πの近似分数について

——MS-DOSとN88-BASICとのベンチマークテスト——

出雲敏彦

“コンピュータは，ただの数を処理するだけの機械にすぎない”という一般の人たちの見方が，いかに狭い考えであるかを示すことにある。より正確にいえば，コンピュータは，シンボルを処理する機械なのである。数値計算と数学の違いについて数学者が注目する最初の相違点は，おそらく，“コンピュータは有限の機械であるが，数学は無限を常用する”ということであろう〔15；pp. 1, 7〕。

1．問題の設定
2．モデルの環境
3．アルゴリズム
4．倍精度πの近似分数
5．添付資料

1．問題の設定

コンピュータによる計算を設計するときに，数学で学ぶアイデアを注意して応用しないと，数学者が得た結果とコンピュータが出した結果との間に，非常なくいちがいが発生してしまうことになる。演算をある一定の有限桁に限定して行なうために生じる“丸めの誤差（roundoff errors）”の影響，無限プロセスを有限表現するために生じる公式誤差とも呼ばれる“打切り誤差（truncation errors）”の影響が実際の計算回数と共に大きくなり，真値と計算値とが乖離することになる。

ここでは，BASIC倍精度に伴う誤差を含む計算値と真値の関係を検証する目的で，円周率πを計算する近似分数プログラムによって，近似の度数分布および処理時間を測定している。コンピュータによる問題解決の手順は，事務計算ではプログラム設計の目的と位置付けを明確にするためにフリーマン（Freeman），メッジャー（Metzger）およびボエム（Boehm）のソフトウェアのライフサイクル〔7；p. 144〕が中心となる。例えば，フリーマン

のモデルは，①要求分析（needs analysis），②仕様決定（specification），③構造設計（architectural design），④詳細設計（detail design），⑤作成（implementation），⑥メンテナンス（maintenance）の各段階に従いシステム開発が行なわれる。このシステム開発の標準化は，ソフトウェアの品質管理，開発過程の作業分担，開発期間の短縮および人の最適配分に役立ち，システムの信頼性，効率の良さ，融通性の高さおよび一般性を保証することになる。

一方，数値計算の問題解法の手順は，①問題の内容を明確にして目的を定義する，②数学的に記述する，③数値解析を行なう，④コンピュータのプログラムを作成する，⑤プログラムをチェックする，⑥計算を実施する，⑦計算結果を解明する，と考えて差し支えない。ここでの命題は限りなく真値に近付けることである。計算誤差の解析は，計算の基礎をなすものであって，一般に手作業で行なうか又はコンピュータを使用して行なっている。入力データは実験又は推定の結果であるから，正確であるが，その計算には各種の誤差が生じている。本稿では，問題の内容を明確にする目的でまず，モデルの環境としてハードウェア，ソフトウェア，システム・フローおよびπの沿革がとり挙げてある。コンピュータの計算能力の興味からも，超越数πの計算はスーパーコンピュータによっていまも，次々に記録が書き換えられている。つぎに，モデルをマーチンの公式によって数学的に記述し，誤差を伴う数値解析にはレジスターの識別を意識したアルゴリズム（算法）が示してある。そして最後に，πの接近法の一つとして，近似分数を作成するプログラムは3700年に渡る有理数の歴史でもあり，その実行結果の分析をしている。その際，N_{88}-BASIC(MS-DOS V.3.1 版）とN_{88}-BASIC(86)とのOS(operating system）については，処理時間を比較する目的でベンチマークテストを行なっている。このπは連分数をつぎつぎに打ち切ることによって得られる分数，有理数の形にして近似分数（convergent）を検索する目的で倍精度の有効桁数16桁まで処理してある。無理数を有理数の形にしたこの近似分数の列は，かなり速く収束することが解かっている。

添付資料は，πの近似分数作成プログラムと1761年πが無理数だということを証明したランベルト（Lambert,J.H.(1728－1777)）のπの逆近似分数をパソコンでシミュレーションした近似分数である〔17；pp. 140－145，187－191〕。

2．モデルの環境

2．1．ハードウェア（PC9801VM2）

CPU（central processing unit）の機能は

1）μ PD70116

2）16ビット　マイクロプロセッサ

3）クロック　10MHz（9.83MHz）

基本クロックサイクル　101.7ns

4）プロセッサバスサイクル　4クロックサイクル

である。そして，主記憶装置はROM(read only memory unit)にはN88-BASIC(86)およびモニタ96Kバイトが内蔵されており，RAM(random access memory unit)の実装容量は384Kバイトである。ここでの処理結果は，フレキシブル・ディスク（通称フロッピ・ディスク）にランダムファイル形式で保存してある。ディスクの仕様は，ディスクインターフェイスとしてμPD765A相当のものであり，5インチ1MBを利用した。

また，ベンチマーク（bench mark）プログラムは，カレンダー時計（μPD1990C相当）とタイマ（PD8253C相当，カウントレート406.9ns／2.4576MHz）とを使用して，テストを行なってある。

2．2．ソフトウェア

ソフトウェアは，要約統計量を加工する目的で，3つの応用プログラムと，ユーティリティプログラムを2つ使用している。

1）πの近似分数作成プログラム〔添付資料1〕	FRACT1．PAI
2）πの近似分数編集印刷プログラム	FRACT4．TIM
3）MS-DOSとN88BASICとのベンチマーク出力プログラム	FRACTLPT．EXE
4）N88／MS-DOSファイルコンバートプログラム〔11〕	FILECONV
5）プログラムリスト編集印刷プログラム	pgmprt．n88

以上のプログラムのうち，2・3・5と精度チェックプログラムは，掲載を省略してある。ただし，FILECONVプログラムはメーカのユーティリティ・プログラムである。

2．3．システム・フロー

本モデルのシステム・フローは，上述2．2．のプログラムを図1のような手順で処理したものである。なお、MS－DOSのプログラムおよびデータはFILECONVプログラムによってコンバートしてあり，添付資料1の掲載プログラムは，pgmprt．n88によって出力したものである。

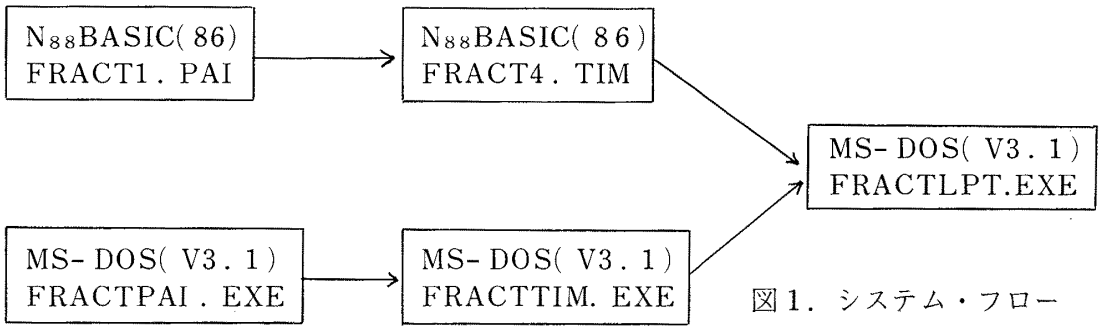

図1．システム・フロー

2．4．π計算の沿革

円周率πとは，Euclid平面上の円周の長さと直径との比，すなわち，$2\int_0^1 dx/SQR(1-x\hat{}2)$の値をいう。その近似値としては，3が古くから用いられている。πの歴史は，人類の

		3.14157<π<3.1416に近似し，有効桁数は5桁確保できる。
紀　元		
500年	祖沖之	3.1415926<π<3.1415927を確定。また，$\pi \fallingdotseq 355/113$=3.141592を求める。既に，有効桁数は7桁を確保している。ローマ帝国の崩壊(476)は，教養ある異邦人の破局であり，それ以後πの暗黒時代でもあった。
1573年	Adriaen, A.	$\pi \fallingdotseq 355/113$=3.141592を見出す。小数点以下6桁まで正しい。πの近似分数中の傑作である。
1596年	Ludolph van Ceulen	π(ルドルフ数)を小数点以下32桁計算。後に35桁まで計算。
1665年	Newton,I	二項定理を使い，πを少なくとも小数点以下16桁計算。ただし，その著書は，ニュートン（1642－1727）の没後1737年に英語訳が出版された。『Method of Fluxions and Infinite Series』, Scotland.
1674年	Leibniz, G.W.	πをarctangentの無限級数(infinite series)で表現する。この級数は収束緩慢で，πの計算には不適当。
1706年	Machin, J.	πを100桁計算。πをarctangentの無限級数で展開し，その等式採用。
1706年	Jones, W.	円周率を記号πで表わす。円周(periphery)の略語と考えられる。
1761年	Lambert, J. H	πが無理数（transfinite number）であることを証明。またπに関する連分数を提示する。
1775年	Euler, L.	πが超越数（transcendental number)であることを示唆。また，連分数を導くことによって，πが無理数であることの研究の基礎を築いた。Lindemann, C. L. F.（1882）はπが超越数であることを証明。
1873年	Shanks, W.	マーチンの公式を用いてπを707桁計算。
1946年	Ferguson, D.F.	シャンクスの値には小数点以下528桁目に誤りがあったことを発見。
1947年	Ferguson, D.F.	卓上計算機で808桁まで計算〔20 ; pp. 186－187〕。
1949年	ENIAC	初めてコンピュータで計算。πをマーチンの公式で2,037桁まで計算。処理には70時間を要し，今日のコンピュータ

		とは比較にならないほど遅い（フォン・ノイマンが処理)。
1986年	マイクロ・コンピュータ	筆者は２．１．ハードウェア（ＰＣ9801ＶＭ２）の環境でマーチンの公式によってN88-BASIC〔16；P.44〕とMS-DOSのTURBO-Pascal〔6；pp.64－66〕との処理をしている。Borland International 社のTURBO-Pascalは，主記憶が64Kbyteの制限があるため１万桁の計算はできなかったが,2,000桁の計算は僅か８分（マシン語で20秒，未検証)，8,000桁の計算は２時間15分で有効な結果が得られている（1986年，検証済)。この値は1958年，パリ・データ処理センターのIBM 704を使った処理能力とほぼ一致している。 BASIC言語については、N88-BASICを通常のインタープリタ形式で利用するケースに対してMS-DOS配下でコンパイル後の実行形式を使用するケースは，処理時間が半分である。TURBO- Pascalに比べると効率は落ちるが実行形式のN88-BASIC(MS-DOS版）は，2,000桁で58分であり，10,000桁では22時間30分であった(1986年，検証済)。 インタープリタ形式のN88-BASIC〔16；p.44〕には，1,000桁で８時間と記述してある。この２年間，パソコンの進歩は目覚ましいものがある。
1954年	NORC	小数点以下 3,089桁計算できるプログラム作成。
1961年	IBM7090	Shanks，D.とWrench，J.W.はπの計算プログラムを改良し，100,000桁計算。'Calculation to 100,000 decimals of π'，『Mathematical Computer V. 16』。ガウスの公式採用。Shanks，D.はShanks，W.とは別人。ガウスの公式は演算速度が早い。
1967年	CDC 6600	500,000桁計算。
1985年	ＮＥＣ	16,777,216桁まで30時間で計算（東京大学；金田康正，田村良明)。
1986年	CRAY-２	29,360,111桁まで27時間で算出（NASA/AMES研究所の２号機；Bailey)。
1987年	NEC SX-２	133,554,000桁まで計算機稼働時間は37時間（うちＣＰＵ使用時間は35時間15分）で画期的な計算をしている（東京大学；金田康正)。使用した日本電気のスーパーコ

表２．分母の桁数と近似分数出現の度数分布

頻度 / 分母の桁数	件数	有効桁数		稼働時間	備考
		最大	内訳：桁数（件数）	HH: MM: SS	
10^0	5	3	1(2), 2(2), 3(1)	00：00：02	
10^1	7	4	3(1), 4(6)	00：00：08	
10^2	2	7	5(1), 7(1)	00：00：09	10秒
10^3	0				
10^4	150	11	7(67), 8(70), 9(9), 10(3), 11(1)	00：20：49	20分
10^5	3	12	12(3)	03：12：11	
10^6	2	14	13(1), 14(1)	05：30：53	6時間
10^7	8	16	14(3), 15(4), 16(1)	81：56：40	3.5日
近似分数出現計	177	——	精度16桁（漸近件数 177件）	81：56：40	

注）本表の各数値は，添付資料３に基づいて加工したものである。又、内訳欄はテューキーの幹葉（stem-and-leaf）表示を意図したものである〔18；pp. 1－26〕。

この表には次のような特徴が読み取れる。近似分数はまず，無作為な頻度で，大半は分母の桁数が５桁のときに出現し，４桁のときは見つからなかった。そして，分母の桁数が３桁で有効桁数７桁の近似値をもつ355／113という分数の発見以後，約1000年に及ぶπの暗黒時代を迎えることになるが，有効桁数８桁の近似値をもつ82件目の74,948／24,175までには，分母の値が 113から24,175と大きな開きのあることと無関係ではないように考えられる。このギャップを乗り越えるには，πが有理数ではなく無理数であるという別のアプローチが必要であった。

そして表２には，本稿のモデル環境という制約条件の下で，計算可能性がもう１つのテーマとしてある。具体的には，稼働時間がとり挙げてあり，ハードウェアの性能とソフトウェアのアルゴリズムとの吟味をすべきである。ＯＳに関しては，添付資料２に掲載してあるようにN$_{88}$-BASIC(86)で処理したとしても，有効桁数16桁の近似分数を検索するには，82時間で収束したので計算は可能であった。そこでアルゴリズムの変更は加えず，サブルーチン・コールはそのまま使用した。倍精度のレジスターは，有効桁数17桁以上の近似分数を検索するにはハードウェアの制約条件に直面することになり，ソフトウェアのアルゴリズムで工夫しなければならない。本モデルでは，有効桁数17桁を有する近似分数は，（４．１）式が妥当性のあるものであれば，１カ月間のうちに検索できると考えてよい。ただし，この稼働時間は計算不可能と識別すべきである。いま，コンパイラー形式のN$_{88}$-BASIC(MS-DOS版)を使用しても，15日間はコンピュータを専有することになるだろう。

N$_{88}$-BASIC(86)とN$_{88}$-BASIC(MS-DOS版）との比較は，ＯＳの相違のことで添付資料２によって出力してある。本稿は今後，BASIC言語教育において，どちらのOSを選択すべきかという別の課題を意識している。最後に，その分析結果を纏めておく。一般的に，この比較はベンチ・マークテストにより行なわれる。ベンチ・マークテストとは，マイクロプロ

ンピュータ（NEC SX-2）は，主記憶256Mバイトと拡張記憶2Gバイトを備えている。プログラミングは言語FORTRANであった。

以上がごく最近までの，π発見の動向である。πの計算はコンピュータの利用と共に，桁数を1桁上げるために指数関数的負荷をメモリと処理時間に与えられることになるが，そのことはコンピュータの性能とプログラミング能力に置き替り，コスト・パフォーマンスの問題になってしまっている。今日社会問題に成ってしまったテクノストレス（"TECHNOSTRESS, 1984"；Brod, Craig）の1つであるテクノ依存症（technocentered）は，πの沿革を見るかぎりコンピュータの過剰適応という症候群が病理現象として現出しても当然と言える。その結果，限りないπ演算の競争が行なわれ，irrationalな問題が生み出したテクノ不安症（technoanxious）に似た途を歩んでいるようにも見える。

3．アルゴリズム

コンピュータの使い方の多くの部分は，課題となっている処理をどのように実行させるかという手順を正確に記述した詳しいアルゴリズム（algorithm）を作ることにある。ある処理のアルゴリズムを作ることがなぜ重要かというと，「ある問題のアルゴリズムを作ることができないということは，これから処理しようとしていることを完全に理解しているかどうか疑わしいということ」と言えるからである。

広く信じられているように，1つの実際問題のアルゴリズムを組織的に作るということは，多くの個々の特別な問題を解くより以上の知的努力が必要である。アルゴリズムの細部を検討するときには，このアルゴリズムを適用しようとする広い範囲の問題について，出現可能なすべてのケースをチェックしなければならないからである。また仮に，効果的なアルゴリズムを仕上げたならば，さらに，別な処理の相対的な効果も調べてみなければならない。要約すれば，アルゴリズムによるアプローチは，システム・フローを理解するための重要な方法であり，問題全体として理解できるようになる〔15；pp. 63－84〕。

ところで，数値計算におけるアルゴリズムは数学的に記述することになる。パソコンの数値計算はハード的には10進法約1桁，汎用コンピュータより精度が良い。商業用に作られた大型コンピュータは，10^±78のように，大きな数を処理するために，16進機になっている。その結果，レジスターは4ビット毎に区切られているので精度が悪くなることがある。数学の世界には数というものは一種類であるが，コンピュータでは整数（integer）と実数（real number）とがある。実数は数学で言う虚数（imaginary number）に対するものではなくて，仮数（mantissa）と指数（exponent）と呼ばれる2つの数を組み合わせた浮動小数点数（floating point number）のことである。また，整数は固定小数点数（fixed point number）を利用することもある。ここで扱う倍精度の実数は，BASICでは仮数部が10進数で16桁を

保証している。この精度を確保するためにプログラミングに際して，計算の手順（アルゴリズム）は重要であり，丸め誤差の影響を少なくするために特別の注意がはらわれ，試行錯誤と共にシステム思考が要求される。換言すると，コンピュータでは，すべて数を有限桁の記数法で表わし，そのうえ，多くの場合に浮動小数点法を用いるので，数学的モデルを使ったときの理想的に正確な数とコンピュータで実際に使われる数との間には，丸めの操作による誤差が生じる。この誤差を減らすためには，実際の場面で役立つ式の変形，級数展開の方法が用いられる。

N88-BASIC（86）の組込関数は単精度（有効桁数7桁，表示桁数6桁）となっている。初めに，モンテカルロ・シミュレーションを用いて，乱数の組込関数をチェックしてみることにする。円の面積は，xとyをともに0から1まで変化させたとき，組込関数の乱数が確率的に一様に分布していれば，x^2＋y^2≦1を4倍すれば求められる。この条件式に該当する件数と乱数の発生回数との比を4倍すれば，円周率となる。N88-BASICのRND関数で計算すると，円周率は乱数の偏りのために10万回発生させて3.14204の値であり，400万回乱数を発生（MS-DOSのN88-BASIC（86）コンパイラーで2時間）させても3.14339の値となって，むしろ悪い結果となってしまう。この場合は，丸めの誤差の影響ではなく，組込関数の偏りあるいは循環の影響によるものである。なお，処理プログラムと処理結果はここに添付することを省いてある。また，ビュッフォン（Buffon, G. L., 1707－1788）の針の問題の場合，πの計算値は与えたπの数値定数よりも当然悪く，組込関数の偏りと丸めの誤差の影響とのために近似も収束もしなかった。条件式は，x＜0.5＊sinϕであり，このときに限り針と線は交わる。そして公式，$\pi = 2 * L/(d * p)$によって，πを計算する。ただし，L：針の長さ，x：線と針の中心の距離，ϕ：線に対する傾き，d：Lと線の間隔，P：交叉する確率である。10万回（15分）針を投げても3.1506の値であった。反復回数は，多くしても結果は変わらないと考えられる。

つぎに，円周率πについての関数ATN(arctangent) 16桁を（3．1）式のように計算した結果は，帰納的に調べてみると，

$$4\#*\mathrm{ATN}(1) = \underline{3.141592741012573}\# \qquad (3．1)$$

となっており，アンダーラインの7桁が有効桁数となっている。そこで，組込関数を用いた反復計算処理は丸めの誤差と打切り誤差の影響を受けることになる。

この有効桁数は5世紀，宋の祖沖之（Tsu-Chung-Chih, 429－500）が見出した355/113と同じ有効桁数である〔10；p. 136, 17；pp. 24, 218〕。この点に関してアルキメデス以来，πの桁数を上げる計算は，純粋に計算能力と忍耐力の問題になってしまっている。祖冲之はこの分数を密率と呼んでいる。この精度はπの沿革で触れたように，ヨーロッパでは16世紀になるまで到達できなかったものである。今日では単に，コンピュータ・プログラミング能力の問題になってしまっている。そして原理的には，計算時間に対して支払える費用の

問題以外の何ものでもなくなってしまった。

コンピュータ・プログラミングにおいて広く採用されている計算式は，マーチン（Machin, J. 1685－1751）の公式である。その等式はつぎの（３．２）式であり，

$$\pi / 4 = 4 * \arctan(1/5) - \arctan(1/239) \qquad (3.2)$$

ただし，arctan x の級数展開はつぎの等式による。

$$\arctan x = x - x^3/3 + x^5/5 - x^7/7 + x^9/9 - \cdots\cdots \qquad (3.3)$$

故に，

$$\pi = 16*(1/5 - 1/(3*5^3) + 1/(5*5^5) - 1/(7*5^7) + \cdots\cdots) - 4*(1/239 - 1/(3*239^3) + 1/(5*239^5) - 1/(7*239^7) + \cdots\cdots) \qquad (3.4)$$

となる。つぎに，コンピュータ・プログラムの作成には，（３．４）式をそのまま使用し，型宣言なしに数値変数および数値定数をコーディングするとレジスター２バイトの単精度，有効桁数７として扱われるので注意が必要である。組込み関数の誤差を最小にするには，（３．５）式のように有効桁数16桁を確保するため４バイト倍精度型のレジスターをコールし，各変数および定数すべてについて倍精度型に宣言するべきである。

$$\pi = 16*\arctan(0.2\#) - 4*\arctan(0.00418410041841\#) \qquad (3.5)$$

この算式は，図２のようなプログラムの場合，型宣言とarctangent のアルゴリズムの違いによって表１に表現されているような処理結果となる。

```
1' save "1:ARCTAN.N98", a
1000 DEFDBL A,X,Y,Z                    ' 数値変数倍精度型宣言
1010 K=12
1020 X=1#/5#                           ' 数値定数倍精度型宣言
1030 GOSUB *SUB.ARC                    ' arctangent subroutine call
1040 Z=Y
1050 X=1#/239#
1060 GOSUB *SUB.ARC
1070 A=16*Z-4*Y                        ' Machin's formula
1080 PRINT "TYPE 1=";A
1090 END
1100 *SUB.ARC                          ' 級数展開処理
1110 Y=0
1120 FOR I=1 TO K
1130   J=2*I-1
1140   IF MOD 2=0 THEN GOTO 1160
1150   Y=Y+X^J/J:GOTO 1170
1160   Y=Y-X^J/J
```

```
1170 NEXT I
1180 RETURN
```

図2．ARCTANGENT級数展開プログラム

このプログラムには（3．1）式の計算処理において，数値変数倍精度型宣言（A，X，Y，Z），数値定数倍精度型宣言（1#／5#，1#／239#）がしてあるが，それぞれの変数定数の宣言を1つ外したケースの結果が表1である。

表1．有効桁数16に含まれる誤差

処理／ケース	数値変数型宣言				数値定数型宣言		計算値
	A	X	Y	Z	1#/5#	1#/239#	
1							3.141592653589793
2	○						3.14159
3		○					3.141592910072929
4			○				3.141592705622315
5				○			3.141592706001467
6					○		3.14159269943952
7						○	3.141592653589793

注）○印の箇所は単精度型宣言し，それ以外は倍精度型宣言してある。

以上の結果から，ケース1とケース7はレジスターとして全て4バイトを使用していることが解かる。それ以外のケースでは，2バイト型のレジスターを代入文や演算途中に使用しているために，丸めの誤差あるいは打切りの誤差が10進数7桁単位で介入していることになり，表1の計算値から絶対誤差の状況が読み取れる。結局のところ，倍精度の有効桁数を確保するためには，図2のようにすべての変数とすべての定数に倍精度型宣言をする必要がある。ただし，1/239#と1#/239#とは同じレジスターを使っていることが，ケース7から検証できる。ところで，（3．5）式は，組込み関数を利用して演算すると，

$$PI\# = 16\# * ANT(1\#/5\#) - 4\# * ATN(1\#/239\#) \qquad (3．6)$$

のような算式となる。全ての変数と定数を倍精度型に宣言しても，PI#の値は，3.14159270 5622315 となり，ちょうどケース4のarctangent演算結果の代理変数Yを単精度にしたときと一致することになる。

レジスターの型宣言に依存したπのアルゴリズムは，図3のようなプログラム〔8；pp. 107－110〕の場合また，組込み関数と倍精度宣言によって処理結果は大きく異なる。

このプログラムは，半径（radius）が1の円の場合，面積と円周率とが等しい性質を利用し，正多角形（regular polygon）の面積を計算して近似するプログラムである。πの近似

```
1   'save "1:FINDPI.N98",a
1000 DEFDBL A,H                    ' double precision declaration
1010 A= 2                          ' Area
1020 H= 1# /2# ^0.5#               ' altitude of each triangle
1030 FOR K= 1 TO 30                ' Double sides
1040    A=A/H
1050    H=((1#+H)/2#)^0.5#
1060    PRINT USING "###)=";K;:PRINT A
1070 NEXT K
1080 END
```

図3．FIND PI プログラム

値は，ＦＯＲ文のLOOP回数によって収束した形で求めることができる。このプログラムを単精度にかえて処理すると，SQRの組込み関数を使用していても，収束した近似値は9回のループで得られる。有効桁数は7桁まで正しい。図3のプログラムは，26回のFOR ループで有効桁数15桁まで近似し，3.141592653589797に収束する。16桁目の誤差は，レジスターの制限によっている。ところが，行番号1020の1#/2#^0.5#の代わりに1#/SQR(2)の組込み関数を使用した場合，30回ループしても3.1415925851327 14となって収束せず，有効桁数は7桁にすぎない。

ここで取り扱ったπのアルゴリズムは，絶対誤差との関係でつぎの4つである。モンテカルロ法を使った円の面積とビュッフォンの針の問題，正多角形の面積計算およびマーチンの公式のアプローチである。単精度である組込み関数は，モンテカルロ法の際に使ったRND関数と正多角形の面積の計算による円の面積に近似したSQR関数を吟味してある。そして，倍精度型レジスターは，マーチンの公式を中心に考察したものである。

4．倍精度πの近似分数

ここで分析した近似分数プログラムの処理結果は，２．３のシステム・フローに従って実行したものである。各プログラムは，ベンチ・マークテストのためにN_{88}-BASIC(86)とN_{88}-BASIC(MS-DOS版）と両方のOSで実行してあるが，はじめにN_{88}-BASIC(86)のOS 配下で動かした状況を吟味する。近似分数作成プログラムは，添付資料１（プログラム名：FRACT 1．N98）を利用してある。このプログラムは，（３．５）式のアルゴリズムを考慮して，プログラミングした図２の級数展開プログラム処理結果を倍精度πの真値として利用している。もともと，すべての有理数（分子，分母が正の整数）は連分数として展開することができるし，無理数はまた無限連分数として展開される。そして，連分数での真値と近

歴史を映し出す小さな鏡である。それはシラキュースの知性的なアルキメデス（B. C. 287－212）の物語でもあれば，現代のコンピュータ物語でもあり，あるいはまた，紀元前３世紀のアレキサンドリア大学の物語であるとともに，愚かにも科学書に火をつけて焼き払った中世の司祭や十字軍の物語でもある。π発見の進退は，その時代の社会的背景と相互依存の関係にある〔17 ; pp. iiv, 233－235〕。

つぎに簡単に，π発見のエポック・メーキングを挙げておくことにする。

紀元前（概数）

2000年　バビロニア人　　π＝3＋1／8を使う。有効桁数は２桁である。

2000年　エジプト人　　π＝(16／9)^2＝3.1605を使う。
4000年後愚かな焚書のため，Heisel, C.T.とFaber, C.T.が，π＝256／81であると大発見したことをつぎの著書で指摘している。『Behold! The Grand Problem, The Circle Squared Beyond Refutation, No Longer Unsolved』, (1931), Ohio.

1200年　中国人　　π＝３を使う。有効桁数は１桁である。

300年　Archimedes　　3＋10／71＜π＜22／7を確立。3.14085＜π＜3.14286，これは正96角形を利用した。また，π≒211875／67441＝3.14163を求める。
この方法は，ニュートンが微積分法を発見するまで，基本的には円に内接，または外接する正ｎ多角形の周長が，ｎを大きくすればそれが円周に近くなることを利用して計算していた。例えば，

内接正　ｎ多角形＝(　n/2)
　　＊sin (　360°　/　n)＜　π

内接正　96多角形＝(　96/2)
　　＊sin(π＊360/180/　96)＝3.13935(単精度)

内接正1024多角形＝(1024/2)
　　＊sin(π＊360/180/1024)＝3.14157(単精度)

外接正　ｎ多角形＝　　n
　　＊tan(　180°　/　n)＞　π

外接正　96多角形＝　　96
　　＊tan(π＊180/180/　96)＝3.14271(単精度)

外接正1024多角形＝　1024
　　＊tan(π＊180/180/1024)＝3.1416（単精度）

即ち，正1024角形の内接・外接アプローチの場合には，

似分数作成ルーチンで求めた近似値とを有効桁数16桁について漸近させたものである。近似分数作成プログラムはもう1つ，近似分数分析の目的で，処理結果をデータ・ファイルにしてある。近似の状況はそのデータ・ファイルを加工して（FRACT 4 . TIM），添付資料3の真値と近似値との差，絶対誤差の欄に表現してある。そこで，π を有理数として表現する場合，絶対誤差の値が分かっていれば，丸めの誤差の影響については推定可能である。

近似分数の処理結果は添付資料3の表頭に，近似分数の分子（numerator）・分母（denominator），有理数の形にして計算した近似値（current valueの欄），有効桁数16桁に出現した近似分数の件数（cnt ; count），絶対誤差から推定した有効桁数（cl ; column），絶対誤差（absolute error）および稼働時間（HH : MM : SS）がとってある。そこでπを有理数化した添付資料3（近似分数分析資料）を分析すると，有効桁数（cl）と近似分数出現との間には一定の規則性が認識でき，つぎの関係式で表現できると考える。

$$(\mathrm{cl}) = (\text{numeratorの桁数}) + (\text{denominatorの桁数}) \pm \alpha \qquad (4 . 1)$$

ただし，$+2 \leqq \alpha \leqq -1$

ランベルトの近似分数のはじめのいくつかは，いろいろな方法で以前に発見されている。近似分数の1件目に表れる$\pi = 3$は，紀元前12世紀にすでに，中国で使っている。（4 . 1）式のαの値は，＋1となる。アルキメデスの発見したπの上限値22／7は，近似分数の5件目に表れるが，αの値は，0である。5世紀に祖冲之，16世紀にアドリアンほか数人が発見した，その14件目に表れる355／113はαの値が－1である。このαの値は他になく，πの有理数としては効率の良いものである。今回の処理は有効桁数16桁について近似分数を検索したが，177件目に絶対誤差はゼロになり，177件の有理数が見つかった。そのなかでαの値は，14件目以外0以上であったことが添付資料3で明らかになった。（4 . 1）式のαの値は，$+2 \leqq \alpha \leqq -1$の制約条件の中でつぎの近似分数が見つかることを意味している。恐らく，近似分数の桁数を増やしても，$\alpha < 0$の値は見つからないと考えて大過ない。（4 . 1）式を許容すれば，355／113という分数は，歴史的発見のみでなく，代数的にも注目すべき有理数である。その近似値は，単精度の変数あるいは組込み関数の精度とほぼ一致しており，3.141592920353928である。そして，15件目の近似分数は，52,163/16,604であり，近似値が3.141592387376536であるので，絶対誤差の値は殆ど変わらないことが解かる。

つぎに，プログラムの稼働時間と近似分数の出現頻度は，指数関数的関係にある。添付資料1のプログラムのアルゴリズムでは，稼働時間と分母の値とは正比例するようにしてある。分母は処理のループ回数であり，その値が大きくなれば，稼働時間に大きな負荷がかかるようになっている。そこで，分母の桁数と近似分数出現の度数分布は，集計すると表2のようになっている。この表の表側には，分母の桁数を採り，桁数1桁から近似値16桁の有効桁数を満足する分母の桁数8桁までに分類してある。一方、表頭には近似分数出現件数，近似値の有効桁数および稼働処理時間に分けて項目を挙げ，度数分析がなされている。

セッサの性能評価の一部，特に実行速度，所要メモリサイズを確認するものである。テストプログラムと現実の応用プログラムとを完全に対応させることは一般に困難であり，どのようなテストプログラムを利用するかによってテストの精度が定まる。テストプログラムは，アドレス関係，レジスター関係，制御関係，演算関係，入出力関係および割込み処理の総合的機能が表わされることが望ましい。添付資料１の近似分数作成プログラムでは，アドレス関係・レジスター関係のものは倍精度型宣言（DEFDBL）であり，制御関係はFOR文，GOSUB文の比較的時間的負荷の必要な命令であり，演算関係はマーチンの公式展開であり，入出力関係は行番号1740から1880のディスク・アクセスや画面出力である。このプログラムには，陽表的に割込み処理命令は使用していない。ただし，GOSUB文の復帰命令は，スタックポインターを利用しているはずなので陰伏的に割込み処理は介入している。

添付資料２に戻ると，ベンチ・マークテストは図１のシステム・フローに添って処理してあるプログラム名 FRACTLPT. EXE は，実行するとベンチ・マークテストの要約統計量即ち，添付資料２が出力される。表頭には，N_{88}-BASIC(86)とN_{88}-BASIC(MS-DOS版)との近似分数作成プログラムの処理時間とその比率，精度にコントロール・ブレークの起きた有効桁数およびその件数，処理上のループ回数が採ってある。入出力関係のアクセス・タイムは355／113の近似分数を検索するまでの処理が該当する。何故ならば，他のベンチ・マークテストの項目は，時間上問題にならないほど，僅かな時間しか作動していない。そこで，N_{88}-BASIC(86)とN_{88}-BASIC(MS-DOS版）とは１対２の比率であり，コンパイラー形式のBASICで処理したにもかかわらず，MS-DOS版は効率が悪い。その理由は，ランダムファイルの一回のディスク・アクセス（セクター単位）において，N_{88}-BASIC(86)は256 バイト単位であるが，N_{88}-BASIC(MS-DOS版）は1024バイト単位にアクセスが行なわれていることに関係あると評価できる。そこで，互換性の高いMS-DOS版は絶えず更新のあるデータ・ファイルに使用するときは，フロッピ・ディスクをハード・ディスクに換えるか，更新頻度を少なくするアルゴリズムにすべきである。逆に，近似分数作成プログラムの近似分数 165件目以降では，その効率は逆転して，２対１の比率になっている（添付資料２および表２参照のこと)。この段階では，ディスク・アクセスは，殆ど問題にならない。ここに至ってはじめて実行形式のBASIC（中間言語）が，処理スピードの点で優位に立つことになっている。π の沿革のところでとり挙げた10,000桁のπ の計算は，この点から実行形式のＢＡＳICを使用することがよい。そして別の言語に視点を移せば，一種のマシン語形式で動き，ベンチ・マークテストの結果が制御関係でよく，スタック・メカニズムの充実しているTURBO-Pascal の方が，それよりも効率の良いことが解かる。ただし，実数型の変数は，実数の範囲が（標準版の場合）11桁の仮数しか扱えず，６バイトのレジスターという制限があり，本稿の近似分数検索には適合しないと言える。ただし，BCD版のTURBO-Pascal は仮数部が最大18桁の有効桁数であるので,（４．１）式に基づく17桁目の処理時間の予想はシステム環境の充実によって検証できるはずである。

添付資料　1

近似分数作成プログラム

```
PROGRAM-ID:[FRACT1.PAI]                                    DATE 87/01/26   PAGE:   1

1 'save "1:FRACT1.PAI",A
1000 '-------------------
1010 ' initial-process     ref. date=86/12/01 time=08:45:00  T.,IZUMO
1020 '-------------------
1030 ' fraction [ PAI ]
1040 CONSOLE 0,25,0,1:CLS 3:WIDTH 80,25:COLOR 7
1050 D87$=DATE$:MID$(D87$,1,2)="87":DATE$=D87$
1060    'ref. (A.C.429-500)      355/113#    3.141592920353982#
1070    '      4#*ATN(1)                     3.141592741012573#
1080    '      52163#/16604#                 3.141592387376536#
1090 LOCATE 7,18:PRINT "***** true value  3.14159265358979323846264338327950288
1971"
1100 DA$=DATE$:TI$=TIME$
1110 DEFDBL A,D,E,L-N,X-Z
1120 COLOR 4
1130 K=11
1140 X=.2#                              ' 4*atn(1/5)
1150 GOSUB *SUB.ARCTAN
1160 Z=Y
1170 X=.00418410041841#                 ' atn(1/239)
1180 GOSUB *SUB.ARCTAN
1190 A=16*Z-4*Y                         ' ->     3.141592653589793   true value [A]
1200      '(4*ATN(1/5)-ATN(1/239))*4#   ->     3.141592741012573#  function value
1210      '                                    -0.000000087422780   absolute error
1220 N=1:D=1
1230 '-------------------
1240 ' convert-process     ( Fraction )
1250 '-------------------
1260 L=A*N
1270 M=INT(L)
1280 IF M+1-L<L-M THEN M=M+1
1290 E=ABS(A-M/N)
1300 IF E>=D THEN GOTO 1510
1310 '-------------------
1320 ' found               ( control-break )
1330 '-------------------
1340 IF N<    115  THEN II=II+1:LOCATE 0,II+3:GOTO 1410
1350 IF N< 16605  THEN LOCATE 0,19    :GOTO 1410
1360 IF N< 40000! THEN LOCATE 0,20    :GOTO 1410
1370 IF N< 70000! THEN LOCATE 0,21    :GOTO 1410
1380 IF N<100000! THEN LOCATE 0,22    :GOTO 1410
1390 IF N<500000! THEN LOCATE 0,23    :GOTO 1410
1400                   LOCATE 0,24    :GOTO 1410
1410 COLOR 4
1420 PRINT USING "##########,/##########, ##.############### ";M,N,M/N;:JJ=JJ+1:
PRINT TAB(60);:PRINT USING "(#### )=&      &";JJ,TIME$
1430 COLOR 6
1440 '
1450 GOSUB *FILE.WRITE
1460 LOCATE 39, 2:PRINT USING "start time    =     &      & &      &";DA$  ,TI$
1470 LOCATE 39, 3:PRINT USING "final time    =     &      & &      &";DATE$,TIME
$
1480 D=E
1490 IF IX > 200 THEN 1600
1500 '-------------------
1510 ' non-found
1520 '-------------------
1530 N=N+1
1540 COLOR 6
1550 IF TH$=MID$(TIME$,4,1) THEN 1580
```

```
1560 LOCATE 39, 1:PRINT USING "current count =(########## )=&      &";N,TIMES
1570 THS=MIDS(TIMES,4,1)
1580 COLOR 7
1590 GOTO 1240
1600 END
1610 '-------------------
1620 *SUB.ARCTAN
1630 '-------------------
1640 '          X  ->  Y
1650 Y=0
1660 FOR I=1 TO K
1670   J=2*I-1
1680   IF I MOD 2 = 0 THEN GOTO 1700
1690   Y=Y+X^J/J:GOTO 1710                ' odd
1700   Y=Y-X^J/J                          ' even
1710 NEXT I
1720 RETURN
1730 '-------------------
1740 *FILE.WRITE
1750 '--------------------
1760 OPEN "2:P70105.dat" AS #1            ' random file
1770 FIELD  #1,8 AS MS,8 AS NS,8 AS NNS,8 AS JJS, 8 AS DXS, 8 AS TXS
1780 NN=M/N
1790 LSET MS=MKDS(M)
1800 LSET NS=MKDS(N)
1810 LSET NNS=MKDS(NN)
1820 LSET JJS=MKDS(JJ)
1830 LSET DXS=DATES
1840 LSET TXS=TIMES
1850 IX=IX+1
1860 PUT #1,IX
1870 CLOSE #1
1880 RETURN
1890 '============= ( end of program ) =============
```

添付資料　2

N$_{88}$-BASIC(86)とN$_{88}$-BASIC(MS-DOS版)との比較

[NBASIC] - [MS-DOS]	[Fraction of Pai		](double precision)	PAGE= 1
HH:MM:SS - HH:MM:SS	[rate.	second	HH:MM:SS](cnt=cl)[	loop cnt]
0: 0: 0 - 0: 0: 0	[	0	0: 0: 0](3= 1)	1
0: 0: 1 - 0: 0: 4	[0.250	-3	-1:59:57](4= 2)	6
0: 0: 2 - 0: 0: 5	[0.400	-3	-1:59:57](5= 3)	7
0: 0: 7 - 0: 0:13	[0.538	-6	-1:59:54](10= 4)	85
0: 0: 9 - 0: 0:20	[0.450	-11	-1:59:49](14= 7)	113
0: 6:31 - 0: 5:46	[1.130	45	0: 0:45](116= 8)	28,017
0: 7:50 - 0: 7:23	[1.061	27	0: 0:27](157= 9)	32,650
0: 8: 0 - 0: 7:35	[1.055	25	0: 0:25](162=10)	33,215
0:20:49 - 0:13:49	[1.507	420	0: 7: 0](164=11)	99,532
1:11:50 - 0:38:21	[1.873	2,009	0:33:29](166=12)	364,913
4:21:45 - 2: 9: 3	[2.028	7,962	2:12:42](168=13)	1,360,120
5:30:53 - 2:41:51	[2.044	10,142	2:49: 2](169=14)	1,725,033
70:51: 4 - 34:10:25	[2.073	132,039	36:40:39](175=15)	22,060,516

添付資料　3

近似分数分析資料

```
[ Fraction of Pai    ]( double precision )                                  PAGE= 1
 numerator/denominator[ current value ](cnt=cl)[ absolute error ] HH:MM:SS

         3/          1 3.000000000000000(   1= 1) 0.141592653589793  0: 0: 0
        13/          4 3.250000000000000(   2= 1)-0.108407346410207  0: 0: 0
        16/          5 3.200000000000000(   3= 2)-0.058407346410207  0: 0: 1
        19/          6 3.166666666666667(   4= 2)-0.025074013076874  0: 0: 1
        22/          7 3.142857142857143(   5= 3)-0.001264489267350  0: 0: 2
       179/         57 3.140350877192982(   6= 3) 0.001241776396811  0: 0: 3
       201/         64 3.140625000000000(   7= 4) 0.000967653589793  0: 0: 4
       223/         71 3.140845070422535(   8= 4) 0.000747583167258  0: 0: 5
       245/         78 3.141025641025641(   9= 4) 0.000567012564152  0: 0: 6
       267/         85 3.141176470588235(  10= 4) 0.000416183001558  0: 0: 7
       289/         92 3.141304347826087(  11= 4) 0.000288305763706  0: 0: 7
       311/         99 3.141414141414141(  12= 4) 0.000178512175652  0: 0: 8
       333/        106 3.141509433962264(  13= 5) 0.000083219627529  0: 0: 9
       355/        113 3.141592920353982(  14= 7)-0.000000266764189  0: 0: 9
    52,163/     16,604 3.141592387376536(  15= 7) 0.000000266213257  0: 3:17
    52,518/     16,717 3.141592390979243(  16= 7) 0.000000262610550  0: 3:19
    52,873/     16,830 3.141592394533571(  17= 7) 0.000000259056222  0: 3:21
    53,228/     16,943 3.141592398040489(  18= 7) 0.000000255549304  0: 3:23
    53,583/     17,056 3.141592401500938(  19= 7) 0.000000252088855  0: 3:25
    53,938/     17,169 3.141592404915837(  20= 7) 0.000000248673956  0: 3:27
    54,293/     17,282 3.141592408286078(  21= 7) 0.000000245303715  0: 3:29
    54,648/     17,395 3.141592411612532(  22= 7) 0.000000241977261  0: 3:31
    55,003/     17,508 3.141592414896048(  23= 7) 0.000000238693745  0: 3:33
    55,358/     17,621 3.141592418137450(  24= 7) 0.000000235452343  0: 3:34
    55,713/     17,734 3.141592421337544(  25= 7) 0.000000232252249  0: 3:36
    56,068/     17,847 3.141592424497114(  26= 7) 0.000000229092679  0: 3:38
    56,423/     17,960 3.141592427616926(  27= 7) 0.000000225972866  0: 3:40
    56,778/     18,073 3.141592430697726(  28= 7) 0.000000222892067  0: 3:42
    57,133/     18,186 3.141592433740240(  29= 7) 0.000000219849553  0: 3:44
    57,488/     18,299 3.141592436745177(  30= 7) 0.000000216844616  0: 3:45
    57,843/     18,412 3.141592439713230(  31= 7) 0.000000213876562  0: 3:47
    58,198/     18,525 3.141592442645074(  32= 7) 0.000000210944719  0: 3:49
    58,553/     18,638 3.141592445541367(  33= 7) 0.000000208048426  0: 3:51
    58,908/     18,751 3.141592448402752(  34= 7) 0.000000205187041  0: 3:53
    59,263/     18,864 3.141592451229856(  35= 7) 0.000000202359937  0: 3:55
    59,618/     18,977 3.141592454023291(  36= 7) 0.000000199566502  0: 3:57
    59,973/     19,090 3.141592456783656(  37= 7) 0.000000196806137  0: 3:59
    60,328/     19,203 3.141592459511535(  38= 7) 0.000000194078258  0: 4: 1
    60,683/     19,316 3.141592462207496(  39= 7) 0.000000191382297  0: 4: 3
    61,038/     19,429 3.141592464872098(  40= 7) 0.000000188717695  0: 4: 5
    61,393/     19,542 3.141592467505885(  41= 7) 0.000000186083908  0: 4: 7
    61,748/     19,655 3.141592470109387(  42= 7) 0.000000183480406  0: 4: 8
    62,103/     19,768 3.141592472683124(  43= 7) 0.000000180906669  0: 4:10
    62,458/     19,881 3.141592475227604(  44= 7) 0.000000178362189  0: 4:12
    62,813/     19,994 3.141592477743323(  45= 7) 0.000000175846470  0: 4:14
    63,168/     20,107 3.141592480230765(  46= 7) 0.000000173359028  0: 4:16
    63,523/     20,220 3.141592482690405(  47= 7) 0.000000170899387  0: 4:18
    63,878/     20,333 3.141592485122707(  48= 7) 0.000000168467086  0: 4:20
    64,233/     20,446 3.141592487528123(  49= 7) 0.000000166061670  0: 4:22
    64,588/     20,559 3.141592489907097(  50= 7) 0.000000163682696  0: 4:24
    64,943/     20,672 3.141592492260062(  51= 7) 0.000000161329731  0: 4:26
    65,298/     20,785 3.141592494587443(  52= 7) 0.000000159002350  0: 4:28
    65,653/     20,898 3.141592496889654(  53= 7) 0.000000156700138  0: 4:30
    66,008/     21,011 3.141592499167103(  54= 7) 0.000000154422690  0: 4:32
    66,363/     21,124 3.141592501420186(  55= 7) 0.000000152169607  0: 4:33
    66,718/     21,237 3.141592503649291(  56= 7) 0.000000149940502  0: 4:35
    67,073/     21,350 3.141592505854801(  57= 7) 0.000000147734992  0: 4:37
    67,428/     21,463 3.141592508037087(  58= 7) 0.000000145552706  0: 4:39
    67,783/     21,576 3.141592510196515(  59= 7) 0.000000143393278  0: 4:41
```

numerator/denominator	[current value](cnt=cl)	[absolute error]	HH:MM:SS
68,138/ 21,689	3.141592512333441(60= 7)	0.000000141256352	0: 4:43
68,493/ 21,802	3.141592514448216(61= 7)	0.000000139141577	0: 4:45
68,848/ 21,915	3.141592516541182(62= 7)	0.000000137048611	0: 4:47
69,203/ 22,028	3.141592518612675(63= 7)	0.000000134977118	0: 4:49
69,558/ 22,141	3.141592520663023(64= 7)	0.000000132926770	0: 4:51
69,913/ 22,254	3.141592522692550(65= 7)	0.000000130897243	0: 4:53
70,268/ 22,367	3.141592524701569(66= 7)	0.000000128888224	0: 4:55
70,623/ 22,480	3.141592526690391(67= 7)	0.000000126899402	0: 4:57
70,978/ 22,593	3.141592528659319(68= 7)	0.000000124930474	0: 4:58
71,333/ 22,706	3.141592530608650(69= 7)	0.000000122981143	0: 5: 0
71,688/ 22,819	3.141592532538674(70= 7)	0.000000121051119	0: 5: 2
72,043/ 22,932	3.141592534449677(71= 7)	0.000000119140116	0: 5: 4
72,398/ 23,045	3.141592536341940(72= 7)	0.000000117247853	0: 5: 6
72,753/ 23,158	3.141592538215735(73= 7)	0.000000115374058	0: 5: 8
73,108/ 23,271	3.141592540071333(74= 7)	0.000000113518460	0: 5:10
73,463/ 23,384	3.141592541908998(75= 7)	0.000000111680795	0: 5:12
73,818/ 23,497	3.141592543728987(76= 7)	0.000000109860806	0: 5:14
74,173/ 23,610	3.141592545531554(77= 7)	0.000000108058239	0: 5:16
74,528/ 23,723	3.141592547316950(78= 7)	0.000000106272843	0: 5:18
74,883/ 23,836	3.141592549085417(79= 7)	0.000000104504376	0: 5:20
75,238/ 23,949	3.141592550837196(80= 7)	0.000000102752597	0: 5:22
75,593/ 24,062	3.141592552572521(81= 7)	0.000000101017272	0: 5:23
75,948/ 24,175	3.141592554291623(82= 8)	0.000000099298169	0: 5:25
76,303/ 24,288	3.141592555994730(83= 8)	0.000000097595063	0: 5:27
76,658/ 24,401	3.141592557682062(84= 8)	0.000000095907731	0: 5:29
77,013/ 24,514	3.141592559353839(85= 8)	0.000000094235954	0: 5:31
77,368/ 24,627	3.141592561010273(86= 8)	0.000000092579520	0: 5:33
77,723/ 24,740	3.141592562651576(87= 8)	0.000000090938217	0: 5:35
78,078/ 24,853	3.141592564277954(88= 8)	0.000000089311839	0: 5:37
78,433/ 24,966	3.141592565889610(89= 8)	0.000000087700183	0: 5:39
78,788/ 25,079	3.141592567486742(90= 8)	0.000000086103051	0: 5:41
79,143/ 25,192	3.141592569069546(91= 8)	0.000000084520247	0: 5:43
79,498/ 25,305	3.141592570638214(92= 8)	0.000000082951579	0: 5:45
79,853/ 25,418	3.141592572192934(93= 8)	0.000000081396859	0: 5:47
80,208/ 25,531	3.141592573733892(94= 8)	0.000000079855901	0: 5:48
80,563/ 25,644	3.141592575261270(95= 8)	0.000000078328523	0: 5:50
80,918/ 25,757	3.141592576775246(96= 8)	0.000000076814547	0: 5:52
81,273/ 25,870	3.141592578275995(97= 8)	0.000000075313798	0: 5:54
81,628/ 25,983	3.141592579763692(98= 8)	0.000000073826101	0: 5:56
81,983/ 26,096	3.141592581238504(99= 8)	0.000000072351289	0: 5:58
82,338/ 26,209	3.141592582700599(100= 8)	0.000000070889194	0: 6: 0
82,693/ 26,322	3.141592584150140(101= 8)	0.000000069439652	0: 6: 2
83,048/ 26,435	3.141592585587289(102= 8)	0.000000068002503	0: 6: 4
83,403/ 26,548	3.141592587012204(103= 8)	0.000000066577589	0: 6: 6
83,758/ 26,661	3.141592588425040(104= 8)	0.000000065164753	0: 6: 8
84,113/ 26,774	3.141592589825951(105= 8)	0.000000063763842	0: 6:10
84,468/ 26,887	3.141592591215085(106= 8)	0.000000062374708	0: 6:12
84,823/ 27,000	3.141592592592592(107= 8)	0.000000060997200	0: 6:13
85,178/ 27,113	3.141592593958618(108= 8)	0.000000059631175	0: 6:15
85,533/ 27,226	3.141592595313303(109= 8)	0.000000058276490	0: 6:17
85,888/ 27,339	3.141592596656791(110= 8)	0.000000056933002	0: 6:19
86,243/ 27,452	3.141592597989218(111= 8)	0.000000055600575	0: 6:21
86,598/ 27,565	3.141592599310720(112= 8)	0.000000054279073	0: 6:23
86,953/ 27,678	3.141592600621432(113= 8)	0.000000052968361	0: 6:25
87,308/ 27,791	3.141592601921485(114= 8)	0.000000051668308	0: 6:27
87,663/ 27,904	3.141592603211009(115= 8)	0.000000050378784	0: 6:29
88,018/ 28,017	3.141592604490131(116= 8)	0.000000049099662	0: 6:31
88,373/ 28,130	3.141592605758976(117= 8)	0.000000047830817	0: 6:33
88,728/ 28,243	3.141592607017668(118= 8)	0.000000046572125	0: 6:35

numerator/denominator	[current value](cnt=cl)	[absolute error]	HH:MM:SS
89,083/ 28,356	3.1415926082663 28(119= 8)	0.000000045323465	0: 6:37
89,438/ 28,469	3.1415926095050 76(120= 8)	0.000000044084717	0: 6:38
89,793/ 28,582	3.1415926107340 28(121= 8)	0.000000042855765	0: 6:40
90,148/ 28,695	3.1415926119533 02(122= 8)	0.000000041636491	0: 6:42
90,503/ 28,808	3.1415926131630 10(123= 8)	0.000000040426783	0: 6:44
90,858/ 28,921	3.1415926143632 65(124= 8)	0.000000039226528	0: 6:46
91,213/ 29,034	3.1415926155541 78(125= 8)	0.000000038035615	0: 6:48
91,568/ 29,147	3.1415926167358 56(126= 8)	0.000000036853937	0: 6:50
91,923/ 29,260	3.1415926179084 07(127= 8)	0.000000035681386	0: 6:52
92,278/ 29,373	3.1415926190719 37(128= 8)	0.000000034517856	0: 6:54
92,633/ 29,486	3.1415926202265 48(129= 8)	0.000000033363245	0: 6:56
92,988/ 29,599	3.1415926213723 44(130= 8)	0.000000032217449	0: 6:58
93,343/ 29,712	3.1415926225094 24(131= 8)	0.000000031080369	0: 7: 0
93,698/ 29,825	3.1415926236378 88(132= 8)	0.000000029951905	0: 7: 2
94,053/ 29,938	3.1415926247578 33(133= 8)	0.000000028831960	0: 7: 4
94,408/ 30,051	3.1415926258693 55(134= 8)	0.000000027720438	0: 7: 6
94,763/ 30,164	3.1415926269725 50(135= 8)	0.000000026617243	0: 7: 8
95,118/ 30,277	3.1415926280675 10(136= 8)	0.000000025522283	0: 7:10
95,473/ 30,390	3.1415926291543 27(137= 8)	0.000000024435466	0: 7:12
95,828/ 30,503	3.1415926302330 92(138= 8)	0.000000023356701	0: 7:14
96,183/ 30,616	3.1415926313038 93(139= 8)	0.000000022285900	0: 7:16
96,538/ 30,729	3.1415926323668 20(140= 8)	0.000000021222973	0: 7:18
96,893/ 30,842	3.1415926334219 57(141= 8)	0.000000020167836	0: 7:20
97,248/ 30,955	3.1415926344693 91(142= 8)	0.000000019120402	0: 7:21
97,603/ 31,068	3.1415926355092 06(143= 8)	0.000000018080587	0: 7:23
97,958/ 31,181	3.1415926365414 84(144= 8)	0.000000017048309	0: 7:25
98,313/ 31,294	3.1415926375663 07(145= 8)	0.000000016023486	0: 7:27
98,668/ 31,407	3.1415926385837 55(146= 8)	0.000000015006038	0: 7:29
99,023/ 31,520	3.1415926395939 09(147= 8)	0.000000013995884	0: 7:31
99,378/ 31,633	3.1415926405968 45(148= 8)	0.000000012992948	0: 7:33
99,733/ 31,746	3.1415926415926 42(149= 8)	0.000000011997151	0: 7:35
100,088/ 31,859	3.1415926425813 74(150= 8)	0.000000011008419	0: 7:37
100,443/ 31,972	3.1415926435631 18(151= 8)	0.000000010026675	0: 7:39
100,798/ 32,085	3.1415926445379 46(152= 9)	0.000000009051847	0: 7:41
101,153/ 32,198	3.1415926455059 32(153= 9)	0.000000008083861	0: 7:43
101,508/ 32,311	3.1415926464671 47(154= 9)	0.000000007122646	0: 7:45
101,863/ 32,424	3.1415926474216 63(155= 9)	0.000000006168130	0: 7:47
102,218/ 32,537	3.1415926483695 48(156= 9)	0.000000005220244	0: 7:48
102,573/ 32,650	3.1415926493108 73(157= 9)	0.000000004278920	0: 7:50
102,928/ 32,763	3.1415926502457 04(158= 9)	0.000000003344089	0: 7:52
103,283/ 32,876	3.1415926511741 09(159= 9)	0.000000002415684	0: 7:54
103,638/ 32,989	3.1415926520961 53(160= 9)	0.000000001493640	0: 7:56
103,993/ 33,102	3.1415926530119 03(161=10)	0.000000000577890	0: 7:58
104,348/ 33,215	3.1415926539214 21(162=10)	-0.000000000331628	0: 8: 0
208,341/ 66,317	3.1415926534674 37(163=10)	0.000000000122356	0:14:24
312,689/ 99,532	3.1415926536189 37(164=11)	-0.000000000029144	0:20:49
833,719/ 265,381	3.1415926535810 78(165=12)	0.000000000008715	0:52:43
1,146,408/ 364,913	3.1415926535914 04(166=12)	-0.000000000001611	1:11:50
3,126,535/ 995,207	3.1415926535886 50(167=12)	0.000000000001143	3:12:11
4,272,943/ 1,360,120	3.1415926535893 89(168=13)	0.000000000000404	4:21:45
5,419,351/ 1,725,033	3.1415926535898 15(169=14)	-0.000000000000022	5:30:53
42,208,400/13,435,351	3.1415926535897 72(170=14)	0.000000000000021	43: 1:48
47,627,751/15,160,384	3.1415926535897 77(171=14)	0.000000000000016	48:35:19
53,047,102/16,885,417	3.1415926535897 81(172=14)	0.000000000000012	54: 8:55
58,466,453/18,610,450	3.1415926535897 84(173=15)	0.000000000000009	59:43:11
63,885,804/20,335,483	3.1415926535897 87(174=15)	0.000000000000006	65:17:25
69,305,155/22,060,516	3.1415926535897 89(175=15)	0.000000000000004	70:51: 4
74,724,506/23,785,549	3.1415926535897 91(176=15)	0.000000000000002	76:23:52
80,143,857/25,510,582	3.1415926535897 93(177=16)	0.000000000000000	81:56:40

参考文献

[1] Hanada, T., (1972), *Numerical Computation,* Occasional Paper.

[2] Helms, H. ed., (1983), *Computer Handbook,* McGraw-Hill.

[3] 細井勉他, (1974), 『ライス コンピュータサイエンス Ⅰ・Ⅱ・Ⅲ』, サイエンス社

[4] 飯尾要, (1986), 『システム思考入門』, 日本評論社。

[5] 石田晴久, (1986), 『MS-DOS』, 産業図書。

[6] 伊藤誠他, (1985), 「パイの多桁計算マーチンの公式を使って」, 『My Computer No. 18』, CQ出版社。

[7] 出雲敏彦, (1985), 「システム要求仕様の標準化について」, 『鈴鹿短期大学紀要 Vol. 5』, 鈴鹿短期大学。

[8] Kemeny. J. G. & Kurtz, T. E., (1985), *Back to BASIC,* Addison-Wesley.

[9] 森口繁一, 伊理正夫, (1985), 『算法通論 第2版』, 東京大学出版会。

[10] 日本数学会編, (1975), 『数学辞典 第2版』, 岩波書店。

[11] 日本電気㈱, (1986), 『N_{88}/MS-DOS FILECONV』, 日本電気株式会社。

[12] 日本電気㈱, (1986), 『PC-9800 Software Library MS-DOS$_{3.1}$』, 日本電気株式会社。

[13] Rohatgi, Vijay K., (1972), *Statistical Inference,* John Wiley & Sons.

[14] 鳳誠三郎編, (1985), 『コンピュータ・システム』, オーム社。

[15] 芹沢正三訳, (1981), 『ハミング 数値計算入門』, みすず書房。

[16] 庄司渉, (1984), 『計算解析のためのBASICテキスト』, 現代数学社。

[17] 田尾陽一・清水韶光訳, (1973), 『ベックマンπの歴史』, 蒼樹書房。

[18] Tukey, J. W., (1977), *Exploratory Data Analysis,* Addison-Wesley.

[19] 渡辺茂監修, (1986), 『マイコン徹底研究』, 日本経済新聞社。

[20] 高木貞治, (1983), 『解析概論 改訂第三版』, 岩波書店。