№20 2021

Annali d'Italia

VOL. 1

ISSN 3572-2436

Annali d'Italia (Italy's scientific journal) is a peer-reviewed European journal covering top themes and problems in various fields of science.

The journal offers authors the opportunity to make their research accessible to everyone, opening their work to a wider audience.

500 copies
Annali d'Italia
50134, Via Carlo Pisacane, 10, Florence, Italy
email: info@anditalia.com
site: https://www.anditalia.com/

CONTENT

BIOLOGICAL SCIENCES

PHYSICAL SCIENCES

TECHNICAL SCIENCES

TECHNICAL SCIENCES

AN ERROR OF 3-DIMENSIONS DIGITAL INTEGRATOR OF SEQUENTIAL CARRY INTERPOLATOR IN TASKS OF COMPUTER GRAPHICS

Denysiuk V.
Doctor of Philosophy
Vinnytsya National Agrarian University

Abstract

A study and analysis of features of recreation of segments of straight lines are undertaken after the method of digital differential analyzer in a linear interpolator, the errors of DDA and 3-dimentions linear interpolation are certain with the use of digital integrator of sequential carry.

Keywords: computer graphics, graphic information, line, 3-dimensions linear interpolation, error, digital differential analyzer, digital integrator of sequential carry, pulse rate multipler

An important task of computer technology is the information service of human life, which is expressed, in particular, in the use of information display devices, which are an integral part of various computer systems, systems and networks. Many multidimensional parameters of real objects require an increase in the number of measurements, increasing the speed of information output and the accuracy of information reproduction by display devices. The most informative and visual way of presenting information is graphical, which is associated with physiological, biological and psychological features of human perception of information as an active part of various computer systems. In addition, there is a tendency to increase the share of information in graphical two-dimensional (2D) and three-dimensional (3D) forms in real time in information measurement systems, research, industry, etc.. All this makes the issue of synthesis of graphic information display devices, which provide the maximum achievable performance with a minimum of hardware costs for their implementation. Large information flows for information display devices require compression, which allows you to optimize the operation of it. Information recovery is based on interpolation algorithms, usually linear. The most widespread elements of the three-dimensional (3D) graphic stages are flat grounds, triangles and segments of straight lines [2, 7, 10, 11]. As a task of recreation of grounds and triangles (both filled by a color and contour [3, 7, 10, 11]) can be broken up on the row of tasks on the recreation of segments of straight lines, then it is possible to consider the segment of straight line the basic element of images.

Raising of task. Among the methods of linear interpolation on the recreation of segments of straight lines most distribution was got by the methods based on the use of digital integrators of sequential carry, and methods that is based on the use of digital integrators of parallel transfer or with the calculation of criterion function, Bresenham's line algorithm (BLA) [2, 6, 9, 11].

One bresenham algorithm for a line in 2D space such [9, 11] :

```
{\\\\\\\\\\\ bresenham algorithm ///////////}
procedure bresenham (x1, y1, x2, y2, value: integer);
var dx, dy, iner1, iner2, d, x, y, xend: integer;
begin
dx := abs ( x2 – x1 );
dy := abs ( y2 – y1 );
d := 2*dy – dx ; { start value of d }
incr1 := 2*dy ; { constant for increment step, if d < 0 }
incr2 := 2*( dy – dx ) ; { constant for increment step, if d >= 0 }
if x1 > x2
then { start at minimum x-value point }
begin
x := x2 ;
y := y2 ;
xend := x1
end
else
begin
x := x1 ;
y := y1 ;
xend := x2
end
write_pixel (x, y, value) ; { first point of line }
while x < xend do
begin
x := x + 1 ;
if d < 0
then
d := d + incr1 { choice of si, y - does not change }
else
begin
y := y + 1 ;   { choice of ti, y – increases }
d := d + incr2
end
write_pixel (x, y, value) { the line chosen near-by is a point }
end { end of while }
end
{/////////// bresenham algorithm \\\\\\\\\\\}
```

Linear interpolation devices, systems of differential equalizations of line based on a decision in a self-reactance kind with the use of digital integrators of parallel transfer, two contain register of increases and two story summators. The module of enumeration of summators equals 2^n, where a n-bit of summators is.

As single increases on coordinates come forward signals of summator repletion signal. The linear interpolator based on the decision of the system of differential equalizations of line in a self-reactance kind with application of digital integrators of sequential carry is contained by two register of increases (RG^x, RG^y), two pulse rate multipler (PRM) and general for both integrators counter – CT (fig.1) [1, 2, 10].

Signals of exits of pulse rate multipliers are the signals of single increases on coordinates. Application of integrators of sequential carry brings in an additional error through the unevenness of passing of impulses on the exit of integrator, that can be decreased, for example, using the combined digital integrators. The above-mentioned methods interpolate the arbitrary segment of line for *2^n* times, where a n-bit of device is. To increase the fast-acting of linear interpolation, change the capacity of counter of binary multiplier depending on the sizes of coordinate increases or carry out the common increase of co-ordinate increases to normalization of one of them (carry out the change of both coordinate increases aside most significant digits, that the most significant digit of one of coordinate increases coincided with the most significant digit of device).

Signals of exits of pulse rate multipliers are the signals of single increases on coordinates. Application of integrators of sequential carry brings in an additional error through the unevenness of passing of impulses on the exit of integrator, that can be decreased, for example, using the combined digital integrators. The above-mentioned methods interpolate the arbitrary segment of line for *2^n* times, where a n-bit of device is. To increase the fast-acting of linear interpolation, change the capacity of counter of binary multiplier depending on the sizes of coordinate increases or carry out the common increase of co-ordinate increases to normalization of one of them (carry out the change of both coordinate increases aside most significant digits, that the most significant digit of one of coordinate increases coincided with the most significant digit of device).

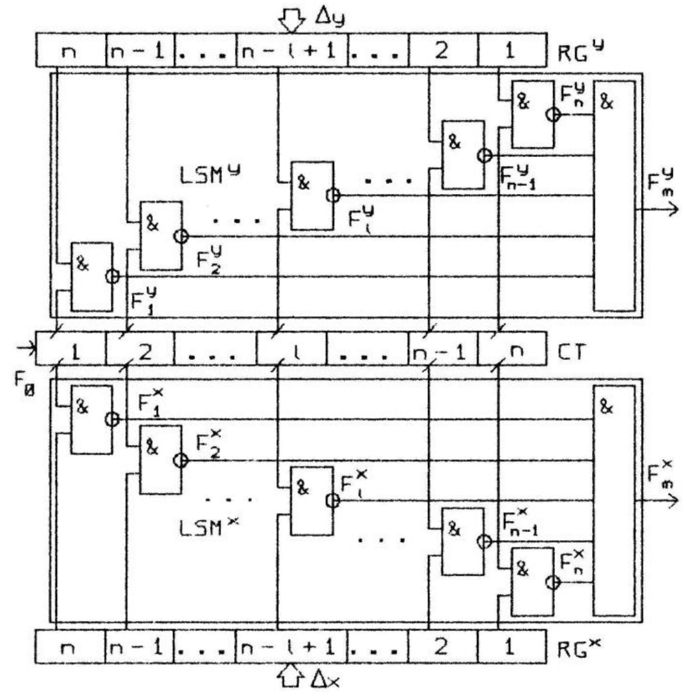

Fig.1. *Two-dimensional linear interpolator is on the basis of digital differential analyzer.*

The feature of BLA-method is an exception of increase operation of functional dependence of X and Y due to the use of step-by-step algorithms that change increase operation in the function of y=f (x) on the step-by-step operations of addition. Interpolation on it method gives a ordinate error that does not exceed the step of discretisation. Well-known BLA-method, that will realize co-ordinate steps as soon as, so diagonal steps [2, 10]. Also there are BLA-methods, that interpolations (not more than 0.5 step of discretisation) give a minimum ordinate error. A method found application with the calculation of two BLA-method functions and side-step less. Methods that will realize two co-ordinate steps are known also.

Research actuality is conditioned by swift development of computer graphics facilities for the recreation of difficult pictures and stages with the large dynamics of image [2,10], in that the various high-performance algorithms of recreation of images elements are widely used with the aim of increase of the productivity and reduction of hardware expenses [2,10].

Exactly defined errors of digital differential analyzer (DDA) and 2D-linear interpolation by digital integrator of sequential carry (DISC) [2, 10].

An aim of the article is an analysis of features of algorithm of recreation of segments of straight lines after the method of DDA for determination of error of linear interpolation at the recreation of information in three-dimensional discrete co-ordinate space.

Analysis of features of digital differential analyzer work.

Digital integrator of sequential carry is known, as pulse rate multipler (PRM) [12] and also, as a digital differential analyzer. The name “pulse rate multipler” arose up in connection with the use of him as a transcoder in frequency.If on the entrance of counter of PRM to give pulse of permanent frequency string, then on the exit of PRM there will be present impulses with a midfrequency, what proportional to the control code.

At the count of impulses amount on going PRM beyond a time that answers beginning and end of period of impulsive sequence domain, there is possible appearance of error on the entrance of counter. An error is related to the unevenness of impulses on a PRM-exit and discrete presentation of count result. Unevenness of passing of impulses on a PRM-exit expressed in that between the following of impulses one is after one on a PRM-exit different intervals at an unchanging control code and even entrance frequency on the entrance of counter. In digital integrators in general and in DISC in particular replacement of integration operation of x-function $(x = f(t))$ on the summarization operation of successive values of $x_k(t_k)$-functions. Successive values are set in the discrete points of t_k, thus $t_{k+1}-t_k=\Delta t$; Δt is permanent, does not depend on t_k and proportional single impulses, id est cost of impulses that act on the entrance of integrator equals "1" [1]. The error of DISC equals a difference between the value of initial size of $\boldsymbol{x}$ at ideal implementation of the set operation of recreation and size of $\boldsymbol{x_0}$, got from signals that seem integrator [1].

Digital integrator gives out a value for the calculation of ordinates of functions in discrete points, that is why an error is examined exactly in these points at the integer values of argument that is measured by the number of impulses [1].

The initial state of DISC-counter (CT) equals "0". In the control code register of DISC (RG) the entered number:

$$\Delta x = \sum_{i=1}^{n} 2^{n-i+1} a_{n-i+1} \;, \quad (1)$$

where $\boldsymbol{a_i}$ - $\boldsymbol{i}$-digit of control code register RG.

The number of impulses, mine-out by the i-chart of coincidence on the receipt of $\boldsymbol{t}$ impulses on the entrance of DISC, equals:

$$t_i = ent\left(\frac{t+2^{i-1}}{2^i}\right), \quad (2)$$

The number of impulses on the DISC-exit equals:

$$x_0 = \sum_{i=1}^{n} t_i a_{n-i+1} \;. \quad (3)$$

After the receipt of $\boldsymbol{t=2^n}$ impulses of DISC pass to the initial state. Equalization of line that passes through points $\boldsymbol{(0, 0)}$ and $\boldsymbol{(2^n, \Delta x)}$:

$$x = \sum_{i=1}^{n} 2^{n-i+1} a_{n-i+1} \cdot t / 2^n \;. \quad (4)$$

Thus, the error of integration of DISC equals $\delta_p = x - x_0$ and looks like :

$$\delta_p = \frac{\sum_{i=1}^{n} 2^{n-i+1} a_{n-i+1}}{2^n} \cdot t - \sum_{i=1}^{n} t_i a_{n-i+1} \;. \quad (5)$$

Consider the three-dimensional (3D) interpolation of the linear function z = f (x, y) on a device that contains three DISC with one common counter (fig.2, 3) [2, 10].

Fig. 4 shows the operation of a 3D linear interpolator on the digital integrator of sequential carry for $\Delta x = 14$, $\Delta y = 7$, $\Delta z = 11$ $(n = 4)$, which explains the process of constructing the interpolation curve $z_o=f(x_o,y_o)$ in 3D discrete coordinate display space.

For the horizontal we take x, for the vertical - y, for the third coordinate, which is directed from the observer into the screen of the information display device (IDD), - z. The coordinate axes Ox, Oy, Oz form the left triple [4, 5, 8, 10].

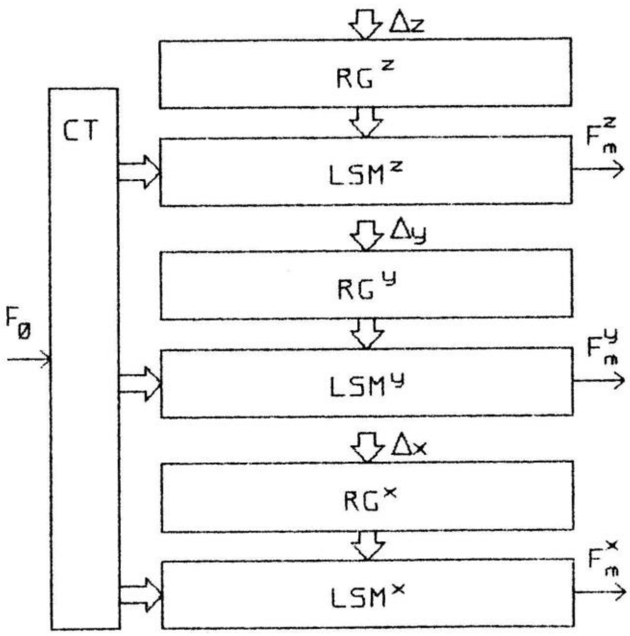

Fig.2. Block diagram of 3D linear interpolator on digital integrator of sequential carry

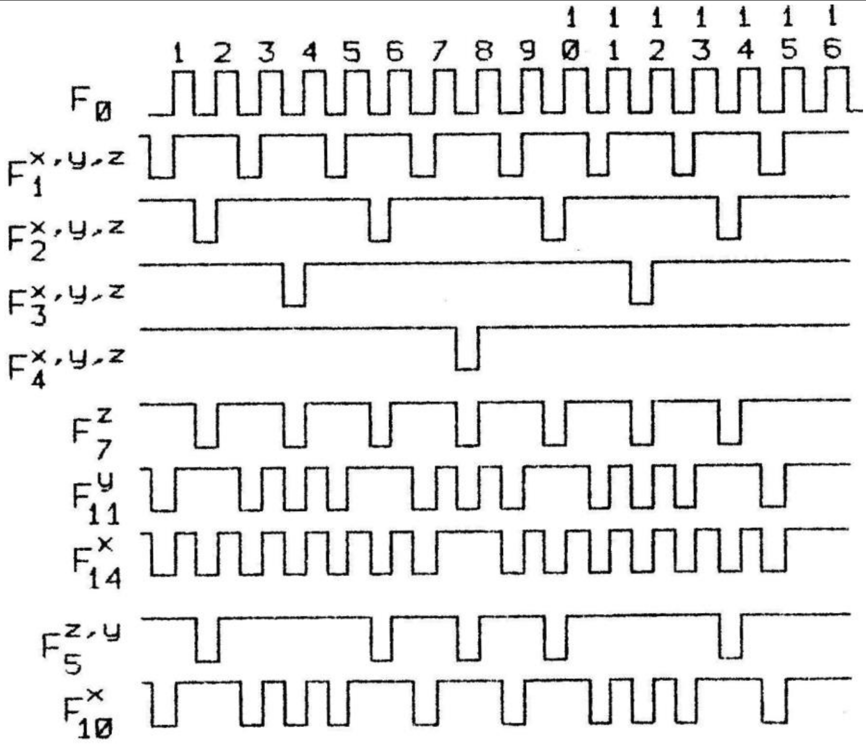

Fig.3. Diagrams signals examply of 3D linear interpolator on digital integrator of sequential carry (n = 4)

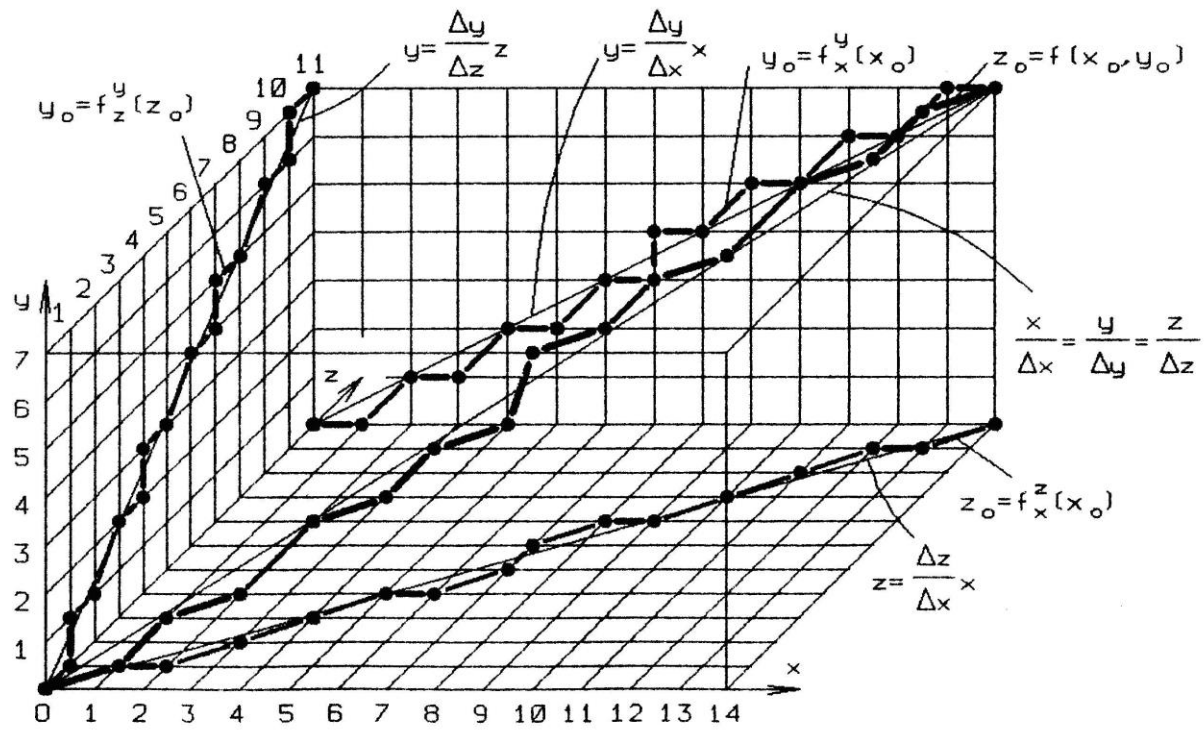

Fig. 4. Operation of a 3D linear interpolator on the digital integrator of sequential carry for $\Delta x = 14$, $\Delta y = 7$, $\Delta z = 11$ (n = 4)

The codes of the corresponding coordinate increments are written in the control registers of the interpolator: Δx - coordinate increment on the x-axis - in RG^x, Δy - on the y-axis - in RG^y , Δz - on the z-axis - in RG^z:

$$\begin{cases} \Delta x = \sum_{i=1}^{n} 2^{n-i+1} a_{n-i+1} \\ \Delta y = \sum_{i=1}^{n} 2^{n-i+1} b_{n-i+1} \\ \Delta z = \sum_{i=1}^{n} 2^{n-i+1} c_{n-i+1} \end{cases} \quad (6)$$

where a_i, b_i or c_i - *i-th* digit of RG^x, RG^y or RG^z , respectively.

The output frequency sequences of the coordinate binary multiplier F_m^x, F_m^y and F_m^z are respectively functions x_0, y_0 and z_0 from the pulses t of the frequency F_o at the input of the common counter (7):

$$x_0 = \sum_{i=1}^{n} t_i a_{n-i+1}; y_0 = \sum_{i=1}^{n} t_i b_{n-i+1}; z_0 = \sum_{i=1}^{n} t_i c_{n-i+1}. \quad (7)$$

As a result, we obtain the function $z_0=f(x_o,y_o)$ in 3D discrete coordinate space (DCS) *(xyz)*. In the following considerations, we assume that Δx> = Δy, Δx> = Δz and Δx contains in the higher digit "1", ie is a normalized value, as the following function is required:

$$\frac{x}{dx} = \frac{y}{dy} = \frac{z}{dz} \quad (8)$$

For the error of 3D linear interpolation at the point $Q_o(x_o,y_o,z_o)=Q(r_0)$ it is accepted to choose the distance from the point Q_o,, which belongs to the 3D discrete coordinate space, on the interpolation curve $z_o=f(x_o,y_o)$, to the line is interpolated and given by expression (8) [2, 4]. That is, for the 3D linear interpolation error δ_o, the value of the normal from the point Q_0 to the interpolated line is chosen. Define the expression for δ_n.

Assume that the point Q_1 (x_1,y_1,z_1) is the starting point of the segment of the interpolated line, given by the coordinate increments Δ*x*, Δ*y*, Δ*z*, and $Q_2(x_2,y_2,z_2)$ is the end point. The equation of the line passing through the two points Q_1 and Q_2 (9):

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1} \quad (9)$$

The distance δ_n from the point $Q_o(x_o,y_o,z_o)$ to the line given by expression (9) is as follows [2, 4]:

$$\delta_n = \frac{\sqrt{\begin{vmatrix} y_2 - y_1 & z_2 - z_1 \\ y_1 - y_0 & z_1 - z_0 \end{vmatrix}^2 + \begin{vmatrix} z_2 - z_1 & x_2 - x_1 \\ z_1 - z_0 & x_1 - x_0 \end{vmatrix}^2 + \begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_1 - x_0 & y_1 - y_0 \end{vmatrix}^2}}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}}. \quad (10)$$

To simplify the calculations, assume that the point Q_1 (x_1,y_1,z_1) coincides with the origin *O* $(x_1=0$ $y_1=0,$ $z_1=0)$. Then, taking into account (8) and (9), expression (10) takes the form:

$$\delta_n = \frac{\sqrt{\begin{vmatrix} \Delta y & \Delta z \\ -y_0 & -z_0 \end{vmatrix}^2 + \begin{vmatrix} \Delta z & \Delta x \\ -z_0 & -x_0 \end{vmatrix}^2 + \begin{vmatrix} \Delta x & \Delta y \\ -x_0 & -y_0 \end{vmatrix}^2}}{\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}} \quad (11)$$

It should be noted that the value of δ_n can be determined from other dependences [1, 5], but they contain ars-trigonometric functions, which makes it difficult to use such dependences in machine analysis. In turn, the use of (11) allows us to determine only the absolute value of δ_n, which fully satisfies the problem of estimating δ_n.

The algorithm for determining δ_n for $i = 1 \div 5$.

```
program lddn;
{ ------------------------ turbo pascal /15.06.2021/ ------------------------ }
{determination of my-mi maximum normal errors}
{ 3d linear interpolator on binary multipliers, for }
{ non-zero values of control code bits, }
{ and the corresponding values of increases on the y and z axes }
{ at normalized x (in senior discharge "1")}
{ sound signal issuance at the end of the table and page }
{ definition}
{absolute time for calculating the page and table }
uses printer,dos,crt;
label aa,bb;

const
rng = 8; { interpolator bit }
bnn = 256; { 2 ** rng }
spn = 2; {number of spaces between lines}
mx = 12; {maximum number of records in a row}
my = 1; {number of maximum errors for dx, dy data}
nl = 30; {number of records per page}
type
vecte =array [1..mx,1..my] of real;
vectz =array [1..mx,1..my] of longint;
```

```
vectorm=array [1..rng,1..bnn] of integer;
vectorb=array [1..rng] of integer; {delete binary code}
vectors = array [1..bnn] of integer; {coordinate steps}
var
lps:integer; { *************************************** }
lpg: integer; {number of lines in result page}
pfn: integer; {number of the first page of the result}
pgn: integer; {page number}
vstx: longint; {initial value of record argument (delta x)}
vspx: longint; {final value of writing argument (delta x)}
vsty: longint; {initial value of record argument (y delta)}
vspy: longint; {final size of record argument (y delta)}
xbin: vectorb; {dx binary code}
ybin: vectorb; {dy body code}
zbin: vectorb; {dz binary code}
ntd: integer; {number of records per line}
h, m, s, f: word; {hours, minutes, sec., hundreds sec. current time}
ph, pm, ps, pf: word; {hours, min., sec., hundred sec. beginning of page }
th, tm, ts, tf: word; {hours, min., sec., hundred sec. home off table }
dh, dm, ds, df: word; {delta hours, minutes, seconds and hundreds of seconds}
vdtx: longint; {number of recording argument values adjustable)}
vdty: longint; {number of recording argument values (adjustable)}
pgd: integer; {number of result pages}
lin: text; {result file}
lct: integer; {line counter in page}
emax: vecte; {ntd for 8 maximum errors}
zmax: vectz; {dz maximum errors}
dt: longint; {number of recording argument values (initial)}
nct: integer; {column counter in records}
dy: longint; {y increase corresponding to max. pogr. }
dz: longint; {increase in z, corresponding to max. pogr. }
dnm: real; {maximum normal error of delta data}
srng: vectorm; {matrix of discharge weight steps}
sx: vectors; {x-step step sign, 0-no step, 1-step}
sy: vectors; {sign of step on the y axis, 0-no step, 1-step}
sz: vectors; {z-axe step sign, 0-no step, 1-step}
x: integer; {record column number}

{\\\\\\\\\\\ degree function with basis 2 ///////////}
function bindegre(nf:integer):longint;
var
nnf:longint;
index:integer;
begin
if nf>0 then
begin
index:=1;
nnf:=1;
while index<=nf do
begin
nnf:=nnf*2;
index:=index+1;
end;
bindegre:=nnf;
end
else
bindegre:=1;
end;
{/////////// degree function with basis 2 \\\\\\\\\\}

{\\\\\\\\\\\ output of the general table cap ///////////}
procedure prntbtit(var bfl:text); {result file}
begin
writeln(bfl,
'result of ldd7 ',
'(turbo pascal version /12.01.2021/)');
writeln(bfl, ' 3d bin rate multiplier ',
' line interpolator ',
my,
' max errors array');
writeln(bfl,' range of interpolator : ',
rng: round(ln(rng)/ln(10)+0.5));
writeln(bfl);
end;
{/////////// output of the general table cap \\\\\\\\\\}

{\\\\ translation into the binary code of the decimal number ////}
procedure decbintr (date: longint; {decimal number}
var bufarray: vectorb); {b-cod border sequence}
var
index: integer; {binary code bit index}
bufdate: longint; {translation buffer}
begin
index:=1;
bufdate:=date;
while index<=rng do
begin
bufarray[index]:=round
((bufdate/2-trunc(bufdate/2))*2);
bufdate:=trunc(bufdate/2);
index:=index+1;
end;
end;
{///////// translation into the binary code of the decimal number \\\\\\\\\}

{\\\\\\\\\\ zeroing of all bits of the binary code //////////}
procedure binreset (var bufarray: vectorb); {binary code sequence}
var
index:integer;
begin
index:=1;
while index<=rng do
begin
```

```
bufarray[index]:=0;
index:=index+1;
end;
end;
{////////// zero all bits of binary code \\\\\\\\\\}

{\\\\\\\\\\ "obtaining a coordinate interpolation sequence //////////}
procedure stepdelt(srng:vectorm;
bufb:vectorb;
var bufs:vectors);
label label1;
var
indexes: integer; {draft weight matrix row index}
index: integer; {binary code column index}
indexl: integer; {step column index or}
{discharge weight matrix column index}
begin
begin
indexl:=1;
while indexl<=bnn do
begin
bufs[indexl]:=0;
inc(indexl);
end;
end;
indexi:=rng;
indexk:=1;
label1: if indexk <=rng then
begin
if bufb[indexk]=1 then
begin
for indexl:=1 to bnn do
begin
bufs[indexl]:=bufs[indexl]+srng[indexi,indexl];
end;
end;
dec(indexi);
inc(indexk);
goto label1
end;
end;
{////////// obtaining a coordinate interpolation sequence \\\\\\\\\}

{\\\\\\\\\\ production of discharge weight sequences //////////}
procedure horsetet (var bufm: vectorm);
var
indexes: integer; {matrix line index}
index: integer; {matrix column index}
bufkk: integer; {intermediate value, kk = (2 ** n / 2 ** i - 1)}
index: integer; {intermediate index}
begin
indexi:=1;
indexj:=1;
for indexi:=1 to rng do
begin
for indexj:=1 to bnn do
begin
bufm[indexi,indexj]:=0;
end;
end;
indexi:=1;
for indexi:=1 to rng do
begin
bufkk:=round(bnn/bindegre(indexi)-1);
for indexk:=0 to bufkk do
begin
bufm[indexi,(bindegre(indexi-1) + bindegre(indexi)*indexk)]:=1;
end;
end;
end;
{////////// getting discharge weight sequences \\\\\\\\\\}

{\\\\\\\\\ increasing binary code by one //////////}
procedure bininc (var bufarray: vectorb) {b-code binder sequence}
label bbii; {procedure label}
var
index: integer; {digital bit code index dy}
begin
index:=1;
while index <= rng do {cycle}
begin {zoom}
if bufarray [index] = 0 then {*****}
begin {binary}
bufarray [index]: = 1; {foot}
goto bbii; {code}
end {bufarray}
else {numbers}
begin {dy}
bufarray [index]: = 0; {on}
index: = index + 1; {unit}
end; {*****}
end; {"+1"}
bbii: end;
{////////// increasing binary code by unit \\\\\\\\\\}

{\\\\\\\\\\\ output page number ///////////}
procedure prnpgnum (ppgn: integer; {page number}
var bfl: text); {result file}
const plps=110;
var
ndex: integer; {auxiliary}
begin
ppgn: = pgn;
index: = round (plps / 2-4-round (ln (plps) / ln (10)) + 0.5);
while index> = 1 do
begin
write (bfl, ''); {spaces}
index: = index-1; {*****}
end;
writeln(bfl, ' - ',
ppgn:round(ln(abs(ppgn))/ln(10)+0.5),
' -');
end;
{/////////// page number output \\\\\\\\\\}

{\\\\\\\\\\ output of the header of records and spaces between them //////////}
```

```
procedure prnnttit (pntd: integer; {number of records in a row}
pvst: longint; {dx first recording in a row}
var bfl: text); {result file}
const nottitle = 'dz err'; {record hat}
var
notcount: integer; {line record counter}
spacecnt: integer; {spacer counter between records}
bufferdx: longint; {dx values for caps}
begin
notcount:=pntd;
bufferdx:=pvst;
while notcount>=1 do
begin
write (bfl, 'dy =', bufferdx: 3); {dx output in record cap}
bufferdx: = bufferdx + 1;
begin
spacecnt: = spn;
while spacecnt> = 1 do
begin
write (bfl, ''); {output spaces between records} spacecnt:=spacecnt-1;
end;
end;
notcount:=notcount-1;
end;
writeln(bfl);}
notcount:=pntd;
while notcount>=1 do
begin
write (bfl, nottitle); {output of the record cap}
begin
spacecnt: = spn;
while spacecnt> = 1 do
begin
write (bfl, ''); {output spaces between records}
spacecnt:= spacecnt-1;
end;
end;
notcount:=notcount-1;
end;
writeln(bfl);
end;
{////////// output of the cap of records and spaces between them \\\\\\\\\

{\\\\\\\\\\ resetting eight maximum errors and their dy //////////}
procedure resvect (var pemax: vecte; {ntd for 8 maximum errors}
var pymax:vectz); { dz }
var
indexx:integer;
indexy:integer;
begin
indexx:=1;
while indexx<=mx do
begin
indexy:=1;
while indexy<=my do
begin
pemax[indexx,indexy]:=0;
pymax[indexx,indexy]:=0;
indexy:=indexy+1;
end;
indexx:=indexx+1;
end;
end;
{////////// zero eight maximum errors and their dy \\\\\\\\\}

{\\\\\\\\\\ sorting 8 max. errors and overwrite them dy //////////}
procedure sortvect (px: integer; {column number}
pdy: longint; {yield increase}
perm: real; {maximum error for dy}
var pemax: vecte; {ntd for 8 maximum errors}
var pymax: vectz); {corresponding to dz}
label sr1;
var
index1: integer; {index of eight maximum errors}
index2: integer; {intermediate index 8-max. errors}
begin
index1:=1;
while index1<=my do
begin
if abs(perm)>=abs(pemax[px,index1]) then
begin
index2:=my;
while index2>=index1 do
begin
pemax[px,index2]:=pemax[px,index2-1];
pymax[px,index2]:=pymax[px,index2-1];
index2:=index2-1;
end;
pemax[px,index1]:=perm;
pymax[px,index1]:=pdy;
goto sr1;
end;
index1:=index1+1;
end;
sr1: end;
{////////// sorting 8 max. errors and overwrite them dy \\\\\\\\\}

{\\\\\\\\\\ print pages with eight maximum errors //////////}
procedure prnsortv (pemax: vecte; {8 maximum errors}
pymax: vectz; {corresponding to dz}
pntd: integer; {number of records per line}
var bfl: text); {result file}
var
ix:integer;
iy:integer;
begin
iy:=1;
while iy<=my do
begin
ix:=1;
while ix<=pntd do
begin
write(bfl,pymax[ix,iy]:3,' ');
```

```
write(bfl,pemax[ix,iy]:1:2,' ');
ix:=ix+1;
end;
writeln(bfl);
iy:=iy+1;
end;
end; {////////// print page with eight maximum errors \\\\\\\\\\}
function factplus(arg:longint):longint;
var
buf:longint;
index1:longint;
index2:longint;
begin
if arg=0 then factplus:=1 else
begin
buf:=0;
index1:=1;
while index1<=arg do
begin
index2:=1;
while index2<=index1 do
begin
inc(index2);
end;
buf:=buf+index2;
inc(index1);
end;
factplus:=buf;
end;
end;

{\\\\\\\\\\\ display of time ///////////}
timwrit procedure (var hour, minute, second, sec100: word);
function lzero (w: word): string; {machine translation function}
{representations of size b}
var {symbol with dynamic long}
s:string; { переменная функции }
begin {------- home body functions -------}
str (w: 0, s); {process of translation into symbol key}
if length (s) = 1 then {if symbol length 1,}
s: = '0' + s; {then increase it by s, i.e. on 1}
lzero: = s; {assign value of function}
end; {------- end of function body -------}
begin {**** home body timwrit procedure ****}
write (lzero (hour), ':', {hours}
lzero (minute), ':', {minutes}
lzero (second), '.', {seconds}
lzero (sec100)); {hundred seconds}
end; {**** end of timwrit body procedure ****}
{/////////// display of time \\\\\\\\\\}

{\\\\\\\\\\\ output time to file ///////////}
procedure prntime(var hour,
minute,
second,
sec100:word;
var bfl:text);
function lzero (w: word): string; {machine translation function}
{representations of size b}
var {symbol with dynamic long}
s: string; { function variable}
begin {------- home body functions -------}
str (w: 0, s); {proc. translation into symbol key}
if length (s) = 1 then {if length of symbol representation 1,}
s: = '0' + s; {then increase it by s, i.e. on 1}
lzero: = s; {assign value of function}
end; {------- end of function body -------}

begin {**** home body procedures prntime ****}
writeln (bfl, lzero (hour), ':', {hours}
lzero (minute), ':', {minutes}
lzero (second), '.', {seconds}
lzero (sec100)); {hundred seconds}
end; {**** end of body procedure prntime ****}
{/////////// output time to file \\\\\\\\\\\}

{\\\\\\\\\\ determination of absolute performance //////////}
procedure dtimer (var h1, m1, s1, f1, {hours, min, sec., sec. / 100 start}
h2, m2, s2, f2, {hours, min, sec, sec / 100 end}
dhp, dmp, dsp, dfp: word); {absolute performance}
var
dhb, dmb, dsb, dfb: integer; {buffer integer variables}
{to define the delta sign}
begin {********* home body of dtimer procedure *********}
dfb: = f2-f1; {hundredth of a second delta sign detection}
if dfb <0 then {if f1> f2, then}
begin {their difference is determined)
df: = 100-f1 + f2; {transfer of 100 second discharges,}
dsb: = s2-s1-1; {unit loans from second discharge}
end {when determining the second delta sign}
else {if f1 <= f2, then}
begin {their difference is determined}
df: = f2-f1; {no loan,}
dsb: = s2-s1; { second delta sign}
end;
if dsb <0 then {if s1> s2, then}
begin {their difference is determined}
ds: = 60-s1 + s2; {transfer of 60 from discharge minutes,}
dmb: = m2-m1-1; {unit loan from minute}
end {when determining the mint sign}
else {if s1 <= s2, then}
begin {if there was a borrow}
if dfb <0 then
ds: = s2-s1-1 {delta seconds including loan}
else {if there was no loan, then}
ds: = s2-s1; {delta second is not corrected}
dmb: = m2-m1; { delta minute sign}
end;
if dmb <0 then {if m1> m2, then}
```

```
begin {their difference is determined}
dm: = 60-m1 + m2; {transfer of 60 from hours}
dhb: = h2-h1-1; {unit loans from hours}
end {when determining the delta sign.}
else {if m1 <= m2, then}
begin {if there was a borrow}
if dsb <0 then {minute must be obtained}
dm: = m2-m1-1 {delta minutes including loan}
else {if there was no loan, then}
dm: = m2-m1; {delta minute is not corrected}
dhb: = h2-h1; { delta clock sign}
end;
if dhb <0 then {if h1> h2, their difference is determined}
dh: = 12-h1 + h2 {including the transfer of 12 from discharges of the day}
else {if h1 <= h2, then}
if dmb <0 then {if there was a loan from the days, it is necessary}
dh: = h2-h1-1 {get the delta of the hours according to the loan}
else {if there was no loan, then the delta clock}
dh: = h2-h1; {no adjustment}
end; { ********* end of the body of the dtimer procedure *********}
{////////// determination of absolute performance \\\\\\\\\}

{\\\\\\\\\\\ determination of the number of decimal positions of the number //////////}
function decposit (anydate: longint): integer;
var
buffer:longint;
begin
if (0<=anydate)and(anydate<=9) then
buffer:=1
else
buffer:=round(ln(abs(anydate))/ln(10)+0.5);
decposit:=buffer;
end;
{////////// determination of the number of decimate positions of the number \\\\\\\\\

{\\\\\\\\\ max. ordinate error linear interpolator /////////}
procedure normerr (dxb, dyb, dzb: longint;
sxb, syb, szb: vectors;
var dnmbuf: real);
var
tx: longint; {current increase coordinates on ox axis}
ty: longint; {current increase coordinates on oy axis}
tz: longint; {current increase coordinates on oz axis}
aq, bq, cq: real;
dnb, dnmax: real; {normal interpolator error}
index:integer;

begin
dnm:=0;
aq:=sqr(dxb)+sqr(dyb)+sqr(dzb);
tx:=0;
ty:=0;
tz:=0;
index:=1;
while index<=bnn do
begin
tx:=tx+sxb[index];
ty:=ty+syb[index];
tz:=tz+szb[index];
bq:=sqr(tx)+sqr(ty)+sqr(tz);
cq:=sqr(dxb-tx)+sqr(dyb-ty)+sqr(dzb-tz);
dnb:=sqrt(abs(bq-(sqr(aq+bq-cq))/(4*aq)));
if dnb>=dnmax then dnmax:=dnb;
inc(index);
end;
dnmbuf:=dnmax;
end;
{///////// max. ordinate error linear interpolator \\\\\\\\\}

begin {\\\\\\\\\ beginning of the main program /////////}
writeln;
writeln ('-------- turbo pascal /15.06.2021/ --------');
writeln ('3d binary rate multipliers line interpolator',
'normal maximum errors array');
writeln;
writeln ('introduce number of first page'); {enter from console number}
write ('->'); {first page, if 1 '',}
readln (pfn); {then the tablet hat is printed,}
writeln; {otherwise-ordinary page. }
writeln ('introduce start increment in x axis'); {console input}
writeln ('(', (bindegre (rng-1)): decposit (bindegre (rng-1)),
'<= start delta x <=', {*****}
(bnn-1): rng, ')'); {left border}
write ('->'); {*****}
readln(vstx); { delta x }
writeln;
writeln ('introduce stop increment in x axis'); {console input}
writeln (',', vstx: decposit (vstx), '<= stop delta x <=', {*****}
(bnn-1): rng, ')'); {right}
write ('->'); {boundaries}
readln(vspx); { delta x }
writeln;
decbintr(vstx,xbin);
vdtx:=vspx-vstx+1;
pgd:=round(factplus(vdtx)/(mx*nl)+0.49);
writeln(' !!!!! - attention - !!!!! ');
writeln(' there are ',pgd:decposit(round(pgd)),
' page of result table ');
writeln;
write(' number of page is ',
(pgd+pfn-1):(decposit(round(pgd+pfn))));
gettime (th, tm, ts, tf); {reversed. to the withdrawal procedure. times}
write (', it is now'); {print text}
```

```
timwrit (th, tm, ts, tf); {display of table start time on screen}
ph: = th; {remembering the beginning hour of page calculations}
pm: = tm; {- "- minute -" -}
ps: = ts; {- "- second -" -}
pf: = tf; {- "- tens of seconds -" -}
pgn:=pfn;
writeln(' , start of ' ,pgn:decposit(round(pgn)), ' page ');
assign(lin,' lin.tab ');
rewrite(lin);
if pgn=1 then
prntbtit(lin)
else
prnpgnum(pgn,lin);
horsetet(srng);
decbintr (vstx, xbin); {translate dx to xbin binary code [i..rng]}
lct: = nl;
while vstx <= vspx do
begin
stepdelt (srng, xbin, sx);
sound (1000); {sound signal}
delay (200); {*****}
nosound; {delta x}
sound (1000); {sound signal}
delay (200); {*****}
nosound; {delta x}
writeln(lin,' dx = ',vstx:3);
vsty:=0;
binreset(ybin);
vspy:=vstx;
aa: if vsty<=vspy then
begin
vdty:=vspy-vsty+1;
bb: if (lct>=1)and(vsty<=vspy) then
begin
if vdty>mx then ntd:=mx
else ntd:=vdty;
nct:=ntd;
x:=1;
prnnttit(ntd,vsty,lin);
resvect(emax,zmax);
while nct>=1 do
begin
dz:=0;
binreset(zbin);
stepdelt(srng,ybin,sy);
while dz<=vsty do
begin
stepdelt(srng,zbin,sz);
normerr(vstx,vsty,dz,sx,sy,sz,dnm);
sortvect(x,dz,dnm,emax,zmax);
inc(dz);
bininc(zbin);
end;
vdty:=vdty-1;
vsty:=vsty+1;
x:=x+1;
bininc(ybin);
nct:=nct-1;
end;
prnsortv(emax,zmax,ntd,lin);
writeln(lin);
lct:=lct-1;
goto bb;
end;
if vsty<vspy then
begin
write (lin, 'end of', pgn: decposit (round (pgn)), 'page,'); {end label output}
write (lin, 'page computating time:'); {pgn pages}
gettime (h, m, s, f);
dtimer (ph, pm, ps, pf, h, m, s, f, dh, dm, ds, df);
ph: = h; {remembering the beginning hour of page calculations}
pm: = m; {- "- minute -" -}
ps: = s; {- "- second -" -}
pf: = f; {- "- tens of seconds -" -}
prntime (dh, dm, ds, df, lin); {output of page time in lin.tab}
write ('number of page is',
(pgd + pfn-1): decposit (round (pgd + pfn)), ',');
write ('it is now');
timwrit (h, m, s, f); {displaying the page end time}
write(' , ');
write(' end of ' ,pgn:decposit(round(pgn)), ' page, ');
sound (750); {sound signal}
delay (500); {at the end}
nosound; {pages}
pgn: = pgn + 1;
writeln ('start of', pgn: decposit (round (pgn)), 'page');
writeln (lin);
writeln (lin, char (12)); {output disconnector output}
prnpgnum(pgn,lin);
lct:=nl;
end;
goto aa;
end;
inc(vstx);
bininc(xbin);
end;
write (lin, 'end of', pgn: decposit (round (pgn)), 'page,'); {pgn page partner label output}
write (lin, 'page computating time:'); {*****}
gettime (h, m, s, f);
dtimer (ph, pm, ps, pf, h, m, s, f, dh, dm, ds, df);
prntime (dh, dm, ds, df, lin); {output of page time in lin.tab}
write ('number of page is',
(pgd + pfn -1): decposit (round (pgd + pfn)), ',');
write ('it is now');
timwrit (h, m, s, f); {displaying the page end time}
write(' , ');
write(' end of ',pgn:decposit(round(pgn)), ' page, ');
writeln('end of table');
sound (750); {sound signal}
delay (500); {at the end}
nosound; {pages}
write (lin, 'end of table,',
'table computating time:'); {table end label output}
```

dtimer (th, tm, ts, tf, h, m, s, f, dh, dm, ds, df);

prntime (dh, dm, ds, df, lin); {output of table time in lin.tab}

writeln (lin);

close (lin); {closing the lin.tab results file on the active disk}

sound (500); {sound signal}

delay (1000); {at the end}

nosound; {tables}

end.

{////////// end of main program \\\\\\\\\\}

The values of δ_i for $i = 1 \div 5$ were determined. It is established that the maximum error δ_n occurs twice in the process of interpolation of the line, and the directions of the vectors of the normal corresponding to δ_n are opposite (one towards the other). The following relations are valid for the parameters Δy, Δz, y_o and z_o:

$$\begin{cases} \Delta y = \Delta z, \\ y_0 = z_0 \end{cases} \quad (12)$$

In addition, it is established that:

$$\left[\begin{array}{l} \Delta x = 2\Delta y + 1, \text{ для n-непарних;} \\ \Delta x = 2\Delta y, \text{ для n-парних.} \end{array} \right. \quad (13)$$

Fig.5 shows the operation of the *3D* linear interpolator on the digital integrator of sequential carry for $\Delta x = 10$, $\Delta y = 5$, $\Delta z = 5$, which corresponds to the maximum interpolation error at n = 4.

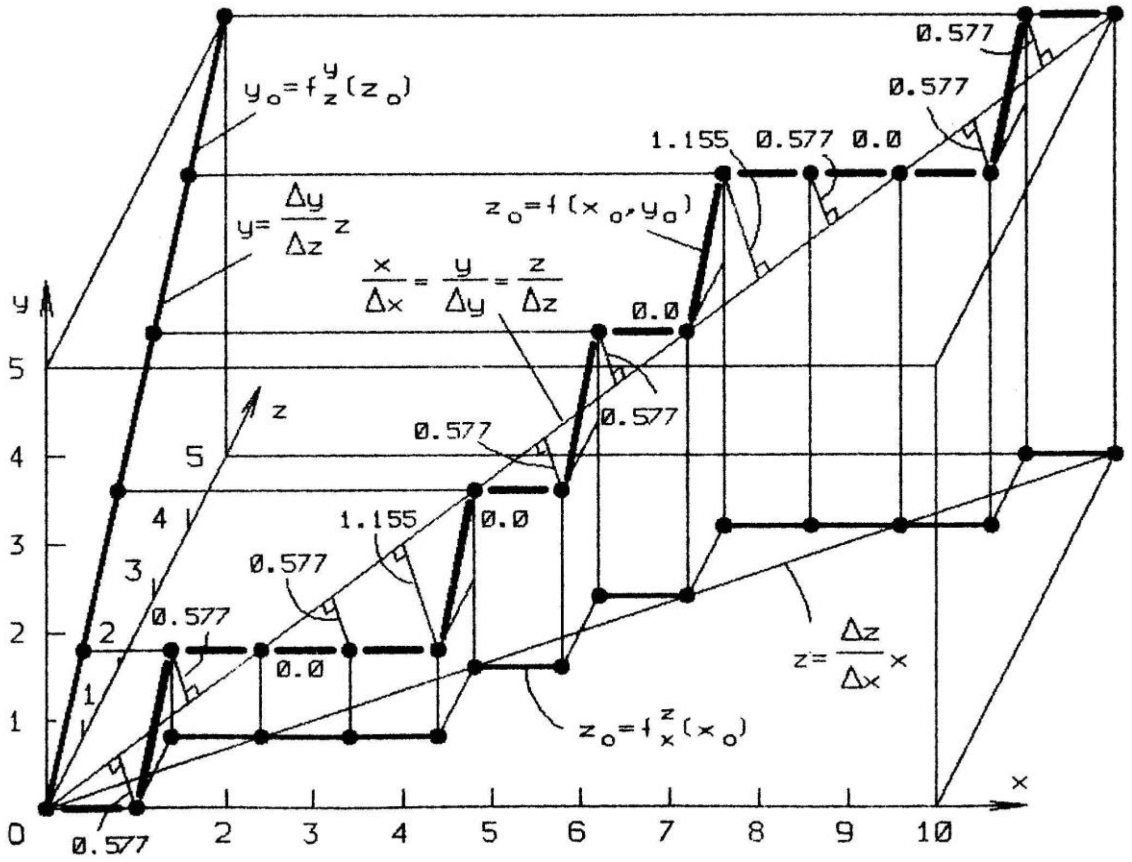

Fig.5. Oeration of the 3D linear interpolator on the digital integrator of sequential carry for $\Delta x = 10$, $\Delta y = 5$, $\Delta z = 5$, $n = 4$

If we conditionally denote the number of pulses *t* at the input of the counter corresponding to the first appearance of the maximum error at *i*-pairs, as C^1i and the error as δ^1_n, and the number of pulses corresponding to the second appearance of the maximum error as C^2_i and the error as δ^2_n, then, taking into account projection of interpolation lines on the plane *(yOz)*, it is proposed to assume that the number of input pulses C^1_i corresponds to a negative error δ^1_n, and C^2_n - a positive error δ^2_n. In the course of the performed analysis and on the basis of the principle of mathematical induction the iterative method of definition of parameters at which there is a maximum error of 3D linear interpolation with use of DISC is offered:

1) if *i*-odd, then $\Delta x_i = 2\cdot\Delta x_{i-1}+1$, $\Delta y_i = \Delta z_i = \Delta x_{i-1}$, a $C_i^1 = 2\cdot C^2_{i-1}+1$,

$C^2_i = C^2_{i-1}$, and $\Delta x_i = C^1_i$, a $\Delta y_i = \Delta z_i = \Delta x_{i-1} = C^2_i$;

2) if i-pair, then $\Delta x_i = 2\cdot\Delta x_{i-1}$, $\Delta y_i = \Delta z_i = \Delta x_{i-1}$, a $C^1_i = C^1_{i-1}$, $C^2_i = 2\cdot C^1_{i-1}$,

and, $\Delta x_i = C^2_i$, a $\Delta y_i = \Delta z_i = \Delta x_{i-1} = C^1_i$.

If for x_0 and y_0, corresponding to the number of pulses at the input C^1_i, take x^1_0 and y^1_0, and for x_o and y_o, corresponding to C^2_1, take x^2_0 and y^2_0, then the analysis follows from the analysis (14):

$$\begin{cases} \left[\begin{aligned} & x_0^1 - 2y_0^1 = 2n-1, \\ & 2y_0^2 - x_0^2 = 2n+1; \end{aligned}\right. \text{ when } i\text{- odd} \\ \left[\begin{aligned} & x_0^1 - 2y_0^1 = n/2, \\ & 2y_0^2 - x_0^2 = n/2; \end{aligned}\right. \text{ at } i\text{-pair} \end{cases} \quad (14)$$

Given (12), the following is true:

$$\begin{vmatrix} \Delta y & \Delta z \\ -y_0 & -z_0 \end{vmatrix} \equiv 0 .$$

Thus, expression (11) will take the form:

$$\delta_n = \frac{\sqrt{2} \cdot (\Delta x \cdot y_0 - \Delta y \cdot x_0)}{\sqrt{\Delta x^2 + 2 \cdot \Delta y^2}} . \quad (15)$$

Taking into account (13) and (14) for pairs *i*, the expression for error (15) takes the form:

$$\delta_n^1 = -\frac{n}{2\sqrt{3}}; \delta_n^2 = \frac{n}{2\sqrt{3}} . \quad (16)$$

Taking into account (13), (14) and on the basis of (15) for odd *i* the following expressions (17) can be written:

$$\delta_n^1 = \frac{1}{\sqrt{3}}\left(\frac{2}{3} - \frac{n+1}{2}\right); \delta_n^2 = \frac{1}{\sqrt{3}}\left(\frac{1}{3} + \frac{n-1}{2}\right) . \quad (17)$$

As a result of the conducted researches and by means of the offered expressions (16) and (17) data on the absolute maximum error of the 3D linear interpolator on DISC (tab. 1 and tab. 2) are received.

Let's analyze the error of the *3D* linear interpolator on the DISC. The value of the relative error of *3D* reproduction of a straight line is $(\delta_0 / \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}) \cdot 100\%$ and is a decreasing value: at *n = 5* it is equal to 5.3%, and at *n = 10* it is already equal to 0.35%.

In other words, the higher the discrete coordinate display space resolution, the less noticeable the absolute interpolation error. For example, for *n = 12* the maximum absolute error of the *3D* interpolator is equal to 3.4641 discrete, the relative error is 0.10% (at *n = 12*, the resolution of the 3D discrete coordinate display space is 4096 × 4096 × 4096 pixels).

The obtained values of the relative errors of *3D* interpolation are much smaller in comparison with, for example, the errors of image output to the information display device screen, reaching values from 0.4 ÷ 0.5% to 4 ÷ 5% [2, 8, 10] and not related to the error interpolation. In addition, when using interpolators on the DISC, error occurs only at intermediate points of the segment of the straight line being reproduced. The beginning and end of the interpolation segment exactly coincide with the ideal boundaries of the segment, ie when constructing, for example, a broken line consisting of a sequence of segments, the error of interpolation coordinates of the beginning of the sides of the broken relative to their ideal coordinates will not accumulate.

Table 1.

Error of the 3D linear interpolator on DISC,
binary code Δx, Δy, Δz and $C_i^{1,2}$ for $n=2 \div 16$

n	The value of Δx, value Δy = Δz		The number of pulses at the input of the counter C_i^1, C_i^2		Error, δ_n
	Binary Seniors---Juniors	Decimal	Binary Seniors-----Juniors	Decimal	
2	10 01	2 1	10 01	1 2	0.5774
3	101 010	5 2	010 101	5 2	0.7386
4	1010 0101	10 5	1010 0101	5 10	1.1547
5	10101 01010	21 10	10101 01010	21 10	1.3406
6	101010 010101	42 21	101010 010101	21 42	1.7321
7	1010101 0101010	85 42	1010101 0101010	85 42	1.9230
8	10101010 01010101	170 85	10101010 01010101	85 170	2.3094
9	101010101 010101010	341 170	101010101 010101010	341 170	2.5015
10	1010101010 0101010101	682 341	1010101010 0101010101	341 682	2.8868
11	10101010101 01010101010	1365 682	10101010101 01010101010	1365 682	3.0791
12	101010101010 010101010101	2730 1365	101010101010 010101010101	1365 2730	3.4641
13	1010101010101 0101010101010	5461 2730	1010101010101 0101010101010	5461 2730	3.6565

14	10101010101010 01010101010101	10922 5461	10101010101010 01010101010101	5461 10922	4.0415
15	101010101010101 010101010101010	21845 10922	101010101010101 010101010101010	21845 10922	4.2339
16	1010101010101010 0101010101010101	43690 21845	1010101010101010 0101010101010101	21845 43690	4.6188

Table 2.

Error of 3D line interpolator on the DISC,
Δx, Δy, Δz, C_i^1, C_i^2, x_o^1 and y_o^2 for $n=3\div16$

n	Δx, Δy=Δz	(1) (2)	C_i^1 C_i^2	x_o^1 — y_o^1 x_o^2 — y_o^2	Error, δ_n
3	5 2	(2) (1)	5 2	1 ——— 1 4 ——— 1	0.7386
4	10 5	(1) (2)	5 10	4 ——— 1 6 ——— 4	1.1547
5	21 10	(2) (1)	10 21	6 ——— 4 15 ——— 6	1.3406
6	42 21	(1) (2)	21 42	15 ——— 6 27 —— 15	1.7321
7	85 42	(2) (1)	42 85	27 —— 15 58 —— 27	1.9230
8	170 85	(1) (2)	85 170	58 —— 27 112 —— 58	2.3094
9	341 170	(2) (1)	170 341	112 —— 58 229 —— 112	2.5015
10	682 341	(1) (2)	341 682	229 —— 112 453 —— 229	2.8868
11	1365 682	(2) (1)	682 1365	453 —— 229 912 —— 453	3.0791
12	2730 1365	(1) (2)	1365 2730	912 —— 453 1818 —— 912	3.4641
13	5461 2730	(2) (1)	2730 5461	1818 —— 912 3643 — 1818	3.6565
14	10922 5461	(1) (2)	5461 10922	3643 — 1818 7279 — 3643	4.0415
15	21845 10922	(2) (1)	10922 21845	7279 — 3643 14566 — 7279	4.2339
16	43690 21845	(1) (2)	21845 43690	14566 — 7279 29124 – 14566	4.6188

Conclusions

After analyzing the features of reproduction of straight line segments by the method of digital differential analyzer, the absolute value of the error of the algorithm for interpolation of straight lines in three-dimensional coordinate space - formulas (16) and (17); and moments of time of occurrence of this error - tables 2 and 3. It is revealed that it is admissible to use algorithm of *3D* interpolators of segments of straight lines by a method of the digital differential analyzer taking into account requirements to accuracy in concrete application, for example, in real-time systems, simulators, in animation, in simulators that require maximum speed when changing the image, ie in the presence of dynamic images without strict requirements for the accuracy of image reproduction.

Literature

1. Воронов А.А. и др. Цифровые аналоги для систем автоматического управления. Москва-Ленинград: Издательство Академии Наук СССР, 1960, 196 с.

2. Денисюк В.А. Исследование и разработка цифровых функциональных генераторов графических примитивов для устройств отображения информации: дис. … канд. техн. наук: 05.13.08 / Денисюк Валерий Александрович, Винница, 1996, 202 с.

3. Компьютерная геометрия и алгоритмы машинной графики: Пособие / Никулин Е.А., СПб: БХВ-Петербург, 2015, 554 с.

4. Корн Г. А. Справочник по математике : Определения, теоремы, формулы : Для научных работников и инженеров / Г. А. Корн, Т. М. Корн , СПб. : Лань, 2003, 831 с.

5. Роджерс Д. Алгоритмические основы машинной графики: Пер. с англ., Москва : Мир, 2006, 640 с.

6. Фоли Дж. Основы интерактивной машинной графики: в 2-х книгах / Фоли Дж., вэн Дэм А.; пер. с англ.- М.: Мир, 1985.

7. Херн Д. Компьютерная графика и стандарт OpenGL / Д. Херн , М. Бейкер; пер. с англ., Москва: Издательский дом "Вильямс", 2005, 1168 с.

8. Эдвард Энджел. Интерактивная компьютерная графика. Вводный курс на базе OpenGL, 2-е изд.: Пер с англ., Москва.: Изд. дом "Вильямс", 2001, 592 с.

9. Bresenham, J.E. Algorithm for computer control of a digital plotter. IBM Systems Journal, 4 (1), 1965, pp. 25-30.

10. Denysiuk V. An error of digital integrator of sequential carry interpolator in tasks of computer graphics/ Denysiuk Valerii// Annali d'Italia, №14, 2020, VOL. 1, PP.41-55.

11. James D. Foley. Fundamentals of Interactive Computer Graphics/ James D. Foley, Andries Van Dam, Addison-Wesley Pub. Co., 1982, p. 664.

12. Meyer M.A., Gordon B.M. Pulse-Rate Multipler. Patent 2910237 USA, Filled Dec. 5, 1952, Patented Oct.27, 1959.

FIRE HAZARD ANALYSIS IN THE BUILDING OF THE MOE "SECONDARY EDUCATIONAL SCHOOL No. 1" SARANSK

Savelyev A.P.,
Doctor of Technical Sciences,
professor of the Life safety department,
Institute of mechanics and power,
Ogarev Mordovia State University, Russia
Morozova E.A.
Master's student,
National Research Mordovian State
University named after N. P. Ogarev. Russia

АНАЛИЗ ПОЖАРНОЙ ОПАСНОСТИ В ЗДАНИИ МОУ «СРЕДНЯЯ ОБРАЗОВАТЕЛЬНАЯ ШКОЛА №1» Г.О. САРАНСК

Савельев А.П.,
Д-р техн. наук, проф.
кафедры безопасности жизнедеятельности,
Национальный исследовательский
Мордовский государственный университет
им. Н. П. Огарёва. Россия
Морозова Е.А.
Магистрант,
Национальный исследовательский
Мордовский государственный университет
им. Н. П. Огарёва. Россия

Abstract

The article is devoted to the problems of fire safety in public buildings.

The solution of fire safety problems is possible only on the basis of a comprehensive analysis of the object of research-identifying factors that affect: the possibility of a fire; the spread and development of a fire; the rescue of people and property; the effectiveness of fire elimination.

Studies have found that the safety of public buildings requires more attention, because, as a rule, these are buildings with a mass presence of people.

Аннотация

Статья посвящена проблематике обеспечения пожарной безопасности в общественных зданиях.

Решение задач по обеспечению пожарной безопасности возможно лишь на основе комплексного анализа объекта исследования - выявления факторов, влияющих: на возможность возникновения пожара; на распространение и развитие пожара; на спасение людей и имущества; на эффективность ликвидации пожара.

Проведенными исследованиями установлено, что безопасность общественных зданий требует более пристального внимания, потому что, как правило, это здания с массовым пребыванием людей.

Keywords: fire safety system, regulatory requirements, automatic fire alarm system, warning system and management of evacuation of people in case of fire, smoke removal system, individual fire risk.

Ключевые слова: система обеспечения пожарной безопасности, нормативные требования, автоматическая пожарная сигнализация, система оповещения и управления эвакуацией людей при пожаре, система дымоудаления, индивидуальный пожарный риск.

№20 2021

Annali d'Italia

VOL. 1

ISSN 3572-2436

The journal is registered and published in Italy.
Articles are accepted each month.
Frequency: 12 issues per year.
Format - A4 All articles are reviewed
Free access to the electronic version of journal

500 copies

Annali d'Italia

50134, Via Carlo Pisacane, 10, Florence, Italy

email: info@anditalia.com

site: https://www.anditalia.com/