实验 2：π的计算

学院：电子信息与电气工程学院　　姓名：王煌基　　学号：519030910100

一、实验目的

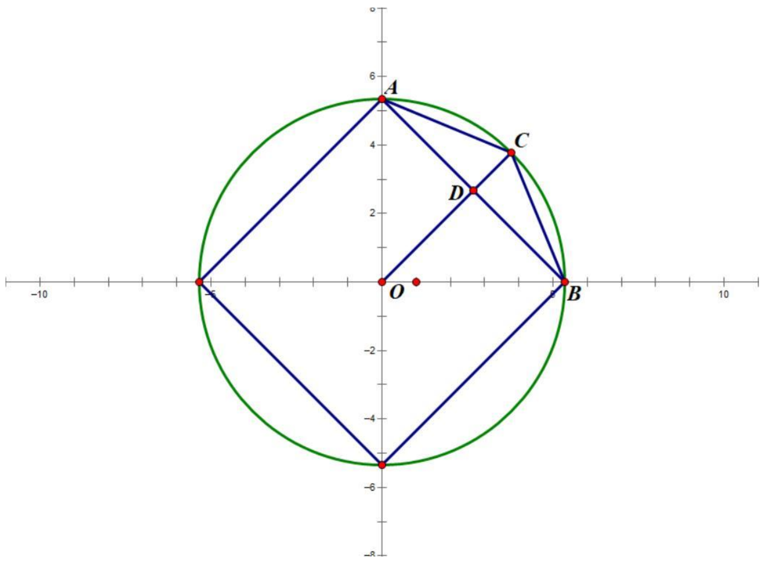
通过相关的微积分、高等数学知识以及历史层面上对重要常量π进行计算，并且在计算过程中考虑其精度问题、差异问题以及效率问题。

二、实验内容

任务一：鲁道夫割圆术求π

【问题】德国人鲁道夫用一生计算圆周率。他同样是用圆的内接多边形逼近圆周，不过他是从正方形开始成倍增加边数。试推导出他计算所采用的递推公式，然后求π的近似值到10位和20位。

【解】



鲁道夫的方法与刘徽的割圆法本质实际上是相同的，只是后者使用的是六边形的切割方法，所以二者原理相同，就是通过多边形边长的无穷增大，使得这个多边形越来越接近圆形，则该多边形的面积便等价于圆的面积，周长等价于圆的周长，便可以用同样的方法去解释鲁道夫割圆术。这里我们用递推法推导出他计算所采用的的递推公式。

设边数为$4*2^{n-1}$的正多边形的边长为a_n，根据我们三角形的勾股关系可以得到：$AC^2=AD^2+DC^2=AD^2+(OC-OD)^2=AD^2+\left(OC-\sqrt{OA^2-AD^2}\right)^2$

然后，我们注意到，在单位圆中，半径长度为1，则上式子可化为：

$$a_{n+1}^2=\left(\frac{a_n}{2}\right)^2+\left(1-\sqrt{1-\left(\frac{a_n}{2}\right)^2}\right)^2=2-\sqrt{4-a_n^2}$$

对于每次所构成的有$4*2^{n-1}$条边的多边形，可以视为由$4*2^{n-1}$个全等的三角形构成，那

么面积自然也可以视为由$4*2^{n-1}$个全等的三角形来构成，故：

$$S_{\triangle AOC}=\frac{1}{2}OC\cdot AD=\frac{1}{2}*1*\frac{a_n}{2}=\frac{a_n}{4}$$

$$S_{n+1}=S_{总}=4*2^{n-1}*S_{\triangle AOC}=2^n a_n$$

因此，综上所述，我们可以得到相应的递推公式：

$$a_n=\begin{cases}\sqrt{2}, n=1\\ \sqrt{2-\sqrt{4-a_n^2}}, n\geq 2 \text{ 且} n\in N^*\end{cases} \text{且} \mathrm{S_n}=\begin{cases}2, n=1\\ 2^{n-1}a_{n-1}, n\geq 2 \text{ 且 } \mathrm{n}\in \mathrm{N}^*\end{cases}$$

利用 matlab 代码实现后得到的 m1_1.m 文件如下（需要保留小数位 10~20 位）：

```
function Rudolf(n)
    a(1) = sqrt(2)  ;
    for i = 2:n
        a(i) = sym((2-(4-a(i-1)^2)^0.5)^0.5)  ;
    end
    PI = 2^(n-1)*a(n-1)  ;
    vpa(PI,10)  %保留 10 位有效数字
    vpa(PI,20)  %保留 20 位有效数字
```

```
>> m1_1(25)

ans =

3.142451272


ans =

3.1424512724941336789
```

然而，令人疑惑的是，当我们输入的代码 m1_1(n)中的 n 在$n\geq 29$的时候出现了较大差异：

```
>> m1_1(29)

ans =

4.0


ans =

4.0
```

```
>> m1_1(30)

ans =

0.0


ans =

0.0
```

这个疑惑待完成任务后再进行总结思考。

任务二：相关公式求 π

【问题】1）用反正切函数的幂级数展开式结合有关公式求 π，若要精确到 40 位、50 位数字，试比较简单公式和 Machin 公式所用的项数.

2）验证公式

$$\frac{\pi}{4}=\arctan\frac{1}{2}+\arctan\frac{1}{5}+\arctan\frac{1}{8}$$

试用此公式右端作幂级数展开完成任务 1）所需的项数。

3）回忆在微积分中学习到的其它级数形式是否可用来求 π 的值到 10 位、20 位、30 位，相应需要级数的多少项？

1.反正切函数展开式：$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots + (-1)^{n-1}\frac{x^{2n-1}}{2n-1} + \cdots$

2.简单公式：$\arctan\frac{1}{2} + \arctan\frac{1}{3} = \frac{\pi}{4}$

3.Machin 公式：$4\arctan\frac{1}{5} - \arctan\frac{1}{239} = \frac{\pi}{4}$

【解】

注意到$\arctan 1 = \frac{\pi}{4}$，容易得到：$\pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots + (-1)^{n-1}\frac{x^{2n-1}}{2n-1} + \cdots)$

故基于提供的公式的基础上，我们可以得到相应的 m 文件 m1_2_1_Arctan.m 如下：

```
function Arctan(n)
    seq(1) = 1;
    for i = 2:n
        seq(i) = seq(i-1) + (-1)^(i-1)/(2*i-1);
    end
    PI = 4 * seq(n)
```

```
>> m1_2_1_Arctan(124000)

PI =

    3.1416
```

对于简单公式和 Machin 公式所用项数的比较，我们引入了 arctan.m 文件来简化过程。

```
function a = arctan(x,n)
    s(1) = x;
    for i = 2:n
        s(i) = s(i-1) + (-1)^(i-1)*(x^(2*i-1))/(2*i-1);
    end
    a = vpa(s(n),52);
```

然后把简单公式、Machin 公式、验证公式放在一起进行比较，可以得到：

```
function Compare(n)
    disp(['When n = ',num2str(n)]);
    %使用简单公式进行运算
    easyn = 1;
    while abs(pi - vpa(4*(arctan(1/2,easyn)+arctan(1/3,easyn)),50)) >= 10^(1-n)
        easyn = easyn + 1;
    end
    disp(['The number of steps using the easy formula is:',num2str(easyn)]);
    %使用 Machin 公式进行运算
    Machinn = 1;
    while abs(pi - vpa(4*(4*arctan(1/5,Machinn)-arctan(1/239,Machinn)),50)) >= 10^(1-n)
        Machinn = Machinn + 1;
    end
    disp(['The number of steps using the Machin formula is:',num2str(Machinn)]);
    %使用测试的公式进行运算
    testn = 1;
```

```
    while abs(pi - vpa(4*(arctan(1/2,testn)+arctan(1/5,testn)+arctan(1/8,testn)),50)) >= 10^(1-n)
        testn = testn + 1;
    end
    disp(['The number of steps using the test formula is:',num2str(testn)]);
```

运行结果为：

命令行窗口

```
>> for t = 1:20
m1_2_1_Compare(t)
end
When n = 1
The number of steps using the easy formula is:1
The number of steps using the Machin formula is:1
The number of steps using the test formula is:1
When n = 2
The number of steps using the easy formula is:2
The number of steps using the Machin formula is:1
The number of steps using the test formula is:2
When n = 3
The number of steps using the easy formula is:3
The number of steps using the Machin formula is:2
The number of steps using the test formula is:3
When n = 4
The number of steps using the easy formula is:4
The number of steps using the Machin formula is:2
The number of steps using the test formula is:4
When n = 5
The number of steps using the easy formula is:6
The number of steps using the Machin formula is:3
The number of steps using the test formula is:6
When n = 6
The number of steps using the easy formula is:7
The number of steps using the Machin formula is:4
The number of steps using the test formula is:7
When n = 7
The number of steps using the easy formula is:9
The number of steps using the Machin formula is:4
The number of steps using the test formula is:9
When n = 8
The number of steps using the easy formula is:10
The number of steps using the Machin formula is:5
The number of steps using the test formula is:10
When n = 9
The number of steps using the easy formula is:12
The number of steps using the Machin formula is:6
fx The number of steps using the test formula is:12
```

```
When n = 10
The number of steps using the easy formula is:13
The number of steps using the Machin formula is:6
The number of steps using the test formula is:13
When n = 11
The number of steps using the easy formula is:15
The number of steps using the Machin formula is:7
The number of steps using the test formula is:15
When n = 12
The number of steps using the easy formula is:17
The number of steps using the Machin formula is:8
The number of steps using the test formula is:17
When n = 13
The number of steps using the easy formula is:18
The number of steps using the Machin formula is:9
The number of steps using the test formula is:18
When n = 14
The number of steps using the easy formula is:20
The number of steps using the Machin formula is:9
The number of steps using the test formula is:20
When n = 15
The number of steps using the easy formula is:21
The number of steps using the Machin formula is:10
The number of steps using the test formula is:21
When n = 16
The number of steps using the easy formula is:23
The number of steps using the Machin formula is:11
The number of steps using the test formula is:23
When n = 17
```

令我感到疑惑的地方是，当 n>=17 的时候，计算过程中的解将不再优化，反而陷入了死循

环，这与我在任务一中遇到的情况是类似的，故暂时放着，后面进行研究。

由于还未接触级数，故级数方面没有提供额外的方法。

任务三：数值积分法求 π

【问题】分别用梯形法和 Simpson 法精确到 10 位有效数字，用 Simpson 法精确到 15 位有效数字。

【原理】

$$A = 4\int_0^1 \frac{1}{1+x^2}dx = \pi\text{,设}y(x) = \frac{1}{1+x^2}\text{且将}[0,1]\text{等分,取}x_k = \frac{k}{n}, y_k = \frac{1}{1+x_k^2}$$

梯形法：$A = \frac{2}{n}[2(y_1 + y_2 + \cdots + y_{n-1}) + y_0 + y_n]$

Simpson 法：

$$A = \frac{1}{6m}[(y_0 + y_{2m}) + 2(y_2 + y_4 + \cdots + y_{2m-2}) + 4(y_1 + y_3 + \cdots + y_{2m-1})]$$

数值积分的关键就是在于 **Riemann sum** 中使得Δx尽可能的小，当足够小的时候，**Riemann sum** 就是所求区域的积分值，这里我们引入（trapezium：梯形）：trp_integral.m 和 Simpson.m：首先对于梯形法进行分析（trp_integral.m）

```
function trapezium()
    n = 1;
    x = 0: 1/n :1;
    y = 1 ./ (1 + x.^2);
    S = 2 * sum(y) - 1 - 0.5;
    S = 2 * S / n;
    vpa(S,50)
    while abs(pi - S) >= 10^(-9)
        n = n + 1;
        x = 0: 1/n :1;
        y = 1 ./ (1 + x.^2);
        S = 2 * sum(y) - 1 - 0.5;
        S = 2 * S / n;
        vpa(S,50);
    end
    n
```

但是在实际运算过程中，这个方法耗费的时间是非常大的，因为假设我们需要得到精确位数的值 n，那么在运算过程中我们不仅每次都是对于 x, y 的数组（大小也为 n）进行操作，而且本身在计算机运算的过程中除法是最消耗时间的，记单步操作花费为 1 单位时间，那么实际上最少要花费$2 * \frac{(1+2+3+\cdots+n)*n}{2} = O(n^2)$（这里用大 O 记号来表示复杂度），那么如果 n 超

过 10000 的时候这个程序将会额外花费很多时间和空间，故这里直接采取二分的思路，简单判断出范围再进行运行会更节省时间（图的上面是 n=16800， 下面是 n=17000）

```
>> trp_integral

ans =

3.1415926529992850291250761074479669332504272460938

>> trp_integral

ans =

3.1415926530130890981240554538089781999588012695312
```

并且当使用此程序的时候，我注意到，实际上这个程序并不能得到正确解，因为当判定标准为小数差距为$\geq 10^{-9}$ 的时候，存在例如 3.14159265299 的值也是正确的解（因为 3.1415926535897 与之的差值显然小于10^{-9}的），故我把 PI 的值进行了一定程度的修改：

```
function trapezium()
    n = 16800;
    x = 0: 1/n :1;
    y = 1 ./ (1 + x.^2);
    S = 2 * sum(y) - 1 - 0.5;
    S = 2 * S / n;
    PI = 3.1415926539999999;
    while abs(PI - S) >= 10^(-9)
        n = n + 1;
        x = 0: 1/n :1;
        y = 1 ./ (1 + x.^2);
        S = 2 * sum(y) - 1 - 0.5;
        S = 2 * S / n;
    end
    n
```

这样便避免了此类型计算过程中产生的误差和尴尬，运算结果如下：

```
>> trp_integral

n =

     16811
```

接下来我们对 Simpson 法进行分析（Simpson.m）：

```
function Simpson()
    m = 1;
    S = 0;
    for i = 1:(2*m)
        x(i) = i/(2*m);
        y(i) = 1/(1+x(i)^2);
```

```
end
for i = 2:2:2*(m-1)
    S = S + 2*y(i);
end
for i = 1:2:2*m-1
    S = S + 4*y(i);
end
S = S + 1 + 0.5;
S = 4 * S / (6*m);
vpa(S,50);
%若判断的是 10 的有效位数，则此处的代码为：
%PI = 3.1415926539999999
%while abs(PI - S) >= 10^(-9)
PI = 3.1415926535897999999;
while abs(PI - S) >= 10^(-14)
    m = m + 1;
    S = 0;
    for i = 1:(2*m)
        x(i) = i/(2*m);
        y(i) = 1/(1+x(i)^2);
    end
    for i = 2:2:2*(m-1)
        S = S + 2*y(i);
    end
    for i = 1:2:2*m-1
        S = S + 4*y(i);
    end
    S = S + 1 + 0.5;
    S = 4 * S / (6*m);
end
m
```

代码长的原因在于这里的 Simpson 法用的累加方式是分奇偶的，故不能直接使用如梯形法一般的写法，由于存在之前提到的用$\geq 10^{-9}$的写法存在较大误差的原因，故这里使用的均为有效位数后面加一定长度的 9 循环来确保正确性，所以判断有效位数 10 位的时候使用 PI = 3.1415926539999999，有效位数 15 位的时候使用 PI = 3.1415926535 8979 99999，结果如下：

```
>> Simpson

m =

    11
```

```
>> Simpson

m =

    77
```

任务四：Monte Carlo 法+ Buffon 实验求 π

【问题】

1) 用 Monte Carlo 法计算 π，除了加大随机数，在随机数一定时可重复算若干次后求平均值，看能否求得 5 位精确数字？

2) 设计方案用计算机模拟 Buffon 实验

【解】

1)首先需要了解 Monte Carlo 法的本质是几何概型的应用，在 π 的计算的应用即通过随机模拟产生大量的满足条件$D=\{0\le x\le 1$ 且 $0\le y\le 1\}$的点上判断满足$D=\{x^2+y^2\le 1\}$的点的个数来计算 π，关键的点在于**只要试验次数很大，该百分比便近似于事件发生的概率。**

$$\frac{Area\ of\ the\ circle}{Area\ of\ the\ square}=\frac{\frac{\pi}{4}}{1}=\frac{\pi}{4}$$

然后我们便可以引入相应的文件（random.m）：

```
function random(n)
    R = rand(2,n);
    m = 0;
    for i = 1:n
        if R(1,i)^2 + R(2,i)^2 <= 1
            m = m + 1;
        end
    end
    vpa(4*m/n,50)
```

运行结果如下：

```
>> random(10000000)

ans =

3.14157760000000001921819148265058174729347229003906
```

实际上在数值较大的时候所计算出来的 π 值确实精确了一些，但是幅度不够，故此处稍微改写一下，重复算若干次后求平均值，引入相应的文件（random2.m），多了重外循环：

```
function random(n)
    S = 0;
    for j = 1:100
        R = rand(2,n);
        m = 0;
        for i = 1:n
            if R(1,i)^2 + R(2,i)^2 <= 1
                m = m + 1;
            end
```

```
        end
        S = S + 4*m/n;
    end
    vpa(S/100,50)
```

运行结果如下：

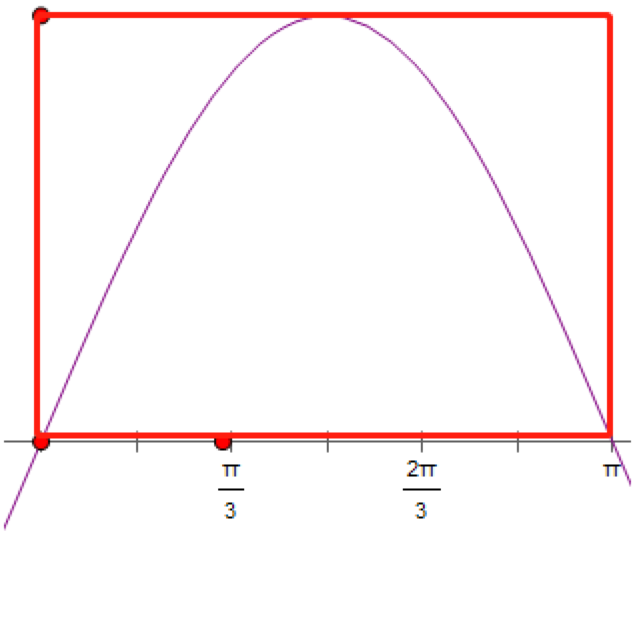
```
>> random2(1000000)

ans =

3.1415851599999999876899892115034162998199462890625
```

可见，这样得到的随机数在增大数据的时候更容易精确，但是并非每次都是精确的，所以如果多次循环求和，累加可以视为是“误差”的累加，那么过程中得到的解反而更加不容易与原始 π 值相符，所以如果单次调出，那么这一次可能就是误差较小的时候，这个时候得到的可能比平均计算得到的更加精确。

2)若需要设计方案用计算机模拟 Buffon 实验，我们首先得构造出 Buffon 实验的数学模型。

【建模】假设平面上存在无数组平行线间距为d，将长度为$l(l \le d)$的针多次随机扔到平面上，设投针次数为n，针与线相交的次数为m，则这个相交概率$p = \frac{m}{n} = \frac{2l}{\pi d}$，从而$\pi = \frac{2ln}{dm}$

【证明】注意到，这个投针过程是随机的，而且平行线有无数多组，故不失一般性，我们通过①针的中点到靠下的平行线的距离a②针与靠下平行线之间所成的夹角θ这两个性质来确定任何一根针的位置，且容易发现，针与平行线有交点的情况必须满足$a \le \frac{l\sin\theta}{2}$，根据三角函数的周期性，我们这里仅考虑$0 \le \theta < \pi$的区域即可。在这个区域内，注意到，必须要满足$a \le \frac{l\sin\theta}{2}$的情况下针才会和平行线相交，相交概率$p = \frac{\int_0^{\pi}\frac{l\sin\theta}{2}d\theta}{\frac{\pi d}{2}} = \frac{2l}{\pi d}$，即用曲线围成的面积比去红色方框得到的面积得到的结果就是相交概率。



下面对其过程进行实现，引入文件（buffon.m）：

```
function buffon(n)
    d = 1;
    l = rand(1);
    m = 0;
    a = rand(1,n)./2;
    theta = rand(1,n)*pi;
    for i = 1:n
        if a(i) < l/2 * sin(theta(i))
            m = m + 1;
        end
    end
    PI = 2*l*n/d/m

```

运行结果：

```
>> buffon(1000000)

PI =

    3.1410
```

这里需要 n 足够大，当 n 不足够大的时候所得到的解将存在非常大的误差。

任务五：用积分公式计算 π

【问题】利用$\boldsymbol{\int_0^{\frac{\pi}{2}} sin^n x dx = \frac{(n-1)!!}{n!!} \cdot \frac{\pi}{2}, n}$为偶数，推导公式：

$$\boldsymbol{\frac{\pi}{2} = \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdots\cdots \frac{2n}{2n-1} \cdot \frac{2n}{2n+1} \cdots}$$

且用此公式计算 π 的近似值，效果如何？

【解】推导过程：

$$\int_0^{\frac{\pi}{2}} \sin^{\mathrm{n}} x\, dx = \frac{(n-1)!!}{n!!} \cdot \frac{\pi}{2} \Leftrightarrow \frac{\pi}{2} = \int_0^{\frac{\pi}{2}} \sin^{\mathrm{n}} x\, dx \cdot \frac{n!!}{(n-1)!!} \Leftrightarrow \frac{\pi}{2} = \int_0^{\frac{\pi}{2}} \sin^{2\mathrm{m}} x\, dx \cdot \frac{(2m)!!}{(2m-1)!!}$$

对于积分式子，我们进行分部积分法处理（记$I(n) = \int_0^{\frac{\pi}{2}} \sin^{\mathrm{n}} x\, dx$）：

$$\int_0^{\frac{\pi}{2}} \sin^{\mathrm{n}} x\, dx = -\int_0^{\frac{\pi}{2}} \sin^{\mathrm{n}-1} x\, d\cos x = -\sin^{n-1} x \cos x \,|_0^{\frac{\pi}{2}} + \int_0^{\frac{\pi}{2}} \cos x\,(n-1) \sin^{\mathrm{n}-2} x \cos x\, dx$$

$$\int_0^{\frac{\pi}{2}} \sin^{\mathrm{n}} x\, dx = 0 + (n-1)\int_0^{\frac{\pi}{2}} (1-\sin^2 x) \sin^{\mathrm{n}-2} x\, dx = (n-1)\int_0^{\frac{\pi}{2}} \sin^{\mathrm{n}-2} x\, dx - (n-1) I(n)$$

所以我们可以得到：

$$I(n)=\int_0^{\frac{\pi}{2}}\sin^{n}x\,dx\ =(n-1)I(n-2)-(n-1)I(n)\Leftrightarrow I(n)=\frac{n-1}{n}I(n-2)$$

注意到，显然，

$$I(1)=\int_0^{\frac{\pi}{2}}\sin x\,dx\ =1\ 且I(2)=\int_0^{\frac{\pi}{2}}\sin^2 x\,dx\ =\frac{2x-\sin 2x}{4}\Big|_0^{\frac{\pi}{2}}=\frac{\pi}{4}$$

由于这里的 n 是每两项间隔一次，故我们大可直接得到 n 的递推关系式：

$$I(n)=\frac{n-1}{n}I(n-2)=\begin{cases}\dfrac{(n-1)!!}{n!!}\cdot\dfrac{\pi}{2}\,(n为偶数)\\ \dfrac{(n-1)!!}{n!!}\,(n为奇数)\end{cases}$$

$$I(2n)=\frac{2n-1}{2n}I(2n-2)=\frac{2n-1}{2n}\frac{2n-3}{2n-2}I(2n-4)$$

$$I(2n+1)=\frac{2n}{2n+1}I(2n-1)=\frac{2n}{2n+1}\frac{2n-2}{2n-1}I(2n-3)$$

且注意到，$\sin^{2n+1}x<\sin^{2n}x<\sin^{2n-1}x\left(x\in(0,\frac{\pi}{2})\right)\to I(2n+1)<I(2n)<I(2n-1)$

那么我们可以得到：

$$\frac{2n!!}{(2n+1)!!}<\frac{(2n-1)!!}{2n!!}\cdot\frac{\pi}{2}<\frac{(2n-2)!!}{(2n-1)!!}\Leftrightarrow\frac{2n!!}{(2n-1)!!}\frac{2n!!}{(2n+1)!!}<\frac{\pi}{2}<\frac{(2n-2)!!}{(2n-1)!!}\frac{2n!!}{(2n-1)!!}$$

由夹逼定理，两边取极限后我们可以得到：

$$\frac{\pi}{2}=\lim_{n\to\infty}\frac{2n!!}{(2n-1)!!}\frac{2n!!}{(2n+1)!!}=\frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\dots\cdot\frac{2n}{2n-1}\cdot\frac{2n}{2n+1}\cdot\dots$$

所以式子得证！

下面我们引入相应的文件来进行 π 近似值计算的分析（formula.m）：

```
function formula(n)
    S = 1;
    for i = 1:n
        tmp = 4*i^2;
        S = S*tmp/(tmp-1);
    end
    vpa(abs(S*2-pi),50)
```

这里的输出结果是我们的方法与实际π值的误差情况（从 n=1 到 n= 1000000000）：

```
>> formula(1)

ans =

0.47492598692312659736103341856505721807479858398438

>> formula(10)

ans =

0.07388884694629549443334326497279107570648193359375

>> formula(100)

ans =

0.007805162961634692919687950052320957183837890625

>> formula(1000)

ans =

0.000784907559405301213928396464325487613677978515625

>> formula(10000)

ans =

0.000078534907871574688442706246860325336456298828125

>> formula(100000)

ans =

0.000007853932479928715792993898503482341766357421875
```

```
>> formula(1000000)

ans =

0.000000785397325397951817649300210177898406982421875

>> formula(10000000)

ans =

0.00000007853828787318661852623336017131805419921875

>> formula(100000000)

ans =

0.000000016582544670740162473521195352077484130859375

>> formula(1000000000)

ans =

0.0000000165800475571131755714304745197296142578125
```

过程中实际上 n 越大精度越高，但是这个 n 大的情况下，我们可以注意到，基本 n 在扩大的情况下，需要 10 倍的扩大量才能换来接近 10 倍的精度改进（可以认为是线性的关系），但是在红框的情况下，我们注意到，这个精度基本没有改变了，所以实际上这个方法在 n 越大的时候越低效，效果不会太佳。

任务六：查找新方法并实践计算 π

【问题】提出其他计算 π 的方法并证明，然后实践该方法，并进行分析讨论。

【解】这里给出一种方法：

$$\frac{\pi^2}{8}=\sum_{0}^{\infty}\frac{1}{(2n+1)^2}$$

证明：注意到反三角函数$\arcsin x$的泰勒展开式为：

$$\arcsin x=\sum_{0}^{\infty}\frac{1\cdot 3\cdot \ldots\cdot(2n-1)}{2\cdot 4\cdot \ldots\cdot(2n)}\cdot\frac{x^{2n+1}}{2n+1}\ (|x|\le 1)$$

这里我们不妨令$x=\sin t\ (t\le\frac{\pi}{2})$，来使得这个式子变得更加简洁可观：

$$t=\sum_{0}^{\infty}\frac{1\cdot 3\cdot \ldots\cdot(2n-1)}{2\cdot 4\cdot \ldots\cdot(2n)}\cdot\frac{\sin^{2n+1}t}{2n+1}$$

然后根据我们在任务 5 中得到的式子：$\int_0^{\frac{\pi}{2}}\boldsymbol{\sin^n x dx}=\frac{(n-1)!!}{n!!}$(n 为奇数)，我们首先的想法

就是对 t 进行积分，这里采取对 t 从 0 到$\frac{\pi}{2}$进行积分，我们可以得到：

$$\int_{0}^{\frac{\pi}{2}} t dt = \sum_{0}^{\infty} \frac{1 \cdot 3 \cdot \ldots \cdot (2n-1)}{2 \cdot 4 \cdot \ldots \cdot (2n)} \cdot \frac{\int_0^{\frac{\pi}{2}} \sin^{2n+1} t\, dt}{2n+1} = \sum_{0}^{\infty} \frac{1 \cdot 3 \cdot \ldots \cdot (2n-1)}{2 \cdot 4 \cdot \ldots \cdot (2n)} \cdot \frac{\frac{2 \cdot 4 \cdot \ldots \cdot 2n}{3 \cdot 5 \cdot \ldots \cdot (2n+1)}}{2n+1}$$

两边同时简化后我们得到：

$$\frac{\pi^2}{8} = \int_{0}^{\frac{\pi}{2}} t dt = \sum_{0}^{\infty} \frac{1 \cdot 3 \cdot \ldots \cdot (2n-1)}{2 \cdot 4 \cdot \ldots \cdot (2n)} \cdot \frac{\frac{2 \cdot 4 \cdot \ldots \cdot 2n}{3 \cdot 5 \cdot \ldots \cdot (2n+1)}}{2n+1} = \sum_{0}^{\infty} \frac{1}{(2n+1)^2}$$

得证。

这里我们引入文件(new.m)：

```
function new(n)
    S = 0;
    for i = 1:n
        S = S + 1/((2*i-1)^2);
    end
    vpa(abs(pi-sqrt(8*S)),50)
```

并重复多次试验，判断其与真实的 π 值的误差情况，如图所示：

```
>> new(1)

ans =

0.31316552884360282504871975106652826070785522460938

>> new(10)

ans =

0.031967195601145359518113764352165162563323974609375

>> new(100)

ans =

0.00318468652270187391195577220059931278228759765625

>> new(1000)

ans =

0.000318325987052947567690353025682270526885986328125

>> new(10000)

ans =

0.000031831149844396833259452250786125659942626953125

>> new(100000)

ans =

0.00000318310044900016464453074149787425994873046875
```

```
>> new(1000000)

ans =

0.000000318309823388318591241841204464435577392578125

>> new(10000000)

ans =

0.00000003183205787848919499083422124385833740234375

>> new(100000000)

ans =

0.000000005738212127681663332623429596424102783203125

>> new(1000000000)

ans =

0.000000005738212127681663332623429596424102783203125
```

注意到，该方法的精确度会比任务五更高一些。由于原理上二者类似，故效率方面二者也

是类似的，均大概提升 10 倍的数值后可以提升 10 倍的精确度（视为线性的），当然，常见的数值过大的时候结果不变的情况仍旧有发生。

任务七：计算 e

【问题】利用下列公式以及其他公式计算 e：

$$e = \lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n$$

$$e = 1 + 1 + \frac{1}{2!} + \cdots + \frac{1}{n!} + \frac{e^{\theta}}{(n+1)!}$$

$$e = \lim_{n\to\infty}\left(1+\frac{1}{n}+\frac{k}{n^2}\right)^n$$

我们这里引入文件（e1.m,e2.m,e3.m）以及结果显示（由于实验原理与之前的类似，故此处不再赘述）：

```
function e1(n)
    t = 1+1/n;
    E = t;
    for i = 2:n
        E = E * t;
    end
    vpa(E,50)
```

```
>> e1(10)

ans =

2.5937424601000023116625925467815250158309936523438

>> e1(100)

ans =

2.7048138294215293697675406292546540498733520507812

>> e1(1000)

ans =

2.7169239322355984711521159624680876731872558593750

>> e1(10000)

ans =

2.7181459268248984173510507389437407255172729492188
```

第二个方法由于最后一个量可以视为无穷小量，故在程序编写时，可以忽略不计。

```
function e2(n)
```

```
    fact = 1;
    E = 1;
    for i = 1:n
        fact = fact * i;
        E = E + 1/fact;
    end
    vpa(E,50)
```

```
>> e2(10)

ans =

2.7182818011463845131459038384491577744483947753906

>> e2(100)

ans =

2.7182818284590455348848081484902650117874145507812

>> e2(1000)

ans =

2.7182818284590455348848081484902650117874145507812

>> e2(10000)

ans =

2.7182818284590455348848081484902650117874145507812
```

这里我们可以注意到，第二个式子的方法的效率非常的高，n=100 的时候精确度已经比第一个式子 n=10000 还要精确，故第二个式子的计算方法非常高效。

```
function e3(n)
    k = 1;
    t = 1 + 1/n + k/(n^2);
    E = t;
    for i = 2:n
        E = E * t;
    end
    vpa(E,50)
```

```
>> e3(100)

ans =

2.731725840049335829462506808340549468994140625

>> e3(1000)

ans =

2.7196394968014705462167057703481987118721008300781

>> e3(10000)

ans =

2.7184177278243217834585720993345603346824645996094

>> e3(100000)

ans =

2.7182954197410422736425061884801834821701049804688
```

第三个式子不仅与 n 相关，而且与 k 相关，当 n 固定的时候（为 1000），程序结果如图：（从上到下分别为 k=0,k=1,k=2），看得出来 e 的值与 k 正相关。

```
>> e3(1000)

ans =

2.716923932235598471152115962468087673187255859375

>> e3(1000)

ans =

2.7182813757516566255389989237301051616668701171875

>> e3(1000)

ans =

2.7196394968014705462167057703481987118721008300781
```

当 k 一定的情况下（为 1），程序结果如图：

```
>> e3(10)

ans =

2.8394209860690176050468380708480253815650939941406

>> e3(100)

ans =

2.7317258400493358294625068083405494689941406 25

>> e3(1000)

ans =

2.7196394968014705462167057703481987118721008300781

>> e3(10000)

ans =

2.7184177278243217834585720993345603346824645996094
```

注意到，第三个程序的准确度也是不错的，但是相比于第二个还略有不足。

综上所述，三个式子的准确度和效率来看，二>三>一。

三、问题思考

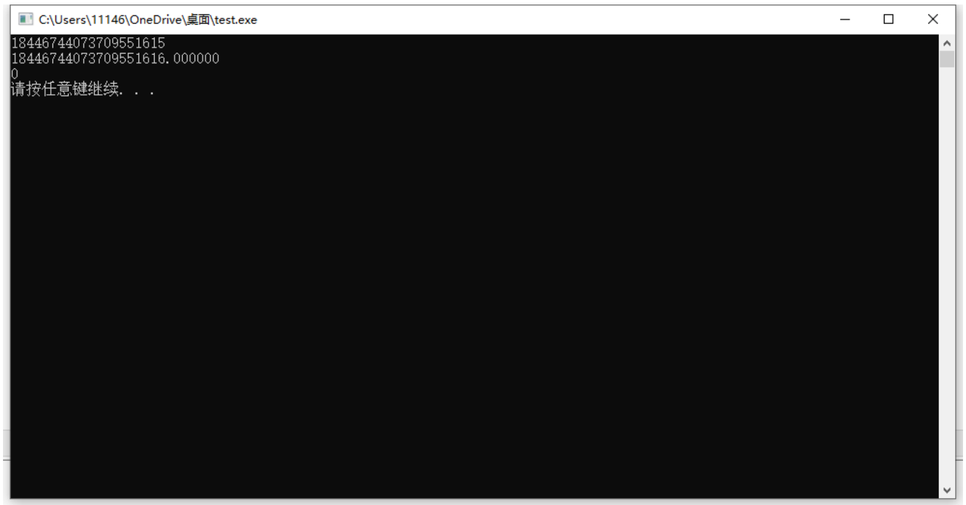
在实验过程中，有一个情况反复出现，就是当精度达到一定的程度，或者计算的值达到一定精度的情况下，我们增大测试的 n 的值程序已经没有变动，保持着之前比较小的 n 的值。

这里，通过查找了相应的资料，以及联系了程序设计课程里所涉及的知识，我得到下面的这个知识点：

大多数计算机内部采用二进制进行存储，由于二进制和十进制的转换并不是“完美”的，即因为计算机的存储位数是固定的，不能达到无限，故某些十进制数并不能转换成有限位的二进制数，例如：$\frac{1}{3}=0*\left(\frac{1}{2}\right)^1+1*\left(\frac{1}{2}\right)^2+0*\left(\frac{1}{2}\right)^3+1*\left(\frac{1}{2}\right)^4+\cdots$，是不能完全地用一个有限的二进制数进行表示的。因此计算机内数字存储，一般采用的是浮点体系，不同的程序、不同的系统采取的具体标准不同，下图展现了 C++和 matlab（test.m）的精度关系分析：

```
#include<iostream>
#include<stdlib.h>
int main(){
```

```
    unsigned long long maxULL=0xffffffffffffffff; //2^64-1=18446744073709551615,
    printf("%llu\n",maxULL);
    double d1=maxULL;                              //20bit Significant,Precision Loss
    printf("%f\n",d1);
    maxULL=d1;
    printf("%llu\n",maxULL);
    return 0;
}
```



在这里，类型一转变，值也随时变得不同了，即存在**精度丢失**现象，这个与它的浮点数规格标准有关系[1]。

```
function test()
    a = 4/3;
    b = a-1;
    c = 3*b;
    d = 1-c
```

本来显然结果是 0 的，但是输出的结果却大相径庭。

```
>> test

d =

   2.2204e-16
```

那怎么样的解决方法可以增加精度呢？这里很重要的一个点就是 matlab 自带的符号计算 sym 函数，这个函数能够保留一个很高的精度，但是同时会对运算速度有较大的影响（当计算量大的情况下）。这里举一个 sym 函数提高精度的例子：

```
function test_sym()
   1/3-0.333333333333
```

[1] 关于 C++ double 浮点数精度丢失的分析：https://www.cnblogs.com/cai2007/p/3679443.html

```
3.     sym(1/3)-0.333333333333
```

```
>> test_sym

ans =

   3.3329e-13


ans =

18013/54043195528445952
```

我们知道，$\frac{1}{3}=0.\dot{3}=0.333333\ldots$，所以应该有$\frac{1}{3}-0.333333333333=0.00000000000033333\ldots$

通过计算，我们发现第二个式子的值是正确的，然而第一个式子出现了 29 这样的数字，说明到后面的位数计算的过程中 matlab 的正常直接计算是会忽略一些具体的值，而保留近似值。

通过对这个原理的了解之后，我就能大致明白，为什么当 n 增大到一定情况的时候会出现“未知错误”、“效率不变”的情况了。这里我觉得应该有几种解决方法，首先是利用 sym 函数进行运算，可以提高一定程度上的精度，但是要注意的是这个精度也是有限的，过多位数的情况下还是不能保证最终结果的有效性；其次是利用高精度算法（在数据结构、算法设计的过程中常用的），对于很大位数的计算，我们可以自己构造一个容量非常大的实数运算集，模拟实数的运算过程，而且可以满足到甚至十万位的精度，而且效率很高，本质上是通过对人类手算加减乘除法的模拟；再者是改变自己的算法，注意到本实验报告中的不同方法的效率、精度都是不同的，所以选择不同的算法进行对比后，择优选择最好的算法进行数据的测试。

因为数字在大多数计算机中的存储本质上是二进制存储，故精度方面的确实无法完全匹配，只能思考如何去优化，去改进。这里我认为，如果能够用好这三种方法，也能够达到非常高的效率。

四、反思体会

由于刚开始学习 matlab，故有些地方的编程思想可能还比较不尽人意，但是在这么七个实验任务的训练下，目前我已经能够较为初步地掌握 matlab 的一些基本应用，并且加深了自己对计算 π 的多种方法的认识和理解。此外，能够把各个科目之间有相通的知识相联系，使得我对于该部分的知识有了更深层次的了解。当然，此次实验还有不足之处，主要在于我对 matlab 的一些语法还不是很熟悉，导致用的方法或者思路并不是最佳思路，希望在接下来的两个月内能够加强我这方面的能力。