COBOL-H の演算精度と速度

清水川緋紗子

穂鷹良介氏の開発された COBOL-H 使用の機会に恵まれたので，このコンパイラーの演算精度と速度について筆者の実験結果を報告する。なお COBOL-H についての有益な助言を与えられた同氏に深謝の意を表する。

目　　次

1. COBOL-H の文法

COBOL-H の文法は基本的には 1961 年の COBOL に拠るが幾つかの機能は拡張，制限または削除されている。具体的内容については参考文献 [1], [2] を参照されたい。[2] は日本電子工業協会から出版された COBOL (OKITAC-5090H 用) を COBOL-H の debug, 機能拡張に伴ない加筆補正したものである。主な変更は，1) MOVE の機能が拡張されてあらゆる data type の組み合せの MOVE が許され，Series Option の MOVE が可能になった。2) FEED 命令 (COBOL-H 固有命令) が追加された。3) ACCEPT, DISPLAY 動詞の解説が増補されたうえ，その機能がコンパイラーにつけ加

わった等である。

所で1961年のCOBOL文法は固定したものではなく，1962年にCOBOL-61 EXTENDED (文献[6]) が発行され大きな変更として 1) Report Writer の機能，2) Sorting の機能の両者がつけ加わった。また，CODASYL のその後の決定に基づいてCOBOL文法の新たな変更を記載した多くの Bulletin が発行されている。1965年，さらに新版が発行された。[7]

2. 演 算 精 度

2. 1. 事務計算の演算について

この節では精度が主関心事であるから，数値データの取り扱いに話を限定する。

COBOL-H の数値 data の入出力は普通次のように行なわれる。計算しようとする数値をその Card File の PICTURE に従って Card にパンチし，これらを data として計算機に読み込む。Card 上にパンチされる数値はプログラマー及び Card を扱う人がわかり易いように10進数で表現された numeric type である。しかし numeric type のまま入力され，かつ出力されることは稀であり，もとの data をそのまま output する場合を除いては作業によって data を変換したり，計算速度と精度をあげるため type の変換を行なったりする。さらに演算途中で使われる WORKING-STORAGE の PICTURE の宣言状態により truncation が行なわれることもある。出力の時は output する File の PICTURE に従い，もう一度変換して numeric type または edited data にしてから output される。

このように data type の変換は PICTURE 部分を変えるだけで自動的に行なわれるので，プログラマーは計算機内部の演算が2進で行なわれていることさえ気付かずにいることもある。

2進法と10進法との変換による最大誤差は[3]によれば10進から2進の数値に変換する場合は 2^{-42} で，逆の2進から10進の数値に変換する場合は

8×10^{-14} である。

さらに浮動小数点計算の仮数部の長さによって正確に表現され得る数値の桁数がきまる。（指数部が計算機の上限，下限を越えると orerflow, underflow が起るが事務計算で扱う普通の計算では起らないようにすべきであろうから省略する）。OKITAC-5090H の浮動小数は single precision で仮数部が 2^{32} 迄，倍精度で 2^{64} 迄表現される。これらは大体 10 進の数値で 9 桁と 19 桁であるから truncation を起さず表現され得る最大の桁数は single precision で 10 進の 9 桁，倍精度で 19 桁である。

OKITAC-5090H は整数型数値を $2^{42}-1$（10 進で 13 桁）迄表現できる。

しかし COBOL が事務用言語であるために科学計算における浮動小数点数値の有効桁，計算誤差の概念が当てはまらず，時には最小単位迄が正確に output されることを要求されることがある。だが，使用する機械のカナモノの条件（2 進か 10 進か，多倍精度の計算の容易，word の bit 構成と長さ）によって最小単位迄求め得るとは限らない。

2. 2. COBOL-H でとり扱える数値の範囲

数値 data の計算機内部の精度は **2. 1.** で述べた通りであるが output された結果で精度を云々する時は DATA DIVISION の PICTURE も精度を左右する。

COBOL-Hの PICTURE の最大の長さは 2 word 分であり WORKING-STORAGE で numeric type で計算する時は有効桁が 10 進 13 桁以上あってはならない。Print Area の numeric display は特に小数点の左右に 13 桁以内の literal があってもよく，9(13). 9(13) が最大桁を表現し得る PICTURE である。従って数値 data が入れるのは 1.0×10^{-13} と 1.0×10^{13} との間である。data が整数であれば有効桁として input 及び output される桁数は 10 進の 13 桁以内であり，10 進の 12 桁迄の整数が正しく計算される。

例 1.　2^n(n＝2, 3, …) と (n＝−2, −3, …) を COBOL-H で計算させ

た結果が figure-1 である。

当然ながら2進数値と10進数値との変換による誤差は生じないし，初め2.0と10進で2桁しか有効桁がなくとも高位の有効桁が生じて 2^{40}（10進の13桁）迄正確にでている。2^{41} の計算で overflow を生じた。途中の計算をComputational type で行なってもまったく同じ結果を得た。

例 2. 小数点をもつ数値 data の入出力

小数点をもつ数値 data の計算精度をみるため DATA DIVISION の PICTURE と入力の方法と数値を変化させたのが figure-2 である。output は LP とし，その PICTURE を 9(13). 9(13) に固定した。

入力は CARD の場合と WORKING-STORAGE に Literal で与える場合の2つの方法をとった。CARD を読みこむ PICTURE を 9V9型の Computational に固定し小数点以下を2づつ増やし 9V9(12) 迄を取扱った。WORKING-STORACTE の PICTURE は 9V9 から 9V9(12) 迄小数点以下を1づつ増し，Computational をかけた場合とかけない場合の両方を行なった。

数値は $2^{-n}(n=1, 2, \cdots 12)$ と $2^{-n}-10^{-n}$ との両方を扱った。これは OKITAC-5090H が内部2進の機械であるから2進10進の変換による誤差のあらわれる様子をみるためである。

以上のいろいろな組合せで次のことを行なった。

1) 読み込んだ，又は literal で置いた data を作業用番地 WM (PICTURE が 9V9(12)) と WC (PICTURE が 9(14)C) へ移してから WM と WC の各々を LP の elementary item へ移した。

2) 1) の WC と WM を使って次の計算をしてから LP の elementay item へ移した。

SUMC=WC * WC/WC+WC−WC

SUMM=WM * WM/WM+WM−WM

ここに SUMC, SUMM は WORKING-STORAGE で定義されており，そ

の PICTURE は 9(14)C と 9V9(12) である。もしも計算誤差がなければ，当然 SUMC は WC に SUMM は WM に一致すべきである。WM の PICTURE が 9V9(12) より短くなれば，それだけ truncation が行なわれ下の桁の精度は失なわれる。小数点の位置が右へずれた時も同様である。

整数部と小数部の両方をもつ一般の数値 data については限られた例ではあるが figure-4 をあげた。小数部分が 2 進でどう表されるかによって numeric の PICTURE でズバリ入るかどうかがきまる。

COBOL-H で計算する数値 data に小数点がある時は WORKING-STORAGE の PICTURE は一般に COMPUTATIONAL と宣言しておくべきである。さもないと知らずに精度を失なうことがある。例えば WS に 0.2 を置く時 9V9 という PICTURE であれば 0.199999... の頭 1 桁をとった 0.1 が入ってしまい，以後の計算結果はすべて誤差をもつ。もしも PICTURE として最大 7 桁をとればよいということが予めわかっている時はこの限りではなく core を節約するために 1 word で最長の PICTURE を numeric type で宣言しておけばよい。

2. 3. 小数位の移動

例 3. 10^n の計算を行ない初めの有効桁以下より精度が保たれないことを figure-3 で示している。B_0 を WS で PICTURE 9V9 V1.0 で与え i の値を順次変えて B_i を求めた。途中の計算の WS の PICTURE は 9(12) V9 で印刷の PICTURE は 9(13).9 である。

3. 演　算　速　度

3. 1. 数式の演算時間

例えば X＝A＋B の計算式においても X, A, B の DATA DIVISION での PICTURE が各々 Computational であるか Numeric type であるかによって $2^3=8$ 通りがある。A と B の位置をかえて X＝B＋A と考えても速

度には差がないと考えられるから table-1 の6通りの組み合せについて演算時間を測定した。変数の添字はその変数の type を表す。N は 9999 type の numeric, C は Computational を表す。

table-1　PICTURE による演算速度の違い

演算式	加　算	乗　算	除　算
$X_N = A_N \cdot B_N$	14	15	15
$X_N = A_N \cdot B_C$	10	11	11
$X_N = A_C \cdot B_C$	7	8	8
$X_C = A_N \cdot B_N$	10	10	11
$X_C = A_N \cdot B_C$	7	7	7
$X_C = A_C \cdot A_C$	3	4	5

• = Arithmetic Operator.
単位 = ms.

当然のことながら特に Computational の計算が速くなっていることがわかる。

3. 2. 転送命令の処理時間

MOVE $\left\{\begin{matrix}\text{data-name-1}\\ \text{literal}\end{matrix}\right\}$ TO data-name-2, ...

program 中でデータの転送を行ない File の中のデータを変えたり，edit をしたりするのは COBOL 言語の特徴の一つである。事務計算ではデータの転送が仕事の大きな部分を占めるから，その program の良否は $\left\{\begin{matrix}\text{data-name-1}\\ \text{literal}\end{matrix}\right\}$ の Source-Area から data-name-2 の Receiving Area への転送状態の効率に左右される。

この命令の処理時間は data-name-i (i=1, 2) と literal の状態の条件によっていろいろ変わる。COBOL-H の data-name-i と literal の条件の組み合せは [1] の MOVE の所で述べている。ここでは 1 item 同志の MOVE に限り，

table-2

PICTURE 行 出　口　先	9 (3)	9 (3) C	X(3)	9 (7)
literal				3
9 (3)	4			4
9 (3) C		1		
X(3)			4	
9 (7)				4
9 (7) C				5
X(7)				4
9 (10)				
9 (10) C				
X(10)				
9 (14)				4
9 (14) C				
X(14)				

literal, numeric, non-numeric の場合を考察する。例えば同じ numeric 同志としてもそれらの data が program の中でどのような PICTURE を宣言されているか，さらにその内容は character 単位でどのように pack されているかにより処理時間の長短がどのように左右されるかをみようとした。

OKITAC-5090H の機械は 1 word (42 bit • binary) であり 6 bit 1 文字で 7 桁の文字が入る。COBOL-H では [1] にも述べられているように番地割付けの際 1 語か 2 語が割当てられるので 7 桁と 14 桁の data が特に変化があるかどうかをみたが，後述のように特に変化はなかった。

data-name-i (i＝1, 2) の条件をいろいろ変えて MOVE の処理時間を測定した。結果は table-2 の通りである。

data の出口も行先も numerie の場合は MOVE 1 命令を実行するのに約 4 ms かかった。この時 data の桁数にはほとんど依存せず，7 桁や 14 桁の時が特に速くなるということはなかった。

行先で Edit (Zero Suppress を実行) を併うと 10 ms [Z(3)], 14 ms [Z(7)], 20 ms [Z(14)] のように numeric の時よりも遅くなり，かつ Zero Suppress されるものの桁数の増加に伴ない処理時間は長くなる。

転送命令の処理時間

MOVE {data-name-1 / literal} TO data-name-2.　　単位 ms

9 (7) C	X(7)	9 (10)	9 (10) C	X(10)	9 (14)	9 (14) C	X(14) C	zero supress
					4			Z (3) —10
	4							
1	4	4		4	4	5 1	4 4	Z (7) —14
	4	4	1	4				Z (10)—17
	4				4 5 4	5 1	4	Z (14)—20

Computational 同志の MOVE では 1 ms より時間がかからない。従って COBOL-H では Computational を旨く使った方が計算速度も，精度も有利である。

尚，table-2 で空白の部分は実験を省略した。

3. 3. FORTRAN との比較

例 3. $V=\sum_{i=1}^{N}(X_i-\bar{X})^2/N$

$X_1=1,\ X_{i+1}=X_i+1,\ N=10000,\ \bar{X}=\frac{1}{N}\sum_{i=1}^{N}X_i$

Vを求めて印刷する。figure-5 参照。

COBOL-H で書くと program step 数は 27，list をとってのコンパイル，プリコンパイル時間の合計が 28 秒，計算時間が 145 秒であった。

これを FORTRAN 型の言語 OKIART で実行させると 14 step，コンパイルとプリコンパイルのための時間が 8.5 秒，計算時間も 8.5 秒であった。

この例題では OKITAC-5090H を使用した時の COBOL-H と FORTRAN の計算速度は約 20：1 であった。

3. 4. 他の COBOL コンパイラーとの比較

[4] に IBM 7070/44, [3] に FACOM を使った COBOL 言語の演算速度が述べられている。他のコンパイラーと比較をする時，当然カナモノによる速度差，機械語の命令の質と数の違いがあるので優劣はつけがたい。

COBOL-H は data 桁数が word の桁数の整数倍や computational 同志の MOVE では IBM 7040/44 のように急速に速くなることはないが，Edit を伴なう MOVE や data の桁数の違う MOVE では約 10^1 の order 遅いのにすぎず，前に述べた種々の条件を考えると IBM コンパイラーと遜色ないものといえよう。

結論　1．a，演算結果が 12 桁以内の四則演算はよい精度が保たれている。

2. a, 速度は Computational type であれば演算及び転送時間が最も早く，Edit または data の type の変換を伴なうとおそくなる。

b, mumeric と non-numeric では転送速度に差がない。

c, data の word length や core への pack 状態は速度に影響を与えない。

figure-1 computation of 2^n

PICTURE of WS	9 (14)	9 (14)
PICTURE of Print	9.9 (13)	Z (14)

	$A_i=2^{-i}$	$B_i=2^i$
i = 2	0.2500000000000	4
i = 3	0.1250000000000	8
i = 4	0.0625000000000	16
i = 5	0.0312500000000	32
i = 6	0.0156250000000	64
i = 7	0.0078125000000	128
i = 8	0.0039062500000	256
i = 9	0.0019531250000	512
i =10	0.0009765625000	1024
i =11	0.0004882812500	2048
i =12	0.0002441406250	4096
i =13	0.0001220703125	8192
i =14	0.0000610351563	16384
i =15	0.0000305175781	32768
i =16	0.0000152587891	65536
i =17	0.0000076293945	131072
i =18	0.0000038146973	262144
i =19	0.0000019073486	524288
i =20	0.0000009536743	1048576
i =21	0.0000004768372	2097152
i =22	0.0000002384186	4194304
i =23	0.0000001192093	8388608
i =24	0.0000000596046	16777216
i =25	0.0000000298023	33554432
i =26	0.0000000149012	67108864
i =27	0.0000000074506	134217728
i =28	0.0000000037253	268435456
i =29	0.0000000018627	536870912
i =30	0.0000000009313	1073741824
i =31	0.0000000004657	2147483648
i =32	0.0000000002328	4294967296
i =33	0.0000000001164	8589934592
i =34	0.0000000000582	17179869184
i =35	0.0000000000291	34359738368
i =36	0.0000000000146	68719476736
i =37	0.0000000000073	137438953472
i =38	0.0000000000036	274877906944
i =39	0.0000000000018	549755813888
i =40	0.0000000000009	1099511627776
i =41	0.0000000000005	overflow

初期値 A_0 を WS (WORKING-STORAGE) に P9V9 V2.0 で与え，$A_1=B_1=1.0$，$A_{i+1}=A_i/A_0$，$B_{i+1}=B_i * A_0$ をで計算した。

figure-2

Literal n	VALUE 2^{-n}	PICTURE IN WS	WC	WM	SUMC	SUMM
1	0.5	9V9	0.5000000000000	0.5000000000000	0.5000000000000	0.5000000000000
2	0.25	9V99	0.2500000000000	0.2500000000000	0.2500000000000	0.2500000000000
3	0.125	9V999	0.1250000000000	0.1250000000000	0.1250000000000	0.1650000000000
4	0.0625	9V9(4)	0.0625000000000	0.0625000000000	0.0625000000000	0.0325000000000
5	0.03125	9V9(5)	0.0312500000000	0.0312500000000	0.0312500000000	0.0112500000000
6	0.015625	9V9(6)	0.0156250000000	0.0156250000000	0.0156250000000	0.0156250000000
7	0.0078125	9V9(7)	0.0078125000000	0.0078125000000	0.0078125000000	0.0078125000000
8	0.00390625	9V9(8)	0.0039062500000	0.0039062500000	0.0039062500000	0.0039062500000
9	0.001953125	9V9(9)	0.0019531250000	0.0019531250000	0.0019531250000	0.0019531250000
10	0.0009765625	9V9(10)	0.0009765625000	0.0009765625000	0.0009765625000	0.0009765625000
11	0.00048828125	9V9(11)	0.0004882812500	0.0004882812500	0.0004882812500	0.0004882812500
12	0.000244140625	9V9(12)	0.0002441406250	0.0002441406250	0.0002441406250	0.0002441406250
	VALUE $2^{-n}-10^{-n}$					
1	0.4	9V9	0.4000000000001	0.4000000000000	0.4000000000001	0.4000000000000
2	0.24	9V99	0.2300000000000	0.2300000000000	0.2300000000000	0.2300000000000
3	0.124	9V999	0.1239999999998	0.1240000000000	0.1239999999998	0.1239999999990
4	0.0624	9V9(4)	0.0623000000001	0.0623000000000	0.0623000000001	0.0623000000000
5	0.03124	9V9(5)	0.0312300000001	0.0312300000000	0.0312300000001	0.0312300000000
6	0.015624	9V9(6)	0.0156229999998	0.0156230000000	0.0156229999998	0.0156229999990
7	0.0078124	9V9(7)	0.0078122999998	0.0078123000000	0.0078122999998	0.0078122999990
8	0.00390624	9V9(8)	0.0039062400001	0.0039062400000	0.0039062400001	0.0039062400000
9	0.001953124	9V9(9)	0.0019531240000	0.0019531240000	0.0019531240000	0.0019531240000
10	0.0009765624	9V9(10)	0.0009765624000	0.0009765624000	0.0009765624000	0.0009765624000
11	0.00048828124	9V9(11)	0.0004882812400	0.0004882812400	0.0004882812400	0.0004882812400
12	0.000244140624	9V9(12)	0.0002441406232	0.0002441406230	0.0002441406232	0.0002441406230
1	0.4	9V9 C	0.4000000000233	0.4000000000230	0.4000000000033	0.4000000000220
2	0.24	9V99 C	0.2399999999907	0.2399999999900	0.2399999999907	0.2399999999890
3	0.124	9V999 C	0.1240000000107	0.1240000000100	0.1240000000107	0.1240000000090
4	0.0624	9V9(4) C	0.0623999999952	0.0623999999950	0.0623999999952	0.0623999999940
5	0.03124	9V9(5) C	0.0312399999966	0.0312399999960	0.0312399999966	0.0312399999960
6	0.015624	9V9(6) C	0.0156239999997	0.0156239999990	0.0156239999997	0.0156239999990
7	0.0078124	9V9(7) C	0.0078123999992	0.0078123999990	0.0078123999992	0.0078123999980
8	0.00390624	9V9(8) C	0.0039062400001	0.0039062400000	0.0039062400001	0.0039062400000

9	0.001953124	9V9(9) C	0.0019531240000	0.0019531240000	0.0019531240000	0.0019531240000
10	0.0009765624	9V9(10) C	0.0009765624000	0.0009765624000	0.0009765624000	0.0009765624000
11	0.00048828124	9V9(11) C	0.0004882812400	0.0004882812400	0.0004882812400	0.0004882812400
12	0.000244140624	9V9(12) C	0.0002441406236	0.0002441406230	0.0002441406236	0.0002441406230
Card	PICTURE IN CARD	PICTURE IN WS	WC	WM	SUMC	SUMM
VALUE が 2^{-n} 型の場合	9V99 C	6V99	0.2500000000000	0.2500000000000	0.2500000000000	0.2500000000000
	9V9(4) C	9V9(4)	0.2500000000000	0.2500000000000	0.2500000000000	0.2500000000000
	9V9(6) C	9V9(6)	0.2500000000000	0.2500000000000	0.2500000000000	0.2500000000000
	9V9(8) C	9V9(8)	0.2500000000000	0.2500000000000	0.2500000000000	0.2500000000000
	9V9(10) C	9V9(10)	0.2500000000000	0.2500000000000	0.2500000000000	0.2500000000000
	9V9(12) C	9V9(12)	0.2500000000000	0.2500000000000	0.2500000000000	0.2500000000000
	9V9(4) C	9V9(4)	0.0625000000000	0.0625000000000	0.0625000000000	0.0625000000000
	9V9(6) C	9V9(6)	0.0625000000000	0.0625000000000	0.0625000000000	0.0625000000000
	9V9(8) C	9V9(8)	0.0625000000000	0.0625000000000	0.0625000000000	0.0625000000000
	9V9(10) C	9V9(10)	0.0625000000000	0.0625000000000	0.0625000000000	0.0625000000000
	9V9(12) C	9V9(12)	0.0625000000000	0.0625000000000	0.0625000000000	0.0625000000000
	9V9(6) C	9V9(6)	0.0156250000000	0.0156250000000	0.0156250000000	0.0156250000000
	9V9(8) C	9V9(8)	0.0156250000000	0.0156250000000	0.0156250000000	0.0156250000000
	9V9(10) C	9V9(10)	0.0156250000000	0.0156250000000	0.0156250000000	0.0156250000000
	9V9(12) C	9V9(12)	0.0156250000000	0.0156250000000	0.0156250000000	0.0156250000000
	9V9(8) C	9V9(8)	0 0039062500000	0.0039062500000	0.0039062500000	0.0039062500000
	9V9(10) C	9V9(10)	0.0039062500000	0.0039062500000	0.0039062500000	0.0039062500000
	9V9(12) C	9V9(12)	0.0039062500000	0.0039062500000	0.0039062500000	0.0039062500000
	9V9(10) C	9V9(10)	0.0009765625000	0.0009765625000	0.0009765625000	0.0009765625000
	9V9(12) C	9V9(12)	0.0009765625000	0.0009765625000	0.0009765625000	0.0009765625000
	9V9(12) C	9V9(12)	0.0002441406250	0.0002441406250	0.0002441406250	0.0002441406250
VALUE が $2^{-n}-10^{-n}$ 型の場合	9V99 C	9V99	0.2300000000000	0.2300000000000	0.2300000000000	0.2300000000000
	9V9(4) C	9V9(4)	0.2399000000000	0.2399000000000	0.2399000000000	0.2399000000000
	9V9(6) C	9V9(6)	0.2399989999999	0.2399990000000	0.2399989999999	0.2399989999990
	9V9(8) C	9V9(8)	0.2399999899999	0.2399999900000	0.2399999899999	0.2399999899990
	9V9(10) C	9V9(10)	0.2399999999002	0.2399999999000	0.2399999999002	0.2399999999000
	9V9(12) C	9V9(12)	0.2399999999989	0.2399999999990	0.2399999999989	0.2399999999980
	9V9(4) C	9V9(4)	0.0623000000001	0.0623000000000	0.0623000000001	0.0623000000000
	9V9(6) C	9V9(6)	0.0623989999999	0.0623990000000	0.0623989999999	0.0623989999990

9V9(8) C	9V9(8)	0.062399899999	0.062399900000	0.062399899999	0.062399899990
9V9(10) C	9V9(10)	0.062399999002	0.062399999000	0.062399999002	0.062399999000
9V9(12) C	9V9(12)	0.062399999989	0.062399999990	0.062399999989	0.062399999980
9V9(6) C	9V9(6)	0.015624000001	0.015624000000	0.015624000001	0.015624000000
9V9(8) C	9V9(8)	0.015624000001	0.015624000000	0.015624000001	0.015624000000
9V9(10) C	9V9(10)	0.015624000001	0.015624000000	0.015624000001	0.015624000000
9V9(12) C	9V9(12)	0.015624000001	0.015624000000	0.015624000001	0.015624000000
9V9(8) C	9V9(8)	0.003906240001	0.003906240000	0.003906240001	0.039062400000
9V9(10) C	9V9(10)	0.003906240001	0.003906240000	0.003906240001	0.039062400000
9V9(12) C	9V9(12)	0.003906240001	0.003906240000	0.003906240001	0.039062400000
9V9(10) C	9V9(10)	0.000976562400	0.000976562400	0.000976562400	0.000976562400
9V9(12) C	9V9(12)	0.000976562400	0.000976562400	0.000976562400	0.000976562400
9V9(12) C	9V9(12)	0.000244140624	0.000244140642	0.000244140642	0.000244140642
9V99 C	9V99	0.239999999998	0.239999999990	0.239999999998	0.239999999980
9V9(4) C	9V9(4)	0.239999999998	0.239999999990	0.239999999998	0.239999999980
9V9(6) C	9V9(6)	0,239999999998	0.239999999990	0.239999999998	0.239999999980
9V9(8) C	9V9(8)	0.239999999998	0.239999999990	0.239999999998	0.239999999980
9V9(10) C	9V9(10)	0.239999999998	0.239999999990	0.239999999998	0.239999999980
9V9(12) C	9V9(12)	0.239999999998	0.239999999990	0.239999999998	0.239999999980
9V9(4) C	9V9(4)	0.062399999998	0.062399999990	0.062399999998	0.062399999980
9V9(6) C	9V9(6)	0.062399999998	0.062399999990	0.062399999998	0.062399999980
9V9(8) C	9V9(8)	0.062399999998	0.062399999990	0.062399999998	0.062399999980
9V9(10) C	9V9(10)	0.062399999998	0.062399999990	0.062399999998	0.062399999980
9V9(12) C	9V9(12)	0.062399999998	0.062399999990	0.062399999998	0.062399999980
9V9(6) C	9V9(6)	0.015624000001	0.015624000000	0.015624000001	0.015624000000
9V9(8) C	9V9(8)	0.015624000001	0.015624000000	0.015624000001	0.015624000000
9V9(10) C	9V9(10)	0.015624000001	0.015624000000	0.015624000001	0.015624000000
9V9(12) C	9V9(12)	0.015624000001	0.015624000000	0.015624000001	0.015624000000
9V9(8) C	9V9(8)	0.003906240001	0.003906240000	0.003906240001	0.039062400000
9V9(10) C	9V9(10)	0.003906240001	0.003906240000	0.003906240001	0.039062400000
9V9(12) C	9V9(12)	0.003906240001	0.003906240000	0.003906240001	0.039062400000
9V9(10) C	9V9(10)	0.000976562400	0.000976562400	0.000976562400	0.000976562400
9V9(12) C	9V9(12)	0.000976562400	0.000976562400	0.000976562400	0.000976562400
9V9(12) C	9V9(12)	0.000244140624	0.000244140642	0.000244140642	0.000244140642

figure-3 shift of dicimal point

	$B_i=B_0 * (10.0)^i$	$B_i=B_0/(0.1)^i$
i = 0	0000000000001.0	0000000000001.0
i = 1	0000000000010.0	0000000000009.9
i = 2	0000000000100.0	0000000000098.9
i = 3	0000000001000.0	0000000000988.9
i = 4	0000000010000.0	0000000009888.9
i = 5	0000000100000.0	0000000098888.9
i = 6	0000001000000.0	0000000988888.9
i = 7	0000010000000.0	0000009888888.9
i = 8	0000100000000.0	0000098888888.9
i = 9	0001000000000.0	0000988888888.9
i =10	0010000000000.0	0009888888888.4
i =11	0100000000000.0	0098888888878.2
i =12	1000000000000.0	0988888888724.4

figure-4

VALUE	PICTURE IN CARD	PICTURE IN WS	WC	SUMC
1.25	9V99 C	9V99	1.2500000000000	1.2500000000000
1.0625	9V9999 C	9V9999	1.0625000000000	1.0625000000000
1.24	9V99 C	9V99	1.2300000000000	1.2300000000000
1.24	9V99 C	9V99 C	1.2399999999998	1.2399999999993
VALUE	PICTURE IN CARD	PICTURE IN WS	WM	SUMM
1.25	9V99 C	9V99	1.2500000000000	1.2500000000000
1.0625	9V9999 C	9V9999	1.0625000000000	1.0625000000000
1.24	9V99 C	9V99	1.2300000000000	1.2300000000000
1.24	9V99 C	9V99 C	1.2399999999990	1.2399999999980

figure-5 caluculation of variance

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
FILF-CONTROL.
    SELECT PRINT ASSIGN TO LP 1.
DATA DIVISION.
FILE SECTION.
FD PRINT LABEL RECORD IS OMITTED DATA RECORD IS P1.
01  P1.
    02  OUT1 P Z(12).99.
WORKING-STORAGE SECTION.
77  N   P 9(14) C.
77  S   P 9(14) C.
77  SS  P 9(14) C.
77  VA P 9(14) C.
PROCEDURE DIVISION.
    OPEN OUTPUT PRINT.
    MOVE 0  TO S, SS, N.
SUM. COMPUTE N=1.0+N.
      COMPUTE S=S+N.  COMPUTE SS=SS+N * N.
    IF N IS L THAN 10000.0 GO TO SUM.
    COMPUTE VA=(SS - S * S/N)/N.
    MOVE VA TO OUT1
    WRITE P1.
    CLOSE PRINT.
    STOP "END OF JOB."
    STOP RUN.
END.
```

昭和42年6月12日

参 考 文 献

[1] 穂鷹良介 Automatic Coding (Ⅲ) その1とその2 商学討究第18巻1号と18巻2号 1967年。

[2] 穂鷹良介 COBOL-H 説明書 1967年7月。

[3] OKITAC-5090H Instruction Manual 沖電気工業株式会社 1966年。

[4] 三井信雄 COBOLに関する諸問題 情報処理 5巻6号。

[5] 渡辺昭雄 コボル入門 日刊工業新聞社 1965年。

[6] EXTENDED SPECIFICATIONS for a COMMON BUSINESS ORIENTED LANGUAGE (COBOL) for Programming Electric Digital Computers, DEPARTMENT OF DEFFENSE, 1962.

[7] COBOL, Edition 1965.
DEPARTMENT OF DEFFENCE, 1965.