Name: 黄弘毅　　student ID:515072910038

Assignment1　 language:matlab

1. $\text{my answer} = (-1)^0 * 2^{14-127} * (1 + 1 * 2^{-1} + 1 * 2^{-3}) = 1.5648181\,\text{e} - 34$

2.

(a) $\ln(x+1) - \ln(x) = \ln\left(\frac{x+1}{x}\right)$

(b) $\sqrt{x^2+1} - x = \frac{1}{\sqrt{x^2+1}+x}$

(c) $\cos^2(x) - \sin^2(x) = \cos(2x)$

(d) $\sqrt{\frac{1+\cos(x)}{2}} = \sqrt{\cos^2\left(\frac{x}{2}\right)} = \left|\cos\left(\frac{x}{2}\right)\right|$

3.my code:

```
%determine the underflow limit, double precision
format long
a=1;
while (a/2)~=a
    a=a/2;
    a
end
```

result:

……

a =

4.9406564584124654e-324

a=

0

```
%determine the overflow limit, double precision
format long
b=1;
while (b*2)~=b
    b=b*2;
    b
end
```

result:

……

a =8.988465674311580e+307

a =Inf

So the underflow limit of double precision storage should be 4.940656458412465e-324, and overflow limit should be 8.988465674311580e+307.

4.my code:

Double precision

```
%determine the machine precision, double precision
mp=1;
one=1;
while (one+mp)~=one
    mp=mp/2;
    (one+mp)-1
end
```

result:

……

ans =

2.220446049250313e-16

ans =

0

So the machine precision is 2.220446049250313e-16 of double precision floats.

Single precision

```
%determine the machine precision
mp=single(1);
one=single(1);
while (one+mp)~=one
    mp=mp/2;
    (one+mp)-1
End
```

result:

……

ans =

1.1920929e-07

ans =

0

So the machine precision is 1.1920929e-07 of single precision floats.

5.

- my algorithm is to combine every two term into one single term according to simple algebra $\frac{1}{n}-\frac{1}{n+2}=\frac{2}{n(n+2)}$, to avoid round-off errors brought by subtraction between two proximate numbers. And once the value of n is set, I'm going to apply a downward calculation to reduce errors brought by addition between one big number and another small number.

$$\pi = 8\left(\frac{1}{(2n-1)(2n-3)}+\frac{1}{(2n-5)(2n-7)}+\cdots+\frac{1}{3*1}\right)$$

- code:

```
%calculate pi
n=40;
mypi=0;
for i=n/2:-1:1 %downward direction
    mypi=mypi+1/(4*i-1)/(4*i-3);
end
mypi=mypi*8
err=(mypi-pi)/pi
```

result:

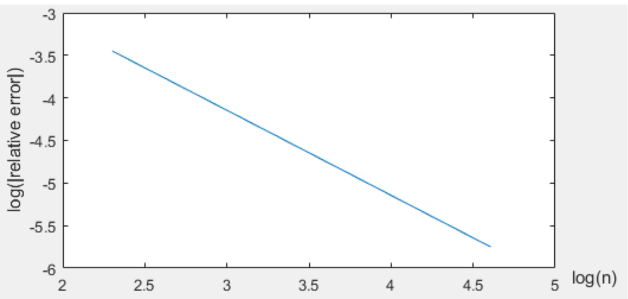
mypi =3.116596556793832

err =-0.007956504726161

So 3.116596556793832 is my result of calculating π, in the case n=40. And relative error is calculated according to the pi stored in matlab, which comes out to be -0.007956504726161.

- here I demonstrate the results in 8 digits for simplicity

n	10	20	40	100
my π	3.0418396	3.0916238	3.1165965	3.1315929
relative error	-0.0317523	-0.0159055	-0.0079565	-0.0031830



- According to theory of errors, errors could come from both approximation error and round-off error, when n is small, approximation error is dominant, when n get big enough, round-off error takes over. And supposedly there will be a optimal n where relative error is minimized. In my case, I choose n to be 10, 20, 40, 100, and the relative error is monotonously decreasing, apparently approximation error is the dominant part in our range of n.

  And if we plot log(relative error) as a function of log(n), it shows a linear relationship, of which slope is about -1.