_{yk}, \quad k = k + 1$.

(This plotting algorithm works for monotonic pixel functions onely as the arithmetic description itself has been defined so far just for such cases.)

In the algorithm presented just now, the variable R, called the main register, plays an important role. This main register indicates always the difference of the not yet plotted next terms of the x and y sequences, since in the starting step it is set so (remember $A_{x0} - A_{y0} = S_{x1} - S_{y1}$!) and in the going on steps (where a pixel is plotted and the corresponding term of either the x or the y sequence is used up) it is modified by the difference of the next and the just plotted terms, being

$$A_{xj} = S_{xj+1} - S_{xj} \text{ and } A_{yk} = S_{yk+1} - S_{yk}.$$

Thus, the main register R tells whether the next term in the common order of the two sequences is out of the x or the y sequence and, consequently, whether the next pixel of the function is adjacent to the last already plotted one in the x or in the y direction.

Comparison of plotting algorithms

In the previous paragraphs two types of plotting algorithms have been considered, one using the basic description and the other using the arithmetic description of the function. There is an important difference between this two types of algorithms regarding the amount of information needed by them.

The "coordinates" of the starting pixel are needed by both types of plotting algorithms, thus it can be disregarded when comparing the amount of information. The algorithm relying on the basic description uses a p_j "neighbourhood bitpair" to plot a new pixel of the pixel function, so the amount of information needed to construct a pixel of the function is 2 bit. The algorithm based on the arithmetic description uses up an A_{xj} or an A_{yk} value for construct-

ing each new pixel of the function. The A_{xj} and A_{yk} values are (generally rather large) integers representing as many bit information as many binary digits are necessary to describe them. Thus, the amount of information needed if using the arithmetic description is several times bigger than in the case of the basic description. The basic description is practically not redundant, contains—broadly speaking—as many information as just needed for selecting a given pixel function out of the manifold of the possible functions and, consequently, the corresponding algorithm seems to be "the best possible". On the other hand, arithmetic descriptions are rather redundant and plotting algorithms relying on them seem to be less efficient, using up much more information to produce the same result.

Really, were all possible pixel functions of the given finite rectangle (i.e. of the given screen) "equal", were not some pixel functions (or better, some groups of pixel functions with some special properties) more important than others in describing natural fenomena, the arithmetic descriptions and the corresponding algorithms had no justification. However, in the previous chapter on the different kinds of pixel functions the formulation of pixel function properties important in expressing natural fenomena has been possible just by taking use of the arithmetic description. In case of such functions with important characteristic properties the A_{xj} and A_{yk} do not appear as independent data but exhibit some regularities, and therefore can be constructed out of a few initial data. In what follows algorithms taking use of these regularities will be presented. The efficiency of such algorithms is rather high, knowing the information concerning the type of the function just a few additional initial data are needed to construct the whole pixel function.

Algorithm for plotting straight lines

Straight lines—as demonstrated earlier—are described by concurrent series of doubly uniform base

$$(i_{x0}, i_{y0})\ A_{x0}, A_x, \ldots A_x, \ldots$$
$$A_y,\ A_y, \ldots A_y, \ldots .$$

Considering this special type of arithmetic description the plotting algorithm of straight lines follows directly from the general case:

— "*Starting step*":

$$R = A_{x0} - A_y,\ i_x = i_{x0}, i_y = i_{y0}.$$

— "*Going on step*":

$$\textit{if } R \leq 0, \textit{ then } i_x = i_x + 1, \quad R = R + A_x;$$
$$\textit{if } R > 0, \textit{ then } i_y = i_y + 1, \quad R = R - A_y.$$

Thus, five numerical data are needed to define a straight line: A_{x0} (or, better the difference $A_{x0}-A_y$), A_x, A_y, and the coordinates of the starting pixel (i_{x0}, i_{y0}). The algorithm for plotting straight lines may be written in an alternative form concentrating data handling in the starting step:

— "*Starting step*":

$$R = A_{x0} - A_y, \quad R_{x1} = A_x, \quad R_{y1} = A_y, \quad i_x = i_{x0}, \quad i_y = i_{y0}.$$

— "*Going on step*":

if $R \leq 0$, *then* $i_x = i_x + 1$, $R = R + R_{x1}$;

if $R > 0$, *then* $i_y = i_y + 1$, $R = R - R_{y1}$.

It should be noted that algorithms of this kind (deduced from different theoretical bases) are well-known and widely used in the practice.

Algorithm for plotting quadratic pixel function

It has been demonstrated that a pixel function of order q is described by a pair of concurrent series with a uniform base with respect to x if the terms of the x series are identical with the terms of an x sequence out of a pair of concurrent sequences with the same uniform base describing a function of order $(q-1)$. Therefore, the pair of concurrent series

$$\begin{array}{l} (i_{x0}, i_{y0})\ A_{x0}, A_{x1}, \ \ldots A_{xj}, \ \ldots \\ \qquad\qquad A_y, \ A_y, \ \ldots A_y, \ \ldots \end{array}$$

describes a pixel function of order 2 if the terms $A_{x1}, \ \ldots \ A_{xj}, \ \ldots$ are equal to the terms

$$S'_{x1} = A'_{x0}, S'_{x2} = A'_{x0} + A_x, \ \ldots \ S'_{xk+1} = A'_{x0} + kA_x, \ \ldots$$

of an x sequence corresponding to the doubly uniform base series description

$$\begin{array}{l} (i_{x0}, i_{y0})\ A'_{x0}, A_x, \ \ldots \ A_x, \ \ldots \\ \qquad\qquad A_y, \ A_y, \ \ldots \ A_y, \ \ldots \end{array}$$

of a linear pixel function. Thus, a quadratic pixel function has a pair of concurrent series in the form

$$\begin{array}{l} (i_{x0}, i_{y0})\ A_{x0}, A_{x1} = A'_{x0}, A_{x2} = A'_{x0} + A_x, \ \ldots \ A_{xk+1} = A'_{x0} + kA_x, \ \ldots \\ \qquad A_y, \qquad\quad A_y \qquad\qquad\quad A_y, \ \ldots \qquad\qquad\qquad A_y, \ \ldots \end{array}$$

and hence the plotting algorithm for quadratic pixel functions is the following:

— "*Starting step*":

$$R = A_{x0} - A_y, R_{x1} = A'_{x0}, R_{x2} = A_x, R_{y1} = A_y, i_x = i_{x0}, i_y = i_{y0}.$$

— *"Going on step"*:

if $R \leq 0$, *then* $i_x = i_x + 1$, $R = R + R_1$, $R_{x1} = R_{x1} + R_{x2}$;

if $R > 0$, *then* $i_y = i_y + 1$, $R = R - R_{y1}$.

Algorithm for plotting pixel functions of order q

An obvious generalisation of the plotting algorithm of the quadratic pixel functions leads to the following algorithm for plotting the general pixel functions of order q:

— *"Starting step"*:

$$R = A_{x0} - A_y,\ R_{x1} = A'_{x0},\ R_{x2} = A''_{x0},\ \ldots\ R_{xq-1} = A_{x0}^{[q-1]},$$
$$R_{xq} = A_x^{[q-1]},\ R_{y1} = A_y,\ i_x = i_{x0},\ i_y = i_{y0}.$$

— *"Going on step"*:

if $R \leq 0$, *then* $i_x = i_x + 1$, $R = R + R_{x1}$, $R_{x1} = R_{x1} + R_{x2}, \ldots$
$R_{xq-1} = R_{xq-1} + R_{xq}$;

if $R > 0$, *then* $i_y = i_y + 1$, $R = R - R_{y1}$.

In accordance with the recursive definition of the pixel function of order q (not considering the starting pixel), the following initial data appear in this algorithm:

$$A_{x0},\ A'_{x0},\ A''_{x0},\ \ldots\ A_{x0}^{[q-1]},$$

i.e., the first terms of the x series of pixel functions of order $[q-1]$, $[q-2]$, $\ldots$, 2, 1, resp.,

$$A_x^{[q-1]},$$

the general x term of the linear pixel function, and the common uniform base

$$A_y.$$

However, for the practical application of this algorithm it is not always important to remember the recursive definition of the pixel function of order q, and to define the initial values of the registers corresponding to this definition; simply all registers need initial values, and if they have got, the algorithm works.

Algorithm for plotting exponential pixel functions

For the exponential pixel function an arithmetic description of uniform base with respect to y has been constructed with

$$S_{xi+1} = c\, S_{xi}$$

for every i with the same constant c. Considering that

$$A_{xi} = S_{xi+1} - S_{xi} = (c-1)S_{xi}, A_{xi+1} = (c^2 - c)S_{xi}, \text{ i.e. } A_{xi+1} = cA_{xi},$$

the following "going on step" can be constructed

— *"Going on step":*

$$\text{if } R \leq 0, \text{ then } i_x = i_x + 1,\ R = R + R_{x1},\ R_{x1} = c\,R_{x1};$$

$$\text{if } R > 0, \text{ then } i_y = i_y + 1,\ R = R - R_{y1}.$$

So far, the plotting algorithms contained additions onely. Therefore it is, expedient to replace the multiplication $c\ R_{y1}$ appearing in the plotting algorithm of the exponential function by additions, too. Assuming that all (initial and subsequent) values of the register R are integer multiples of $R_{y1} = A_y$, and assuming further that $R_{x2} = (c-1)\ A_y$ is an integer (as we have seen, c may be a fraction), a plotting algorithm for the exponential pixel fuction can be constructed in the following form:

— *"Starting step":*

$$R = 0,\ R_{x1} = A_{x0},\ R_{y1} = A_y,\ R_{x2} = (c-1)A_y,\ i_x = i_{x0},\ i_y = i_{y0}.$$

— *"Going on step":*

$$\text{if } R \leq 0, \text{ then } i_x = i_x + 1,\ R = R + R_{x1};$$

$$\text{if } R > 0, \text{ then } i_y = i_y + 1,\ R = R - R_{y1},\ R_{x1} = R_{x1} + R_{y2}.$$

The structure of this plotting algorithm has much in common with procedures presented earlier for the linear, quadratic and general polynomial pixel functions: it realizes the plotting of the exponential pixel function by using register additions onely. However, there is a characteristic difference: while in case of polynomials all registers with subscript x have to be modified in the x branch of the going on step, in case of he exponential function the register R_{x1} is modified in the y branch, i.e. the handling of the registers with subscripts x and y is not strictly seperated any more. In our method of discussing functions this expresses the essential difference between "polynomials" and "transcendent" functions.

Finally some practical remarks on plotting exponential pixel functions. Remember that the last version of the plotting algorithm has been based on assumptions concerning the divisibility of the R register values by R_{y1}, and R_{y1} by the denominator of c (if c is a fraction). Obviously, these stipulations always can be met using common multiples, but for that somtimes rather big (long) integers are needed. If (e.g. in order to avoid a slowing down of the algorithm) the integers are not used in their full length required by the divisibility assumptions, the algorithm will be of an approximative character. There is the possibility of using a pair of such "fast" approximative algorithms paralel,

one of them giving a lower the other one an upper bound of the function to be plotted and as far as the two approximations coincide, the result is exact. Sometimes such a pair of approximative algorithms works faster than a single exact algorithm using long integers.

Systematization of pixel functions. The function number

In the foregoing chapters—somewhat as examples—plotting algorithms have been given for polynomial and exponential pixel functions. The structure of this algorithms lends itself to classification of the different types of pixel functions. The "starting step" of the algorithms is irrevelant from this point of view; in this step the i_x and i_y "coordinates", the main register R, and all the x and y registers used in the algorithm just get their initial values. The type of the function is defined by the structure of the "going on" step, which is always divided into two branches depending on the conditions whether $R \leq 0$ or $R > 0$, resp., the first one being the x branch, the other one the y branch. In the x branch i_x, in the y branch i_y is increased by one. These operations are common for any type of (monotonic) pixel functions. Beside these coordinate modifications the values of some registers may be modified too by adding the content of an other register to the existing value. The type of the pixel function is defined just by these register modifications: which register is modified, by which other register and in which branch. A substantial and unambiguous definition of the different types of pixel functions is possible by a concise description of the register modifications in the two branches of their plotting algorithms.

In this concise description the two branches will be separated by a colon " : ", preceded by the list of register modifications in the x branch, and followed by the list of register modifications in the y branch. (The explicite indication of the coordinate modifications is not needed, they are common in every plotting algorithm.) The register modifications themselves can be expressed in an unambiguous, equivalent shorter form, too. If the main register is understood with the subscript zero, the letter symbol R may be omitted using the subscripts onely. Remember that in general the subscript of a register contains one of the letters x or y and a positive integer number. Agreeing that the subscript integer is taken with positive sign if the subscript letter is the same as the letter symbol of the branch in which it appears, and with a negative sign in the opposite case, the explicite indication of the subscript letter my be omitted, too. In the assignment statements of register modifications the register to be modified appears on the left hand side of the equation sign "=", but it appears once more on the right hand side too, indicating that the modification has to be added to (or substracted from) the existing value. In a concise description of the register modifications this repeated writing can be

avoided by agreeing that the symbol $i \mid j$ is equivalent to the assignment statement $R_{xi} = R_{xi} + R_{xj}$ (if it precedes the :) or to $R_{yi} = R_{yi} \pm R_{yj}$ (if it follows the :). (Remember that in the assignment statements of the plotting algorithm the modification is always an addition, only in $R = R - R_{y1}$ appears a subtraction !) Using the introduced conventions, the plotting algorithms of some types of pixel functions have the following concise form

$0 \mid 1 : 0 \mid 1$	(straight line),
$1 \mid 2,\ 0 \mid 1 : 0 \mid 1$	(parabola),
$(q-1) \mid q \ldots 1 \mid 2,\ 0 \mid 1 : 0 \mid 1$	(polynomial of order q),
$0 \mid 1 : 0 \mid 1,\ -1 \mid 2$	(exponential).

In case of many important functions, series of assignment statements can be found where the subscript of the modifier register is in each statement by one greater than that of the register to be modified and the modifier of the first assignment statement is modified by the next statement of the series, and so on. Such series of assignment statements appear e.g. in the plotting algorithms of polynomials. For these type of series of assignment statements a concise notation can be introduced by naming the lowest subscript to be modified, and indicating the number of assignment statements belonging to the series as a superscript, as an "exponent". Thus, e.g. the concise writing of the plotting algorithm of a polynomial of order q is

$$0^q : 0^1.$$

Note that if the superscript is 1, this notation is equivalent to a single assignment statement. Let us agree that, as in the case of "normal" exponents, the explicite writing out of the superscript 1 may be omitted.

By introducing the "power type" notation with "bases" and "exponents", the naming of registers has been somewhat automatized: onely the lowest subscript is mentioned explicitly as the "base". For avoiding ambiguities, some strict rules for giving subscripts to registers have to be established. Let consider the assignment statements always starting from the colon, i.e. on the right-hand side in the "normal way" (from left to right), but on the left-hand side of the colon in the opposite way (from left to right). Each register gets its letter subscript corresponding to the branch in which it appears first as a modifier, and gets as serial number subscript the next free integer, not used yet with the specified letter subscript (corresponding to the just established order of the assignment statements). Every modifier register involved by a "power type" notation is considered as a new register, being not identical with any previously specified one. (If it happens that this is not the case, the concise "power type" notation must not be used, and the corresponding assignment statement has to be written out explicitly.) Finally one can observe that the assignment statement $0 \mid 1$ or 0^1 appears in both branches of any plotting algorithm, and there-

fore it must be always included even if it is not writen out explicitly. All these rules define a very concise, informative, and in the same time strictly unambiguous notation of pixel functions (through their plotting algorithms) which will be called "function number".

Some important function numbers

The function numbers of function types discussed in detail in the

$0^1 : 0^1$ or	$0 : 0$	(linear),
	$0^2 : 0$	(quadratic or parabola with a vertical axis),
	$0^q : 0$	(polynomial of order q),
$0^1 : 0^1, -1^1$ or	$0 : -1$	(exponential).

The function numbers of inverse functions can be obtained by interchanging the two branches in the plotting algorithm, i.e. by interchanging the two sides of the colon in the function numbers:

$0 : 0^2$	(square root or parabola with a horizontal axis),
$-1 : 0$	(logarithmic).

Concerning the above mentioned pixel functions their definition has been given in the foregoing chapters, the equivalence of the plotting algorithm and the given definition has been demonstrated, the mutual relation of the analytic and pixel functions has been studied, thus the legitimacy of the function numbers has been somewhat established. Now some more interesting function numbers will be listed without proof:

$0^2 : 0^2$	circle, ellipse, parabola, hyperbola with principal axes parallel to the coordinate directions,
$-1, \quad 0^2 : 0^2, -1$	conic sections, as above, but with arbitrary (skew) axes, plus spirals, etc.,
$-2, 0^1 : 0^2$ or $-2 : 0^2$	the basic trigonometric and hyperbolic functions (sin, cos, sh, ch).

A pixel function is defined by its function number and by the initial values of all registers involved in it (including i_x and i_y). Note that there are registers which appear as modifiers onely. If the initial value of such a register is set to 0, the assignment statements involving this regigster may be omitted. i.e. the function number may be reduced to a simpler form. For example if

$R_{y2} = 0$ in a pixel function with function number $0^2 : 0^2$, then it reduces to the $0^2 : 0^1$ parabola.

A function number comprises a more or less wide range of pixel functions, depending—roughly speaking—on the number of the involved registers. For example the function number -2^2, -1^2, $0^3 : 0^3$, -1^2, -2^2 comprizes all pixel functions mentioned in this paper plus the pixel equivalents of many "special functions" having important applications in natural sciences and technology.

Possibilities of generalisation and other complementary remarks

Non-monotonic functions

In this section some remarks will be presented concerning the possible generalizations, the indispensable further investigations and the expectable fields of practical application. Many aspects will be just hinted at without detailed investigation and justification, leaving these to future papers.

The most restrictive assumption used till now has been, no boubt, that of the monotonic character of the considered pixel functions. For instance, the algorithms implied by the function numbers of the quadratic or of the sine pixel functions (at least in the form as they have been presented in the foregoing chapters) are able to plot with full certainity the monotonic sections of these functions onely. The possibilities of releasing this restriction will be studied on the case of the quadratic pixel function as the simplest characteristic example.

The quadratic pixel function consists of two monotonic parts, one of them being an increasing, the other one a decreasing one. These two parts are joined by the extremum point of the function (being a minimum or a maximum depending on the sequential order of the increasing and decreasinging parts). The plotting algorithm has been constructed directly for the increasing part of the function, but—in a slightly modified form—it can describe the decreasing part, too, namely in the y steps the i_y coordinate instead of being increased has to be decreased by one. Thus both for the increasing part and for the decreasing part there is a plotting algorithm available. At least formally, it is easy to combine these two separate algorithms into a single one. There is no change in the starting step of such a common plotting algorithm, while the combined "going on step" will be as follows:

— *"Going on step":*

$$\text{if } R \leq 0, \text{ then } i_x = i_x + sgn(R_{y1}),\ R = R + |\, R_{x1} \,|\,,$$
$$R_{x1} = R_{x1} + sgn(R_{y1}).\ R_{x2};$$
$$\text{if } R > 0, \text{ then } i_y = i_y + sgn(R_{x1}),\ R = R - |\, R_{y1} \,|\,.$$

In this combined algorithm the two monotonic parts of the function are characterized by the sign of the R_{x1} register: a positive value in the R_{x1} means that the function increases, a negative one that it decreases. It is interesting to note that the main register is modified always by the absolute value of the registers R_{x1} or R_{y1} corresponding to the fact that the terms in any pair of concurrent series describing a pixel function are always positive.

The above form of the algorithm is somewhat more general, than needed to describe the two monotonic parts of the quadratic pixel function, namely it allows for a change in sign not onely in case of the register R_{x1} but also in case of the register R_{y1}. This means that the parabola can be plotted not onely from left to right but, by giving a negative value to R_{y1}, from right to left, too. In case of parabolas this is not really important, but for continuous drawing of closed curves (e.g. circles) such a reverse plotting is inevitable on some parts of the curve. In general, the sign of the main register R specifies whether an x or a y step is needed to plot the next pixel of the function, while the sign of the registers R_{x1} and R_{y1} choses between the up—down or right—left neighbours.

No doubt, the combined algorithm can plot two monotonic parts, one after the other. It has been strictly demonstrated that both of these parts are parabolic. However, it can be asked (1) whether these two parts do belong to the same parabola or not, and (if they do) (2) whether the two plotted parts do contain all the pixels of the whole parabola (including its extremal point, too) or not? The answer to the first question is yes, but concerning the second question there is no definite positive answer. The combined algorithm plots two monotonic parts (with adjoining pixels) of the same theoretical parabola, but it may happen in some ill-conditioned situations that the pixel containing the theoretical extremum is missing (even together with some of its neighbours in the same column). The probability of such ill-conditioned situations and the practical significance of the accidentally missing pixels can always be diminished by using a finer mash in the proximity of the extrema i.e. for low absolute values of the register R_{x1}.

Pixel curves in the three-dimensional space

The pixel functions investigated in the foregoing chapters are the pixel counterparts of the analytic plane curves. The concept of the pixel functions can be generalized for the three-dimensional space, too and one can define the pixel counterparts of the 3D curves and 3D surfaces as well. For doing that, first of all the finite plane rectangle with its finite subdivisions has to be generalized into a finite hexahedron divided by three appropriate sets of planes into elementary hexahedra. Such an elementary hexahedron is a "3D pixel". A 3D pixel has—in general—six neighbours, and can be characterized by a triplet of integer numbers (i_x, i_y, i_z). Using this form, its six neighbours are

$$(i_x + 1, i_y, i_z), \qquad (i_x - 1, i_y, i_z),$$
$$(i_x, i_y + 1, i_z), \qquad (i_x, i_y - 1, i_z),$$
$$(i_x, i_y, i_z + 1), \qquad (i_x, i_y, i_z - 1).$$

A pixel curve in the three-dimensional space is a strict analogue of the plane pixel functions: one has to choose a 3D starting pixel and to procede by stepping from neighbour to neighbour. The arithmetic description of the monotonic pixel functions is valid, too, onely one has to use (instead of a pair) a triplet of concurrent sequences or series. The three sequences correspond to the three coordinate directions, and one has to chose the neighbouring pixels in the x, y or z direction as the x, y or z terms follow each other in the common order of the three sequence. This very straightforward method of generalisation indicates that the concept of pixel function can be easily extended not just for the 3D case but for an arbitrary n-dimensional space, too.

The arithmetic description of monotonic pixel functions by means of monotonic concurrent sequences and series can be interpreted in many different forms. A possible interpretation, expedient from the point of view of further generalizations, is the following. Let us consider a starting pixel (i_{x0}, i_{y0}, i_{z0}), and a general pixel in the form

$$(i_x, i_y, i_z) = (i_{x0} + i, i_{y0} + j, i_{z0} + k), \text{ with } i, j, k > 0.$$

In the x, y and z sequences the correspondig terms are

$$S_{xi}, S_{yj}, S_{zk},$$

and these, together with their predecessors define three interwalls of natural numbers

$$S_{xi-1} \leq N \leq S_{xi}, S_{yj-1} \leq N \leq S_{yj}, S_{zk-1} \leq N \leq S_{zk}.$$

Let us associate with the pixel $(i_x, i_y, i_z) = (i_{x0} + i, i_{y0} + j, i_{z0} + k)$ the just described three corresponding intervals. The pixel belongs to the 3D pixel curve if all three associated intervals have common parts, i.e. if the three associated intervals are mutually overlapping intervals. This definition is, obviously, nothing else than an alternative formulation of the arithmetic description rule. (Note that the aim of this presentation is just to outline some possibilities of generalization and many details need a more exact formulation: e.g. what is the predecessor of the first term of a sequence? how to avoid ambiguity if two terms of different sequences happen to be equal? etc. To give exact answers to all these questions is out of the scope of the present paper.)

The definition of pixel curves by three associated overlapping intervals has been introduced in the previous paragraph as an alternative interpretation of the generalised arithmetic description. On the other hand, it can be considered as an independent method of describing pixel curves, too. In this case simply

three intervals of natural numbers are associated to each pixel of the finite hexahedron, and those pixels constitute a 3D pixel function for which the three associated intervals mutually overlap. Whether these "generalised" pixel functions are really pixel curves in the sense of the "step from neighbour to neighbour" definition or not, depends on the rule by means of which the intervals are associated to the individual pixels. To any 3D pixel curve there are several different sets of association rules describing it, as there are several equivalent arithmetic descriptions, too. In case of any set of association rules describing a real 3D pixel curve a corresponding arithmetic description can be found.

Pixel surfaces

The method of associated overlapping intervals can be used readily to describe 3D pixel surfaces, too. In this case two intervals of natural numbers are assigned to each 3D pixel, and the pixel surface is the set of those pixels for which the two associated intervals overlap. In general, a subspace of dimension m of an euclidean space of dimension n can be specified by assigning $(n+1-m)$ intervals to each n-dimensional pixel and by finding those pixels for which all $(n-m+1)$ intervals have common parts, i.e. all possible pairs of the $(n-m+1)$ intervals are overlapping.

The method of associated overlapping intervals lends itself for solving different problems of surface geometry. Let us consider, for example, two surfaces, each having two sets of associated intervals. If one of the sets of associated intervals of the first surface is identical with one of the sets of associated intervals of the second surface then there are out of the four original sets of associated intervals onely three which are really different, and these three different sets of associated intervals define (under certain conditions) a 3D curve, being, obviously, the line of intersection of the two surfaces. But there are always a great many equivalent sets of associated intervals for any surface, and it is always possible to find for any two surface associated overlapping interval descriptions with identical pairs of interval sets. Thus the intersection problem of surfaces is essentially solved. For taking practical use of this theoretical possibility the transformation rules between equivalent descriptions have to be established and thereby the forming of descriptions with identical pairs of interval sets facilitated.

The association rules of the overlapping intervals are, in case of practically important surfaces, very similar to the "plotting algorithms" of plane pixel curves, as expressed by the function numbers. This means with other words that the function numbers themselves can be generalized for describing 3D surfaces and curves, and the rules of identical transformations can be formulated for these generalised function numbers. Under such circumstances,

the construction of descriptions with identical pairs of associated intervals and therefore the solution of the intersection problem of surfaces is a matter of "function number arithmetic". Many other problems of surface geometry, such as finding curves drawn on a given surface, constructing 2D projections of 3D surfaces (including the determination of contours), etc. can be treated in a similar way.

The generalized multiplication-division

The plotting algorithms (in the form as they have been presented in the foregoing chapters) are able to find all pixels of a pixel function, but onely one after the other, stepping from neighbour to neighbour. If we ask (e.g. in 2D) what is the i_y coordinate of a pixel belonging to a given function and lying in the column i_x (if there is any), all intermediate pixels between the starting pixel and the column i_x have to be generated. To avoid this a more general form of the algorithm is needed.

Let us consider first a linear pixel function given by its starting pixel (i_{x0}, i_{y0}) and by its doubly uniform arithmetic description. While proceeding from the starting pixel to the column i_x, the plotting algorithm adds the A_x value $(i_x - i_{x0})$ — times to the initial value of the main register, in between always substracts the value A_y as soon as possible and counts how many times can the substraction be fulfilled. The number of the possible substractions plus i_{y0} gives the required i_y coordinate. Working in a binary (or in any other positional) notation, the repeated addition can be replaced by multiplication and the repeated substraction by (integer) division:

$$i_y = i_{y0} + [R_{init.} + (i_x - i_{x0})\, A_x] \div A_y,$$

i.e. the i_y value belonging to a given i_x can be computed by means of a multiplication and a subsequent division (at least in the case of linear pixel functions).

No doubt, this pair of a multiplication and a division operation takes much less computer time on the usual arithmetic devices than the repeated addition and substraction process prescribed by the original plotting algorithm. For example a binary multiplication is a set of at most as many additions as many digits are in the binary form of the multiplier $(i_x - i_{x0})$. The possible terms of additions are

$$T_k = 2^{k-1} A_x (k = k_{\max}, k_{\max.} - 1, \ldots 2, 1),$$

and such a term really appears in the addition if the k^{th} digit (counted from the right) in the binary form of the multiplier is *1*, and does not appear if the corresponding digit of the multiplier is *0*. A similarily well-known analogous algorithm is available for the binary division, working with the

$$D_l = 2^{l-1} A_y (l = l_{\max}, l_{\max.} - 1, \ldots 2, 1)$$

subtrahends, and setting the corresponding l^{th} digit of the quotient to *1*, if the substraction can take place (with a positive remainder), and to 0, if not.

This method of computing an i_y value corresponding to a given i_x is valid in the given form onely in the case of linear pixel functions. However, it can be generalised for other types of pixel functions preserving its basic structure, onely the construction rules of the T_k additive terms and D_l substrahends have to be modified (the simple "doubling" and "halving" does not work in the general case). Such a pair of a "multiplication" and a "division" operation but with generalized rules for the T_k additive terms and D_l substrahends is called a generalized multiplication-division.

The plotting algorithms are needed if we want to represent a real figure on a graphic device, and for this purpose they are really efficient. But for constructing a curve or surface corresponding to some prescribed conditions or for solving some other problems of computational geometry such computations are needed for which the plotting algorithms in their original form are less efficient, and one has to have recourse to generalized multiplications-divisions. From a plotting algorithm (or, with other words, from a function number) the corresponding generalized multiplication-division rules can be deduced unambiguously.

Some theoretical problems

The aim of this paper is a brief presentation of some ideas, possibly useful in pixel graphics and computational geometry. Thus, many problems had to be left open not onely in this last complementary chapter over some possible generalizations, but in the course of the more detailed investigations, too. Now we want to mention some of these open problems explicitly.

The different types of 2D pixel functions have been defined by means of specific stipulations applied on their derivatives (e.g. a parabola is a pixel function whose derivatives are linear). However, it has not been demonstrated whether these stipulations are really sufficient, free of contradictions, etc. Of course, for the few pixel function types which have been studied in detail, the fact that they could have been constructed on the basis of the given stipulations shows that these stipulations are sensible, but a more general and profound investigation is desirable.

In case of linear pixel functions it has been demonstrated that by means of the corresponding doubly uniform arithmetic description any linear pixel function possible on the given screen can be described and an upper limit has been found for the maximum values of integers needed in such an arithmetic description. For the other types of pixel functions it has been demonstrated onely that the corresponding arithmetic descriptions really produce a pixel function of the given type; the problem whether any arbitrary pixel function

of the given type possible on the given screen size can be constructed this way and if yes how long integers are needed to do this, has been left open, although this problem is very important both from theoretical and from practical point of view.

Practical applications

To conclude this paper some possibilities of practical applications should be mentioned, starting with the immediate (partly already realized) applications and proceeding towards the prospective (and more general) ones.

- Even the simplest PCs have some curve-drawing facilities. The drawing algorithms presented in this paper can be realized in the machine language of many microprocessors in general use with a high efficiency. In the same time they are versatile enough to cover a wide variety of curves appearing in different branches of plane geometry. These properties turn out to be rather advantageous even in many simple hobby and educational applications.
- The 2D representation of 3D objects may be facilitated using the function number type description of 3D curves and curved surfaces. Efficient 3D systems can be realized on this basis for relatively low cost computers.
- The theoretical ideas of this paper can be utilized on the field of free formed curves and surfaces, too, both for efficient realization of the well-known methods and for inducing new approaches to this important problem.
- It seems to be likely that some methods for describing 3D curves may be useful in the robotics, too. There is no difficulty of introducing the time variable as a fourth coordinate and the description methods of the 3D curves could be readily generalised for describing the motions of a point in the 3D space as a "4D curve" in the 4D space-time.
- If some ideas of this paper seem to be useful as a theoretical basis of efficient softwares on the different fields of application just mentioned, they can be even more efficient in form of hardware realizations. The possibilities in this respect are rather diverse. A hardware realization can be based on the plotting algorithms or it can follow more directly the properties of the finite rectangle with finite subdivision and the definition of the pixel function derivatives. Both possibilities point towards modern and promising parallel processor architectures, the second one being, no doubt, the more radical one.
- Some theoretical ideas of this paper may have eventual a more general interpretation, even outside the scope of the computational geometry. If we ask e.g. whether two different pixels do define a straight line or not, the answer is that they define a well circumscribed manifold of straight lines,

not a single one. Even this very elementary example demonstrates that the problems of the traditional analytic geometry in this approach exhibit such properties which are in the mathematical analysis normally hidden, but in the applications (in the natural sciences and technology) may be meaningful. These properties are studied traditionally in the "arithmetics of intervals" but with the methods presented in this paper this study, hopefully, can be facilitated and promoted, possibly conducing to theoretical and practical results.

Dr. József PEREDY H-1521, Budapest