Solomon I. Khmelnik, Inna S. Doubson

Computing of the complex variable functions

Published by "MiC" - Mathematics in Computers Comp.
BOX 15302, Bene-Ayish, Israel, 60860
Fax: ++972-8-8691348
solik@netvision.net.il
Printed in United States of America, Lulu Inc., ID **10803662**, 2011
ISBN 978-1-257-82660-5

Israel
2011

Annotation

Hardware algorithms for computing of all elementary complex variable functions are proposed.

Contents

Introduction

In connection with existence of codes of complex numbers [1, 2] arise , naturally, new opportunities for functions computation of complex argument and the decision of the equations with complex variables.

In the given book methods of the decision of these problems, named by a method «digit-by-digit» is described. This method is known as applied to real numbers [3]. It is shown below, that it can be generalized on codes of complex numbers by introduction of some generalized operation of comparison of codes.

The hardware for these calculations is described in [4]

1. A method «digit-by-digit»

Let *Z* - some complex number. We shall consider the sequence of complex numbers $Z_h \; (h=1,\ldots,m)$:

$$Z_1,\; Z_2,\; Z_3,\; \ldots Z_h,\; Z_{h+1},\; \ldots Z_m,$$

the sequence of complex codes $K(Z-Z_h)$:

$$K(Z-Z_1),\; K(Z-Z_2),\; \ldots K(Z-Z_h),\; \ldots K(Z-Z_m)$$

Let us suppose that the complex number code has a certain characteristic, further referred to as **code size**. We shall express this code characteristic of the complex number *Z* by the symbol, $NK(Z)$. *The highest significant digit number* or *the modulus* of the encoded number can be used as code size. Let us consider the sequence of sizes, $NK(Z-Z_h)$:

$$NK(Z-Z_1),\; NK(Z-Z_2),\; \ldots NK(Z-Z_h),\; \ldots NK(Z-Z_m)$$

We will refer to the numbers sequence Z_h as the generating sequence $\{Z_h\}$, if with a random $h=1,\ldots,m$ the following condition is satisfied:

$$NK(Z-Z_{h+1}) \leq NK(Z-Z_h) \qquad (1)$$

The method of comparison of the size of codes is considered further.

The numbers Z_h are formed in such a manner as to satisfy the recurrent condition:

$$Z_{h+1} = \Phi\left[Z_h,\; \varepsilon_{h+1},\; \rho^{h+1}\right] \qquad (2)$$

where $\varepsilon_h = 0,\ 1,\ 2,...;\ \rho$ - is the basis of the encoding system. The expression ρ^h will be further used to express the basic function $f(\rho,\ h)$, which (as shown above) does not always equal the value of ρ^h.

Considering (1) and (2) simultaneously, we find that:

$$NK\left[Z - \Phi\left(Z_h,\ \varepsilon_{h+1},\ \rho^{h+1}\right)\right] \le NK\left[Z - Z_h\right]. \qquad (3)$$

For codes of real numbers on the real radix the inequality (3) is equivalent to an inequality with modules

$$\left|Z - \Phi\left(Z_h,\ \varepsilon_{h+1},\ \rho^{h+1}\right)\right| \le \left|Z - Z_h\right|,$$

as in this case the code with big number of the senior meaning category has the greater module and on the contrary.

Let's return again to generating sequence $\{Z_h\}$. Everyone *h*-member Z_h this sequence represents number *Z* with some accuracy, namely to within the maximal module of the code having the size $NK(Z - Z_h)$. Hence, representation of number *Z* with an allowable maximal absolute error Δ equivalently to calculation such *m*-member Z_m of generating sequence at which $NK(Z - Z_m)$ has the size equal to the maximal size of number with the module, not exceeding Δ.

Let us denote:

$$NK(Z - Z_m) = H(m). \qquad (4)$$

2. Decomposition

2.1. Introduction

Let us proceed to describing the Z_m calculation algorithm, for which purpose we have first to consider the transition process from Z_h to Z_{h+1}. The *h* generator of Z_h is known. In order to determine the *(h+1)* generator of Z_{h+1} we must find the maximum value, a_{h+1}, at which Z_{h+1} calculated using formula (1.2) will still satisfy condition

(1.1). Obviously, $Z_{h+1}^{(0)} = Z_h$ satisfies this condition if $a_{h+1} = 0$, where $a_{h+1} = 1$. The value, $\overline{Z}_{h+1}^{(1)} = \Phi\left[Z_h,\ 1,\ \rho^{h+1}\right]$ should be tested to see if it satisfies condition (1.1). The test consists in comparing the code values of numbers $\left(Z - Z_h\right)$ and $\left(Z - \overline{Z}_{h+1}^{(1)}\right)$ There are three possible options in this comparison:

a) $NK\left[Z - \overline{Z}_{h+1}^{(1)}\right] < NK\left[Z - Z_h\right]$ - this inequality indicates that $a_{h+1} = 1$ and $Z_{h+1} = Z_{h+1}^{(1)}$;

b) $NK\left[Z - \overline{Z}_{h+1}^{(1)}\right] > NK\left[Z - Z_h\right]$ - this inequality indicates that $a_{h+1} = 0$ and $Z_{h+1} = Z_h$;

c) $NK\left[Z - \overline{Z}_{h+1}^{(1)}\right] = NK\left[Z - Z_h\right]$- this inequality indicates that $a_{h+1} \geq 1$ and values $\overline{Z}_{h+1}^{(2)} = \Phi\left[Z_h,\ 2,\ \rho^{h+1}\right]$ or $\overline{Z}_{h+1}^{(2)} = \Phi\left[Z_{h+1}^{(1)},\ 1,\ \rho^{h+1}\right]$should be tested for satisfying condition (1.3). This test is carried out in a similar way by comparing the codes of numbers $\left(Z - Z_{h+1}^{(1)}\right)$ and $\left(Z - Z_{h+1}^{(2)}\right)$ The test results either in the determination of the values of $a_{h+1} = 1$ or $a_{h+1} = 2$ and $Z_{h+1} = \overline{Z}_{h+1}^{(1)}$ or $Z_{h+1} = \overline{Z}_{h+1}^{(2)}$, or in the transition to code inquiry of numbers $\left(Z - \overline{Z}_{h+1}^{(2)}\right)$ and $\left(Z - \overline{Z}_{h+1}^{(3)}\right)$ where $\overline{Z}_{h+1}^{(3)} = \Phi\left[Z_{h+1}^{(2)},\ 1,\ \rho^{h+1}\right]$

Consequently, as a result of the consistent use of testing with respect to code values of $\left(Z - \overline{Z}_{h+1}^{(\mu-1)}\right)$ and $\left(Z - \overline{Z}_{h+1}^{(\mu)}\right)$ where

$\mu = 0,\ 1,\ 2,\ldots a_{h+1}$, the value, $Z_{h+1} = \overline{Z}_{h+1}^{(a_{h+1})}$, is determined based on the known value Z_h. Obviously, the calculation process of the *(h + 1)* generator by the known *h* generator may be initiated from the first Z_1 generator and finished with the *m*^{*th*} generator of Z_m. The algorithm describing the *h*^{*th*} calculation cycle of Z_m looks like this:

2.2. Algorithm of decomposition

In the future calculation of the generating sequence will be called the decomposition, ie the representation of a complex number as a sum or product of some other famous numbers - the elements of the expansion.

The algorithm, describing h-cycle of calculation Z_m, has the following appearance.

The difference is known previous generatrix $\overline{Z}_h^{(\mu)} = Z_{prev}\ (\mu = 0,1,2,\ldots)$, $H(m),\ h$ and difference $E_{prev} = Z - Z_{prev}$.

1. The next expected generator $\overline{Z}_h^{(\mu+1)} = Z_{next}$ is determined using the formula

$$Z_{next} = \Phi\left[Z_{prev},\ 1,\ \rho^{h+1}\right] \qquad (1)$$

2. Calculate the difference $E_{next} = Z - Z_{next}$.
3. Compare values $H_{prev} = NK\left(E_{prev}\right)$ and $H_{next} = NK\left(E_{next}\right)$. Then, according to the comparison results of H_{prev} and H_{next}:
4. Determine the number $h' = \begin{cases} h \text{ if } H_{next} = H_{prev} \\ h+1 \text{ if } H_{next} \neq H_{prev} \end{cases}$ of the next cycle.

5. Determine the value of the previous generator $Z'_{prev} = \begin{Bmatrix} Z_{prev} \text{ if } H_{next} > H_{prev} \\ Z_{next} \text{ if } H_{next} \le H_{prev} \end{Bmatrix}$ for the next cycle.
6. Determine the difference $E'_{prev} = Z - Z'_{prev}$ for the next cycle.
7. Examine the fulfillment of the condition $H_{next} = H(m)$: if this equality is satisfied, it indicates that the *m^th^* generator has been found for Z_m, i.e. the calculation is finished; if it is not satisfied, then the h' calculation cycle is carried out.

Table 1

Result of check	h'	Z'_{prev}	$Z - Z'_{prev}$	W'_{prev}
$H_{next} < H_{prev}$	$h+1$	Z_{next}	$Z - Z_{next}$	W_{next}
$H_{next} = H_{prev}$	h	Z_{next}	$Z - Z_{next}$	W_{next}
$H_{next} > H_{prev}$	$h+1$	Z_{prev}	$Z - Z_{prev}$	W_{prev}

Items 4, 5, 6 are carried out according to table 1 (*without a column* W'_{prev}). Thus if $NK(Enext) \le NK(Eprev)$, then corresponding element is included in structure of decomposition, i.e. the number a_h increases on 1; besides value of size *E* is updated $(Eprev{=}Enext)$ and carried out transition to the following step with former value *h*. Otherwise transition to the following step with reduced on 1 value *h* and value $a_h = 0$ is carried out.

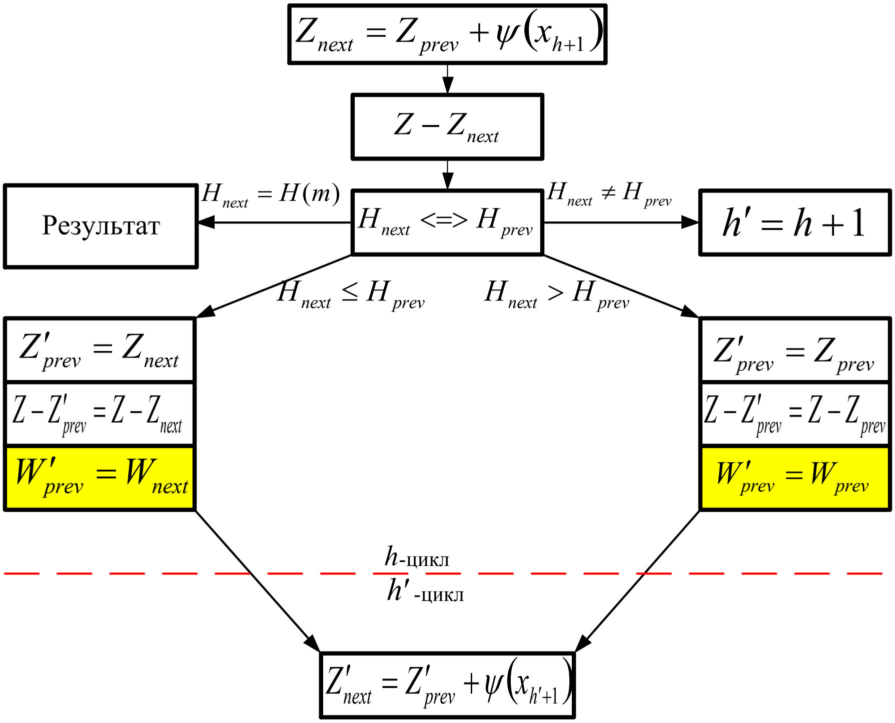

Fig. 1.

The sequence of numbers a_h grows out decomposition. Each potential element of decomposition can be absent in concrete decomposition or be present at it, repeating a_h time, i.e. some cycles can have same number (at $a_{h+1} > 1$); therefore the number of cycles Ω can surpass number of generatrixes *m*.

Thus, on algorithm of decomposition the generating sequence $\{Z_h\}$ is calculated. This algorithm contains only elementary instructions of addition, subtraction and comparison of the sizes and consequently is easily realized in the arithmetic unit. More evidently the sequence of elementary operations of algorithm of decomposition is represented on fig. 1 (*without the allocated block*).

2.3. Variants of decomposition

Table 2 and table 3 lists the types of decomposition, which are used later, and the formula for calculating them.

Table 2

#	Designation	Application
D2	DecompLogar	Decomposition of the sum of logarithms
D3	DecompBinom	Decomposition of the product binomials
D32	DecompBinomInv	Decomposition into an inverted product binomials
D4	DecompBinom2	Decomposition of the product of the squares of binomials
D42	DecompBinom2inv	Decomposition into product of inverse squares of binomials
D2R	DecompLogarReal	Decomposition of the sum of the logarithms of real numbers (h - even)
D3R	DecompBinomReal	Decomposition of the product of real binomials (h - even)
D32R	DecompBinomInvReal	Decomposition into an inverted product of real binomials (h - even)
D4R	DecompBinom2real	Decomposition into product of squares of real binomials (h - even)
D42R	DecompBinom2invReal	Decomposition into product of inverse squares of real binomials (h – even)

Detailed algorithms for these decomposition are resulted further. In these algorithms the following designations are accepted:

Compar– function of comparison of complex numbers on the module,

Accum– function of record in the given category of the accumulator,

Even– function of check of parity of number,

Logar – function of a choice of a constant $\ln\left(1+\rho^{-h}\right)$,

Z– given complex number,

Eprev– previous value of result,

Enext– next value of result,

DZ– increment,

H – current number of the category, Hmin =< H =< Hmax

Iter– counter of iterations,

R– current value of the category of a collector,
A, B –intermediate results (complex numbers).

Table 3.

#	Decomposition	Zo	Enext =	Eo
D2	$Z=\sum a_h \ln\left(1+\rho^{-h}\right)$	0	$\text{Eprev}-\ln\left(1+\rho^{-h}\right)$	Z
D3	$Z=\prod\left(1+\rho^{-h}\right)^{a_h}$	1	$\text{Eprev}-(Z-\text{Eprev})\rho^{-h}$	Z-1
D32	$\frac{1}{Z}=\prod\left(1+\rho^{-h}\right)^{a_h}$	Z	$\text{Eprev}-(1-\text{Eprev})\rho^{-h}$	1-Z
D4	$Z=\prod\left(1+\rho^{-h}\right)^{2\cdot a_h}$	1	$Z-(Z-\text{Eprev})\left(1+\rho^{-h}\right)^2$	Z-1
D42	$\frac{1}{Z}=\prod\left(1+\rho^{-h}\right)^{2\cdot a_h}$	Z	$1-(1-\text{Eprev})\left(1+\rho^{-h}\right)^2$	1-Z
D2R	$Z=\sum a_h \ln\left(1+\rho^{-h}\right)$	0	$\text{Eprev}-\ln\left(1+\rho^{-h}\right)$	Z
D3R	$Z=\prod\left(1+\rho^{-h}\right)^{a_h}$	1	$\text{Eprev}-(Z-\text{Eprev})\rho^{-h}$	Z-1
D32R	$1=Z\cdot\prod\left(1+\rho^{-h}\right)^{a_h}$	Z	$\text{Eprev}-(1-\text{Eprev})\rho^{-h}$	1-Z
D4R	$Z=\prod\left(1+\rho^{-h}\right)^{2\cdot a_h}$	1	$Z-(Z-\text{Eprev})\left(1+\rho^{-h}\right)^2$	Z-1
D42R	$1=Z\cdot\prod\left(1+\rho^{-h}\right)^{2\cdot a_h}$	Z	$1-(1-\text{Eprev})\left(1+\rho^{-h}\right)^2$	1-Z

```
Function DecompBinom (Z)
Function DecompBinom2 (Z)
        H = Hmax
        Iter = 0
Begin:
        R = 0
        If H = Hmax Then Eprev = Z - 1
        Im(A)=0
        If Even(H) Then Re(A) = (-2)^H Else Re(A) = (-2)^(H-1)
Compar:
        B = Z - Eprev
        If Even(H)  Then        ' shift on 2H categories to the right
                DZ = A
        Else    ' shift on 2 (H-1) categories to the right and multiplication on j
                DZ = j * A
```

```
          End If
          B = B + DZ * B
          B = B + DZ * B          ‘ is present only at function DecompBinom2
          Enext = Z - B
          Iter = Iter + 1
          If Compar(Enext, Eprev) = 1 Then      ‘ Enext=<Eprev
                    R = R + 1
                    Eprev = Enext
                    GoTo Compar
          Else                                  ‘ Enext>Eprev
                    Enext = Eprev
          End If
          Accum(-H + Hmax) = R
          H = H - 1
          If H = Hmin Then GoTo End Else GoTo Begin
End:
```

<u>Function DecompBinomReal (Z)</u>
<u>Function DecompBinom2Real (Z)</u>

```
          H = Hmax
          Iter = 0
Begin:
          R = 0
          If H = Hmax Then Eprev = Z - 1
          Im(A)=0
          Re(A) = (-2)^H
Compar:
          B = Z - Eprev
          DZ = A                        ' shift on 2H categories to the right
          B = B + DZ * B
          B = B + DZ * B          ‘ is present only at function DecompBinom2Real
          Enext = Z - B
          Iter = Iter + 1
          If Compar(Enext, Eprev) = 1 Then               ‘ Enext=<Eprev
                    R = R + 1
                    Eprev = Enext
                    GoTo Compar
          Else                                           ‘ Enext>Eprev
                    Enext = Eprev
          End If
          Accum(Hmax - H) = R
          H = H - 2
          If H <= Hmin Then GoTo End Else GoTo Begin
End:

Function DecompLogar (Z)
Function DecompLogarReal (Z)
          H = Hmax
          Iter = 0
Begin:
          R = 0
          If H = Hmax Then     Eprev = Z Else Eprev=Enext
          DZ = Logar(H)
          Enext=Eprev-DZ
Compar:
          Iter = Iter + 1
          If Compar(Enext, Eprev) = 1 Then     'Enext=<Eprev
                    R = R + 1
                    Eprev = Enext
                    Enext = Eprev - DZ
                    GoTo Compar
          Else                                 'Enext>Eprev
                    Enext = Eprev
          End If
          Accum(-H + Hmax) = R
          H = H – 1                   'for function DecompLogar
          H = H – 2                   'for function DecompLogarReal
          If H = Hmin Then GoTo End Else GoTo Begin
End:
```

Fig. 2 shows a diagram of decomposition algorithms *DecompBinom* and *DecompBinom2*. Highlighted in this figure refers only to block the last of these decompositions. Fig. 3 shows the decomposition algorithms *DecompBinomReal* and *DecompBinom2Real*. Highlighted in this figure refers only to block the last of these decompositions.

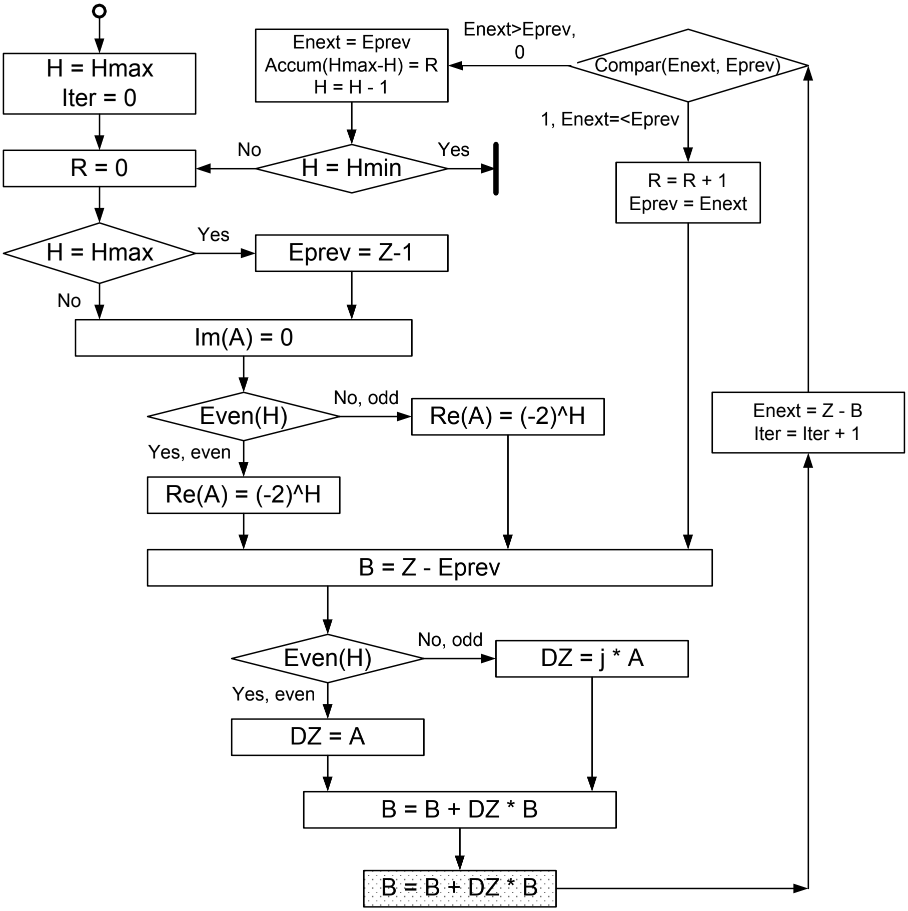
H = Hmax
Iter = 0
R = 0
H = Hmax
Yes
No
Eprev = Z-1
Im(A) = 0
Even(H)
No, odd
Yes, even
Re(A) = (-2)^H
Re(A) = (-2)^H
B = Z - Eprev
Even(H)
No, odd
Yes, even
DZ = j * A
DZ = A
B = B + DZ * B
B = B + DZ * B
Enext = Z - B
Iter = Iter + 1
Compar(Enext, Eprev)
Enext>Eprev,
0
1, Enext=<Eprev
R = R + 1
Eprev = Enext
Enext = Eprev
Accum(Hmax-H) = R
H = H - 1
H = Hmin
No
Yes

Fig. 2

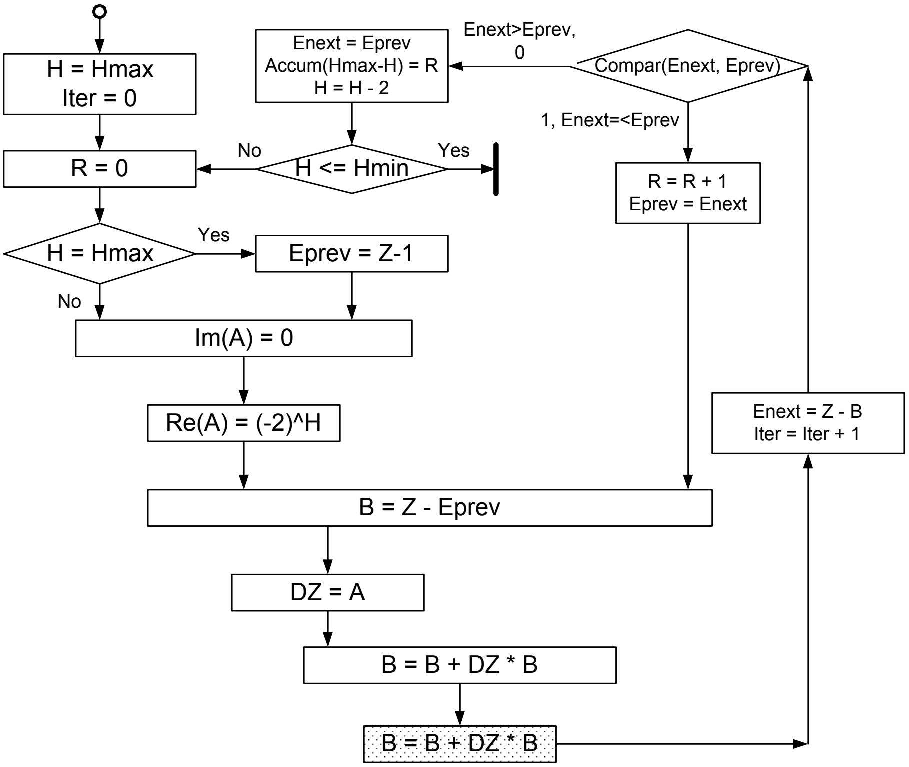

Fig. 3

Fig. 4 shows a algorithm diagram of decompositions *DecompBinomLogar* and *DecompBinomLogarReal*. Highlighted in this figure blocks «H-1» and «H-2» relate only to the decomposition *DecompBinomLogar* and *DecompBinomLogarReal* respectively.

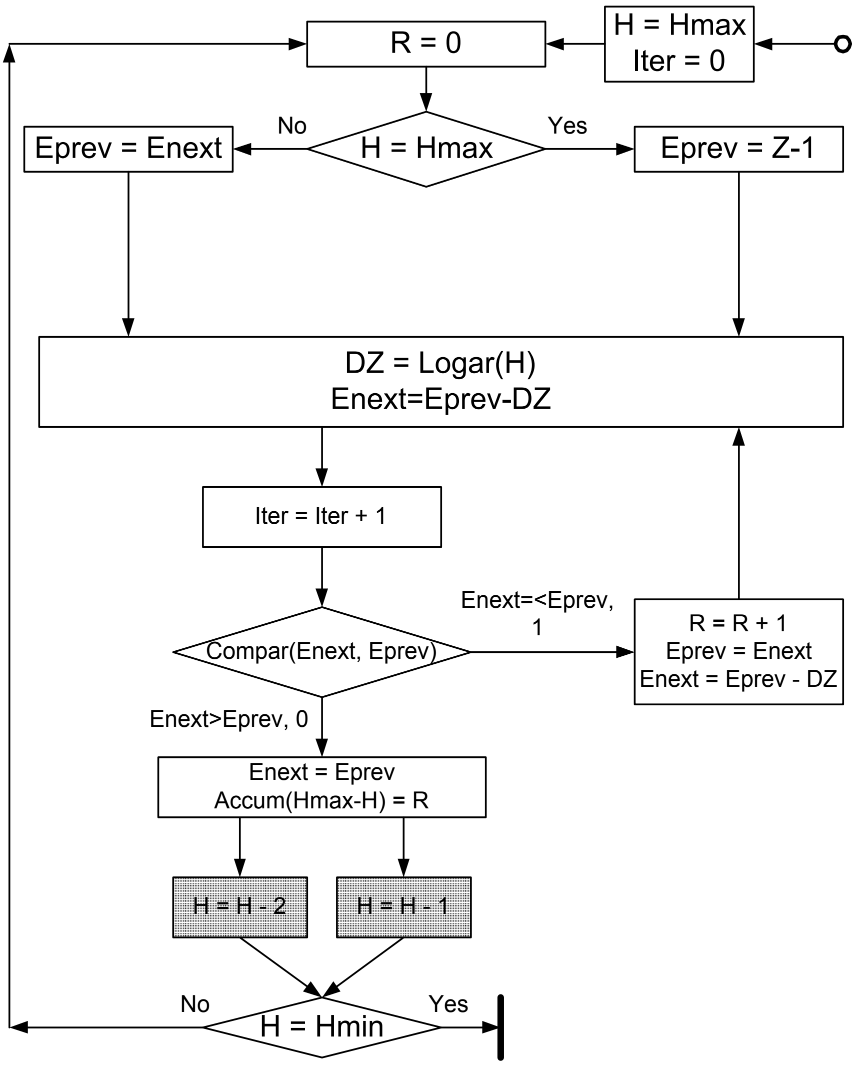

Fig. 4

3. Compositions

As stated earlier, the result of decomposition is the a_h sequence of numbers. Obviously, it is possible to restore the number that was decomposited from this sequence. We are going to refer to this calculation as **composition**. This operation is the opposite of decomposition and consists in calculating a complex number as a sum or product of certain other known numbers that are elements of decomposition. In this context such representation of a number will be called **composition**. It should also be noted that in the process of composition the elements of decomposition may be converted or substituted by other elements. That is the essence of composition and we will use this method later on.

Table 4.

№	Designation	Application	Formula
C2	CompBinom	Composition binomials	$W=\prod\left(1+\rho^{-h}\right)^{a_h}$
C2A	CompBinomA	Composition binomials	$W=A\cdot\prod\left(1+\rho^{-h}\right)^{a_h}$
C3	CompLogar	Composition for the logarithm	$W=\sum a_h\ln\left(1+\rho^{-h}\right)$
C4	CompLogarAngle	Composition for the argument of complex number	$W=-j\cdot\mathrm{Im}\sum a_h\left[\ln\left(1+\rho^{-h}\right)\right]$
C5	CompLogarModul	Composition for the logarithm of the modulus	$W=\mathrm{Re}\sum a_h\left[\ln\left(1+\rho^{-h}\right)\right]$
C7	CompBinomConjug	Composition for the adjoint of the square root	$W=\prod\left(1+\tilde{\rho}^{-h}\right)^{a_h}$
C8	CompBinomModul	Composition for the module	$W=\prod\left[\left(1+\rho^{-h}\right)\left(1+\tilde{\rho}^{-h}\right)\right]^{a_h}$
C8A	CompBinomModul	Composition for the module	$W=A\cdot\prod\left[\left(1+\rho^{-h}\right)\left(1+\tilde{\rho}^{-h}\right)\right]^{a_h}$
C9	CompBinom2	Composition binomials squared	$W=\prod\left(1+\rho^{-h}\right)^{2a_h}$
C9A	CompBinom2A	Composition binomials squared	$W=A\cdot\prod\left(1+\rho^{-h}\right)^{2a_h}$
C4 and C9A		Argument and the module (once)	$\arg(Z)=-2j\cdot\mathrm{Im}\left(\sum\ln\left(1+\rho^{-h}\right)\right)$ $\lvert Z\rvert=\prod\left(1+\rho^{-h}\right)\left(1+\tilde{\rho}^{-h}\right)$

In the table 4 variants of compositions are submitted. The composition represents iterative process, where on each step the number a_h is known and change of the previous value of required complex number under the formulas resulted in table 1 is carried out. Each potential element of decomposition can be absent in concrete decomposition or be present at it, repeating a_h time.

Detailed algorithms for these composition are resulted further. In these algorithms the following designations are accepted:

A – given complex number; argument of some compositions (is present at compositions with the termination 'A'; in other compositions A=1),

Accum– function of record in the given category of the accumulator,

Even– function of check of parity of number,

Logar – function of a choice of a constant $\ln\left(1+\rho^{-h}\right)$,

Unit – function of a choice of a constant «1 in the h-category» or shift of complex number A on h-categories (at And <> 1),

Wprev– previous value of result,

Wnext– next value of result,

DW– increment,

H – current number of the category, Hmin =< H =< Hmax

Iter– counter of iterations,

k – counter of value of the category of a collector,

R– current value of the category of a collector,

B – intermediate variable (complex number).

Fig. 5 shows a chart of algoritm *CompBinom*.
Fig. 6 shows a chart of algoritm *CompLogar*.

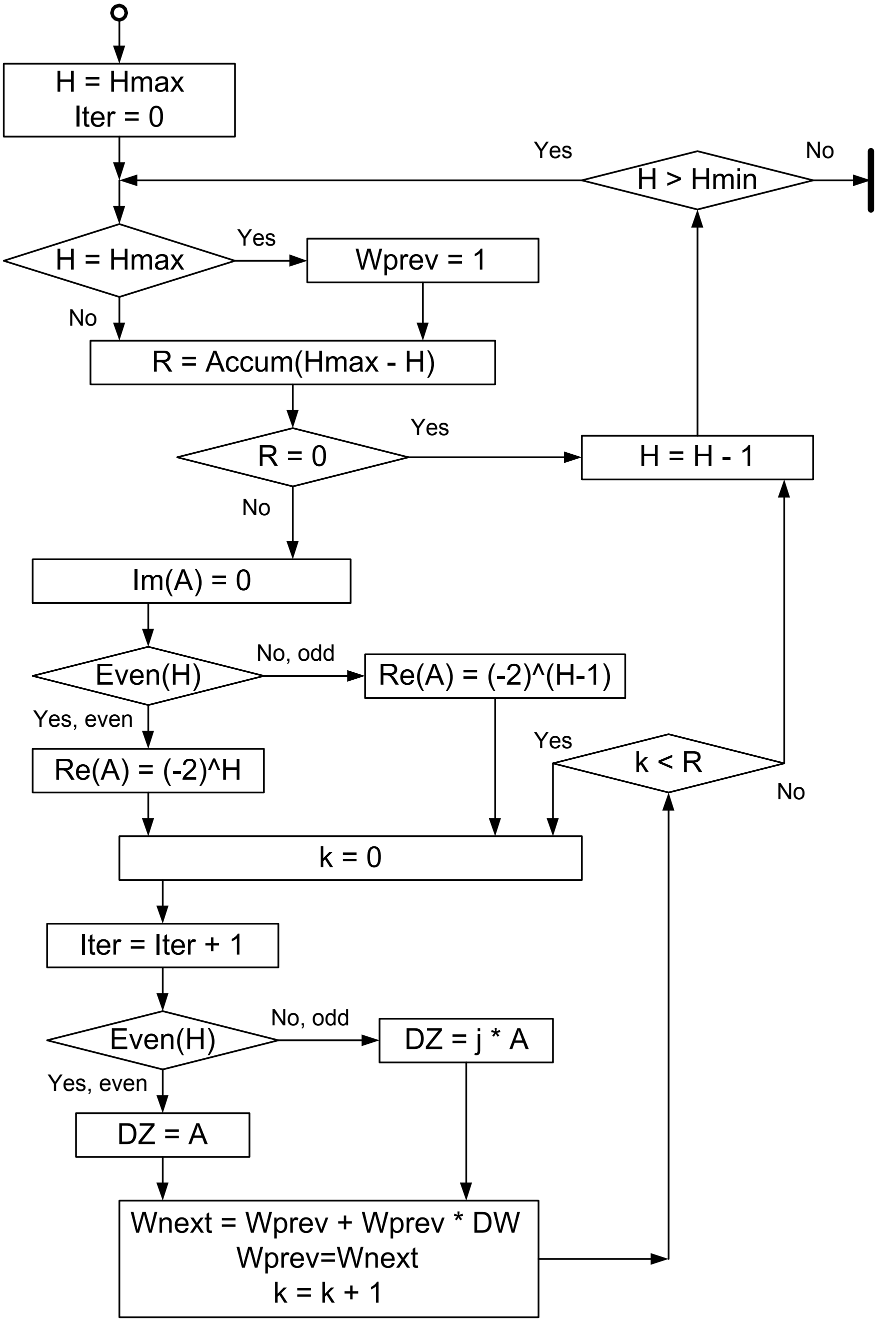
H = Hmax
Iter = 0
Yes
H > Hmin
No
H = Hmax
Yes
Wprev = 1
No
R = Accum(Hmax - H)
Yes
R = 0
H = H - 1
No
Im(A) = 0
Even(H)
No, odd
Re(A) = (-2)^(H-1)
Yes, even
Yes
k < R
Re(A) = (-2)^H
No
k = 0
Iter = Iter + 1
Even(H)
No, odd
DZ = j * A
Yes, even
DZ = A
Wnext = Wprev + Wprev * DW
Wprev=Wnext
k = k + 1

Fig. 5.

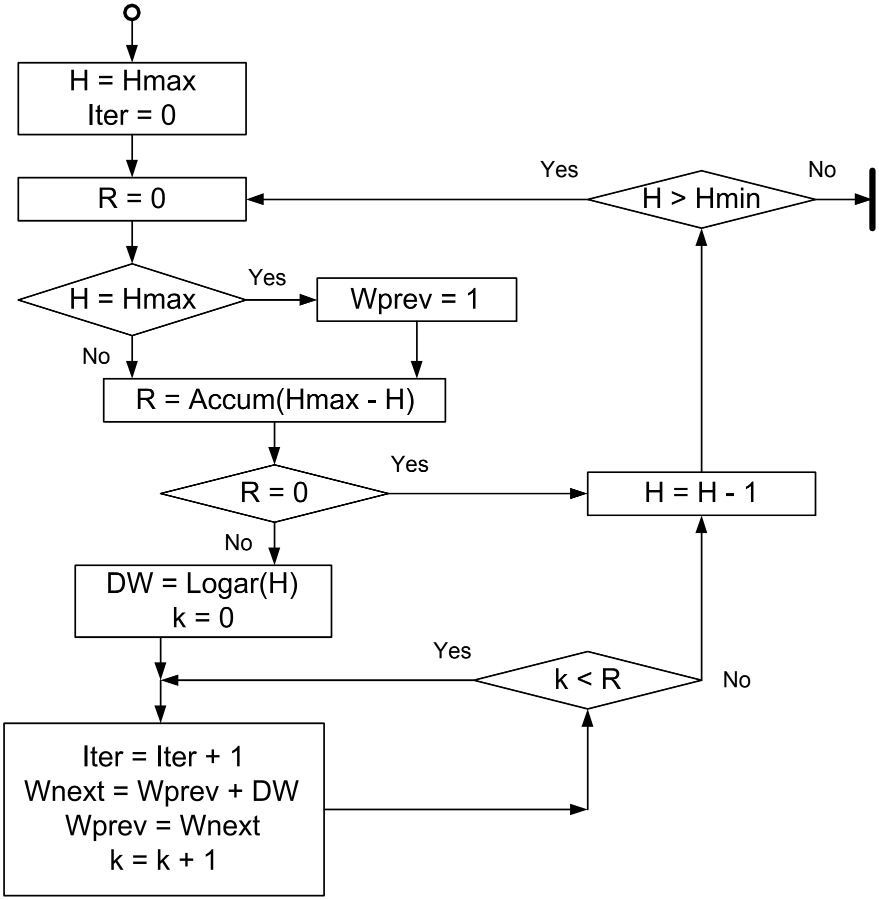

Fig. 6.

```
Function CompBinom ()
          H = Hmax
          Iter = 0
Begin:
          If H = Hmax Then Wprev=1
          R = Accum(Hmax - H)
          If R = 0 Then GoTo Lab2
          Im(A)=0
          If Even(H) Then Re(A) = (-2)^H Else Re(A) = (-2)^(H-1)
          k = 0
Lab1:
          Iter = Iter + 1
          If Even(H)  Then        ' shift on 2H categories to the right
                    DW = A
          Else       ' shift on 2 (H-1) categories to the right and multiplication on j
                     DW = j * A
          End If
          Wnext = Wprev + Wprev * DW
          Wprev=Wnext
          k = k + 1
          If k < R Then GoTo Lab1
Lab2:
          H = H - 1
          If H > Hmin Then GoTo Begin
End:

Function CompLogar ()
          H = Hmax
          Iter = 0
Begin:
          R = 0
          If H = Hmax Then     Wprev = 1
          R = Accum(Hmax - H)
          If R = 0 Then GoTo Lab1
          DW = Logar(H)
          k = 0
Lab2:
          Iter = Iter + 1
          Wnext = Wprev + DW
          Wprev = Wnext
          k = k + 1
          If k < R Then GoTo Lab2
Lab1:
          H = H - 1
          If H > Hmin Then GoTo Begin
End:
```

4. Two-step Operations

4.1. Introduction

Let us now consider a certain function, $W = f(Z)$. If the generating sequence $\{Z_h\}$ for the number Z is known, then the generating sequence $\{W_h\}$ of the number W can be calculated, since $W_h = f(Z_h)$. Specifically, the previous generator, $W_{prev} = f(Z_{prev})$, and the next generator, $W_{next} = f(Z_{next})$, or taking into account the formula (1.2),

$$W_{next} = f\left\{\Phi\left[Z_{prev},\ 1,\ \rho^{h+1}\right]\right\} \tag{1}$$

It is often possible to represent the expression (1) as follows:

$$W_{next} = \Psi\left[W_{prev},\ 1,\ \rho^{h+1}\right] \tag{2}$$

Thus, calculation *m*-generatrix W_m, representing function $W = f(Z)$ with an allowable maximal absolute error Δ', is reduced to algorithm of decomposition and calculations under the formula (2). The number generatrixes *m* is defined, still, from a condition (1.4), where Z_m - *m*-generatrix of argument, having an absolute error Δ. The necessary value of an error Δ is defined by value of an error Δ' and a kind of function $W = f(Z)$.

4.2. Algorithm of function evaluation

So, calculation of value of function with an allowable error can be made on algorithm, which basically coincides with algorithm of decomposition and describes *h*-cycle of calculation. Difference consists only that still the value W_{prev} in the beginning is known, and the item 5 of algorithm has the following view:

5. Value previous generatrix Z'_{prev} and W'_{prev} for the following cycle, where $Z'_{prev} = \left\{\begin{matrix} Z_{prev} \text{ if } H_{next} > H_{prev} \\ Z_{next} \text{ if } H_{next} \le H_{prev} \end{matrix}\right\}$ and

$W'_{prev} = \begin{cases} W_{prev} \text{ if } H_{next} > H_{prev} \\ W_{next} \text{ if } H_{next} \leq H_{prev} \end{cases}$, is defined. Thus it

W_{next} is calculated (if it is necessary) under the formula (2).

Items 4, 5, 6 are carried out according to table 1 (*with a column* W'_{prev}). More evidently the sequence of elementary operations of algorithm of decomposition is represented on fig. 1 (*with the allocated block*).

The expediency of function computing on this algorithm is completely defined by complexity of calculations under formulas (1) and (2), i.e. a kind of function $W = f(Z)$. In the event that calculations under formulas (1) and (2) are reduced to such operations which are simply realized on digital schemes, application of a method «digit-by-digit» for function $W = f(Z)$ computing it is effect. This method can be realized both program, and hardware way. The set of operations of the processor should contain operation of comparison of the sizes of codes, after which signals of conditional transition on this or that direction in the program of calculations are developed, in the first case. The hardware way can be used, if calculations under formulas (1) and (2) are made only by means of so-called "short" or elementary machine operations: shift, addition, subtraction.

4.3. About hardware realization

At hardware realization of the specified algorithm, as a rule, it is expedient to carry out it in two stages:

1) decomposition of argument *Z*, i.e. calculation of sequence of numbers a_h,
2) composition, i.e. iterative function $W = f(Z)$ calculation under the formula

$$W_{next} = \Psi\left[W_{prev},\ a_h,\ \rho^{h+1}\right]$$

So, hardware calculation of some functions $W = f(Z)$ will consist of two steps with use of a collector (described above):

1. decomposition of number *Z* and filling of a collector,
2. composition of number *W* at the given filling a collector.

Table 5.

Operation	D	C
$\lvert Z\rvert = \prod \left(1+\rho^{-h}\right)\left(1+\tilde{\rho}^{-h}\right)$	**D4**	**C8**
$\sqrt{Z} = \prod \left(1+\rho^{-h}\right)$	**D4**	**C2**
$\sqrt{\tilde{Z}} = \prod \left(1+\tilde{\rho}^{-h}\right)$	**D4**	**C7**
$e^{Z} = \prod \left(1+\rho^{-h}\right)$	**D2**	**C2**
$\ln(Z) = \sum \ln\left(1+\rho^{-h}\right)$	**D3**	**C3**
$\ln(Z) = 2\cdot\sum \ln\left(1+\rho^{-h}\right)$	**D4**	**C3**
$\arg\left(Z\right) = -j\cdot \mathrm{Im}\sum \ln\left(1+\rho^{-h}\right)$	**D3**	**C4**
$\arg\left(Z\right) = -2j\cdot \mathrm{Im}\sum \ln\left(1+\rho^{-h}\right)$	**D4**	**C4**
$\ln\lvert Z\rvert = \mathrm{Re}\sum \ln\left(1+\rho^{-h}\right)$	**D3**	**C5**
$\frac{1}{Z} = \sum \left(\rho^{-h}\right)$	**D1**	**C1**
$Z = \lvert Z\rvert\cdot e^{j\varphi} = \lvert Z\rvert\cdot\prod \left(1+\rho^{-h}\right)$	**D2**	**C2A**
$\arg(Z) = -2j\cdot \mathrm{Im}\left(\sum \ln\left(1+\rho^{-h}\right)\right),$ $\lvert Z\rvert = \prod \left(1+\rho^{-h}\right)\left(1+\tilde{\rho}^{-h}\right)$	**D4**	**C4 and C8**

In table 5 such operations are listed. Communication is specified in table 6 between functions, compositions and decomposition.

Table 6.

	D1	**D2**	**D3**	**D4**
C1	Division			
C2		Exponentiation	**Identically**	Square-rooting
C3		**Identically**	Logarithm	
C4			Argument	
C5			Modulus Logarithm	
C7				Conjugated Square-rooting
C8				Modulus
C9				**Identically**

5. Taking the logarithm

5.1. Definition of the natural logarithm of complex number (a variant 1).

Taking the logarithm it is based on the following: if $y=\prod\left(1+\rho^{-h}\right)^{ah}$, then $\ln(y)=\sum a_h\ln\left(1+\rho^h\right)$. The algorithm of definition of the natural logarithm of complex number will consist in the following:

1. The number $Z=(-2)^k y$, where *y* - mantissa, *k* – exponent, is given.
2. Decomposition under the formula $y=\prod\left(1+\rho^{-h}\right)^{ah}$ - "*DecompBinom*".
3. Composition under the formula $\ln(y)=w=\sum a_h\ln\left(1+\rho^h\right)$ - "*CompLogar*".
4. Calculation of $\ln(Z)=\left[k\cdot\ln(-2)+\ln(y)+S\cdot\ln(-1)\right]$.

Here $\ln(-2)=\ln(-1)+\ln(2)=j\pi+\ln(2)$.

Thus, $\ln(Z)=k\cdot\ln(2)+w+(k+S)j\pi=\operatorname{Re}Z+\operatorname{Im}Z$,

where $\operatorname{Re}Z=k\cdot\ln(2)+\operatorname{Re}w,\ \operatorname{Im}Z=j\pi(k+S)+\operatorname{Im}w$.

5. Definition of a principal value of the natural logarithm. In it the imaginary part is in limits $-\pi\le\operatorname{Im}Z\le\pi$. Therefore the imaginary part of a principal value is defined under the formula:

$$g=\operatorname{int}\left[\frac{\operatorname{Im}Z}{2\pi}\right],\ (\operatorname{Im}Z)_{main}=\left\{\begin{array}{l}g\ \text{if}\ g\le\pi,\\ g-2\pi\ \text{if}\ g>\pi.\end{array}\right\}.$$

6. Normalization of result.

5.2. Calculation of the logarithm of the module of complex number

The algorithm of calculation of the logarithm of the module of complex number is based that $\ln(|Z|)=\operatorname{Re}\ln(Z)$, and differs from algorithm taking the logarithm by a variant 1 only that in item 3 instead of a composition “*CompLogar*” the composition “*CompLogarModul*” is used:

3. Composition under the formula $W = \mathrm{Re}\sum a_h \left[\ln\left(1+\rho^{-h}\right)\right]$- “*CompLogarModul*”.

5.3. Definition of the natural logarithm of complex number (a variant 2)

Taking the logarithm in this case it is based on the following: if $y = \prod\left(1+\rho^{-h}\right)^{2\cdot a_h}$, then $\ln(y) = 2\cdot\sum a_h \ln\left(1+\rho^{h}\right)$ Accuracy taking the logarithm by this variant is less than accuracy taking the logarithm by the first variant, but as it will be clear from the further, this variant is well combined with other calculations. The algorithm will consist in the following:

1. The number $Z = (-2)^k y$, where *y* - a mantissa, *k* – exponent, is given.

2. Decomposition under the formula $y = \prod\left(1+\rho^{-h}\right)^{2\cdot a_h}$ - (“*DecompBinom2*”).

3. Composition under the formula $w = \sum a_h \ln\left(1+\rho^{h}\right)$ - (“*CompLogar*”).

4. Calculation of $\ln(Z) = \left[k\cdot\ln(-2)+\ln(y)\right]$.

Here $\ln(-2) = \ln(-1)+\ln(2) = j\pi+\ln(2)$. Thus, $\ln(Z) = k\cdot\ln(2)+Y+kj\pi = \mathrm{Re}Z+\mathrm{Im}Z$, where

$$\mathrm{Re}Z = k\cdot\ln(2)+\mathrm{Re}Y,\ \ \mathrm{Im}Z = kj\pi+\mathrm{Im}Y\cdot$$

5. Definition of a principal value of the natural logarithm. In it the imaginary part is in limits $-\pi \le \mathrm{Im}Z \le \pi$. Therefore the imaginary part of a principal value is defined under the formula:

$$g = \mathrm{int}\left[\frac{\mathrm{Im}Z}{2\pi}\right],\ \left(\mathrm{Im}Z\right)_{main} = \left\{\begin{matrix} g \text{ if } g \le \pi, \\ g-2\pi \text{ if } g > \pi. \end{matrix}\right\}.$$

6. Normalization of result.

5.4. Definition of the natural logarithm of positive real number

The algorithm of definition of the natural logarithm of a positive real number will consist in the following:

1. The number $Z=(-2)^k y$, where *y>0* - a mantissa, *k* – exponent, is given.
2. Decomposition of number *y>0* under the formula $y=\prod\left(1+\rho^{-h}\right)^{a_h}$ - "*DecompBinomReal*".
3. Composition under the formula $w=\sum a_h \ln\left(1+\rho^h\right)$ - ("*CompLogar*").
4. Calculation of $\ln(Z)=[k\cdot\ln(-2)+w]$. Here,
   - if k – even, then $\ln(Z)=k\cdot\ln(2)+w$,
   - if k – odd, then $\ln(Z)=\pi+k\cdot\ln(2)+w$.
5. Normalization of result.

5.5. Definition of the natural logarithm of negative real number

The algorithm of definition of the natural logarithm of a negative real number will consist in the following:

1. The number $Z=(-2)^k y$, where *y<0* - mantissa, *k* – exponent, is given.
2. Decomposition of number *(-y)>0* under the formula $y=\prod\left(1+\rho^{-h}\right)^{a_h}$ - ("*DecompBinomReal*").
3. Composition under the formula $w=\sum a_h \ln\left(1+\rho^h\right)$ - ("*CompLogar*").
4. Calculation of $\ln(Z)=[k\cdot\ln(-2)+w+\ln(-1)]$. Here,
   - if k – even, then $\ln(Z)=\pi+k\cdot\ln(2)+w$,
   - if k – odd, then $\ln(Z)=k\cdot\ln(2)+w$.
5. Normalization of result.

5.6. Definition of the natural logarithm of real number

Calculation in this case begins with the analysis of a sign taking the logarithm numbers and the reference to taking the logarithm positive or a negative number.

6. Potentiation

6.1. Potentiation of complex number - *Potentiating*

Potentiation will consist in definition of number $Z = e^X$ in a complex degree $X = \left(-2\right)^p m$, where *m* - mantissa, *p* - exponent. The algorithm of potentiation is based on the following: if $X = \sum a_h \ln\left(1+\rho^h\right)$then $e^X = \prod \left(1+\rho^{-h}\right)^{a_h}$. We shall notice, that

$$e^{\beta \ln(-2)} = (-2)^\beta,\ \ e^{\beta \ln(2)} = (2)^\beta,$$

$$\ln\left(-2\right) = \ln(-1) + \ln(2) = j\pi + \ln(2).$$

Let's transform the given number *X*:

$$X = \operatorname{Re} X + j \operatorname{Im} X = 2x' + \beta \ln(-2) + j \operatorname{Im} X =$$

$$= 2x' + \beta \ln(-2) + j(2x'' + \mu \frac{\pi}{2}) =$$

$$= 2x' + \beta \ln(2) + j\beta\pi + j(2x'' + \mu \frac{\pi}{2}) =$$

$$= \beta \ln(2) + j\frac{\pi}{2}(2\beta + \mu) + 2\left(x' + jx''\right).$$

Thus,

$$\operatorname{Re} X = \beta \ln(2) + 2x',\ \ j \operatorname{Im} X = j\left(\frac{\pi}{2}(2\beta + \mu) + 2x''\right).$$

In these parities

β - the whole part from quotient $X' = \dfrac{\operatorname{Re} X}{\ln(2)}$,

μ - the whole part from quotient $X'' = \dfrac{\operatorname{Im} X}{\pi/2}$.

$2x'$ - the fractional part from quotient $X' = \dfrac{\operatorname{Re} X}{\ln(2)}$,

$2x''$ - the fractional part from quotient $X'' = \dfrac{\operatorname{Im} X}{\pi/2}$.

From here follows, that

$$\exp(X)=\exp\left[\left(\beta\ln(2)+2x'\right)+j\left(\frac{\pi}{2}(2\beta+\mu)+2x''\right)\right]=$$

$$=(-2)^{\beta}e^{2\left(x'+jx''\right)}e^{j\frac{\pi}{2}(2\beta+\mu)}=(-2)^{\beta}e^{2y}e^{j\frac{\pi}{2}\gamma}\beta_0=$$

$$=(-2)^{\beta}e^{2y}\theta,$$

where $y=\left(x'+jx''\right)$, $\gamma=\left(2\beta+\mu\right)$, $\vartheta=e^{j\frac{\pi}{2}\gamma}\beta_0$,

$$\beta_0=\left\{\begin{array}{l}1 \text{ if } \beta-\text{even}\\ -1 \text{ if } \beta-\text{odd}\end{array}\right\}.$$

Here $\vartheta=\left\{\begin{array}{l} j \text{ if } \eta\cdot\beta_0=1\\ 1 \text{ if } \eta\cdot\beta=0\\ -j \text{ if } \eta\cdot\beta=-1\\ -1 \text{ if } \eta\cdot\beta=-2\end{array}\right\}$, where $\eta=\{-2,-1,0,1\}$ - the rest about divisions of an integer μ on 4.

The algorithm of potentiation will consist in the following:

1. The number $X=(-2)^{p}m$, where *m* - mantissa, *p* - exponent, is given.
2. If $X=0$, then $Z=1$.
3. Calculation of $X'=\dfrac{\operatorname{Re}X}{\ln(2)}$, $X''=\dfrac{\operatorname{Im}X}{\pi/2}$.
4. Allocation from numbers X', X'' the whole β, μ and fractional x', x'' parts accordingly.

5. Calculation of fractional parts $y' = x'\frac{\ln(2)}{2}$, $y'' = x''\frac{\pi}{4}$ and formation of a code of number $y = \left(y' + jy''\right)$, which does not contain exponent.
6. Decomposition of number $y = \left(y' + jy''\right)$ under the formula $y = \sum a_h \ln\left(1 + \rho^h\right)$- "*DecompLogar*".
7. Composition under the formula $W = \prod \left(1 + \rho^{-h}\right)^{a_h}$ - "*CompBinom*".
8. Calculation of value θ under the above-stated formula depending on younger categories of codes of numbers μ, β.
9. Definition of $\exp(X) = (-2)^{\beta} \cdot \theta \cdot W^2$
10. Normalization of result.

6.2. Potentiation of real number - *PotentiatingReal*

Potentiation will consist in definition of number $Z = e^X$ in a real degree $X = \left(-2\right)^p m$, where *m* - mantissa, *p* - exponent. By analogy with previous we shall notice, that

$$e^{\beta \ln(-2)} = (-2)^{\beta},\;\; e^{\beta \ln(2)} = (2)^{\beta},$$
$$\ln\left(-2\right) = \ln(-1) + \ln(2) = j\pi + \ln(2).$$

Let's transform the given number *X*:

$$X = x' + \beta \ln(-2) = x' + \beta \ln(2) + j\beta\pi.$$

In these parities

β - the whole part from quotient $X' = \frac{X}{\ln(2)}$,

y - the fractional part from quotient $X' = \frac{X}{\ln(2)}$.

From here follows, that

$$\exp(X) = \exp\left[\left(\beta \ln(2) + 2x'\right) + j\pi\beta\right] =$$
$$= 2^{\beta} e^{2x'} e^{j\pi\beta} = (-2)^{\beta} e^{2x'} e^{j\pi\beta} \beta_0 = (-2)^{\beta} e^{2x'} \theta,$$

where $\vartheta = e^{j\pi\beta}\beta_0$, $\beta_0 = \begin{cases} 1 \text{ if } \beta - \text{even} \\ -1 \text{ if } \beta - \text{odd} \end{cases}$. Obviously, $\vartheta = 1$.

Hence, $\exp(X) = (-2)^{\beta} e^{y}$.

The algorithm of potentiation of real number will consist in the following:

1. The number $X = (-2)^p m$, where *m* - mantissa, *p* - exponent, is given.
2. If $X = 0$, then $Z = 1$.
3. Calculation $X' = \dfrac{X}{\ln(2)}$.
4. Allocation from number X' the whole β and fractional *y* parts.
5. Decomposition of number y under the formula $y = \sum a_h \ln\left(1 + \rho^h\right)$- "*DecompLogarReal*".
6. Composition under the formula $W = \prod\left(1 + \rho^{-h}\right)^{a_h}$ - "*CompBinom*".
7. Definition $\exp(X) = (-2)^{\beta} \cdot W$
8. Normalization of result.

7. Operations with logarithmic forms

7.1. Logarithmic form representation

The logarithmic form is submitted on fig. 7 and used for representation of complex number as $Z = 2^{\omega} m \cdot e^{j\varphi}$, where ω - an integer, *m* – fractional positive number, φ - a real number, a principal value of argument which to be in limits $(-\pi) \le \varphi \le \pi$. At the same time,

- ω is written to the exponent,
- $\log_2(m)$is written to the real part of the mantissa,

- number $j\phi = \frac{j\varphi}{16}$ is written to the imaginary part of the mantissa, while $\frac{-\pi}{16} \le \phi \le \frac{\pi}{16}$.

63	62	....	54	53	52	....	2	1	0
Exponent ω				Logarithm of mantissa $\log_2(M)$					

Fig. 7.

7.2. Formations of the logarithmic form - *FormLogar*

1. The number $Z = (-2)^k \cdot M$ is given.
2. It will be transformed to a kind $$Z = 2^{\omega} z, \text{ where } z = \frac{M}{d}, \begin{cases} \omega = k, \; d = 1 \text{ if } k - \text{even} \\ \omega = k + 1, \; d = -2 \text{ if } k - \text{odd} \end{cases}$$
3. Taking the logarithm mantissas Z - calculation the natural logarithm $\ln(z) = \ln(|z|) + j\varphi$; in the process, *the principal value of argument* of the mantissa logarithm, which has to fall within the range $(-\pi) \le \varphi \le \pi$, *is written* to the imaginary part
4. Calculation $m = \log_2(|z|) = \ln(|z|) \cdot \frac{1}{\ln(2)}$
5. Writing the number ω to the exponent.
6. Writing the number $m = \log_2(|z|)$ to the real part.
7. Writing the number $j\phi = \frac{j\varphi}{16}$ to the imaginary part.

Addressing the general algorithm of taking the logarithm, let us review the algorithm of taking the logarithm of the value Z, which involves the following:

1. Transformation of the number *M* into a form that would enable the decomposition required for taking the logarithm. As a result of the transformation, the number *S* and the complex number *y* = *Q* **z* are determined

2. Decomposition formula $y = \prod \left(1 + \rho^{-h}\right)^{ah}$ - "*DecompBinom*".

3. Composition formula $\ln(y) = w = \sum a_h \ln\left(1 + \rho^h\right)$- "*CompLogar*".

4. Calculation $\ln(Z) = \operatorname{Re} Z + \operatorname{Im} Z$, where

$$\operatorname{Re} Z = \operatorname{Re} w, \ \operatorname{Im} Z = jS\pi + \operatorname{Im} w \cdot$$

5. Determination of the principal value of the natural logarithm. Its imaginary part falls within the range $-\pi \le \operatorname{Im} Z \le \pi$. Therefore, the imaginary part of the principal value is determined by the formula:

$$g = \operatorname{int}\left[\frac{\operatorname{Im} Z}{2\pi}\right], \ (\operatorname{Im} Z)_{main} = \left\{\begin{matrix} g \ \text{if} \ g \le \pi, \\ g - 2\pi \ \text{if} \ g > \pi. \end{matrix}\right\}.$$

7.3. Return from the logarithmic form – *RetLogar*

1. The given number is $V = \omega + \log(w) + j\phi$.
2. Allocation of number ω from exponent.
3. Allocation of number $m = \log_2(w)$ from real part.
4. Allocation of number $j\phi = \dfrac{j\varphi}{16}$ from imaginary part.
5. Calculation $Z = q(-2)^{\omega} M$, где
$$M = \exp[\ln(2) \cdot m + 16j\phi],$$
$$q = \left\{\begin{matrix} 1, \ \text{if} \ \omega - \text{even} \\ -1, \ \text{if} \ \omega - \text{odd} \end{matrix}\right\}$$

Addressing to the general algorithm exponentiation, we shall consider algorithm exponentiation number *X*, which will consist in the following:

1. If $X = 0$, then $M = 1$.

3. Calculation $X' = \dfrac{\operatorname{Re} X}{\ln(2)} = m, \ X'' = \dfrac{\operatorname{Im} X}{\pi/2} = \dfrac{32\phi}{\pi}$.

4. Allocation from among X', X'' the whole $\beta = 0$, μ and fractional $x' = m$, x'' parts accordingly. Allocation of the whole and fractional parts is carried out in such a manner, that fractional parts of number $y = (x' + jx'')$ are in the following limits:

$$-\frac{2\ln(2)}{3} \le x' \le \frac{\ln(2)}{3},\ -\frac{\pi}{3} \le x'' \le \frac{\pi}{6}.$$

5. Decomposition of the number $y = (x' + jx'')$ under the formula $y = \sum a_h \ln\left(1+\rho^h\right)$- "*DecompLogar*".

6. Composition by the formula $W = \prod\left(1+\rho^{-h}\right)^{a_h}$ - "*CompBinom*".

7. Definition $\exp(X) = \theta \cdot W$. Here $\vartheta = \left\{\begin{matrix} j \text{ if } \eta = 1 \\ 1 \text{ if } \eta = 0 \\ -j \text{ if } \eta = -1 \\ -1 \text{ if } \eta = -2 \end{matrix}\right\}$, where η is the remainder from dividing an integer μ by 4.

7.4. Algebraic addition of logarithmic forms

Multiplication and division of the numbers, submitted in the logarithmic form, is equivalent to algebraic addition of these forms. Really, if $Z' = 2^{\omega'} m' \cdot e^{j\varphi'}$ and $Z'' = 2^{\omega''} m'' \cdot e^{j\varphi''}$, then $Z'Z'' = 2^{\omega' + \omega''} m'm'' \cdot e^{j(\varphi' + \varphi'')}$. In this group the following operations are stipulated:

- Addition - *AddLog*
- Subtraction - *SubLog*
- Inversion - *InvLog*

Algebraic addition of logarithmic forms is carried out under the following scheme:

- Algebraic addition of exponents

- Algebraic addition of mantissas

7.5. Multiplication of the logarithmic form by an integer - *PowerLogar*

At a positive integer this operation is equivalent to erection in the whole degree. At a negative integer this operation is equivalent to division «1» on a root of the whole degree. Operation is carried out under the following scheme:

- Multiplication of exponent of the logarithmic form on an integer
- Multiplication of mantissa of the logarithmic form to an integer

In this group the following operations are stipulated separately:

- Squaring - *QuadrLogar*
- Division «1» on a square root - *MinusQuadrLogar*

7.6. Overflow

At algebraic addition of logarithmic forms and multiplication of the logarithmic form to an integer can arise overflow.

- At overflow of the logarithm $m = \log_2(|z|)$ on the real integer s we have: $\log_2(|z|) = [s + \log_2(|z'|)]$ and $\omega' = [\omega + s]$, that is at occurrence of carry S from the real part of a mantissa this carry develops with exponent.
- At occurrence of carry S from an imaginary part of a mantissa from it the number is subtracted $s\pi/8$.

8. Extraction of a square root

8.1. Extraction of a square root from complex number

Extraction of a square root it is based on the following: if $X = \prod\left(1+\rho^{-h}\right)^{2 \cdot a_h}$, then $\sqrt{X} = \prod\left(1+\rho^{-h}\right)^{a_h}$. More, this operation is considered a [5]. The algorithm will consist in the following:

1. The number $Z = (-2)^k \cdot M$, where mantissa, *k* – exponent, is given.

2. Reduction of the given number a kind $Z = (-2)^{2m} \cdot a \cdot M$. Here,

   if *k* - even, then *2m=k, a=1;*
   if *k* - odd, then *2m=k-1, a=-2.*

3. Decomposition under the formula $X = \prod \left(1 + \rho^{-h}\right)^{2 \cdot a_h}$ - "*DecompBinom2*".

4. Composition under the formula $W = \prod \left(1 + \rho^{-h}\right)^{a_h}$ - "*CompBinom*".

5. Formation of result $\sqrt{Z} = \left(-2\right)^m Y$.

6. Normalization of result.

8.2. Extraction of a square root from conjugation number

The algorithm of extraction of a square root from the conjugation complex number differs from algorithm of extraction of a square root from complex number only that in item 5 instead of a composition "*CompBinom*" the composition "*CompBinomConjug*" is used:

5. Composition under the formula $W = \prod \left(1 + \tilde{\rho}^{-h}\right)^{a_h}$ - "*CompBinomConjug*".

8.3. Extraction of a root from a positive real number

The algorithm of extraction of a square root will consist of a positive real number in the following:

1. The number $Z = (-2)^k \cdot M$, where mantissa, *k* – exponent, is given.

2. Decomposition under the formula $X = \prod \left(1 + \rho^{-h}\right)^{2 \cdot a_h}$ - "*DecompBinom2real*".

3. Composition under the formula $W = \prod \left(1 + \rho^{-h}\right)^{a_h}$ - "*CompBinom*".

4. Formation of result $\sqrt{Z} = \left(-2\right)^m Y$.

5. Normalization of result.

9. Polar coordinates

Below algorithms of calculation of polar coordinates, and also some auxiliary algorithms are described.

9.1. Calculation of the module of complex number

The algorithm of calculation of the module of complex number differs from algorithm of extraction of a square root from complex number only that in item 5 instead of a composition "*CompBinom*" the composition "*CompBinomModul*": is used:

5. Composition under the formula $W=\prod\left[\left(1+\rho^{-h}\right)\left(1+\tilde{\rho}^{-h}\right)\right]^{a_h}$ - "*CompBinomModul*".

It follows from the formula $\left|Z\right|=\sqrt{Z\cdot\tilde{Z}}$

9.2. Calculation of argument of complex number (a variant 1)

The algorithm of calculation of argument of complex number differs from algorithm taking the logarithm by a variant 1 only that in item 4 instead of a composition "*CompLogar*" the composition "*CompLogarAngle*" is used:

3. Composition under the formula $W=-j\cdot\mathrm{Im}\sum a_h\left[\ln\left(1+\rho^{-h}\right)\right]$ - "*CompLogarAngle*".

It follows from the formula $\arg(Z)=-j\cdot\mathrm{Im}\left(\ln(\mathrm{Z})\right)$.

9.3. Calculation of argument of complex number (a variant 2) - *AngleSqr*

Calculation of argument of complex number is based on the formula $\arg(Z)=-j\cdot\mathrm{Im}\left(2\cdot\ln\left(\sqrt{\mathrm{Z}}\right)\right)$ The algorithm will consist in the following:

1. The number $Z=(-2)^k\cdot M$, where mantissa, *k* – exponent, is given.
2. Reduction of the given number a kind $Z=(-2)^{2m}\cdot a\cdot M$. Here,

   if *k* - even, then *2m=k, a=1;*

   if *k* - odd, then *2m=k+1, a=-1/2.*
3. Transformation $X=f_1\left(a\cdot M\right)$, where *X* is in the first semi-quadrant - *see operation «Segmentation before extraction of a root».*
4. Decomposition under the formula $X=\prod\left(1+\rho^{-h}\right)^{2\cdot a_h}$ - ("*DecompBinom2*").

5. Composition under the formula $W = -2j \cdot \mathrm{Im} \sum a_h \left[\ln\left(1 + \rho^{-h}\right) \right]$ - “*CompLogarAngle*”.

6. Transformation $Y = f_3\left(W_a\right)$ - *see operation «Segmentation for taking the logarithm»*. Here $Y = \mathrm{Im}(\ln(M))$.

7. Calculation of $\arg(Z) = \mathrm{Im}(\ln(Z)) = \mathrm{Im}[2m \cdot \ln(-2) + \ln(M)]$. Here $\ln(-2) = \ln(-1) + \ln(2) = j\pi + \ln(2)$. So $\arg(Z) = Y + 2mj\pi$. Composed, multiple 2π, it is possible to reject. Therefore $\arg(Z) = Y$.

8. Normalization of argument $\arg(Z) = Y$.

9.4. Calculation of polar coordinates - CartesianToPolar

Transformation of rectangular coordinates in polar is equivalent to calculation of argument and the module of complex number. The corresponding algorithm will consist in the following:

1. The number $Z = (-2)^k \cdot M$, where mantissa, *k* – exponent, is given.

2. Reduction of the given number a kind $Z = (-2)^{2m} \cdot a \cdot M$. Here,

   if *k* - even, then *2m=k+2, a=1/4;*
   if *k* - odd, then *2m=k+3, a=-1/8.*

3. Transformation $X = f_1(a \cdot M)$, where *X* is in the first semi-quadrant - *see operation «Segmentation <u>before</u> extraction of a root».*

4. Decomposition under the formula $X = \prod \left(1 + \rho^{-h}\right)^{2 \cdot a_h}$ - “*DecompBinom2*”.

5. Composition under the formula $W_a = -2j \cdot \mathrm{Im} \sum a_h \left[\ln\left(1 + \rho^{-h}\right) \right]$ - “*CompLogarAngle*”.

6. Composition under the formula $W_m = \prod \left[\left(1 + \rho^{-h}\right)\left(1 + \tilde{\rho}^{-h}\right) \right]^{a_h}$ - “*CompBinomModul*”.

7. Transformation $V = f_2\left(W_m\right)$ – *see operation «Segmentation <u>after</u> extraction of a root».*

8. Formation of result as $|Z| = 2^{m+3} V$ - see item 4.

9. Transformation $Y = f_3(W_a)$ – *see operation «Segmentation for taking the logarithm»*. Here $Y = \mathrm{Im}(\ln(M))$.

10. Calculation of $\arg(Z) = \mathrm{Im}(\ln(Z)) = \mathrm{Im}[k \cdot \ln(-2) + \ln(M)]$. So $\ln(-2) = \ln(-1) + \ln(2) = j\pi + \ln(2)$. Therefore, $\arg(Z) = Y + kj\pi$.

11. Definition of a principal value of argument. In it the imaginary part is in limits $-\pi \le \mathrm{Im}Z \le \pi$. Therefore the principal value of argument is defined under the formula:

$$g = \mathrm{int}\left[\frac{\arg Z}{2\pi}\right],\ (\arg Z)_{main} = \begin{cases} g \text{ if } g \le \pi, \\ g - 2\pi \text{ if } g > \pi. \end{cases}.$$

9.5. Return from polar coordinates - *PolarToCartesian*

Transformation of polar coordinates in rectangular will consist in calculation under the formula $X = |X| \cdot e^{j\varphi}$, where $|X|$, φ - polar coordinates, real numbers.

The algorithm will consist in the following:

1. The number $X = |X| \cdot e^{j\varphi}$, submitted by a mantissa *m* and exponent *p* is given. Thus $|X| = (-2)^p \cdot \mathrm{Re}(m),\ \mathrm{Im}(m) = j\varphi/16$.
2. If $|X| = 0$, then $Z = 1$.
3. Formation of number $j\varphi = (-2)^4 \cdot \mathrm{Im}(m)$.
4. Calculating of $e^{j\varphi}$ (operation *Ort*).
5. Multiplication $Z = |X| \cdot e^{j\varphi}$.

9.6. Calculation of a sine and cosine a real number – *Ort*

This problem will consist in definition of number $Z = e^{jX}$, where a real number $X = (-2)^p m$, *m* - mantissa, *p* - exponent. Number $Z = \mathrm{Cos}X + j\mathrm{Sin}X$ received in result. This calculation in many respects is similar to potentiation.

Let's transform the given number: $jX = j\left(\frac{\pi}{2}\mu + 2x''\right)$.

In these parities

μ - the whole part from quotient $X'' = \frac{X}{\pi/2}$.

$2x''$ - the fractional part from quotient $X'' = \frac{X}{\pi/2}$.

From here follows, that

$$\exp(jX) = \exp\left[j\left(\frac{\pi}{2}\mu + 2x''\right)\right] = e^{2(jx'')}e^{j\frac{\pi}{2}\mu} = e^{2y}\theta,$$

where $y = (jx'')$, $\vartheta = e^{j\frac{\pi}{2}\mu}$.

Here $\vartheta = \left\{\begin{array}{l} j \text{ if } \eta = 1 \\ 1 \text{ if } \eta = 0 \\ -j \text{ if } \mu = -1 \\ -1 \text{ if } \mu = -2 \end{array}\right\}$, where $\eta = \{-2,-1,0,1\}$ - the rest about divisions of an integer μ on 4.

The algorithm will consist in the following*:*

1. The number $X = (-2)^p m$, where *m* - mantissa, *p* - exponent, is given.
2. If $X = 0$, then $Z = 1$.
3. Calculation of $X'' = \frac{X}{\pi/2}$.
4. Allocation from number X'' the whole μ and fractional x'' parts accordingly.
5. Calculation of fractional parts $y = x''\frac{\pi}{4}$ and formation of a code of number y, which does not contain exponent.
6. Decomposition of number y under the formula $y = \sum a_h \ln\left(1 + \rho^h\right)$- “*DecompLogar*”.

7. Composition under the formula $W = \prod \left(1 + \rho^{-h}\right)^{ah}$ - “*CompBinom*”.

8. Calculation of value θ under the above-stated formula depending on younger categories of codes of number μ.

9. Definition of $\exp(X) = \theta \cdot W^2$

10. Normalization of result.

9.7. Definition semi-quadrant - *Semiquadrant*

In this operation number of semi-quadrant, in which there is a complex number - see fig. 8 is defined. Definition carries out «Block of definition of semi-quadrant».

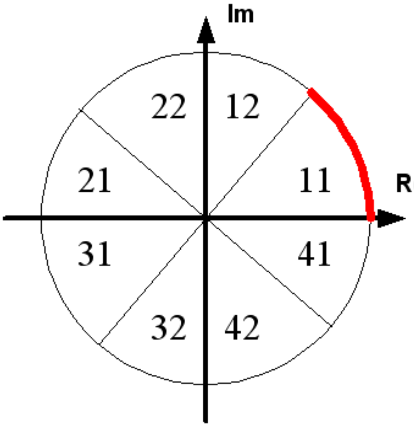

Fig. 8.

9.8. Segmentation before extraction of a root - *SemiquadrantTransOne*

In this operation such transformation complex numbers $X = |X|e^{j\varphi}$, at which it moves in semi-quadrant 11 (see table 7), is carried out. Further it is transformations we shall designate as $f_1(X)$. At this transformation the number X will be transformed to number $X' = |X|e^{j\vartheta}$ with argument $45^{\square} \geq \vartheta \geq 0$.

Table 7.

Site of number X		Re*X*	Im*X*	Parity between |Re| and |Im|	$f_1(X)$
Quad-rant	Semi-quad-rant				
1	11	Re*X>0*	Im*X>0*	$\lvert\mathrm{Re}\rvert > \lvert\mathrm{Im}\rvert$	Without changes
	12			$\lvert\mathrm{Re}\rvert < \lvert\mathrm{Im}\rvert$	Mirror display concerning a bisector of the *first* quadrant
2	21	Re*X<0*	Im*X>0*	$\lvert\mathrm{Re}\rvert > \lvert\mathrm{Im}\rvert$	Mirror display concerning an axis of ordinates
	22			$\lvert\mathrm{Re}\rvert < \lvert\mathrm{Im}\rvert$	Turn on (-90)
3	31	Re*X<0*	Im*X<0*	$\lvert\mathrm{Re}\rvert > \lvert\mathrm{Im}\rvert$	Turn on (-180)
	32			$\lvert\mathrm{Re}\rvert < \lvert\mathrm{Im}\rvert$	Mirror display concerning a bisector of the *fourth* quadrant
4	41	Re*X>0*	Im*X>0*	$\lvert\mathrm{Re}\rvert > \lvert\mathrm{Im}\rvert$	Mirror display concerning an axis of ordinates
	42			$\lvert\mathrm{Re}\rvert < \lvert\mathrm{Im}\rvert$	Turn on (+90)

9.9. Segmentation after extraction of a root - *SemiquadrantTransTwo*

In this operation transformation complex numbers $\sqrt{X'} = \sqrt{|X|} \cdot e^{j\vartheta/2}$ is carried out, where $X' = f_1(X)$. At this transformation the number $\sqrt{X'}$ moves to that semi-quadrant, where its argument equal $\frac{\varphi}{2}$, i.e. the number accepts value $\sqrt{X} = \sqrt{|X|} \cdot e^{j\varphi/2}$ - see table 8. Further it is transformations we shall designate as $f_2(X)$.

Table 8. Functions of transformation $f_1 \to f_2$.

Site of number X		ReX	ImX	Parity between $\lvert\mathrm{Re}\rvert$ and $\lvert\mathrm{Im}\rvert$	$f_2(X)$
Quadrant	Semiquadrant				
1	11	ReX>*0*	ImX>*0*	$\lvert\mathrm{Re}\rvert > \lvert\mathrm{Im}\rvert$	Without changes
	12			$\lvert\mathrm{Re}\rvert < \lvert\mathrm{Im}\rvert$	Turn on (+45) and mirror display concerning a bisector of the *first* quadrant
2	21	ReX<*0*	ImX>*0*	$\lvert\mathrm{Re}\rvert > \lvert\mathrm{Im}\rvert$	Mirror display concerning a bisector of the *first* quadrant
	22			$\lvert\mathrm{Re}\rvert < \lvert\mathrm{Im}\rvert$	Turn on (+45)
3	31	ReX<*0*	ImX<*0*	$\lvert\mathrm{Re}\rvert > \lvert\mathrm{Im}\rvert$	Turn on (-90)
	32			$\lvert\mathrm{Re}\rvert < \lvert\mathrm{Im}\rvert$	Turn on (+135) and mirror display concerning a bisector of the *second* quadrant
4	41	ReX>*0*	ImX>*0*	$\lvert\mathrm{Re}\rvert > \lvert\mathrm{Im}\rvert$	Mirror display concerning an axis abstsiss
	42			$\lvert\mathrm{Re}\rvert < \lvert\mathrm{Im}\rvert$	Turn on (-45)

9.10. Segmentation for taking the logarithm - *SemiquadrantTransThree*

In this operation transformation complex numbers $\ln X' = \ln\left(\sqrt{|X|}\cdot e^{j\vartheta/2}\right) = \ln\left(\sqrt{|X|}\right) + j\vartheta/2$, is carried out, where $X' = f_1(X)$. Thus the number is formed

$$\ln X = 2\cdot\ln\left(\sqrt{|X|}\cdot e^{j\vartheta/2}\right) = \ln(|X|) + j\varphi\cdot$$

Transformation is described by table 9. Further it is transformations we shall designate as $f_3(X)$.

Table 9. Functions of transformation $f_1 \rightarrow f_3$.

Site of number X		ReX	ImX	Parity between |Re| and |Im|	$f_3(X)$
Quad-rant	Semi-quad-rant				
1	11	Re*X>0*	Im*X>0*	|Re| > |Im|	$\mathrm{Re} f_3(X) = 2\,\mathrm{Re}\,X$ $\mathrm{Im} f_3(X) = 2\,\mathrm{Im}\,X$
	12			|Re| < |Im|	$\mathrm{Re} f_3(X) = 2\,\mathrm{Re}\,X$ $\mathrm{Im} f_3(X) = \frac{\pi}{2} - 2\,\mathrm{Im}\,X$
2	21	Re*X<0*	Im*X>0*	|Re| > |Im|	$\mathrm{Re} f_3(X) = 2\,\mathrm{Re}\,X$ $\mathrm{Im} f_3(X) = \pi - 2\,\mathrm{Im}\,X$
	22			|Re| < |Im|	$\mathrm{Re} f_3(X) = 2\,\mathrm{Re}\,X$ $\mathrm{Im} f_3(X) = \frac{\pi}{2} + 2\,\mathrm{Im}\,X$
3	31	Re*X<0*	Im*X<0*	|Re| > |Im|	$\mathrm{Re} f_3(X) = 2\,\mathrm{Re}\,X$ $\mathrm{Im} f_3(X) = -\pi + 2\,\mathrm{Im}\,X$
	32			|Re| < |Im|	$\mathrm{Re} f_3(X) = 2\,\mathrm{Re}\,X$ $\mathrm{Im} f_3(X) = -\frac{\pi}{2} - 2\,\mathrm{Im}\,X$
4	41	Re*X>0*	Im*X>0*	|Re| > |Im|	$\mathrm{Re} f_3(X) = 2\,\mathrm{Re}\,X$ $\mathrm{Im} f_3(X) = -2\,\mathrm{Im}\,X$
	42			|Re| < |Im|	$\mathrm{Re} f_3(X) = 2\,\mathrm{Re}\,X$ $\mathrm{Im} f_3(X) = -\frac{\pi}{2} + 2\,\mathrm{Im}\,X$

10. Operations with polar forms

10.1. The polar form of representation of complex number

On fig. 9 the polar form of a complex тгьиук $2^{\omega} \cdot D \cdot \exp(j\varphi)$ is submitted. This form is used for representation of complex number in polar coordinates, where $2^{\omega} \cdot D$ - module, real number, φ - real number, a principal value of argument, which to be in limits $(-\pi)\leq\varphi\leq\pi$.

63	62	....	54	53	52	....	2	1	0
Exponent ω				Logarithm of mantissa *M*					

Fig. 9.

Thus

- in an exponent enters ω,
- in a real part of a mantissa the number $\operatorname{Re}M=D$ enters,
- in an imaginary part of a mantissa the number $j\phi=-\frac{j\varphi}{8}$ enters; here $\frac{-\pi}{8}\leq\phi\leq\frac{\pi}{8}$.

10.2. Multiplication of polar forms

In this operation operands are submitted in the indicative form of a kind $2^{\omega}\cdot D\cdot\exp(j\varphi)$, i.e. in polar coordinates. Multiplication of two numbers submitted in such form will consist in the following:

- multiplication of real parts of mantissas - modules;
- addition an exponents;
- addition of imaginary parts of mantissas - arguments; at occurrence of carry S from an imaginary part of a resulting mantissa from it the number $-s\pi/4$ is subtracted.

10.3. Turn of the polar form

In this operation *first* operand are submitted in the indicative form of a kind $2^{\omega}\cdot D\cdot\exp(j\varphi)$, i.e. in polar coordinates. The *second* operand is submitted as an imaginary part of a mantissa, which has value ϑ and represents a corner of turn ψ as $j\vartheta=-\frac{j\psi}{8}$, where $\frac{-\pi}{8}\leq\vartheta\leq\frac{\pi}{8}$. Thus, the *second* operand also is submitted in the indicative form of number $\exp(j\psi)$. Turn of complex number *A* on a corner ψ will consist in the following:

- transfer of the real part of a mantissa of an operand A to the real part of a mantissa of result;
- transfer exponent of an operand A in an exponent of result;
- addition of imaginary parts of mantissas of the first and second operands; at occurrence of carry $\boldsymbol{S}$ from an imaginary part of a resulting mantissa from it the number is subtracted . $-s\pi/4$

References

1. Khmelnik S.I. **Coding of Complex Numbers and Vectors**, publ. «Mathematics in Computers», Israel, 2004, ISBN 978-0-557-74692-7, http://mic34.com/Magazine/94846.pdf (in Russan)
2. Khmelnik S.I., Doubson I.S. **Positional codes of complex numbers and vectors,** publ. «Mathematics in Computers», Israel, Printed in USA, Lulu Inc., ID 10793760, 2011.
3. Baikov V.D., Smolov V.B. **The specialized processors: iterative algorithms and structures**, publ. "Radio I swiaz", Moscow, 1985 (in Russian).
4. Khmelnik S.I. **Computer Arithmetic of Complex Numbers and Vectors. Theory, Hardware, Modeling**, publ. «Mathematics in Computers», Israel, Printed in USA, Lulu Inc., ID 560836, 2006., (in Russan)
5. Khmelnik S.I. **Complex Numbers Square-Rooting Unit**, publ. «Mathematics in Computers», Israel, Printed in USA, Lulu Inc., ID 2607144, Second Edition, 2011, ISBN: 978-1-4583-9202-2.