2014

π Computation

INDIVISUAL PROJECT

ZHAOFAN QIU (MSR STUDENT-PERSON CONSULTING)

一、 时间安排

作为我们高级软件工程（ASE）课程的个人项目作业，圆周率的计算是一个十分具有挑战性的课题。为了帮助自己更好的进行本次作业完成，对接下来一周的时间进行分配如下：

时间	事宜	内容
9.3 ~ 9.5	课题调研	以互联网检索作为基本方式进行课题调研
9.6 ~ 9.7	代码编写	完成实现代码，确保正确性
9.8 ~ 9.9	性能优化	对算法进行性能优化并撰写报告

其中课题调研花费的时间较长，原因如下：

1、 具体算法是限制该程序性能的最主要因素，需要一定时间调研及阅读论文来分析各类算法的优劣。

2、 平时还有部分 mentor 布置的任务需要完成，不适宜进行大工作量任务(如代码编写)。

3、 调研过程中同时完成代码框架设计和接口定义的工作，为后期代码编写过程和最后代码的可读性将会带来极大的便利。

在 9.6 ~ 9.7 两天完成代码编写的工作之后，将会集中进行代码效率提升的工作，使用算法改进、指令集优化和 VS 性能分析等工具，在保证正确

性的前提下，对程序进行性能提升，以完成快速计算的需求。其中具体优化项目如下：

1、 算法优化；

2、 指令集优化；

3、 并行化优化；

4、 减少内存申请优化；

5、 减少变量传递优化；

6、 细节优化。

通过以上步骤的优化，最终实现对 π 的小数点后 1M 位的精确计算可以在 12.245 s 完成，速度较为理想。具体性能如下:

有效位数（n）	时间（计算时间/计算+打印时间）
1000（1K）	0.015 / 0.015
10000（10K）	0.031 / 0.031
100000（100K）	0.249 / 0.328
1000000（1M）	5.218 / 12.245

二、 课题调研

作为最基本的课题调研方式，通过对互联网检索和论文检索的调研，总结出以下几种传统利用程序计算π的方法（代码均用 Matlab 实现）。

1、使用 Matlab 自带的 pi

```
digits(100)
vpa(pi)
```

2、刘徽割圆法

```
function y=calpi(n)
syms a;
for i=1:n
   a=sqrt(2-sqrt(4-a^2));
end
a=subs(a,'a','1');
y=3*2^n*vpa(a,n+5);
```

3、反正切级数

反正切级数 1

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots + (-1)^{n-1}\frac{x^{2n-1}}{2n-1} + \cdots$$

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \cdots + (-1)^{n-1}\frac{1}{2n-1} + \cdots$$

```
function y=calpi1(k)
for n=1:k
   a(n)=(-1).^(n-1)./(2*n-1);
end;
4*sum(a)
```

反正切级数 2

$$\arctan\frac{1}{2}+\arctan\frac{1}{3}=\frac{\pi}{4}$$

$$\pi = 4[\frac{1}{2}-\frac{1}{3}(\frac{1}{2})^3+\frac{1}{5}(\frac{1}{2})^5-\cdots+\frac{(-1)^{n-1}}{2n-1}(\frac{1}{2})^{2n-1}+\cdots$$
$$+\frac{1}{3}-\frac{1}{3}(\frac{1}{3})^3+\frac{1}{5}(\frac{1}{3})^5-\cdots+\frac{(-1)^{n-1}}{2n-1}(\frac{1}{3})^{2n-1}+\cdots]$$

```
function y=calpi2(k)
for n=1:k
   a(n)=(-1).^(n-1)*(1/2).^(2*n-1)./(2*n-1)+(-1).^(n-
1)*(1/3).^(2*n-1)./(2*n-1);
end;
vpa(4*sum(a))
```

4、数值积分法

$$A=4\int_0^1\frac{1}{1+x^2}dx=\pi$$

```
function y=fun(x)
y=4./(1+x.^2);
```

```
function y=calpi3(k)
for n=1:k-1
    a(n)=2*fun(n/k);
end;
vpa(1/(2*k)*(sum(a)+fun(0)+fun(1)))
```

5、蒙特卡洛方法

使用求圆面积的蒙特卡洛方法。

```
function y=calpi4(k)
m=0;
for n=1:k
    if rand(1)^2+rand(1)^2<=1
        m=m+1;
    end;
end;
4*m/k
```

以上调研的方法，除了使用 Matlab 自带的 π 之外，运行速度均较慢，难以满足高速计算 π 的需求，但是上述算法的思想均有可以借鉴之处，有利于我们更深入理解对 π 的计算。

为了完成更高效率的高精度 π 的计算，调研总结出以下三个较为理想的实现算法。

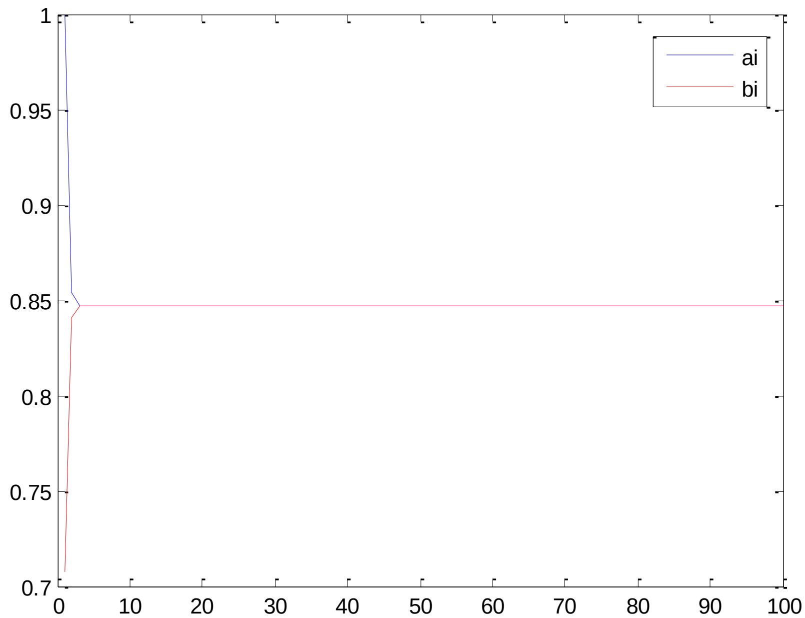
6、算数几何平均方法（AGM）

算数几何平均定义如下：

$$a_n = \tfrac{1}{2}(a_{n-1} + b_{n-1}), \qquad b_n = (a_{n-1} b_{n-1})^{1/2}$$

$$c_n = ½(a_{n-1} - b_{n-1})$$

算数几何平均收敛极快，下图为$a_0 = 1$，$b_0 = \frac{\sqrt{2}}{2}$时收敛图像，



横坐标为迭代次数，纵坐标为两者取值，可以看出，收敛极快，利用这一点可以快速计算π的取值。

$$\mathrm{agm}(a_0, b_0) = \lim a_n = \lim b_n$$

$$\pi = \frac{4\,\mathrm{agm}(1, k)\mathrm{agm}(1, k')}{1 - \sum_{j=1}^{\infty} 2^j(c_j^2 + c_j'^2)}$$

利用该公式的级数形式

$$\pi_{nn'} = \frac{4a_{n+1}a'_{n'+1}}{1 - \sum_{j=1}^{n} 2^j c_j^2 - \sum_{j=1}^{n'} 2^j c_j'^2}$$

即可以近似求出 π，而且需要的级数项数非常少，有效位数计算公式如下

$$-\log_{10}|\pi - \pi_n| > (\pi/\log 10)2^{n+1} - n\log_{10}2 - 2\log_{10}(4\pi/\text{agm})$$

总体上有效位数随迭代次数呈指数级增长。

```
k = 100000;
digits(k);
syms a0 b0 up down ai bi ci p pp
a0 = 1;
b0 = vpa(1/sqrt(2));
pp = 0;
down = 1;
cf = vpa(2);
while(1)
    ai = (a0 + b0) / 2;
    bi = sqrt(a0 * b0);
    ci = (a0 - b0) / 2;

    cf = cf * 2;
    up = 4 * ai * ai;
    down = down - cf * ci * ci;
    p = up / down;
    if (vpa(p, k) == vpa(pp, k))
        break;
    end
    pp = p;
    a0 = ai;
    b0 = bi;
end
```

7、Chudnovsky 公式

Chudnovsky Formula: ***O(n log(n)³)***

$$\frac{1}{\pi}=\frac{\sqrt{10005}}{4270934400}\sum_{k=0}^{\infty}(-1)^k\frac{(6k)!}{(k!)^3(3k)!}\frac{(13591409+545140134k)}{640320^{3k}}$$

8、Ramanujan 公式

Ramanujan's Formula: ***O(n log(n)³)***

$$\frac{1}{\pi}=\frac{2\sqrt{2}}{9801}\sum_{k=0}^{\infty}\frac{(4k)!}{k!^4}\frac{(1103+26390k)}{396^{4k}}$$

以上两个级数公式是计算 π 的级数中收敛较快的两个，经常用于大规模 π 有效位数的计算，由于这两个公式在收敛速度、计算复杂度、实现复杂度上均没有太大区别，故技术法的实现均考虑 Chudnovsky 公式。

级数法和 AGM 迭代法的区别在于，级数法的求和量较大，随 n 线性增长，但是每个求和项计算较为便捷；AGM 迭代收敛速度快，迭代次数非常少，但是每次迭代的计算量较大。从以往的实现情况和论文中的描述来看，级数法会相对占有一定的优势。所以在之后的实现过程中优先实现级数法进行 π 的计算。

```
function [Pab, Qab, Tab] = binarysplitting(a,b)
    if b - a == 1
        sa = vpa(a);
        if a == 0
            Pab = vpa(1);
            Qab = vpa(1);
        else
            Pab = (6*sa-5)*(2*sa-1)*(6*sa-1);
            Qab = sa*sa*sa*(vpa(640320)^3)/24;
        end
        Tab = Pab * (13591409 + 545140134*sa);
        if (bitand(a,1))
            Tab = -Tab;
        end
    else
        m = uint32((a + b) / 2);
        [Pam, Qam, Tam] = binarysplitting(a, m);
        [Pmb, Qmb, Tmb] = binarysplitting(m, b);
        Pab = Pam * Pmb;
        Qab = Qam * Qmb;
        Tab = Qmb * Tam + Pam * Tmb;
    end
end


k = 1000;
digits(k);
C3_OVER_24 = vpa(640320)^3/24;
DIGITS_PER_TERM = log(C3_OVER_24/6/2/6) / log(10);
N = uint32(k/DIGITS_PER_TERM + 1);

[P, Q, T] = binarysplitting(0, N);
res = 426880 * vpa(sqrt(10005)) * Q / T;
```

三、具体实现

为了使用 C++实现以上调研中选定的算法（Chudnovsky 公式），将模块设计分为以下几个部分：

1、 数字表示部分；

2、 高精度乘法部分；

3、 平方根倒数部分；

4、 除法部分；

5、 进制转化部分。

为了完成 Chudnovsky 公式中级数求和的部分，采用 Binary Splitting 的方法，对求和公式进行拆分。具体拆分方式参考网页（http://www.craig-wood.com/nick/articles/pi-chudnovsky/）。

1、数字表示

使用高精度整数和高精度浮点数两个类型表示数字，具体方式如下：

$$BigInteger = A[0] + A[1]*base^1 + A[2]*base^2 + A[3]*base^3 + \dots$$

$$BigFloat = sign * (A[0] + A[1]*base^1 + A[2]*base^2 + A[3]*base^3 + \dots) * base^{exp}$$

在实现时由于机器为 64 位系统采用$base = 2^{64}$进制，以加快运算速度。

2、高精度乘法

常用的高精度乘法列表及运算效率如下：

Algorithm	**Complexity**
Basecase (or grade-school) Multiplication	$O(n^2)$
Karatsuba Multiplication	$O\left(n^{\frac{\log(3)}{\log(2)}}\right) \sim (n^{1.585})$
k-way Toom-Cook Multiplication	$O\left(c(k)\, n^{\frac{\log(2k-1)}{\log(k)}}\right)$
Floating-Point Fast Fourier Transform (FFT)	$\sim O\left(\frac{n}{w-\log(n)} \log\left(\frac{n}{w-\log(n)}\right)\right)$
Schönhage–Strassen Algorithm (SSA)	$O(n\log(n)\log(\log(n)))$
Small Primes Number-Theoretic Transform (NTT)	$O(n\log(n))$

在实现过程中，实现了 Basecase、Karatsuba、FFT 三种乘法算法，分别适用于 n 较小、n 适中、n 较大三种情况，根据 n 的大小选择算法。

3、平方根倒数算法

使用牛顿迭代法求解 x 的平方根倒数。

Find:

$$r = \frac{1}{\sqrt{x}}$$

Start with an initial guess:

$$r_0 \cong \frac{1}{\sqrt{x}}$$

Iterate:

$$r_{n+1} = r_n - \frac{r_n^2 x - 1}{2} r_n$$

通过以上迭代算法，得到 x 的平方根倒数。精确位数随迭代次数呈指数增长。

4、除法算法

将除法算法转换为求倒数算法，同样使用牛顿迭代法。

Find:

$$r = \frac{1}{x}$$

Start with an initial guess:

$$r_0 \cong \frac{1}{x}$$

Iterate:

$$r_{n+1} = r_n - (r_n x - 1)\, r_n$$

5、进制转换

进制转换使用在将$base = 2^{64}$进制转换到 10 进制输出的过程中，所以进行转换的数字是纯小数，采用连续乘法取整的方法进行进制转换，思想如下：

Working Example:

$x = 0.4125de4_{16}$

$x = x * 10 = 2.8b7aae8_{16}$
out[0] = Floor(x) = 2

$x = x * 10 = 5.72cad10_{16}$
out[1] = Floor(x) = 5

$x = x * 10 = 4.72cad10_{16}$
out[2] = Floor(x) = 4

$x = x * 10 = 4.d739a40_{16}$
out[3] = Floor(x) = 4

$x = x * 10 = 8.6840680_{16}$
out[4] = Floor(x) = 8

$x = x * 10 = 4.1284100_{16}$
out[5] = Floor(x) = 4

$x = x * 10 = 0.b928a00_{16}$
out[6] = Floor(x) = 0

out = {2,5,4,4,8,4,0} = 0.2544840

实现了以上五个部分，带入 Binary Splitting 公式后，即可求出 π 的精准小数。

四、部分结果展示

n = 10，

```
$ CutiePie.exe 10
1415926535
```

n = 100，

```
$ CutiePie.exe 100
1415926535
8979323846
2643383279
5028841971
6939937510
5820974944
5923078164
0628620899
8628034825
3421170679
```

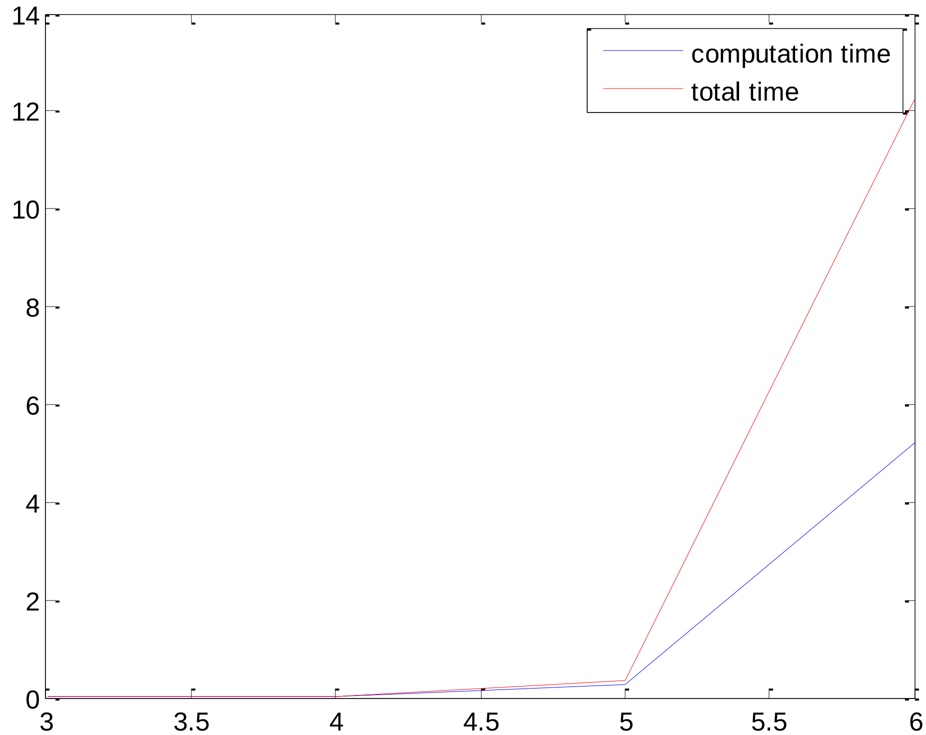
n = 1000，

```
$ CutiePie.exe 1000
1415926535
8979323846
2643383279
5028841971
6939937510
5820974944
5923078164
0628620899
8628034825
```

……

```
7669147303
5982534904
2875546873
1159562863
8823537875
9375195778
1857780532
1712268066
1300192787
6611195909
2164201989
```

运行效率，



横坐标为有效位数$log_{10}(n)$， 纵坐标为运行时间。

五、引用

[1] Yee, Alexander, and Shigeru Kondo. "10 trillion digits of pi: A case study of summing hypergeometric series to high precision on multicore systems." (2011).

[2]Salamin, Eugene. "Computation of π using arithmetic-geometric mean." Pi: A Source Book. Springer New York, 1997. 418-423.

[3] Brent, Richard P., and Paul Zimmermann. Modern computer arithmetic. No. 18. Cambridge University Press, 2011.

[4] Bernstein, Daniel J. "Scaled remainder trees." URL: http://cr. yp. to/papers. html# scaledmod. ID e2b8da026cf72d01d97e20cf2874f278. Citations in this document 18 (2004).

[5] http://www.numberworld.org/y-cruncher/

[6] http://www.craig-wood.com/nick/articles/pi-chudnovsky/