進数と補数と多桁計算について　　澤見英男

１．進数と四則演算について

n 桁の r 進数を用いた整数の数詞表現と数値との対応は以下のようになる(**r; radix**)。

数詞表現：$d_{n-1}\, d_{n-2}\, d_{n-3} \ldots\, d_2\, d_1\, d_0$
数値：$d_{n-1} * r^{n-1} + d_{n-2} * r^{n-2} + d_{n-3} * r^{n-3} + \cdots + d_2 * r^2 + d_1 * r^1 + d_0$
ただし　$d_k \in \{0,1,2,\ldots,\mathrm{r}-1\}$　for　k=0,1,2,…, $n-1$　である（ゼロの威力)。

整数の加算 $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{y}$ $(= x_{n-1}\, x_{n-2}\, x_{n-3} \ldots x_2\, x_1\, x_0 \;+\; y_{n-1}\, y_{n-2}\, y_{n-3} \ldots y_2\, y_1\, y_0$) は
$\boldsymbol{z} \coloneqq z_n\, z_{n-1}\, z_{n-2}\, z_{n-3} \ldots z_2\, z_1\, z_0$ とすると桁上げ(carry) を考慮し次のように表現できる。

$$
\begin{array}{r}
c_n\, c_{n-1}\, c_{n-2}\, c_{n-3} \ldots c_2\, c_1\, c_0 \\
\hline
x_{n-1}\, x_{n-2}\, x_{n-3} \ldots x_2\, x_1\, x_0 \\
+\; y_{n-1}\, y_{n-2}\, y_{n-3} \ldots y_2\, y_1\, y_0 \\
\hline
=\; z_n\, z_{n-1}\, z_{n-2}\, z_{n-3} \ldots z_2\, z_1\, z_0
\end{array}
$$

$z_k = mod(x_k + y_k + c_k\ ,\ \mathrm{r})\ ,\ c_k = quotient(\ x_{k-1} + y_{k-1} + c_{k-1}\ ,\ \mathrm{r}\ )\ ,\ z_n = c_n\ ,\ c_0 = 0$

ただし *quotient* は r を用いた除算の商を *mod* は余りを表している。
多桁の四則演算は, 各桁を配列(セル)に記憶し, 上記のように行うことができる(３参照)。

n 桁の整数の乗算 $\boldsymbol{z} = \boldsymbol{x} * \boldsymbol{y} = x_{n-1}\, x_{n-2}\, x_{n-3} \ldots x_2\, x_1\, x_0 \;*\; y_{n-1}\, y_{n-2}\, y_{n-3} \ldots y_2\, y_1\, y_0$　は
$\boldsymbol{z} \coloneqq z_{2n}\, z_{2n-1}\, z_{2n-2}\, z_{2n-3} \ldots z_2\, z_1\, z_0$ とすると桁上げ(carry) も考慮し次のように表現できる。

$$
\begin{array}{r}
c_{2n-1}\, c_{2n-2}\, c_{2n-3} \quad \ldots \quad c_n\, c_{n-1}\, c_{n-2}\, c_{n-3} \ldots c_2\, c_1\, c_0 \\
\hline
x_{n-1}\, x_{n-2}\, x_{n-3} \ldots x_2\, x_1\, x_0 \\
*\;\; y_{n-1}\, y_{n-2}\, y_{n-3} \ldots y_2\, y_1\, y_0 \\
\hline
y_0(x_{n-1}\, x_{n-2}\, x_{n-3} \ldots x_2\, x_1\, x_0) \\
+\, y_1(x_{n-1}\, x_{n-2}\, x_{n-3} \ldots x_2\, x_1\, x_0) \\
\cdots\cdots\cdots\cdots\cdots\cdots \\
+\, y_{n-1}(x_{n-1}\, x_{n-2}\, x_{n-3} \ldots x_2\, x_1\, x_0) \\
\hline
=\; z_{2n-1}\; z_{2n-2}\; z_{2n-3} \quad \cdots \quad z_n\, z_{n-1}\, z_{n-2}\, z_{n-3} \ldots z_2\, z_1\, z_0
\end{array}
$$

ここで z_k は k 桁目の項全てを加算した値の下１桁 , c_k はこの総和を r で除して商にも配慮した桁上げであり, $z_0 = mod(y_0 * x_0\ ,\ r)$, $c_0 = 0$ などとなる。ただし r=2 （2 進数）に

すると$z_0 = x_0$もしくはゼロになるだけではなく，乗算$y_k(x_{n-1}\,x_{n-2}\,x_{n-3}\ldots x_2\,x_1\,x_0)$の結果も$(x_{n-1}\,x_{n-2}\,x_{n-3}\ldots x_2\,x_1\,x_0)$もしくはゼロという簡単な値になってしまう（$y_k \in \{0,1\}$）。
さらに２進数（r=2）を用いた場合には，除算の計算手順が主としていわゆる桁シフトと減算によって簡単に構成できてしまうことを注記しておく。

２．補数について

一方，整数の減算 $\boldsymbol{z = x - y}$ は r の補数 $\boldsymbol{t}$ を用いた加算 $\boldsymbol{z = x + t}$ により実行できる。

補数：$\boldsymbol{t} \coloneqq t_{n-1}\,t_{n-2}\,t_{n-3}\ldots t_2\,t_1\,t_0$
内訳：$t_k = r - 1 - y_k$，ただし $t_0 = r - y_0$（r の補数） または $t_0 = r - 1 - y_0$（r－1 の補数）

減数 $\boldsymbol{y}$ の最下位桁(y_0)が零でなければ，補数の最下位桁(t_0) は r および r－1 の補数いずれの場合でも r 以下となり，桁上げ処理が不要になることから，どの桁からでも計算することができる。身近に感じるであろうと考えられるプチ応用例として，釣銭を計算する際の減算(r=10)を上の桁から行うという計算手順を以下に示す。

千円で１２３円の商品を購入した際のつり銭の計算を重要な上位桁から行う：
１２３の１０の補数は各桁の合計が９になる様にして最下位桁（$y_0 = 3 \neq 0$ に注意）に１を加えると得られ，８７６＋１＝８７７となる。これが釣銭の額であるが，補数を使うことによって，経済価値の高い上位桁から計算できるようになる。

3. 多桁計算について

多桁計算の簡単な例として，表計算ソフト「エクセル」を用いた，被乗数を多桁の百億進数で表現した階乗の計算方法と，計算結果を多桁の１億進数で表現したネピア数の計算手順を取り上げる（サーバにて提供されているファイル参照)。なおどちらの場合も，得られる結果が数万桁であったとしても，乗数と除数は１桁で十分であることを注記しておく。

3.1　百億進数を用いた階乗の計算

１０桁の百億進数を用いて階乗（ｎ！）を計算する。ワークシートのセルＡ２からＡ７２にはｎとして０から７０までを入力しておく。百億進数１０桁目のセルＬ２には０！として１を入力しておく。このようにした場合，被乗数は１０桁の百億進数となることから，階乗計算を十進数百桁まで計算することが可能となる。なおＭ欄は値を零に初期化しておく。

	A	B	C	D	E	F	G	H	I	J	K	L
1	n	!=	1*2*3*・・・*n		Computation of the factorial n! using 10places number system with radix 10000000000							by H.Sawami
2	0	!=										1
3	1	!=										
4	2	!=										
5	3	!=										
6	4	!=										
7	5	!=										
8	6	!=										
9	7	!=										
10	8	!=										
11	9	!=										
12	10	!=										
13	11	!=										
14	12	!=										
15	13	!=										
16	14	!=										
17	15	!=										
18	16	!=										
19	17	!=										
20	18	!=										
21	19	!=										
22	20	!=										

図１　百億進数の各桁などを対応するセルに記憶し階乗計算の準備をする

階乗を百億進数として記憶しているセルの最下位桁から桁上げを考慮した乗算を行う。n=1 の場合，以下のように最下位桁になるセルＬ3 を計算し，百億進数１桁分の結果を得る。

=MOD(QUOTIENT(M2*$A3,10000000000)+L2*$A3,10000000000)

	A	B	C	D	E	F	G	H	I	J	K	L
1	n	!=	1*2*3*・・・*n		Computation of the factorial n! using 10places number system with radix 10000000000							by H.Sawami
2	0	!=										1
3	1	!=										1
4	2	!=										2
5	3	!=										6
6	4	!=										24
7	5	!=										120
8	6	!=										720
9	7	!=										5040
10	8	!=										40320
11	9	!=										362880
12	10	!=										3628800
13	11	!=										39916800
14	12	!=										479001600
15	13	!=										6227020800
16	14	!=									8	7178291200
17	15	!=									130	7674368000
18	16	!=									2092	2789888000
19	17	!=									35568	7428096000
20	18	!=									640237	3705728000
21	19	!=									12164510	0408832000
22	20	!=									243290200	8176640000

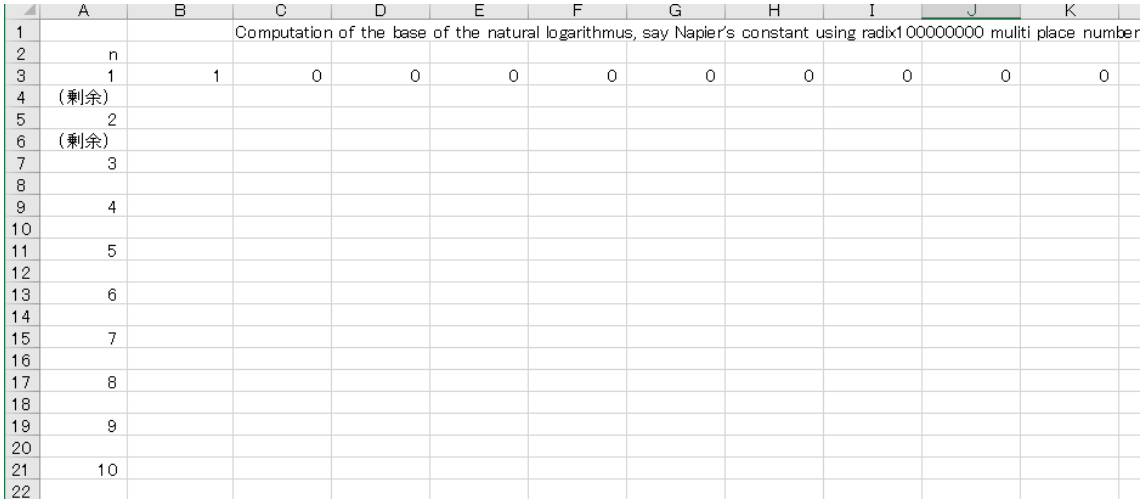
図２　百億進数の最下位桁から乗算と桁上げ処理をし階乗の計算を実行する

次にこの計算式を上位桁セル K3,・・・，C3 へと左方向にコピーすることにより百億進数１０桁による１！の計算結果を得ることができる。以降は百億進数１０桁分のセルを下方向にコピーすることにより，ｎ！の値を求めることができる。階乗の多桁計算では乗算結果の桁上げだけを考慮すれば良いことから，意外なほどに簡単だったのではないだろうか。

3.2　１億進数を用いたネピア数の計算

自然対数の底でもあるネピア数を０からｎまで階乗分の１の総和の極限として計算する。

ここでは例として１７桁の１億進数を用い，階乗分の１と総和を交互に計算している。



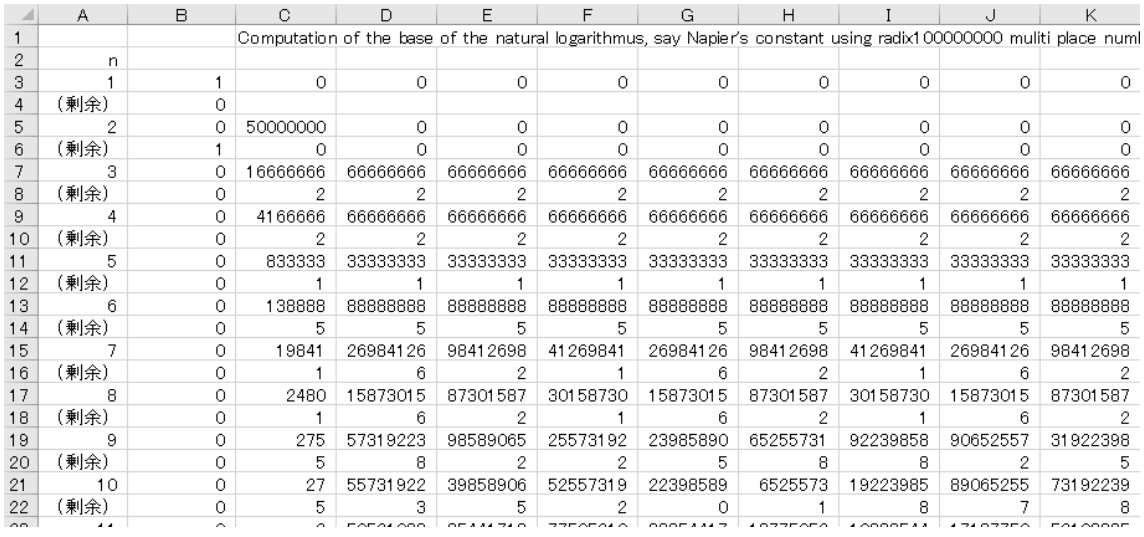
	A	B	C	D	E	F	G	H	I	J	K
1			Computation of the base of the natural logarithmus, say Napier's constant using radix100000000 muliti place number								
2	n										
3	1	1	0	0	0	0	0	0	0	0	0
4	(剰余)										
5	2										
6	(剰余)										
7	3										
8											
9	4										
10											
11	5										
12											
13	6										
14											
15	7										
16											
17	8										
18											
19	9										
20											
21	10										
22											

図３　１億進数で表記した数値１を対応するセルに記憶し除算の準備をする

ゼロの階乗分の１は１なので，ｎ＝１の階乗分の１から計算する。階乗分の１を求めたセルのひとつ下には計算をより簡単に進めて行けるよう剰余を記憶するようにする。初期値としてＡ欄にｎと３行目に１７桁の１億進数相当の１を設定しておく。n=2 の場合，階乗分の１を最上位桁から 5 行目のセルに求め，6 行目のセルにその際の桁ごとの剰余を記憶する。

=QUOTIENT(C3+100000000*B6,$A5)

=MOD(C3+100000000*B6,$A5)



	A	B	C	D	E	F	G	H	I	J	K
1			Computation of the base of the natural logarithmus, say Napier's constant using radix100000000 muliti place numl								
2	n										
3	1	1	0	0	0	0	0	0	0	0	0
4	(剰余)	0									
5	2	0	50000000	0	0	0	0	0	0	0	0
6	(剰余)	1	0	0	0	0	0	0	0	0	0
7	3	0	16666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666
8	(剰余)	0	2	2	2	2	2	2	2	2	2
9	4	0	4166666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666
10	(剰余)	0	2	2	2	2	2	2	2	2	2
11	5	0	833333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333
12	(剰余)	0	1	1	1	1	1	1	1	1	1
13	6	0	138888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888
14	(剰余)	0	5	5	5	5	5	5	5	5	5
15	7	0	19841	26984126	98412698	41269841	26984126	98412698	41269841	26984126	98412698
16	(剰余)	0	1	6	2	1	6	2	1	6	2
17	8	0	2480	15873015	87301587	30158730	15873015	87301587	30158730	15873015	87301587
18	(剰余)	0	1	6	2	1	6	2	1	6	2
19	9	0	275	57319223	98589065	25573192	23985890	65255731	92239858	90652557	31922398
20	(剰余)	0	5	8	2	2	5	8	8	2	5
21	10	0	27	55731922	39858906	52557319	22398589	6525573	19223985	89065255	73192239
22	(剰余)	0	5	3	5	2	0	1	8	7	8

図４　１億進数で表記した数値１を階乗による除算処理する

次にこの計算式を右方向にコピーし，１億進数１７桁分のセル２行を下方向に拡張するこれにより階乗分の１(n=1,2,…)が求まる。こうして得られた各項目を桁ごとに加算し最上位桁には１を加えておく。１億進数表示にした結果は，桁ごとの計算結果を最下位桁から

QUOTIENT 関数を用いて統合(unification)すればネピア数(e)を求めることができる。

133	66	0	0	0	0	0	0	0	0	0	0	0	0	1837
134		0	0	0	0	0	0	0	0	0	0	0	0	4
135	67	0	0	0	0	0	0	0	0	0	0	0	0	27
136		0	0	0	0	0	0	0	0	0	0	0	0	28
137	68	0	0	0	0	0	0	0	0	0	0	0	0	0
138		0	0	0	0	0	0	0	0	0	0	0	0	27
139	69	0	0	0	0	0	0	0	0	0	0	0	0	0
140		0	0	0	0	0	0	0	0	0	0	0	0	0
141	70	0	0	0	0	0	0	0	0	0	0	0	0	0
142		0	0	0	0	0	0	0	0	0	0	0	0	0
143	71	0	0	0	0	0	0	0	0	0	0	0	0	0
144		0	0	0	0	0	0	0	0	0	0	0	0	0
145	72	0	0	0	0	0	0	0	0	0	0	0	0	0
146		0	0	0	0	0	0	0	0	0	0	0	0	0
147	73	0	0	0	0	0	0	0	0	0	0	0	0	0
148		0	0	0	0	0	0	0	0	0	0	0	0	0
149	74	0	0	0	0	0	0	0	0	0	0	0	0	0
150		0	0	0	0	0	0	0	0	0	0	0	0	0
151	75	0	0	0	0	0	0	0	0	0	0	0	0	0
152		0	0	0	0	0	0	0	0	0	0	0	0	0
153	76	0	0	0	0	0	0	0	0	0	0	0	0	0
154		0	0	0	0	0	0	0	0	0	0	0	0	0
155	77	0	0	0	0	0	0	0	0	0	0	0	0	0
156		0	0	0	0	0	0	0	0	0	0	0	0	0
157	78	0	0	0	0	0	0	0	0	0	0	0	0	0
158		0	0	0	0	0	0	0	0	0	0	0	0	0
159	79	0	0	0	0	0	0	0	0	0	0	0	0	0
160		0	0	0	0	0	0	0	0	0	0	0	0	0
161	80	0	0	0	0	0	0	0	0	0	0	0	0	0
162		0	0	0	0	0	0	0	0	0	0	0	0	0
163	81	0	0	0	0	0	0	0	0	0	0	0	0	0
164		0	0	0	0	0	0	0	0	0	0	0	0	0
165														
166	summation	1	71828178	484590442	1035360274	1347135253	1324977552	2047093683	1695957475	2169676254	2324076601	2935354729	3045713792	2978525131
167	unification		71828182	84590452	1035360287	1347135266	1324977572	2047093699	1695957496	2169676277	2324076630	2935354759	3045713821	2978525166
168	result	e=2.	71828182	84590452	35360287	47135266	24977572	47093699	95957496	69676277	24076630	35354759	45713821	78525166
169		(integer)	(fraction)											
170														
171		自然対数の底(e=Σ1/k! = 2.7182818…)ネイピア数をここでは17桁の1億進数(radix100000000)で表した被除数に対する除算と商の加算により求めている(除数は1桁)												
172		多桁の1億進数で表した被除数1を1桁の除数1，2，3，・・・により順次除算し階乗分の1を計算し得られた商を桁毎に加算してから1億進数として整理する(以下参照)												
173														
174			summation では各桁の階乗分の1の総和を求めている											
175			unification では1億進数のルールに従って下の桁からの桁上がり分(1億を超える値・・・1億で除した商)を加えている											
176			result では桁上げ処理を済ませた後、桁上げ処理を済ませた1億を超える値を(1億のmodを計算することで)取り除き1億進数として整理する											

図５　階乗による除算の総和を求め１億進数として整理する

n=81 までの階乗による除算結果を見ると，１億進数１５桁目までゼロであることから，この計算例により得られたネピア数，は十進数で小数点以下 120 桁程度の精度になっているものと考えられる。ここで示したような多桁計算により，例えば，良く知られているマチンの式を用いると，高精度な円周率の値を求めることもできる。

$$\pi/4 = 4\arctan(1/5) - \arctan(1/239)$$　　マチンの式

この場合，以下の漸化式を多桁計算することで，百万桁精度の円周率が得られる。

$$\arctan(1/5) = \sum_{n=0}^{715372} (-1)^n \, (1/5)^{2n+1}/(2n+1) \quad ,$$

$$\arctan(\frac{1}{239}) = \sum_{n=0}^{210236} (-1)^n \, (1/239)^{2n+1}/(2n+1)$$

以上，進数と補数についての解説と，高精度な計算結果を得るための多桁計算の原理について簡単な紹介をしてきた。なお円周率の計算に特化した高速かつ高精度なアルゴリズムとしては，高速離散フーリエ変換を応用したものが良く知られていることを注記しておく。