The Computation of the STAFF Cross-correlation

Draft

R. Manning, M. Belkacemi, C.C. Harvey

Contents

1 The Distribution of Cross-correlation Samples

The STAFF correlator computes aboard Cluster the Hermitian cross-spectral matrix. The off-diagonal elements are normalised to the diagonal elements, $c_{ij} = S_{ij}/\sqrt{(S_{ii}S_{jj})}$ so that, by Schwarz's inequality, $0 \leq c_{ij} \leq 1$. The square of the real $r_{ij} = [\mathcal{R}e(c_{ij})]^2$ and the square of the imaginary $i_{ij} = [\mathcal{I}m(c_{ij})]^2$ parts of c_{ij} are telemetered to the ground using three bits for each, the remaining two bits of the eight-bit word being used for the signs of $\mathcal{R}e(c_{ij})$ and of $\mathcal{I}m(c_{ij})$. As the magnitudes of r_{ij} and of i_{ij} are carried by only three bits, each can take one of only $2^3 = 8$ values. The problem is to optimise the distribution (between the limits of 0 and 1) of the ranges corresponding these eight allowable values. Two possible distributions have been proposed: (approximately) logarithmic, and linear.

The choice of distribution must be driven by the scientific objectives, which are:
1) determination of the magnitude of the correlation, and
2) determination of the phase of the correlation for all magnitudes.

In each quadrant of the complex c_{ij} plane there are $8 \times 8 = 64$ bins for the complex correlation coefficient. Figures 1 and 2 show respectively the first quadrant of the complex C_{ij} plane for two different distributions of sampling bins. A number is printed at the position of the outer corner of each of the 64 bins, its value being the phase angle corresponding to that corner. Note that the magnitude of the correlation coefficient is the radial distance from the origin of coordinates.

Fig. 1 is for a logarithmic distribution of digitisation ranges, as originally proposed and currently implemented, except for one small modification. The smallest of the inter-range separation limits has been lowered, to modify the relative sizes of the lowest two ranges. The effect is to reduce the size of the bins next to the real axis and increase the size of the bins on the row above, *i.e.*, to make the two rows more nearly equal (Fig. 1); and the same thing along the imaginary axis.

The approximation, discussed in sections 4 and 5, used to calculate the r_{ij} and i_{ij} is such that it is equally easy to calculate real and imaginary parts of c_{ij} itself (*i.e.*, square roots can be taken). It has been proposed to use a linear distribution of ranges for sampling $\mathcal{R}e(c_{ij})$ and $\mathcal{I}m(c_{ij})$; Fig. 2 shows the distribution of bins in the first quandrant of the complex c_{ij} plane for this distribution.

Note that a logarithmic distribution of can be applied to either r_{ij} or c_{ij}. The base of the logarithm used to draw Fig. 1 has been optimised; it corresponds to $\log_2$ for r_{ij} or $\log_{\sqrt{2}}$ for c_{ij}.

These figures were drawn using the programme on page 10, which produced a LaTeXsource file. This was subsequently used to produce the plots. Note that the scale used to draw the first figure is exactly logarithmic except for p=7; the scale used for the Cluster hardware will be the same, according to the digital algorithms described in sections 4 and 5 are used.

Note that, for a given magnitude of the correlation coefficient, the complex correlation coefficient must lie on a quadrant of a circle centered on the origin and with radius equal to the magnitude of the correlation coefficient.

It can be seen that:

- The logarithmic distribution gives a phase resolution which has a much better distribution in the complex c_{ij} plane, giving phase resolution for small correlations; although less good than for high correlations, it is physically acceptable. Only one pair of values for the real and imaginary parts of c_{ij} can occurs (*i.e.*, is totally wasted).

- The corners of all the "quantisation cells" (except where modified close to the real and imaginary axes) lie on lines of constant phase given by $\tan^{-1}(2^{i/2})$ where i is an integer in the range $-7 \leq i \leq +7$. This is why the phase resolution is conserved for different magnitudes of correlation.

- The linear distribution of values of the real and imaginary parts of c_{ij} gives no phase information at all for correlations less than 10%, and only comparable (but more uniform) phase resolution for high (~ 1) corelation magnitudes.

- The reason for this loss is information is that 8 bins (*i.e.*, 8 of the 64 possible pairs of values for the real and imaginary parts of c_{ij}) can never occur because, for physical reasons, the magnitude of the correlation can never be larger than unity.

2 Scientific Conclusions

1. The logarithmic distribution is superieur to the linear distribution, because:
   - It has both magnitude and phase resolution over the entire complex c_{ij} plane;
   - In particuler, the logarithmic distribution handles the phase determination for small correlations.
2. The behaviour near the real and the imaginary axes (the boundary of the smallest cells on each axis) has been modified (from strictly logarithmic) to give more uniform phase resolution for small phase angles.

3 Implementation

In the processing, real numbers are represented by 8 bits of exponent with value e and 8 bits of mantissa with value m, which represent the value

$$N = 2^{(e-8)} \times (2^8 + m). \tag{1}$$

Thus, with

$$0 \leq e \leq 255 \quad \text{and} \quad 0 \leq m \leq 255,$$

we have

$$1 \leq N \leq 2^{247} \times (2^9 - 1) \simeq 2^{256} \simeq 10^{77}. \tag{2}$$

If we denote the reduced mantissa m' by

$$m' = m/2^8, \tag{3}$$

then

$$0 \leq m' \leq 1$$

and

$$N = 2^e \times (1 + m'). \tag{4}$$

The product of two numbers N_1 and N_2 is

$$N_1 \times N_2 = 2^{e_1}(1 + m_1') \times 2^{e_2}(1 + m_2') = 2^{(e_1+e_2)} \times (1 + m_1' + m_2' + m_1'm_2')$$

This is very close to the number represented by the expression $2^{(e_1+e_2)}(1 + m_1' + m_2')$ obtained by simply adding the exponents amd mantissas.

4 Numerical considerations

It would be is computationally very convenient to treat these exponent+mantissa representations as if they are true logarithms, provided that the consequent errors are acceptable. We denote the mantissa in eq. (1) by M, that is, $M = (256 + m)$ and lies in the range $256 \leq M < 511$. If the 16-bit representation of a number were a true logarithmic representation, then the eight right-most bits would vary logarithmically over the same range of values as the approximate mantissa in Eq. 1, that is, between 256 when $M = 256$ and 512 when $M = 512$. The logarithmic function satisfying these two conditions is $256 \times \log_2(M/256)$, instead of $m = M - 256$ with our exponent+mantissa representation. The error committed in treating the 16-bits of exponent+mantissa as the true logarithm is just the error introduced by representing $256 \times \log_2(M/256)$ by $M - 256$ in the range $256 \leq M < 511$, that is, by representing $\log_2 M'$ by $M' - 1$ in the range $1 \leq M' < 2$.

For any value of the "reduced" mantissa $x = N/256$ in the range $1 \leq x < 2$, the value of m' is $x - 1$. If m' is now interpreted as being the "true" logarithm (to base 2) of the mantissa, the corresponding value for the mantissa would be $2^{m'} = 2^{(x-1)}$. Therefore, the ratio of the true value of x to the value obtained by interpreting the approximation as the true logarithm is

$$f(x) = \frac{x}{2^{(x-1)}} \quad \text{for} \quad 1 \leq x < 2.$$

The function $f(x)$ take thes following values in the range $1 \leq x < 2$

```
  x  = 1.00  1.250  1.375  1.4427  1.500  1.750  2.00
f(x) = 1.00  1.051  1.060  1.0615  1.061  1.041  1.00
```

$f(x)$ is unity at the ends of the range, within which it is always greater than unity, with a maximum value of 1.06148 occurring when $x = \log_2 e = 1.4427$. Thus use of the expression exponent+mantissa as the logarithm leads to an under-estimation of the correct numerical value by an amount which is always less than 6.15%.

Of course, it would be possible to remove this error completely by representing numbers by their "true" logarithm. This would be possible by replacing the eight bits of our mantissa by the corresponding eight bits of the true logarithm, for example, by using the look-up table shown in Table 1.

5 Calculation of the r_{ij} and i_{ij}

Having decided to use the exponent+mantissa as if it is the true logarithm, the calculation of r_{ij} and i_{ij} is rather easy.

$$r_{ij} = [\mathcal{R}e(c_{ij})]^2 = [\mathcal{R}e(S_{ij})]^2/(S_{ii}S_{jj})$$
$$i_{ij} = [\mathcal{I}m(c_{ij})]^2 = [\mathcal{I}m(S_{ij})]^2/(S_{ii}S_{jj})$$

The logarithms of r_{ij} is simply

$$\log_2(r_{ij}) = 2\log_2(\mathcal{R}e(S_{ij})) - \log_2 S_{ii} - \log_2 S_{jj}$$

There is a slight complication caused by the fact that r_{ij} (and i_{ij}) are both less than unity, while numbers less than unity are not defined on our exponent+mantissa representation of eqs. (1) or (3). Therefore we compute the logarithm of the inverse,

$$\log_2(1/r_{ij}) = \log_2 S_{ii} + \log_2 S_{jj} - 2\log_2(\mathcal{R}e(S_{ij})) \tag{5}$$

and the same thing for the imaginary part,

$$\log_2(1/i_{ij}) = \log_2 S_{ii} + \log_2 S_{jj} - 2\log_2(\mathcal{I}m(S_{ij})) \tag{6}$$

Finally, it is easy to realise the distribution of samples on the logarithmic scale recommended in section 2. If the 16-bit word written

$$b_{15}b_{14}b_{13}b_{12}b_{11}b_{10}b_9b_8b_7b_6b_5b_4b_3b_2b_1b_0$$

represents the value x, then

0000000000000000	represents	$x =$	1
0000000100000000	represents	$x =$	2
0000001000000000	represents	$x =$	4
0000001100000000	represents	$x =$	8
0000010000000000	represents	$x =$	16
0000010100000000	represents	$x =$	32
0000011000000000	represents	$x =$	64
0000011100000000	represents	$x =$	128

Provided that

$$b_{15}b_{14}b_{13}b_{12}b_{11} = 00000,$$

we may take (using a mask) only the three bits $b_{10}b_9b_8$; the values of these three bits represent values of x in the ranges shown below; the last column shown the value actually sent to the telemetry.

000 →	1	$\leq x <$	2	TM = 7
001 →	2	$\leq x <$	4	TM = 6
010 →	4	$\leq x <$	8	TM = 5
011 →	8	$\leq x <$	16	TM = 4
100 →	16	$\leq x <$	32	TM = 3
101 →	32	$\leq x <$	64	TM = 2
110 and 111 →	64	$\leq x <$	256	TM = 1

The last line deviates from the geometric progression because, as was shown in section 2, it is useful to deviate from a true logarithmic scale for the last two bins, to obtain more uniform phase resolution for small phase angles. In the case

$$b_{15}b_{14}b_{13}b_{12}b_{11} \neq 00000, \quad \text{we have} \quad 256 \leq x \quad \text{and} \quad \text{TM} = 0$$

which completes the modified logarithmic scale used to draw Fig. 1.

6 Linear distribution

$1/r_{ij}$ is greater than 1, and we start by calculating two's compliment of the representation; this is equivalent to dividing x into 2^{2^8}, so that the result $y = 2^{2^8}/x$ lies in the range $1 \leq y > 10^{-77}$. In particular,

1111111111111111	represents	$y =$	nearly 1
1111111110000000	represents	$y =$	$1/\sqrt{2}$
1111111100000000	represents	$y =$	$1/2$
1111111010000000	represents	$y =$	$1/\sqrt{8}$
1111111000000000	represents	$y =$	$1/4$
1111110110000000	represents	$y =$	$1/\sqrt{32}$
1111110100000000	represents	$y =$	$1/8$
1111110010000000	represents	$y =$	$1/\sqrt{128}$
1111110000000000	represents	$y =$	$1/16$
1111101100000000	represents	$y =$	$1/32$
1111101000000000	represents	$y =$	$1/64$
1111100100000000	represents	$y =$	$1/128$
1111100000000000	represents	$y =$	$1/256$

To obtain $\mathcal{R}e(c_{ij} = \sqrt{(}r_{ij})$ it is necessary to take the square root, which is equivalent to shifting the whole 16-bit word one bit to the right. *For reasons which I have not quite understood*, by taking the bits $b_9b_8b_7$ from the above table, we have

111 →	$1/\sqrt{8}$	$\leq x <$	1	TM = 7
110 →	1/2	$\leq x <$	$1/\sqrt{8}$	TM = 6
101 →	1/4	$\leq x <$	8	TM = 5
100 →	1/8	$\leq x <$	16	TM = 4
011 →	1/16	$\leq x <$	32	TM = 3
010 →	1/32	$\leq x <$	64	TM = 2
001 →	1/64	$\leq x <$	64	TM = 2
000 →	32	$\leq x <$	64	TM = 2

7 Precision of the Results

It was shown above that the use of the exponent+mantissa as if it were the true logarithm leads to an error which is less than 6.15%. We have not worked out the distribution of the error, but we do know that it will always be in the same sense (the represented value will be too small). However, in eqs (5) and (6) we have two logarithms which are added and two (actually, the same one twice over) which are subtracted, so the error will not accumulate systematically. Therefore the maximum error (worst possible case) is $1.0615^2 = 1.127$, or 12.7%. The probable error will be smaller.

The precise estimation of the standard error needs further study, but the result is certainly compatible with our quasi-logarithmic distribution of telemetered samples, which works only to within a factor of 2.

C.C. Harvey
Meudon, le 30 avril 1993

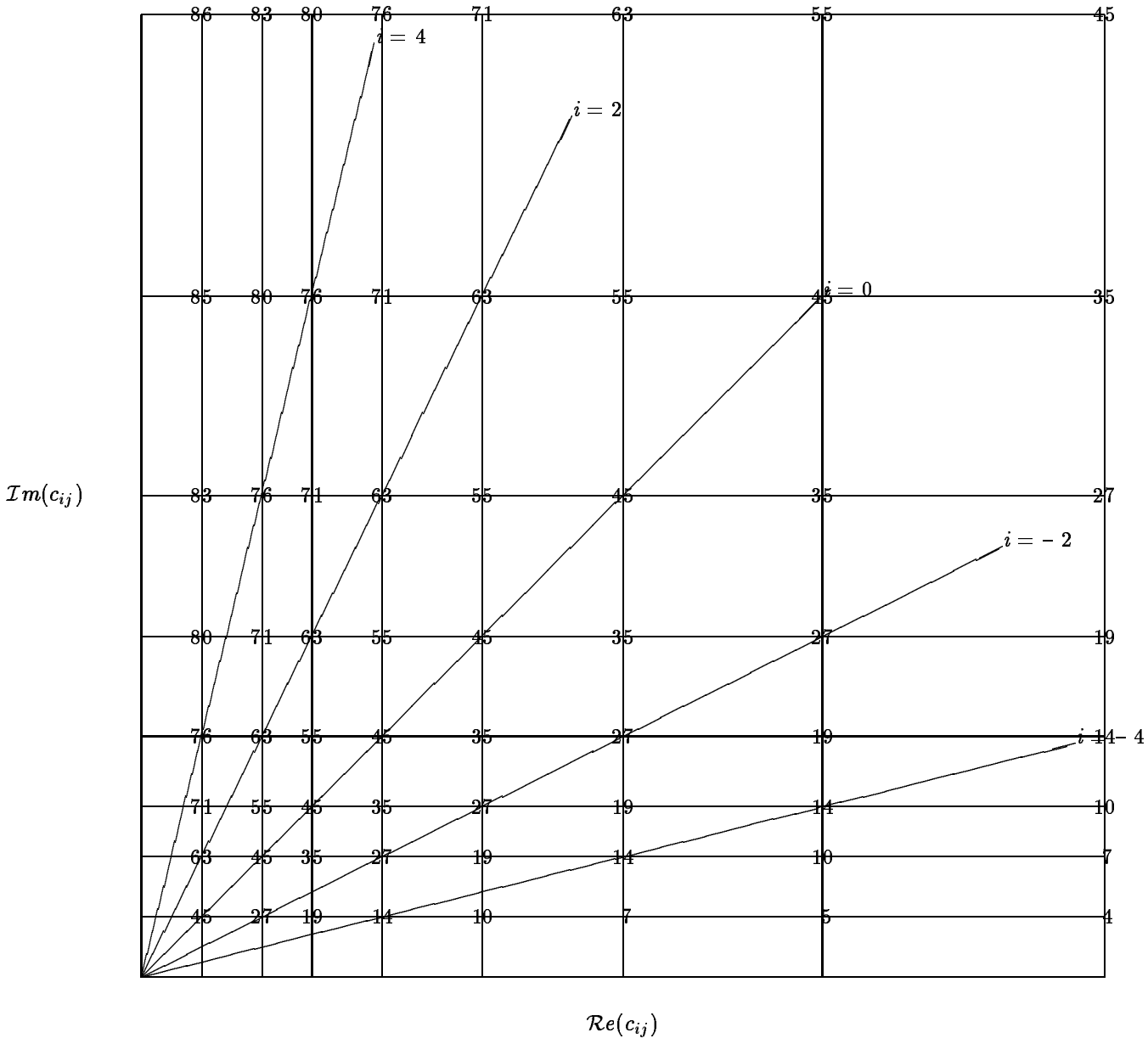

Figure 1. The distribution of cells in the complex c_{ij} plane when using logarithmic quantisation. Each printed number is the phase of the corner of the bin. The logarithmic distribution used for this following plot is

$$C = 2^{-p} \quad \text{for} \quad 0 \leq p \leq 6$$

and

$$C = 2^{-p-1} \quad \text{for} \quad p = 7$$

which is exactly logarithmic except for $p = 7$.

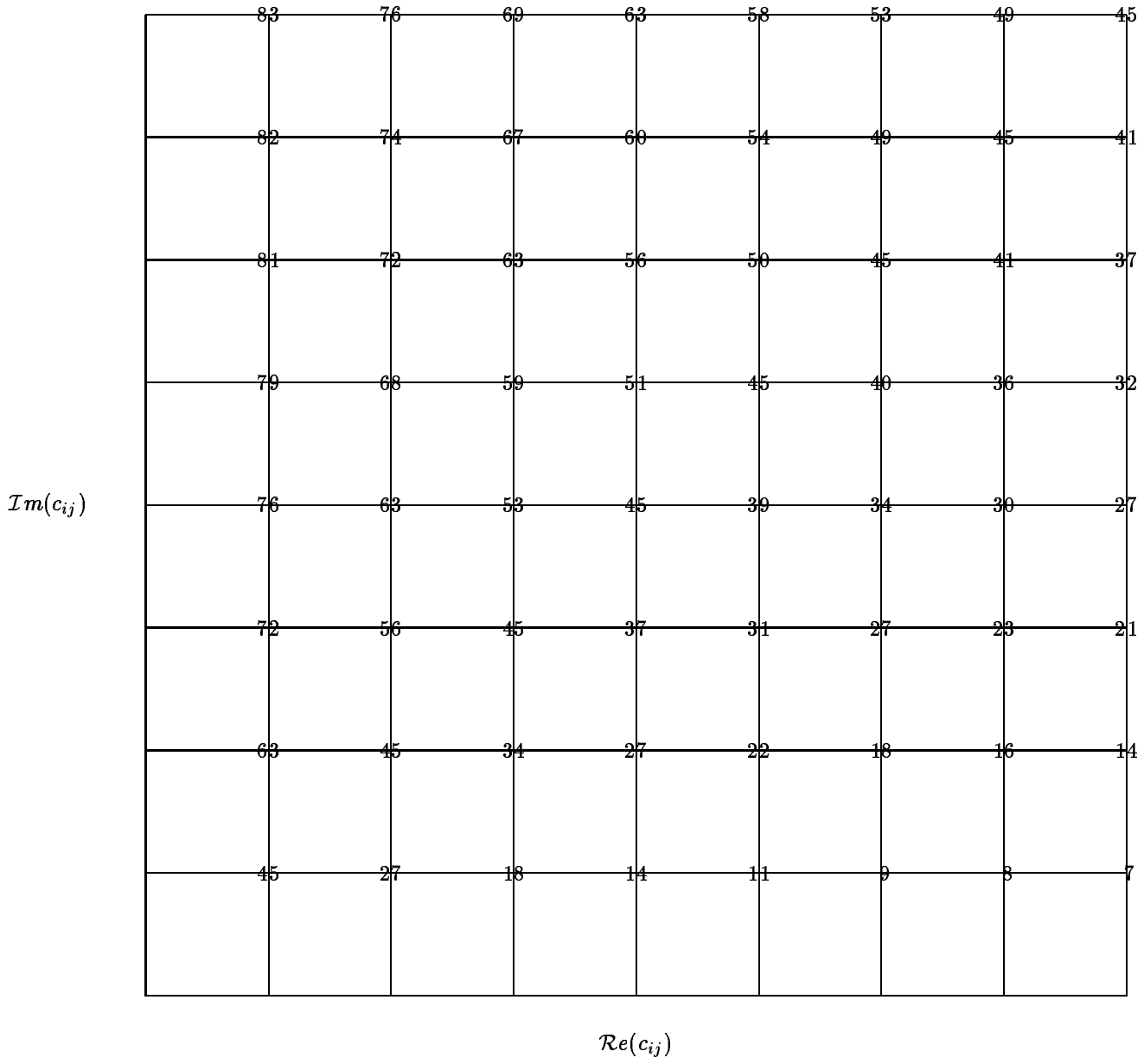

Figure 2. The distribution of cells in the complex c_{ij} plane when using linear quantisation.

0	0	32	44	64	82	96	118	128	150	160	179	192	207	224	232
1	1	33	45	65	84	97	119	129	151	161	180	193	208	225	233
2	3	34	46	66	85	98	120	130	152	162	181	194	208	226	234
3	4	35	47	67	86	99	121	131	153	163	182	195	209	227	234
4	6	36	49	68	87	100	122	132	154	164	183	196	210	228	235
5	7	37	50	69	88	101	123	133	155	165	184	197	211	229	236
6	9	38	51	70	89	102	124	134	155	166	185	198	212	230	237
7	10	39	52	71	90	103	125	135	156	167	185	199	212	231	238
8	11	40	54	72	92	104	126	136	157	168	186	200	213	232	238
9	13	41	55	73	93	105	127	137	158	169	187	201	214	233	239
10	14	42	56	74	94	106	128	138	159	170	188	202	215	234	240
11	16	43	57	75	95	107	129	139	160	171	189	203	216	235	241
12	17	44	59	76	96	108	130	140	161	172	190	204	216	236	241
13	18	45	60	77	97	109	131	141	162	173	191	205	217	237	242
14	20	46	61	78	98	110	132	142	163	174	192	206	218	238	243
15	21	47	62	79	99	111	133	143	164	175	192	207	219	239	244
16	22	48	63	80	100	112	134	144	165	176	193	208	220	240	244
17	24	49	65	81	102	113	135	145	166	177	194	209	220	241	245
18	25	50	66	82	103	114	136	146	167	178	195	210	221	242	246
19	26	51	67	83	104	115	137	147	168	179	196	211	222	243	247
20	28	52	68	84	105	116	138	148	169	180	197	212	223	244	247
21	29	53	69	85	106	117	139	149	169	181	198	213	224	245	248
22	30	54	71	86	107	118	140	150	170	182	198	214	224	246	249
23	32	55	72	87	108	119	141	151	171	183	199	215	225	247	249
24	33	56	73	88	109	120	142	152	172	184	200	216	226	248	250
25	34	57	74	89	110	121	143	153	173	185	201	217	227	249	251
26	36	58	75	90	111	122	144	154	174	186	202	218	228	250	252
27	37	59	77	91	112	123	145	155	175	187	203	219	228	251	252
28	38	60	78	92	113	124	146	156	176	188	203	220	229	252	253
29	40	61	79	93	114	125	147	157	177	189	204	221	230	253	254
30	41	62	80	94	116	126	148	158	178	190	205	222	231	254	255
31	42	63	81	95	117	127	149	159	178	191	206	223	231	255	255

Table 1. This table shows the value (right-hand column) which should be used to replace the value (left-hand column) of the 8-bit mantissa in order to obtain the true logarithm of the value to be represented.

```
proc lindist
  let x=1
  while x>0
    lprint "\put(";dec(x,4,6);",0){\line(0,1){1}}"
    lprint "\put(0,";dec(x,4,6);"){\line(1,0){1}}"
    let y=1
    while y>0
      let a=deg(atn(y/x))
      lprint "\put(";dec(x,4,6);",";dec(y,4,6);"){\makebox(0,0){";num(a,2);"}}"
      let y=y-0.125
      endwhile
    let x=x-0.125
    endwhile
  endproc
proc logdist
  let i=0
  while i<8
    if i=7: let i=8: endif
    let x=2^(-i)
    let sx=sqr(x)
    lprint "\put(";dec(sx,4,6);",0){\line(0,1){1}}"
    lprint "\put(0,";dec(sx,4,6);"){\line(1,0){1}}"
    let j=0
    while j<8
      if j=7: let j=8: endif
      let y=2^(-j)
      let sy=sqr(y)
      let a=deg(atn(sy/sx))
      lprint "\put(";dec(sx,4,6);",";dec(sy,4,6);"){\makebox(0,0){";num(a,2);"}}"
      let j=j+1
      endwhile
    let i=i+1
    endwhile
  endproc
proc logtable
  rem ----- programme to produce the "log-table"
  let ln2=ln(2)
  let k=0
  while k<32
    let j=0
    while j<8
      let i=32*j+k
      let x=256*ln(1+i/256)/ln2
      lprint num(i,4);num(x,4);"";
      let j=j+1
      endwhile
    lprint ""
    let k=k+1
    endwhile
  endproc
```