Quora ZOOM科学会

円周率とコンピュータ技術

～ 円周率の魅力に迫る ～

2020年10月10日

Tokieda Yukinobu

Agenda

円周率計算のトレンドの変遷, 説明者の円周率計算実績 を説明/紹介し, 円周率の魅力に迫る。

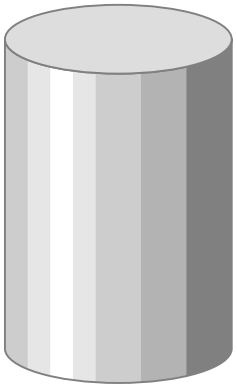
1. はじめに
2. 正多角形によるアプローチ
3. 逆正接関数によるアプローチ
4. 算術幾何平均によるアプローチ
5. Binary Splitting Method
6. 円周率の魅力

1. はじめに

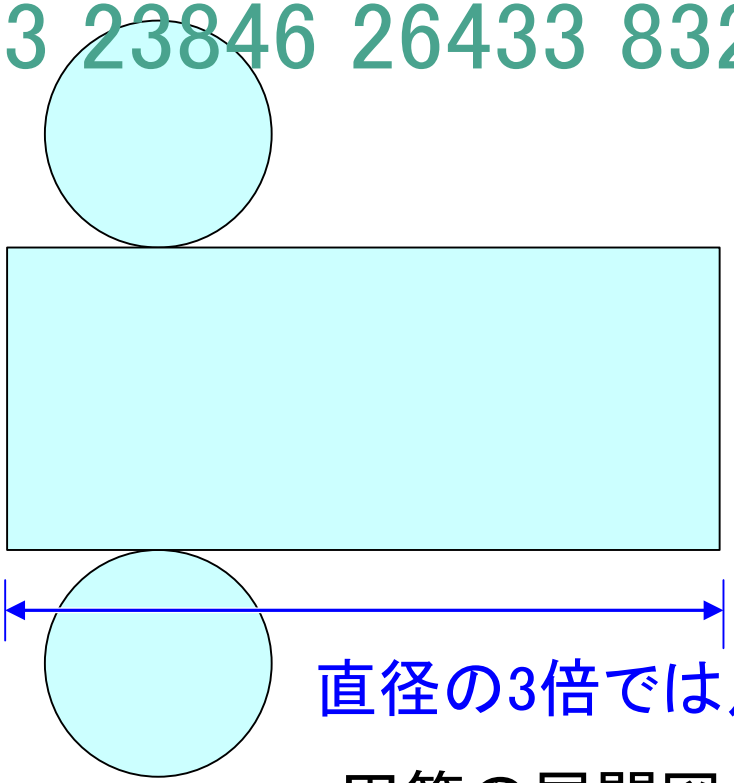
円周率とは

円周の長さを直径で割った値。(円周の長さ と 直径の比)

3.14159 26535 89793 23846 26433 83279 ...



円筒



直径の3倍では足りない...

円筒の展開図

「円周率を1億桁まで計算」ってどういうことだろう???

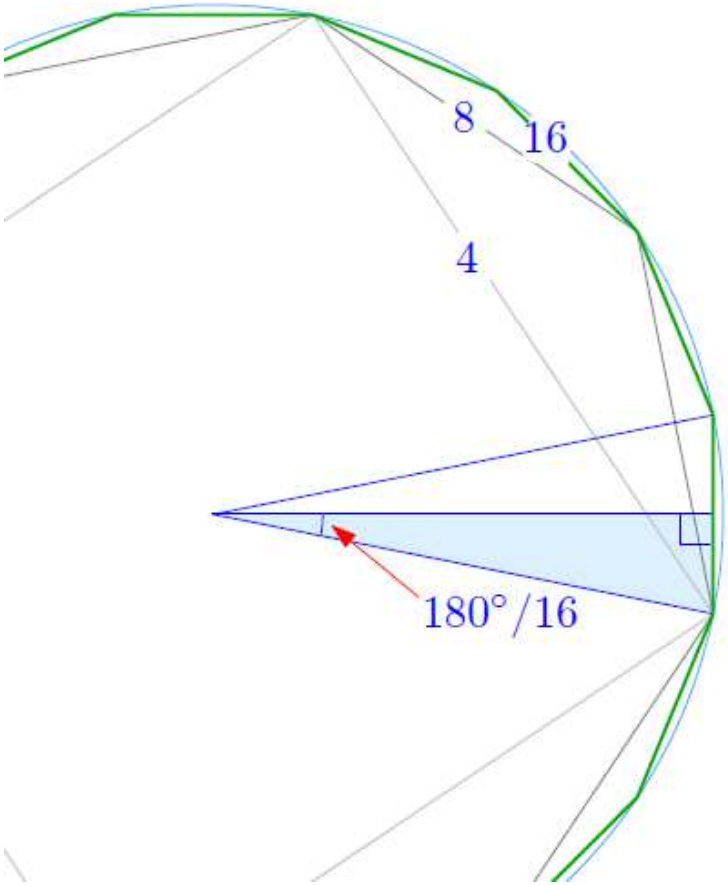
1. はじめに

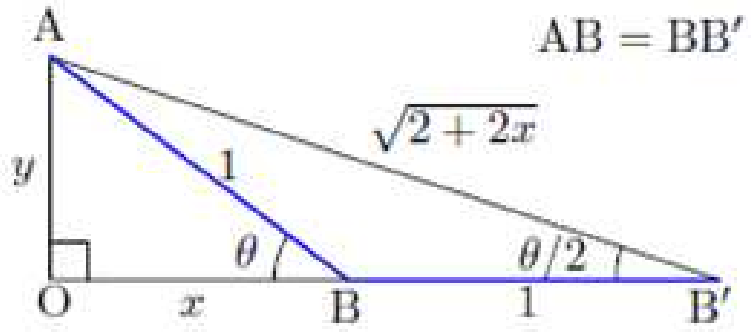
古典における円周率特定

年代	地域・人名	特定した値
BC 2000	バビロニア	$3 + \frac{1}{8}$ (= 3.125)
BC 3世紀	ギリシア アルキメデス	$3 + \frac{10}{71} < \pi < 3 + \frac{1}{7}$ (3.140845... < π < 3.142857...)
2世紀	ギリシア プロトレマイオス	$\frac{377}{120}$ (= 3.14166...)
5世紀	隋 (中国) 祖沖之	$\frac{355}{113}$ (= 3.141592920...)
1220年	イタリア フィボナッチ	$\frac{864}{275}$ (= 3.141818...)

アルキメデスは, 正96角形の周囲長を評価して
円周率の範囲を特定した。

2. 正多角形によるアプローチ





「2分の1倍の角度をつくる」

高校数学の参考書「なべつぐの あすなろ数学」演習問題より

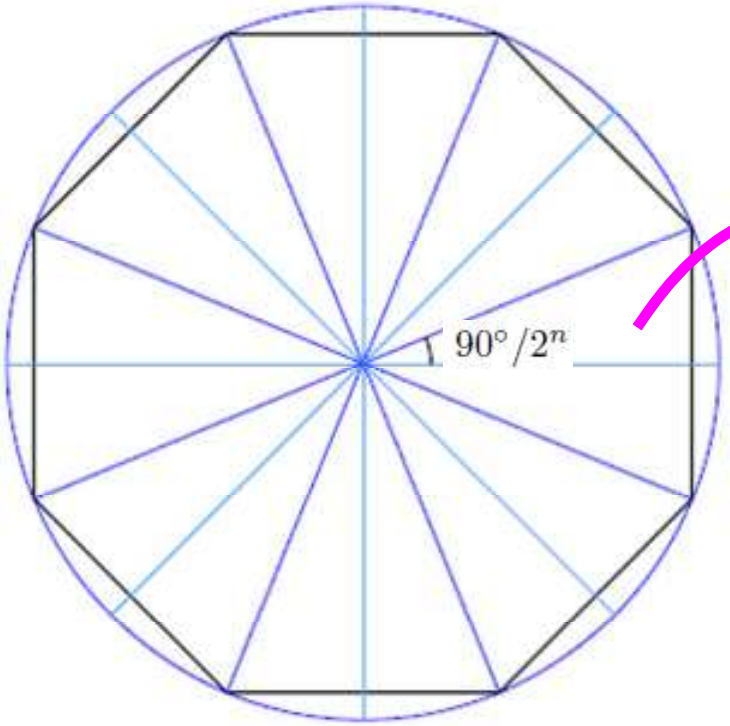
$$\sin\frac{\theta}{2}=\sqrt{\frac{1-x}{2}} \qquad \cos\frac{\theta}{2}=\sqrt{\frac{1+x}{2}}$$

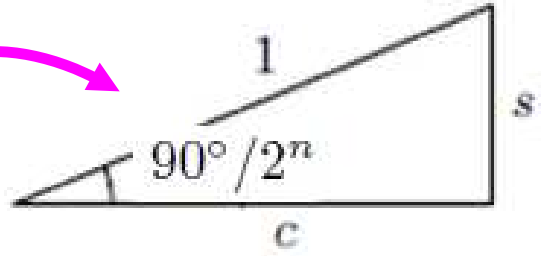
半角の公式

$$\sin\frac{90^\circ}{2^n}=\frac{\sqrt{2-\sqrt{2+\sqrt{2+\cdots+\sqrt{2}}}}}{2} \qquad \cos\frac{90^\circ}{2^n}=\frac{\sqrt{2+\sqrt{2+\sqrt{2+\cdots+\sqrt{2}}}}}{2}$$

根号の中に「2」が n 個

2. 正多角形によるアプローチ





半径1の円（直径2の円）に内接する
正 2^{n+1} 角形の周囲長 = $2^{n+2}s$

円周率 $\pi \simeq 2^{n+1}s = 2^{n+1}\sin\dfrac{90°}{2^n}$

$$= 2^n\sqrt{2-\sqrt{2+\sqrt{2+\cdots+\sqrt{2}}}}$$

根号の中に
「2」が n 個

2. 正多角形によるアプローチ

MATLAB code

```
a=sqrt(2.0);
for k = 1: 20;
    b=sqrt(2-a);  a=sqrt(2+a);
    str = sprintf('%15.15f', b*2^(k+1));
    disp(str);
end;
```

実行結果

```
3.061467458920718
3.121445152258053
3.136548490545941
3.140331156954739
3.141277250932757
3.141513801144145
3.141572940367883
3.141587725279961
3.141591421504635
3.141592345611077
3.141592576545004
3.141592633463248
3.141592654807589
3.141592645321215
3.141592607375720
3.141592910939673
3.141594125195191
3.141596553704820
3.141596553704820
3.141674265021758
```

このアルゴリズムは桁落ちしやすい。

逆数をとると...

$$\frac{2}{\pi} = \sqrt{\frac{1}{2}}\sqrt{\frac{1}{2}+\frac{1}{2}\sqrt{\frac{1}{2}}}\sqrt{\frac{1}{2}+\frac{1}{2}\sqrt{\frac{1}{2}+\frac{1}{2}\sqrt{\frac{1}{2}}}}\cdots$$

ヴィエトの公式（桁落ちしにくい）

25回反復結果 → 3.141592653589793

日本では関孝和が使用した... はず。

3. 逆正接関数によるアプローチ

解析的アプローチで円周率が計算できる。円周率計算の革命

$$\arctan x = \int \frac{\mathrm{d}x}{1+x^2} = \int (1 - x^2 + x^4 - x^6 + x^8 - \cdots)\,\mathrm{d}x$$

$$= x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

ライプニッツ級数
収束が著しく遅い

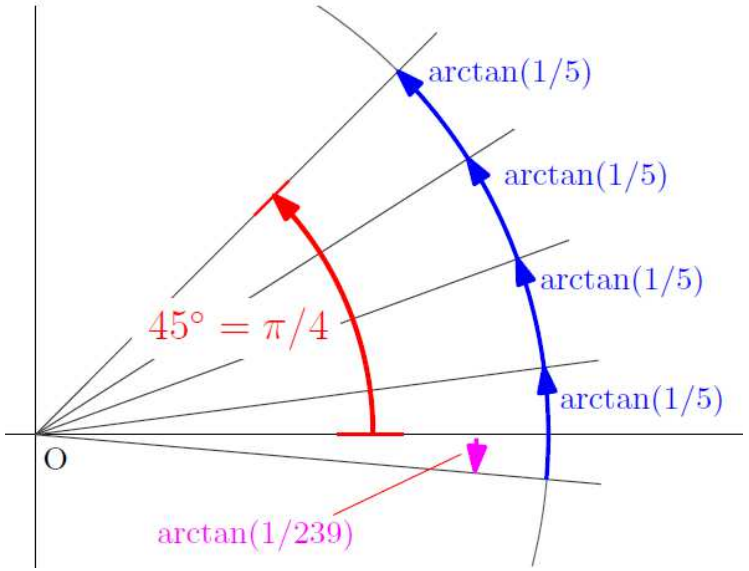
単純に応用すると…

$$\frac{\pi}{4} = \arctan 1 = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$$

倍角の公式
加法定理

$$\frac{\pi}{4} = 4\arctan\frac{1}{5} - \arctan\frac{1}{239}$$

マチンの公式



3. 逆正接関数によるアプローチ

マチンの公式は単純で収束が速い。

$$\frac{\pi}{4} = 4\arctan\frac{1}{5} - \arctan\frac{1}{239}$$

$$= \frac{4}{5}\left(1 - \frac{1}{3\cdot 25} + \frac{1}{5\cdot 25^2} - \cdots\right) - \frac{1}{239}\left(1 - \frac{1}{3\cdot 57121} + \frac{1}{5\cdot 57121^2} - \cdots\right)$$

項を追加するたびに
1.40桁の精度改善

項を追加するたびに
4.76桁の精度改善

年代	計算者	桁数	備考
1706年	ジョン・マチン	100桁	
1789年	ユリ―・ベガ	137桁	140桁まで計算
1850–1873年	ウィリアム・シャンクス	527桁	707桁まで計算
1947年	ENIAC	2,037桁	

3. 逆正接関数によるアプローチ

Y. Tokieda による計算

1988年 NEC PC-8801 mkII SR　CPU 4MHz
マチンの公式　アセンブリ言語
計算桁 2,000桁　　計算時間　2時間

衝撃の事実
乗算命令・除算命令がない。
仕方ないのでアセンブリ言語で自作。

1991年 大学の計算機を使って
(C言語) で2000桁
1997年 Windows で
C言語 + インラインASM

2020年 右プログラム →
を再実行。

Core i7-8565U（1.8GHz）

10万桁	8.6秒
20万桁	34.5秒
100万桁	916.8秒

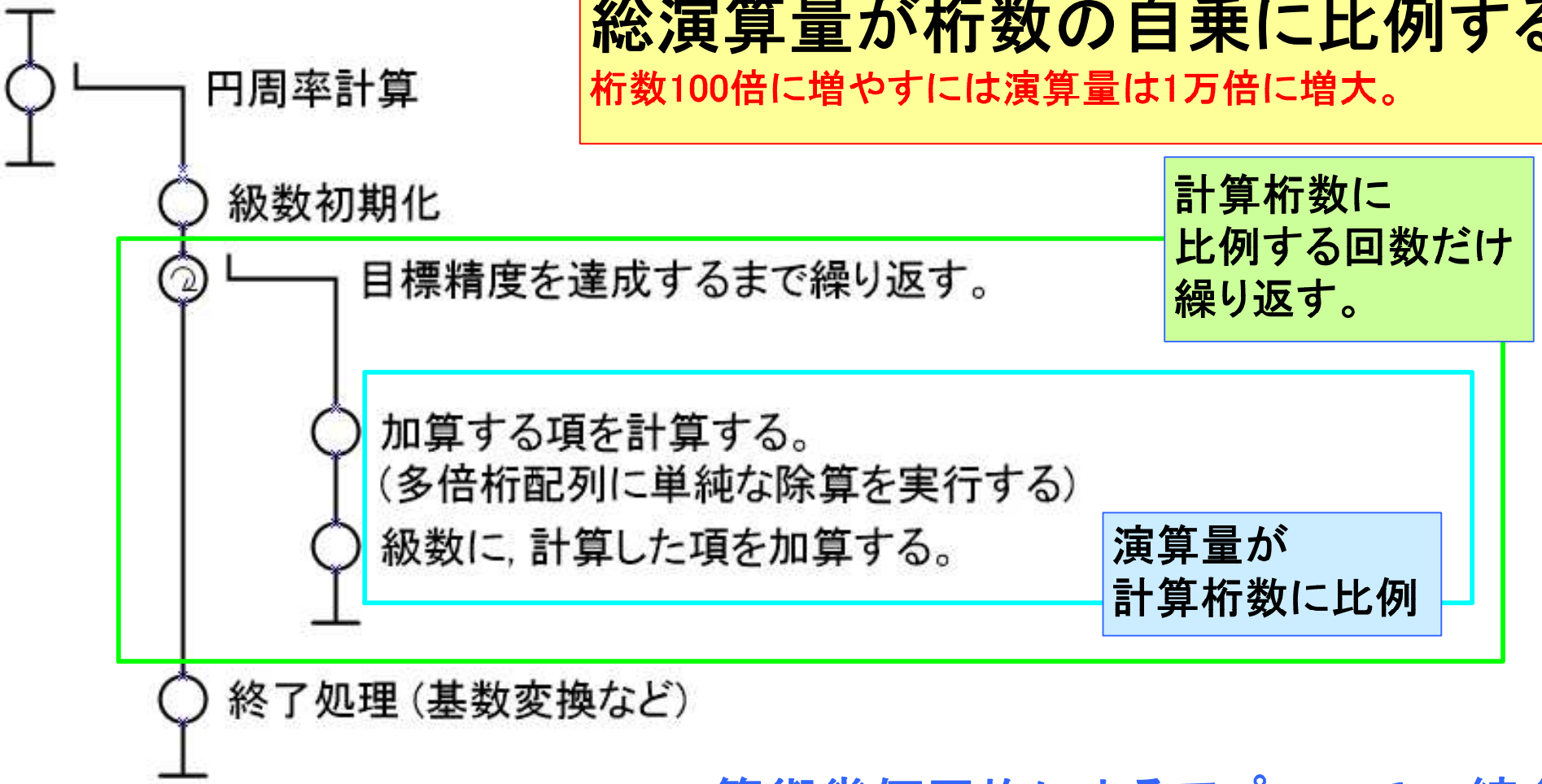
1997年版プログラムの一部

```
    } /* <----- end _asm */
}
void Sub(DWORD *lpdwDst, DWORD *lpdwSrc, DWORD dwNum)
{
    _asm {
        mov     edi, DWORD PTR lpdwDst  ; edi = lpdwDst;
        mov     esi, DWORD PTR lpdwSrc  ; esi = lpdwSrc;
        mov     ecx, DWORD PTR dwNum    ; CF  = 0;
        sub     edi, 4
        or      eax, eax                ; for (ecx=dwNum; ecx != 0; ecx--) {
LP00:   lea     edi, [edi+4]            ;     edi += 4;
        mov     eax, DWORD PTR [esi]    ;     eax = *((DWORD *)esi);
        lea     esi, [esi+4]            ;     esi += 4;
        sbb     DWORD PTR [edi], eax    ;     *((DWORD *)edi) -= (eax + CF);
        dec     ecx                     ; }
        jnz     LP00                    ;
    } /* <----- end _asm */
}
void Mul(DWORD *lpdwDst, DWORD *lpdwSrc, DWORD dwMul, DWORD dwNum)
{
    _asm {
        mov     edi, DWORD PTR lpdwDst  ; edi = lpdwDst;
        mov     esi, DWORD PTR lpdwSrc  ; esi = lpdwSrc;
        mov     ecx, DWORD PTR dwNum    ; edx = 0;
        sub     edi, 4
        sub     edx, edx                ; for (ecx=dwNum; ecx != 0; ecx--) {
LP00:   mov     eax, DWORD PTR dwMul    ;     eax = dwNum;
        add     edi, 4                  ;     edi += 4;
        mov     ebx, edx                ;     ebx = edx;
        mul     DWORD PTR [esi]         ;     edx:eax = *((DWORD *)esi) * eax;
        add     esi, 4                  ;     esi += 4;
        add     eax, ebx                ;     edx:eax += ebx;
        adc     edx, 0                  ;
        mov     DWORD PTR [edi], eax    ;     *((DWORD)edi) = eax;
        dec     ecx                     ;
        jnz     LP00                    ; }
```

3. 逆正接関数によるアプローチ

円周率の桁数合戦にアルゴリズムがついていけなくなった。
世界記録が 100万桁に達すると(1973年), 計算桁数が伸びなくなった。

総演算量が桁数の自乗に比例する。
桁数100倍に増やすには演算量は1万倍に増大。



計算桁数に
比例する回数だけ
繰り返す。

演算量が
計算桁数に比例

算術幾何平均によるアプローチへ続く。

4. 算術幾何平均によるアプローチ

1973年, サラミンとブレントが独立に **ガウス・ルジャンドル法** を提唱した。そのアルゴリズムが, 1980年代以降, 円周率の計算桁を飛躍的に増加させることになる。(楕円積分におけるルジャンドルの関係式を利用)

ガウス・ルジャンドル法の基本形

- 初期値として $a_0 = 1, b = 1/\sqrt{2}$ を設定する。
- 次の漸化式 (算術幾何平均) を用いて, a_n, b_n を更新する。

$$a_{n+1} = \frac{a_n + b_n}{2}, \qquad b_{n+1} = \sqrt{a_n b_n}.$$

- 十分に更新したら次の数式で円周率を計算する。

$$\pi_n = \frac{2a_n^2}{1 - \sum_{i=0}^{n} 2^i (a_i^2 - b_i^2)},$$

漸化式を適用するたびに有効桁が2倍増大する。

4. 算術幾何平均によるアプローチ

基本形の実装による
ガウス・ルジャンドル法による演算経過

Loop 1: pi = 4.00000 00000 00000 00000 00000 00000 00000 00000 00000 00000
00000 00000 00000 00000 00000 00000 00000 00000 00000 00000

Loop 2: pi = 3.18767 26427 12108 62720 19299 70525 36923 26510 53571 85936
92264 87633 98627 51228 32528 12233 01147 28610 66016 17974

Loop 3: pi = 3.14168 02932 97653 29391 80704 24560 00938 27957 19438 81540
28326 44189 46319 56630 01010 25531 93888 89427 51526 46103

Loop 4: pi = 3.14159 26538 95446 49600 29147 58818 04348 61088 79237 26131
15896 51101 35768 46530 79503 08650 17740 97586 28986 31570

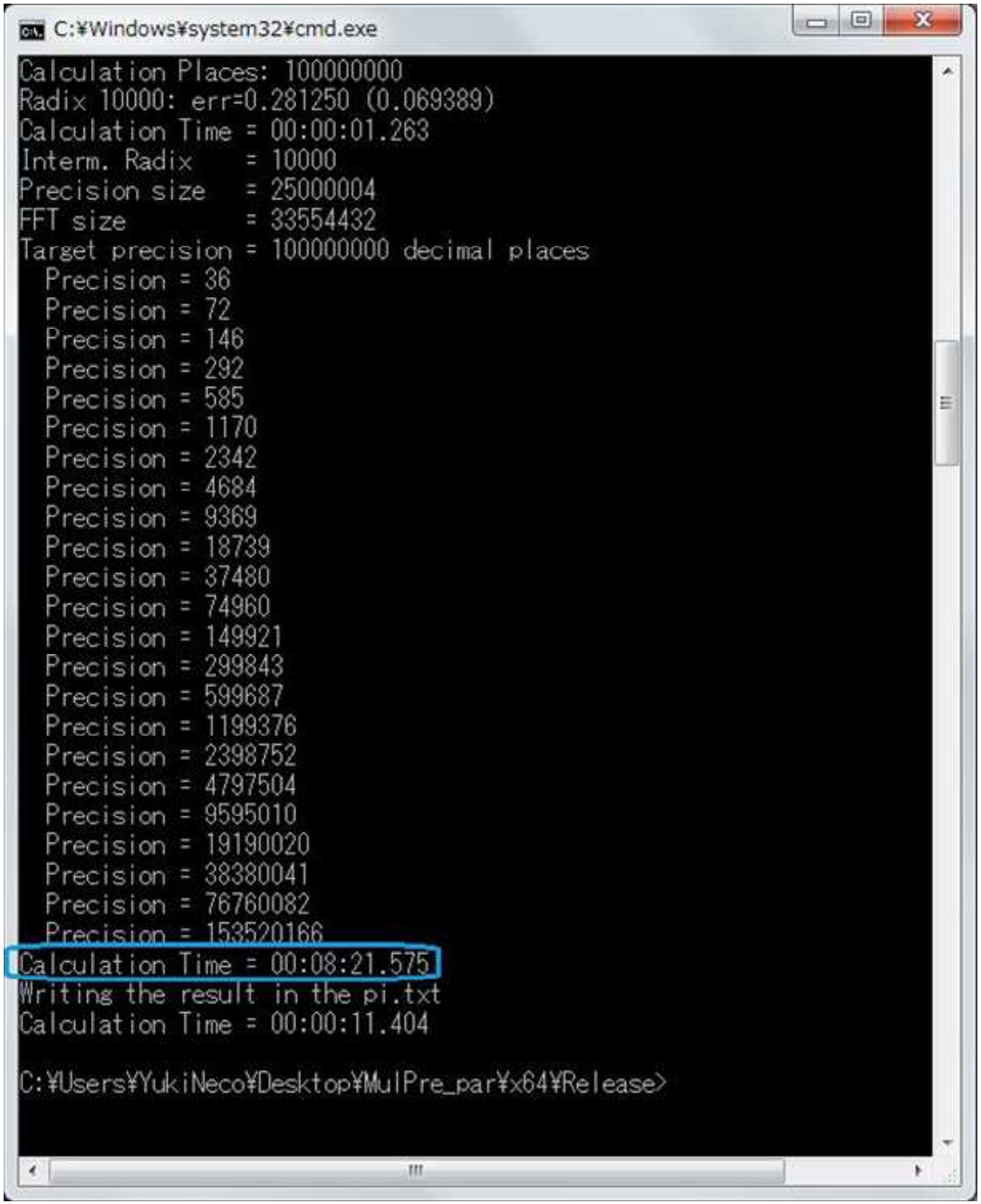
Loop 5: pi = 3.14159 26535 89793 23846 63606 02706 63132 17577 02411 34242
93564 86846 01523 84109 48606 92775 82680 62200 73327 62131

Loop 6: pi = 3.14159 26535 89793 23846 26433 83279 50288 41971 69949 16472
66058 34696 12594 87480 06095 32900 58518 51575 93171 01939

Loop 7: pi = 3.14159 26535 89793 23846 26433 83279 50288 41971 69399 37510
58209 74944 59230 78164 06286 20899 86280 46852 22865 41150

Loop 8: pi = 3.14159 26535 89793 23846 26433 83279 50288 41971 69399 37510
58209 74944 59230 78164 06286 20899 86280 34825 34211 70679

4. 算術幾何平均によるアプローチ



Y. Tokieda による
1億桁の円周率計算

改良型 ガウス・ルジャンドル法
京都大学 大浦拓哉先生の方法

最初の漸化式で36桁精度。

漸化式1回で精度は2倍。

1億桁を8分21秒で計算。

Core i7 Extreme (6 Core)
2012年

4. 算術幾何平均によるアプローチ

$$a_{n+1} = \frac{a_n + b_n}{2}$$

$$b_{n+1} = \sqrt{a_n b_n}$$

ガウス・ルジャンドル法の実装は大掛かり

多倍長変数の乗算

筆算アルゴリズムでは，演算量が桁数の自乗に比例する。フーリエ変換を応用し，$O(n \log n)$ の演算量で乗算を実現する。

平方根

ニュートン法を利用し，目標桁まで急速に収束させる。
しかも，除算を使わないアルゴリズムを使う。

除算

除算は「逆数を乗じる」と考える。
ニュートン法を利用し，目標桁まで
急速に収束させる。

$$\pi_n = \frac{2a_n^2}{1 - \sum_{i=0}^{n} 2^i (a_i^2 - b_i^2)}$$

4. 算術幾何平均によるアプローチ

フーリエ変換の高速化

高速アルゴリズム採用

Split Radix アルゴリズムを採用し，
乗算回数を基数2アルゴリズムから
20%削減。

SIMD命令による複素演算の高速化

実部と虚部の演算並列化。
データ転送高速化。

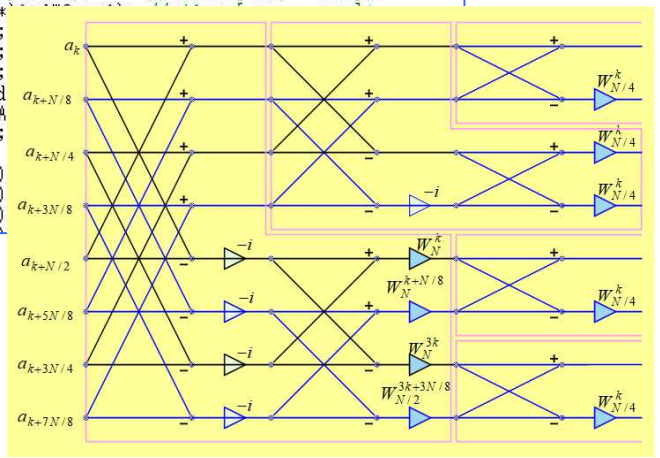
```
    i2 = i1 + uLenH;
    i3 = i2 + uLenH;

    mdA0 = _mm_load_pd(&lpdfC[i0]);
    mdA1 = _mm_load_pd(&lpdfC[i1]);
    mdA2 = mdA0;
    mdA3 = mdA1;
    mdA3 = _mm_shuffle_pd(mdA3, mdA3, 0x01);
    mdA3 = _mm_xor_pd(mdA3, mdPm);

    mdX0 = _mm_add_pd(mdA0, mdA1);
    mdX1 = _mm_sub_pd(mdA0, mdA1);
    mdX2 = _mm_add_pd(mdA2, mdA3);
    mdX3 = _mm_sub_pd(mdA2, mdA3);

    mdA0 = _mm_loaddup_pd((double*)&mdW1);     // A0 = [ wx,   wx ]
    mdA1 = _mm_loaddup_pd((double*)&mdW1 + 1); // A1 = [ wy,   wy ]
    mdA0 = _mm_xor_pd(mdA0, mdMm);              // A0 = [-wx,  -wx ]
    mdA0 = _mm_mul_pd(mdA0, mdX2);              // A0 = [-wx x,  -wx y ]
    mdX2 = _mm_mul_pd(mdX2, mdA1);              // X  = [ wy x,   wy y ]
    mdA0 = _mm_shuffle_pd(mdA0, mdA0, 0x01);    // A0 = [-wx y,  -wx x ]
    mdX2 = _mm_addsub_pd(mdX2, mdA0);           // X  = [ wy x + wx y,   wy
    mdA0 = _mm_loaddup_pd((double*)&mdW3);     // A0 = [ wx,   wx ]
    mdA1 = _mm_loaddup_pd((double*
    mdA0 = _mm_xor_pd(mdA0, mdMm);
    mdA0 = _mm_mul_pd(mdA0, mdX3);
    mdX3 = _mm_mul_pd(mdX3, mdA1);
    mdA0 = _mm_shuffle_pd(mdA0, md
    mdX3 = _mm_addsub_pd(mdX3, mdA
    mdX3 = _mm_xor_pd(mdX3, mdMm);

    _mm_store_pd(&lpdfC[i0], mdX0)
    _mm_store_pd(&lpdfC[i1], mdX1)
    _mm_store_pd(&lpdfC[i2], mdX2)
```




```
    s_aThdPrm[2].uLen = uLen;
    s_aThdPrm[2].lpfnProc = fft_Exec;

    // fft_Exec (lpdfC + 6 * uLen, uLen, lpdfW + uLen);
    s_aThdPrm[3].lpdfC = lpdfC + 6 * uLen;
    s_aThdPrm[3].lpdfW = lpdfW + uLen;
    s_aThdPrm[3].uLen = uLen;
    s_aThdPrm[3].lpfnProc = fft_Exec;

    PostThreadMessage(s_adwThdId[0], WM_USER, 0, 0);
    PostThreadMessage(s_adwThdId[1], WM_USER, 0, 0);
    PostThreadMessage(s_adwThdId[2], WM_USER, 0, 0);
    PostThreadMessage(s_adwThdId[3], WM_USER, 0, 0);

    /**** Sleeps until bufferfly computation is done. ****/
    WaitForMultipleObjects(4, s_ahEvent, TRUE, 1000);
    ResetEvent(s_ahEvent[0]);
    ResetEvent(s_ahEvent[1]);
```

積極的なマルチスレッド化

バタフライ演算を並列化し，
大幅な高速化を図る。

フーリエ変換で乗算が実現できる説明は長くなるので，残念ながら割愛。

5. Binary Splitting Method

2000年あたりで，再び，級数展開に基づくアルゴリズムが浮上した。その代表がチュドノフスキーの公式。

$$\frac{1}{\pi} = 12\sum_{k=0}^{\infty} \frac{(-1)^k\,(6k)!\,(13591409 + 545140134k)}{(3k)!\,(k!)^3\,640320^{3k+3/2}}$$

- 新たなアプローチでは，分数のまま計算する。
- トーナメント戦の要領で級数を計算する。(これが Binary Splitting)
- 不必要に大きな桁で計算しない。
- 計算の進行とともに計算桁が大きくなるので，総演算量が抑えられる。

年代	計算者	桁数	備考
2009年	ベラール	2.7兆桁	PC利用
2013年	近藤，イー	12.1兆桁	PC利用
2019年	Google	31.4兆桁	クラウド利用
2020年	マリカン	50兆桁	PC利用

6. 円周率の魅力

数値そのものよりも, 円周率特定のために様々な技術が得られることが大きな魅力。(私自身, 円周率の数値に興味はない ^^;)

演算精度 (誤差解析)

雑音解析との類似性 → 通信システム etc.

新アルゴリズムの獲得

高速フーリエ変換 → 画像処理, レーダ, etc.
巨大桁の乗算・除算 → 暗号技術

並列演算・コンピュータ アーキテクチャ

効率的な演算振り分け高速化
→ マルチスレッド, GPU, FPGA, カスタムCPU

人類の円周率への挑戦は, 技術進展に貢献。
桁数の伸展とともにトレンドが変わる。だから面白い。

円周率への挑戦は, **ロマン** にあふれている。

ご興味ありましたら,
こちら↓ もよろしくお願いします。 ^___^

Quora で検索

キーワード: 円周率

ライター: Tokieda

Scientific Doggie
数理の楽しみ

http://www.wannyan.net/scidog/