数理解析研究所講究録
第1677巻 2010年 253-258

パスカルの3角形と和算

神谷徳昭 (Noriaki Kamiya)　鈴木大郎 (Taro Suzuki)
公立大学法人会津大学 (University of Aizu)

この話はパスカルの三角形と現在呼ばれているものがいたるところに出てきますという話です。特に和算の中にも存在しましたという話題提供と、計算機による計算結果についてお話しします。

— 関連する分野 —
高校数学, 数学の歴史, 日本の和算 (関孝和、松永良弼), 計算機アルゴリズム

数学のみを教育するのではなくいろいろな知識を教育する手段として数学を用いたいと考えます。またアルゴリズムを計算する箇所もありますので、数理科学のコンピュータ教育を学ぶ学生にも役立つと思います。数学史、アルゴリズム、教育の観点から少し述べさせていただきます。

パスカルの三角形は、2項定理との関連でよく知られています。下図に示したように、三角形の上から $n+1$ 段目の数列が $(a+b)^n$ の係数の列になっています。

$$(a+b)^n = \sum_{k=1}^{n} {}_nC_k a^k b^{n-k}$$

						1							
$a+b$ の係数					1		1						
$(a+b)^2$ の係数				1		2		1					
$(a+b)^3$ の係数			1		3		3		1				
$(a+b)^4$ の係数		1		4		6		4		1			
$(a+b)^5$ の係数	1		5		10		10		5		1		
$(a+b)^6$ の係数		1		6		15		20		15		6		1
$(a+b)^7$ の係数	1		7		21		35		35		21		7		1
	↙		↙		↙		↙		↙		↙		↙	
	$(1-x)^{-1}$	$(1-x)^{-2}$		$(1-x)^{-3}$		$(1-x)^{-4}$		$(1-x)^{-5}$		$(1-x)^{-6}$		$(1-x)^{-7}$	

一方、右上から左下へと続く数列を見ると、左上から n 個目の数列は $(1-x)^{-n}$ を母関数とするべき級数の係数の列になっています。

$$\frac{1}{(1-x)^{n+1}} = \sum_{i=n}^{\infty} {}_iC_n\, x^{i-n} \quad (n \geq 0)$$

江戸時代の和算家である松永良弼は、$(1-x)^{-(p+1)}$ $(p \geq 1)$ を展開して得られるべき級数を利用して、べき級数 $1^p + 2^p x + 3^p x^2 + 4^p x^3 + \cdots$ の母関数を求める方法について考察しています。そのことに関する解説を参考文献 [1] から引用します (わかりやすさのために、原文で使っている和算固有の用語を言い換えている箇所があります)。

帰除得商 (和算 松永良弼)

$\frac{1}{(1-x)^i}$ $(i = 1, 2, \ldots)$ を巾級数に展開したときの係数が作る数列を利用して, 特殊な級数の総和法を考察している. 即ち $\frac{1}{(1-x)^i}$ を巾級数に展開すると, 係数が作る数列は次のようになる.

$i = 1$ のとき $1, 1, 1, \ldots$
$i = 2$ のとき $1, 2, 3, 4, \ldots$ (底子)
$i = 3$ のとき $1, 3, 6, 10, \ldots$ (圭垛)
$i = 4$ のとき $1, 4, 10, 20, \ldots$ (三角衰垛)

であり, 以下, 再乗衰垛, 三乗衰垛, 四乗衰垛,... と続く. これらの数列を基にすると,

$$\begin{aligned} \frac{1}{(1-x)^2} &= 1 + 2^1 x + 3^1 x^2 + 4^1 x^3 + \cdots \\ \frac{1}{(1-x)^3} + \frac{x}{(1-x)^3} &= 1 + 2^2 x + 3^2 x^2 + 4^2 x^3 + \cdots \\ \frac{1}{(1-x)^4} + \frac{4x}{(1-x)^4} + \frac{x^2}{(1-x)^4} &= 1 + 2^3 x + 3^3 x^2 + 4^3 x^3 + \cdots \end{aligned}$$

とかける. 一般には,

$$\frac{A_0 x^0}{(1-x)^{p+1}} + \frac{A_1 x}{(1-x)^{p+1}} + \frac{A_2 x^2}{(1-x)^{p+1}} + \cdots + \frac{A_{p-1} x^{p-1}}{(1-x)^{p+1}} = 1 + 2^p x + 3^p x^2 + \cdots$$

とおくとき, p と A_i との間に次の表のような関係が成立する.

p	A_0	A_1	A_2	A_3	A_4	$\cdots$	右辺の係数が作る数列
2	1	1					平方垛
3	1	4	1				立方垛
4	1	11	11	1			三乗垛
5	1	26	66	26	1		四乗垛
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

このような公式はすでに古い時代から 得られているかもしれませんが和算の中にもあります.

松永良弼はもと寺内平八郎良弼と言い, 後, 寺内姓を松永と改めました。権平、安右衛門などと称し、号には東岡、龍池、葆真斎、探玄子、東溟、源翼などを用いました。関孝和から始まる関流和算の確立に貢献した人物であり、円周率を少数第 51 位までを算出 (第 49 位までは合致) する方法を考案したことでも知られています。延享元年 (西暦 1744 年) 6 月 23 日に江戸で没したと言われています。

西洋と江戸時代では次の人たちの時代です。

テイラー (1685–1731)、マクローリン (1698–1746)、オイラー (1707–1783)
1674(延宝２年) 関孝和 (発微算法)、 1716(享保元年) 吉宗の改革始まる

松永氏の結果を補足, 発展させますと

$$\frac{A_0x^0}{(1-x)^{p+1}}+\frac{A_1x}{(1-x)^{p+1}}+\frac{A_2x^2}{(1-x)^{p+1}}+\cdots+\frac{A_{p-1}x^{p-1}}{(1-x)^{p+1}}=1+2^px+3^px^2+\cdots$$

の式において $p=6$ とおくと、$(1-x)^{-7}=1+7x+28x^2+84x^3+210x^4+462x^5+\cdots$ の式より

$$\frac{1}{(1-x)^7}+\frac{57x}{(1-x)^7}+\frac{302x^2}{(1-x)^7}+\frac{302x^3}{(1-x)^7}+\frac{57x^4}{(1-x)^7}+\frac{x^5}{(1-x)^7}=1+2^6x+3^6x^2+4^6x^3+\cdots$$

が得られます。

また、上の式において $p=7$ とおくと、$(1-x)^{-8}=1+8x+36x^2+120x^3+330x^4+792x^5+1716x^6+\cdots$ の式より

$$\frac{1}{(1-x)^8}+\frac{120x}{(1-x)^8}+\frac{1191x^2}{(1-x)^8}+\frac{2416x^3}{(1-x)^8}+\frac{1191x^4}{(1-x)^8}+\frac{120x^5}{(1-x)^8}+\frac{1x}{(1-x)^8}$$

$$=\quad 1+2^7x+3^7x^2+4^7x^3+5^7x^4+\cdots$$

が得られます。

p の値をさらに大きくすると手計算ではたいへんなので、係数の列を計算するアルゴリズムを利用したプログラムを作りました。これらの係数は、Haskell というプログラミング言語を用いて計算しました。付録にプログラムを示します。計算機を用いて $p=8$ から $p=20$ までを求めた結果の表は以下の様になります。

$p=8$: 1, 247, 4293, 15619, 15619, 4293, 247, 1

$p=9$: 1, 502, 14608, 88234, 156190, 88234, 14608, 502, 1

$p=10$: 1, 1013, 47840, 455192, 1310354, 1310354, 455192, 47840, 1013, 1

$p=11$: 1, 2036, 152637, 2203488, 9738114, 15724248, 9738114, 2203488, 152637, 2036, 1

$p=12$: 1, 4083, 478271, 10187685, 66318474, 162512286, 162512286, 66318474, 10187685, 478271, 4083, 1

$p=13$: 1, 8178, 1479726, 45533450, 423281535, 1505621508, 2275172004, 1505621508, 423281535, 45533450, 1479726, 8178, 1

$p=14$: 1, 16369, 4537314, 198410786, 2571742175, 12843262863, 27971176092, 27971176092, 12843262863, 2571742175, 198410786, 4537314, 16369, 1

$p=15$: 1, 32752, 13824739, 848090912, 15041229521, 102776998928, 311387598411, 447538817472, 311387598411, 102776998928, 15041229521, 848090912, 13824739, 32752, 1

$p=16$: 1, 65519, 41932745, 3572085255, 85383238549, 782115518299, 3207483178157, 6382798925475, 6382798925475, 3207483178157, 782115518299, 85383238549, 3572085255, 41932745, 65519, 1

$p=17$: 1, 131054, 126781020, 14875399450, 473353301060, 5717291972382, 31055652948388, 83137223185370, 114890380658550, 83137223185370, 31055652948388, 5717291972382, 473353301060, 14875399450, 126781020, 131054, 1

$p=18$: 1, 262125, 382439924, 61403313100, 2575022097600, 40457344748072, 285997074307300, 1006709967915228, 1865385657780650, 1865385657780650, 1006709967915228, 285997074307300, 40457344748072, 2575022097600, 61403313100, 382439924, 262125, 1

$p=19$: 1, 524268, 1151775897, 251732291184, 13796160184500, 278794377854832, 2527925001876036, 11485644635009424, 27862280567093358, 37307713155613000, 27862280567093358, 11485644635009424, 2527925001876036, 278794377854832, 13796160184500, 251732291184, 1151775897, 524268, 1

256

$p = 20$: 1, 1048555, 3464764515, 1026509354985, 73008517581444, 1879708669896492, 21598596303099900, 124748182104463860, 388588260723953310, 679562217794156938, 679562217794156938, 388588260723953310, 124748182104463860, 21598596303099900, 1879708669896492, 73008517581444, 1026509354985, 3464764515, 1048555, 1

p が増えるにつれて、急速に係数が大きくなっていくのがわかります。また，係数列 $A_0, A_1, \ldots, A_{p-1}$ は対称、すなわち $0 \leq i \leq \lfloor (p-1)/2 \rfloor$ をみたす任意の i について $A_i = A_{p-i-1}$ となっていることがわかります。

最後に、$p = 100$ のときの最初の 10 個の係数 $A_0 \sim A_9$ を示します。

A_0 : 1,

A_1 : 126765060022822940149670320 5275,

A_2 : 515377520732011203003750506714451721535083784075,

A_3 : 160693804420693714594802895430811555956843372279823116256 1425,

A_4 : 78886088899093755865619243027863479807547976736296853433607667554792 24,

A_5 : 65331782675056474791189022332543454184193492190167780995769976161231163185 2456,

A_6 : 32344105244836219596289968744828696579088764165630728396702235487202770702414227 50600,

A_7 : 20367092975062717197499859649643254849715739148636494923684379721764713884744979 37125232600,

A_8 : 26540826457626248433269837748243195063629116117580136893814757276310090737549735 9328820538770500,

A_9 : 99731832736161942945728688300737467829635974446029797343503599604425666285939543 17781079447309908844

あとがき

和算の中にどのような形であれパスカルの三角形と関連する研究がなされていたのは興味あることです。そしてたくさんの項の係数を求めるのには現代の計算機科学の方法が適していると考えます。最後に、パスカルが計算機の原理を最初に考えた一人と言われているのは、このようなアルゴリズム的な計算を求めたかったのではないかと 300 年後の我々は推察します。

— パスカルと松永の精神を現代化する —

謝辞 竹之内脩、藤井康生先生を始め、研究会の折には諸先生方から貴重な御助言を頂きました。ここに感謝申し上げます。

参考文献

1] 松永良弼 全集, 1987. 東京法令出版株式会社

2] 神谷徳昭 Report of Study for Mathematical Education, Lecture note of Aizu Univ, 2009.

3] W.Dunham, Journey through genius, 1990, John Wiley.

4] S.Peyton Jones(eds.), Haskell 98 Language and Libraries: The Revised Report, 2003, Cambridge University Press. (PDF 版が http://haskell.org/definition/haskell98-report.pdf から取得可能)

257

付録

ここでは、プログラミング言語 Haskell により記述した、上記の係数列を求める計算アルゴリズムについて述べます。

$$\frac{A_0x^0}{(1-x)^{p+1}}+\frac{A_1x}{(1-x)^{p+1}}+\frac{A_2x^2}{(1-x)^{p+1}}+\cdots+\frac{A_{p-1}x^{p-1}}{(1-x)^{p+1}}=1+2^px+3^px^2+\cdots$$

と

$$\frac{1}{(1-x)^{p+1}}={}_pC_p+{}_{p+1}C_px+{}_{p+2}C_px^2+\cdots+{}_{p+i}C_px^i+\cdots$$

より、

$$\begin{array}{llllllll}
{}_pC_pA_0 & +\,{}_{p+1}C_pA_0x & +\,{}_{p+2}C_pA_0x^2 & +\cdots+ & {}_{p+i}C_pA_0x^i & +\cdots+ & {}_{2p-1}C_pA_0x^{p-1} & +\cdots \\
 & {}_pC_pA_1x & +\,{}_{p+1}C_pA_1x^2 & +\cdots+ & {}_{p+i-1}C_pA_1x^i & +\cdots+ & {}_{2p-2}C_pA_1x^{p-1} & +\cdots \\
 & & {}_pC_pA_2x^2 & +\cdots+ & {}_{p+i-2}C_pA_2x^i & +\cdots+ & {}_{2p-3}C_pA_2x^{p-1} & +\cdots \\
 & & & \ddots & \vdots & & \vdots & \\
 & & & & {}_pC_pA_ix^i & +\cdots+ & {}_{2p-i-1}C_pA_ix^{p-1} & +\cdots \\
 & & & & & \ddots & \vdots & \\
+) & & & & & & {}_pC_pA_{p-1}x^{p-1} & +\cdots \\
\hline
1^p & +\ 2^px & +\ 3^px^2 & +\cdots+ & (i+1)^px^i & +\cdots & p^px^{p-1} & +\cdots
\end{array}$$

が得られます。上式から x の係数列を取り出し、${}_pC_p=1$ を用いると、

$$\begin{pmatrix} A_0\\A_1\\A_2\\A_3\\\vdots\\A_{p-1}\\0\\0\\\vdots \end{pmatrix}=\begin{pmatrix} 1^p\\2^p\\3^p\\4^p\\\vdots\\p^p\\(p+1)^p\\(p+2)^p\\\vdots \end{pmatrix}-\left[A_0\begin{pmatrix} 0\\{}_{p+1}C_p\\{}_{p+2}C_p\\{}_{p+3}C_p\\\vdots\\{}_{2p-1}C_p\\{}_{2p}C_p\\{}_{2p+1}C_p\\\vdots \end{pmatrix}+A_1\begin{pmatrix} 0\\0\\{}_{p+1}C_p\\{}_{p+2}C_p\\\vdots\\{}_{2p-2}C_p\\{}_{2p-1}C_p\\{}_{2p}C_p\\\vdots \end{pmatrix}+A_2\begin{pmatrix} 0\\0\\0\\{}_{p+1}C_p\\\vdots\\{}_{2p-3}C_p\\{}_{2p-2}C_p\\{}_{2p-1}C_p\\\vdots \end{pmatrix}+\cdots+A_{p-2}\begin{pmatrix} 0\\0\\0\\0\\\vdots\\{}_{p+1}C_p\\{}_{p+2}C_p\\{}_{p+3}C_p\\\vdots \end{pmatrix}+\cdots\right]$$

が得られます。ただし、列 $X=(x_1,x_2,\cdots)$, $Y=(y_1,y_2,\cdots)$ と数 a について、$X+Y=(x_1+y_1,x_2+y_2,\ldots)$、$X-Y=(x_1-y_1,x_2-y_2,\ldots)$ (列の加減算)、および $aX=(ax_1,ax_2,\ldots)$ (列のスカラー倍) とします。

上式に基づき、p を与えられたときに係数列 $A_0,\ldots,A_{p-1}$ を求める Haskell のプログラムは以下のようになります。

```
factors p = take p facs
  where facs = 1 : subSequences powers (sums facs)
        powers = [ n^p | n <- [2..] ]
        series = [ binomial n p | n <- [(p+1)..] ]
        sums (a:as) = s : addSequences ss (sums as)
          where (s:ss) = multiplyScalarSequence a series
```

ここで、binomial n k は ${}_nC_k$ を、addSequences、subSequences、multiplyScalarSequence は列の加減算とスカラー倍を、facs、powers、series はそれぞれ無限数列 $A_0,A_1,\ldots,A_{p-1},0,\ldots$ と $2^p,3^p,4^p,\ldots$、および ${}_{p+1}C_p,{}_{p+2}C_p,{}_{p+3}C_p,\ldots$ を表します。sums facs は上式の [] の中の列から先頭の要素 (必ず 0) を除いたものに対応します。その他、: は $x_1:(x_2,x_3,\ldots)=(x_1,x_2,x_3,\ldots)$ をみたす演算子、take p facs は無限数列 facs の最初の p 個の要素を値とする式です。

258

このように、Haskell は無限級数の係数列のような無限のデータを直接表現することができるので、上式のような無限に続く列を含む問題をほぼそのまま記述できるという利点があります。

入出力と補助関数 binomial、addSequences、subSequences、multiplyScalarSequence の定義を加えたプログラム全体は以下のようになります。

```
module Main(main,factors) where

import System

main = do p <- getFirstArg
          printList (factors p)
  where getFirstArg = do (s:_) <- getArgs
                         return (read s)
        printList [] = return ()
        printList (x:xs) = do print x
                              printList xs

factors :: Int -> [Integer]
factors = ...

binomial :: Int -> Int -> Integer
binomial m 0 = 1
binomial 0 n = 0
binomial m n =
  ((binomial (m-1) (n-1)) * (fromIntegral m)) `div` (fromIntegral n)

addSequences = zipWith (+)
subSequences = zipWith (-)
multiplyScalarSequence x = map (x*)
```