Cube Root 2.01.07

J = 018
K = 017

```
0000000101083e8,
019135f19c004,
00d135f01c016,
001135f01c017,
00080003ff35d,
000c8003ff016,
000835d3ff35e,
00083ff3ff35c,
029135e1f835c,
00083ee3ff35d,
100335c018800,
021180029835d,
031135e31235d,
025135f19c015,
000835d3ff35f,
0002800800800,
000335e800800,
000a80035f800,
0003800018800,
000880035f35d,
000a35f800800,
000f00080000e,
000835d3ff000,
0005000000000,
002c000000000,
```

CHECK WITH GMc 9-22-59

1. IDENTIFICATION

a. File number: 2.01.07

b. Title: Cube Root Routine (CUR)

c. Author: Donald E. Coon d. Accepted:

2. PURPOSE

$r = \sqrt[3]{b}$

3. LINKAGE AND STORAGE

a. Routine linkage: Standard L and L + 1 orders. The link word is:

Order #	Type	A	B	C
L + 2	[e]	[b]	[r]	[β]

b. Adaptation link word:

L + 2	[24]	[W_i]	[23]	[β]

c. Storage information: 24 storage locations are required and 4 OPSTO (860 - 863)

4 METHOD AND OPERATION

a. Operation: Floating point

b. Additional routines required: None

c. Range and form of data: b must be real and normalized. Its range: $9 \times 10^{-75} \leq |b| \leq 9 \times 10^{75}$

d. Accuracy: Max. 1×2^{-40} of the significant number, otherwise as determined by e.

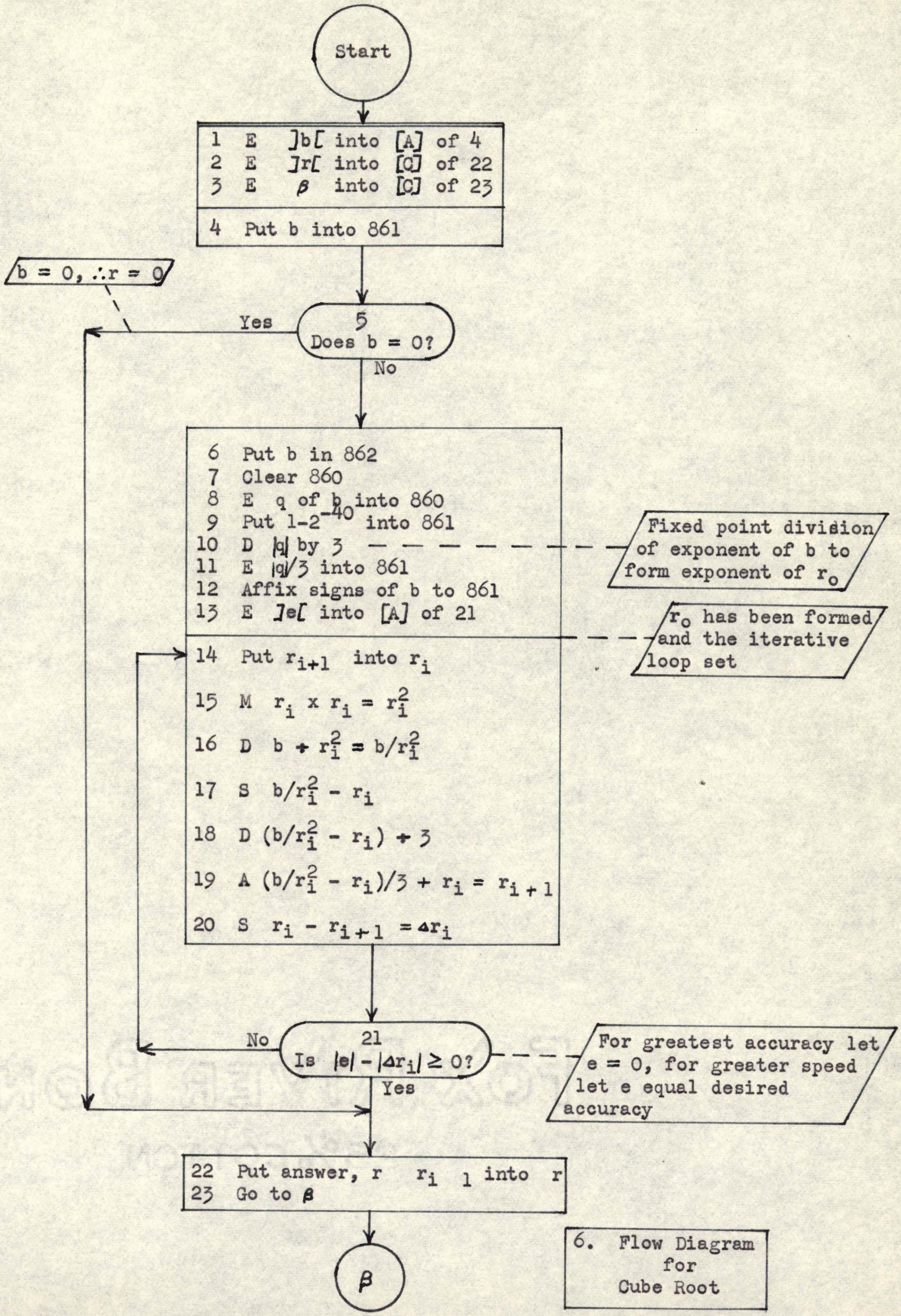
e. Method: The method used is the Newton - Raphson iteration:

$$r_{i+1} = r_i + (b/r_i^2 - r_i)/3$$

5. PROGRAMMING INFORMATION

a. Performance time: The time is dependent on the number of iterations as follows:

Number of iterations	1	4	5	6	7
Time in seconds	0.780	1.380	1.560	1.800	1.980



6. Flow Diagram for Cube Root

2.01.07

7. **WISCoding for** Cube Root of b

By Donald E. Coon **Date** 5 - 30 - 59 **Page** 1 **of** 1

FLOW #	ORDER #	X	TYPE	A	B	C	#	HEXADECIMAL X	T	A	B	C
	1	25	E	Lo [863]	[25,12]	[4]	1	019	1	35f	19c	004
	2	13	E	Lo [863]	[1,12]	[22]	2	00d	1	35f	01c	016
	3	1	E	Lo [863]	[1,12]	[23]	3	001	1	35f	01c	017
	4		A	b [(b)] +	0 [1023] → r_{i+1}	[861]	4	000	8	000	3ff	35d
	5		TZ	r_{i+1} [\ ] -	0 [1023]	[22]	5	000	c	800	3ff	016
	6		A	r_{i+1} [861] +	0 [1023] → N	[862]	6	000	8	35d	3ff	35e
	7		A	0 [1023] +	0 [1023] → Q	[860]	7	000	8	3ff	3ff	35c
	8	41	E	N [862]	q [31,8] → Q	[860]	8	029	1	35e	1f8	35c
	9		A	.111.. [1006] +	0 [1023] → r_{i+1}	[861]	9	000	8	3ee	3ff	35d
	10	100	D	Q [860] ÷	3 [24] →	[\ ]	a	100	3	35c	018	800
	11	33	E	[\ ]	$\frac{1}{3}$ [29,8] → r_{i+1}	[861]	b	021	1	800	298	35d
	12	49	E	N [862] signs	[49,2] → r_{i+1}	[861]	c	031	1	35e	312	35d
	13	37	E	Lo [863]]e	[25,12]	[21]	d	025	1	35f	19c	015
	14		A	r_{i+1} [861] +	0 [1023] → r_i	[863]	e	000	8	35d	3ff	35f
	15		M	r_i [\ ] ×	r_i [\ ]	[\ ]	f	000	2	800	800	800
	16		D	N [862] ÷	r_i^2 [\ ]	[\ ]	10	000	3	35e	800	800
	17		S	[\ ] -	r_i [863]	[\ ]	11	000	a	800	35f	800
	18		D	[\ ] ÷	3 [24]	[\ ]	12	000	3	800	018	800
	19		A	[\ ] +	r_i [863] → r_{i+1}	[861]	13	000	8	800	35f	35d
	20		S	r_i [863] -	[\ ]	[\ ]	14	000	a	35f	800	800
	21		TNA	e [(e)] -	[\ ]	[14]	15	000	f	(000)	800	00e
	22		A	r_{i+1} [861] +	0 [1023] → r	[(r)]	16	000	8	35d	3ff	(000)
	23		TU	[−]	[−]	[(β)]	17	000	5	000	000	(000)
	24		3	Constant	[ ]	[ ]	18	002	c	000	000	000
				[ ]	[ ]	[ ]						

8. ANALYSIS

a. Program

The cube root, r, of the normalized binary number b is found by the iteration:

$$r_{i+1} = r_i + (b/r_i^2 - r_i)/3$$

The routine is so written that the iteration ends when $r_i - r_{i+1} \leq e$ where e is zero for greatest accuracy but may be any positive number in floating point form.

The form of b is:

$$b = \pm p \times 2^{\pm q}$$

The initial guess, r_0, is formed by extracting the exponent q into bit positions 31 thru 38 of a cleared location, performing a fixed point divide by three, and extracting bits 33 thru 40 of the result into the exponemt position of $.1111...1_2$. The sign of b and the sign of q are then extracted into the signs position of the above number.

Thus if q is evenly divisible by 3:

$$r_o = \pm(1-2^{-40}) \times 2^{\pm q/3}$$

and if there is a remainder of 1:

$$r_o = \pm(1-2^{-40}) \times 2^{\pm(q-1)/3}$$

and if the remainder is 2:

$$r_o = \pm(1-2^{-40}) \times 2^{\pm(q-2)/3}$$

the following cases may arise:

(1) $b > 0$, $r_o = 1 \times 2^{\pm q/3}$ then:

$$b = r^3 = p \times 2^{\pm q} = p \times r_o^3$$

and $r_o/r = (p)^{-1/3}$

but since $0.5 \leq p < 1$,

$$1 < r_o/r \leq (2)^{1/3}$$

and the maximum number of iterations to find r is 7.

(2) $b > 0$, $q > 0$, $r_o = 1 \times 2^{(q-1)/3}$ then:

$b = r^3 = p \times 2^q = 2p \times r_o^3$

and $r_o/r = (2p)^{-1/3}$

but since $1 \leq 2p < 2$,

$$(2)^{-1/3} < r_o/r < 1$$

and the maximum number of iterations to find r is 6.

(3) $b > 0$, $q < 0$, $r_o = 1 \times 2^{-(q-1)/3}$ then:

$b = r^3 = p \times 2^{-q} = p/2 \times r_o^3$

and $r_o/r = (2/p)^{1/3}$

but since $2 < 2/p \leq 4$,

$$(2)^{1/3} < r_o/r \leq (4)^{1/3}$$

and the maximum number of iterations to find r is 7.

(4) $b > 0$, $q > 0$, $r_o = 1 \times 2^{(q-2)/3}$ then:

$b = r^3 = p \times 2^q = 4p \times r_o^3$

and $r_o/r = (4p)^{-1/3}$

but since $2 \leq 4p < 4$

$$(4)^{-1/3} < r_o/r \leq (2)^{-1/3}$$

and the maximum number of iterations to find r is 7.

(5) $b > 0$, $q < 0$, $r_o = 1 \times 2^{-(q-2)/3}$ then:

$b = r^3 = p \times 2^{-q} = p/4 \times r_o^3$

and $r_o/r = (4/p)^{1/3}$

but since $4 < 4/p \leq 8$

$$(4)^{1/3} < r_o/r \leq 2$$

and the maximum number of iterations to find r is 7.

A similar analysis holds for $b < 0$ since the sign of b did not enter the analysis.

It is interesting to note the way in which the routine depends on the initial value r_o. If $r_o/r = K_o$, $r_1/r = K_1$, $r_2/r = K_2$, etc., then since:

$$r_1 = r_o + (b/r_o^2 - r_o)/3$$

it follows that:

$$K_1 = K_o + (1/K_o^2 - K_o)/3$$

and the iteration is finished when $K_j \doteq K_{j+1}$. The number of iterations required for several values of K_o are given below.

K_o	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.4	1.6	1.8	2.0
N	7	7	6	5	1	6	6	7	7	7	7