# 第一章作业

贺自怡 3140100524

## NAVIGATOR



## MAIN BODY

1 问题分析

- 分别以单精度和双精度数据类型用以下近似算法分别计算π的近似值

$$\pi = 4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\cdots\right)$$

$$\pi = 6\left(0.5+\frac{0.5^3}{2\times 3}+\frac{3\times 0.5^5}{2\times 4\times 5}+\frac{3\times 5\times 0.5^7}{2\times 4\times 6\times 7}+\cdots\right)$$

  - 要求结果具有4位有效数字；
  - 如果采用单精度数据类型要求计算结果达到机器精度，此时结果如何？采用双精度数据类型达到单精度机器精度要求以及更高的精度要求，计算结果如何？（测试机器精度：满足$1+\varepsilon>1$的最小浮点数）

这个问题可以分为分别用单、双精度使用两种算法计算两种精度的结果 8 个部分。

以 Matlab 内置的$\pi = 3.141592653589793$作为真实值。

1. 要达到 4 位有效数字，可以建立结果向量，记录每次运算结果值，对两种精度两个算法的计算结果进行对比。

   循环停止条件： $当前误差|\varepsilon_a| < 0.005\%$

   $$\varepsilon_a = \frac{当前近似值-前一近似值}{当前近似值} * 100\%$$

   $$(|\varepsilon_a| < (0.5e(2 - 有效数字))\%)$$

2. 单精度达到机器精度，双精度达到单精度机器精度及以上。首先要单精度的机器精度，因为双精度机器精度较难达到。

## 2 代码实现

2.1 精确度 4 位有效数字

### 2.1.1 算法一

#### 2.1.1.1 单精度

Code：

```
function a1single
result(1)=0;
%数组前两项定义，用于进入 while 循环
result(2)=4;
result=single(result);
%定义数组为单精度
flag=single(1);
%定义每项符号变化初值
n=2;
%定义数组下标初值
while(abs((result(n)-result(n-
1))/result(n))>=0.005e-2)
%判断是否达到 4 位有效数字
    flag=-flag;
%符号变化
 result(n+1)=result(n)+4*flag*1/(2*n-1);
%计算新的结果值
    n=n+1;
%下标增 1
end;
result(n)
%输出符合 4 位有效数字的结果
n-2
%输出循环次数
end
```

计算结果

#### 2.1.1.2 双精度

Code：

```
function a1double
result(1)=0;
%定义数组前两项，并为双精度型，用于进入
while 循环
result(2)=4;
flag=1;
%定义每项符号变化初值
n=2;
%定义数组下标初值
while(abs((result(n)-result(n-
1))/result(n))>=0.005*10^-2)
%判断是否达到 4 位有效数字
    flag=-flag;
%符号变化
 result(n+1)=result(n)+4*flag*1/(2*n-1);
%计算新的结果
    n=n+1;
%下标增 1
end;
result(n)
%输出符合 4 位有效数字的结果
n-2
%输出循环次数
end
```

计算结果

### 2.1.2 算法二

#### 2.1.2.1 单精度

Code：

```
function a2single
a=single(1); b=single(1);
ka=single(1);kb=single(1);
%设置各变量为单精度
result(1)=0;
result(2)=3;
%定义数组的前两项
result=single(result);
%设置结果数组类型为单精度
n=2;
%设置结果数组下标初值
while(abs((result(n)-result(n-
1))/result(n))>=0.005e-2)
%进入 while 循环的条件
    ka=ka*a;
    kb=kb*(b+1)*(b+2)/b;
    a=a+2; b=b+2;
%a、b 新值
 result(n+1)=result(n)+6*ka*0.5^a/kb;
%运算结果数组新值
    n=n+1;
%数组下标增 1
end;
result(n)
%输出符合具有 4 位有效数字的结果
n-2
%输出循环次数
end
```

计算结果

#### 2.1.2.2 双精度

Code：

```
function a2double
a=1;b=1;
ka=1;kb=1;
result(1)=0;
result(2)=3;
%定义数组的前两项
n=2;
%设置结果数组下标初值
while(abs((result(n)-result(n-
1))/result(n))>=0.005*10^(2-4))
%进入 while 循环的条件
ka=ka*a;
kb=kb*(b+1)*(b+2)/b;
a=a+2;  b=b+2;
%a、b 新值
result(n+1)=result(n)+6*ka*0.5^a/kb;
```

```
%运算结果数组新值
n=n+1;
%数组下标增 1
end;
result(n)
%输出符合具有 4 位有效数字的结果
n-2
%输出循环次数
end
```

计算结果

2.2 精确度达单机器精度

计算精度程序：

```
function e = mech_single
e=single(1);            %定义数据类型为单精度，使后续程序按单精度进行计算
a=1+e;                  %定义 sum 作为与 1 进行比较的和数
i=0;
while(a>1);             %当 a 大于 1 时进行运算求得机器精度
    e=e/2;
    i=i+1;
    a=1+e;
end;
e=e*2                   %得到计算机单精度机器精度
i                       %循环次数
end
```

结果

e = 1.1920929e-07

2.2.1 算法一

2.2.1.1 单精度

Code：

```
function a1_mech_single
e = mech_single;                %得到计算机单精度机器精度
result(1)=0;
%数组前两项定义
result(2)=4;
result=single(result);
%定义数组为单精度
```

```
flag=single(1);
%定义每项符号变化初值
n=2;                %定义数组下标初值
while(abs((result(n)-result(n-
1))/result(n))>=e);
%判断是否达到单精度机器精度
flag=-flag;
%符号变化
result(n+1)=result(n)+4*flag*1/(2*n-1);
%计算新的结果值
n=n+1;
%下标增 1
end;
result(n)
%输出符合条件的结果
n-2
```

计算结果

2.2.1.2 双精度

Code：

```
function a1_mech_double
e=mech_single;
%得到计算机单精度机器精度
result(1)=0;            %定义数组前两项
result(2)=4;
flag=1;      %定义每项符号变化初值
n=2;            %定义数组下标初值
while(abs((result(n)-result(n-
1))/result(n))>=e);
%判断是否达到机器精度
flag=-flag;
%符号变化
    result(n+1)=result(n)+4*flag*1/(2*n-1);
%计算新的结果
n=n+1;
%下标增 1
end;
result(n)
%输出符合条件的结果
n-2
%输出循环次数
end
```

计算结果

2.2.2 算法二

2.2.2.1 单精度

Code：

```
function a2_mech_single
e = mech_single; %得到计算机单精度机器
精度
a=single(1);b=single(1);ka=single(1);kb=sin
```

```
gle(1);
 %设置各变量为单精度
result(1)=0;
result(2)=3;        %定义数组的前两项
result=single(result);                    %设
置结果数组类型为单精度
n=2;       %设置结果数组下标初值
while(abs((result(n)-result(n-
1))/result(n))>=e); %进入 while 循环的条件
ka=ka*a;
kb=kb*(b+1)*(b+2)/b;
a=a+2; b=b+2;
%a、b 新值
result(n+1)=result(n)+6*ka*0.5^a/kb;        %
运算结果数组新值
n=n+1; %数组下标增 1
end;
result(n)
%输出符合条件的结果
n-2
%输出循环次数
```

计算结果

2.2.2.2 双精度

Code：

```
function a2_mech_double
e=mech_single;%得到计算机单精度机器精
度
a=1;b=1;ka=1;kb=1;
%设置各变量为单精度
result(1)=0;result(2)=3;
%定义数组的前两项
n=2;       %设置结果数组下标初值
while(abs((result(n)-result(n-
1))/result(n))>=e);
 %进入 while 循环的条件
ka=ka*a;
kb=kb*(b+1)*(b+2)/b;
a=a+2;b=b+2; %a、b 新值

result(n+1)=result(n)+6*ka*0.5^a/kb;
%运算结果数组新值
n=n+1; %数组下标增 1
end;
result(n)
%输出符合条件的结果
n-2
%输出循环次数
end
```

计算结果

2.3 更高精确度

2.3.1 算法一

**Code:**

```
function a1_improve_double
e = [1e-6;1e-7;1e-8;1e-9;1e-10;1e-11;1e-12];
num = size(e,1);
Result = zeros(num,2);
for i=1:num
    i
    tic
    result(1)=0;
%定义数组前两项，并为双精度型，便于进入 while 循环
    result(2)=4;
    flag=1;
%定义每项符号变化初值
    n=2;
%定义数组下标初值
    while(abs((result(n)-result(n-1))/result(n))>=e(i))
%判断是否达到机器精度
flag=-flag;
%符号变化
    result(n+1)=result(n)+4*flag*1/(2*n-1); %计算新的结果
        n=n+1; %下标增 1
    end
    Result(i,1) = result(n);
% Result(:,1)表示每种精度计算结果
    Result(i,2) = n-2;
% Result(:,2)表示每种精度迭代次数
    toc
end
figure;
plot(1:4,Result(1:4,1));
% 4 是经试验得出 4 之后的精度计算过慢而取的值
figure;
plot(1:4,Result(1:4,2));
end
```

计算结果

### 2.3.2 算法二

Code：

```
function a2_improve_double
e = [1e-6;1e-8;1e-10;1e-12;1e-14;1e-16;1e-18;1e-20];
num = size(e,1);
Result = zeros(num,2);
for i=1:num
    i
    tic
    a=1;b=1;
    ka=1;kb=1;
    result(1)=0;
    result(2)=3;
 %定义数组的前两项，且为双精度，便于进入 while 循环
    n=2;
 %设置结果数组下标初值
```

```
    while(abs((result(n)-result(n-
1))/result(n))>=e(i))      %进入 while 循环的
条件
        ka=ka*a;
        kb=kb*(b+1)*(b+2)/b;

a=a+2;
%a、b 新值
        b=b+2;

result(n+1)=result(n)+6*ka*0.5^a/kb;
%运算结果数组新值
        n=n+1;
%数组下标增 1
    end;
    Result(i,1) = result(n);
    Result(i,2) = n-2;
    toc
end
Result(:,1) = Result(:,1)-pi;
figure;
plot(1:num,Result(:,1));
figure;
plot(1:num,Result(:,2));
end
```

计算结果

## 3　计算结果

3.1 4 位有效数字精度

3.1.1　算法一

3.1.1.1 单精度

计算结果：(single) 3.1415219

循环次数：12739

相对误差：-0.002252156711418%

3.1.1.2 双精度

计算结果：3.141671189676991

循环次数：12732

相对误差：0.002499881297731%

3.1.2　算法二

3.1.2.1 单精度

计算结果：(single) 3.1415765

循环次数：5

相对误差：-0.0005141847328437621%

3.1.2.2 双精度

计算结果：3.141576715774867

循环次数：5

相对误差：-0.00050731640551251 46%

3.2 机器精度

3.2.1 算法一

3.2.1.1 单精度

计算结果：（single）3.1415968

循环次数：5592405

相对误差：0.00013198433609787 54%

3.2.1.2 双精度

计算结果：3.141592840843156

循环次数：5340354

相对误差：0.00000596045966760 2996%

3.2.2 算法二

3.2.2.1 单精度

计算结果：（single）3.1415923

循环次数：9

相对误差：0.00001125511267658055%

3.2.2.2 双精度

计算结果：3.141592622870617

循环次数：9

相对误差：-0.00000097782174454 29433%

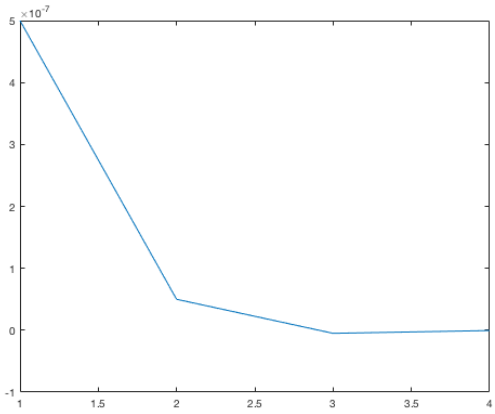
3.3 提高

3.3.1 算法一

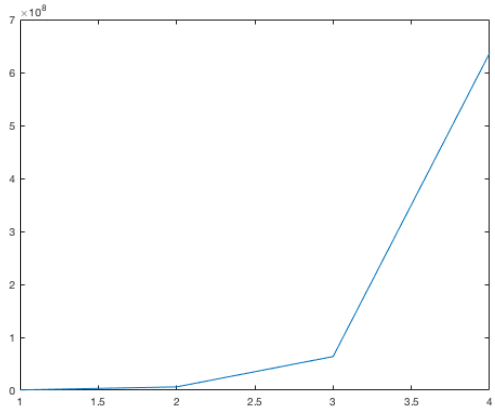
| 精度 | 1e-6 | 1e-7 | 1e-8 | 1e-9 |
|---|---|---|---|---|
| 计算时间/s | 0.106633 | 0.894212 | 7.529037 | 118.691378 |
| 循环次数 | 636620 | 6366198 | 63661977 | 636619795 |
| 相对误差 | 4.9999900e-6 | 5.0000008e-8 | 4.9999136e-9 | 5.002581e-10 |

当需要计算 1e-10 精度时，Matlab 会报错需要 8G 以上的 array。

相对误差变化曲线（横坐标 1234 分别为 1e-6 1e-7 1e-8 1e-9）：



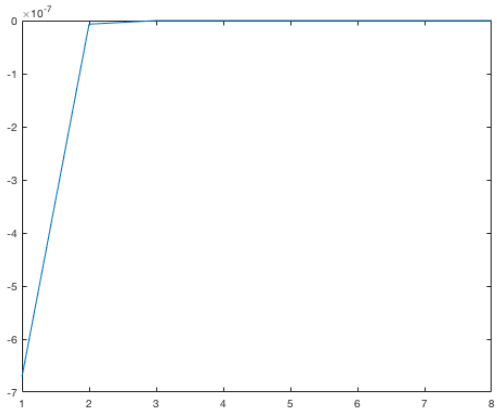

循环所需次数变化曲线：



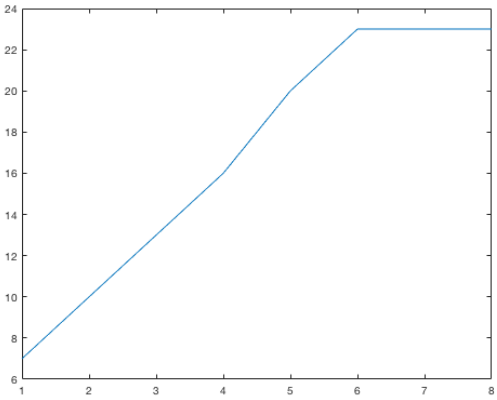

### 3.3.2 算法二

| 精度 | 1e-6 | 1e-8 | 1e-10 | 1e-12 | 1e-14 | 1e-16 | 1e-18 |
|---|---|---|---|---|---|---|---|
| 计算时间 | 0.005894 | 0.001295 | 0.000054 | 0.000035 | 0.000057 | 0.000034 | 0.000028 |
| 循环次数 | 7 | 10 | 13 | 16 | 20 | 23 | 23 |
| 相对误差 | -6.712314e-7 | -6.714232e-9 | -7.4456e-11 | -8.81e-13 | -3e-15 | 0 | 0 |

相对误差变化曲线（横坐标为 1e-6 到 1e-18）：




循环所需次数变化曲线：




## 4　分析

从 4 位有效数字的试验可以看出，使用单双精度计算结果误差差距不大，算法二明显优于算法一，无论是在速度还是准确性上。

当精度扩展到单精度机器精度时，双精度误差明显小于单精度，算法二误差小于算法一，且循环次数也较少。

用算法一可提升精度较少，到 1e-9 之后已经无法计算，用算法二可计算出绝对误差为 0 的解（仍有截断误差）。

综上，双精度优于单精度，算法二优于算法一。