# Determination of π Revisited

Supposed we were to use cosmic rays to measure π using the square with an inscribe circle technique used in experiment 2. We measure the cosmic rays with precise silicon detectors. There are two ways we can take the data: 1) take a fixed number of cosmic rays (in this case 5000) and stop to analyze the data, or 2) wait for a fixed time (with a mean of 5000) cosmic rays and stop to analyze the data. The error we would get in the two case is very different!

For 1) the number of events is fixed so the error is binomial $\sigma_\pi = \frac{4\sqrt{\frac{\pi}{4}(1-\frac{\pi}{4})}}{\sqrt{5000}} = 0.0232$

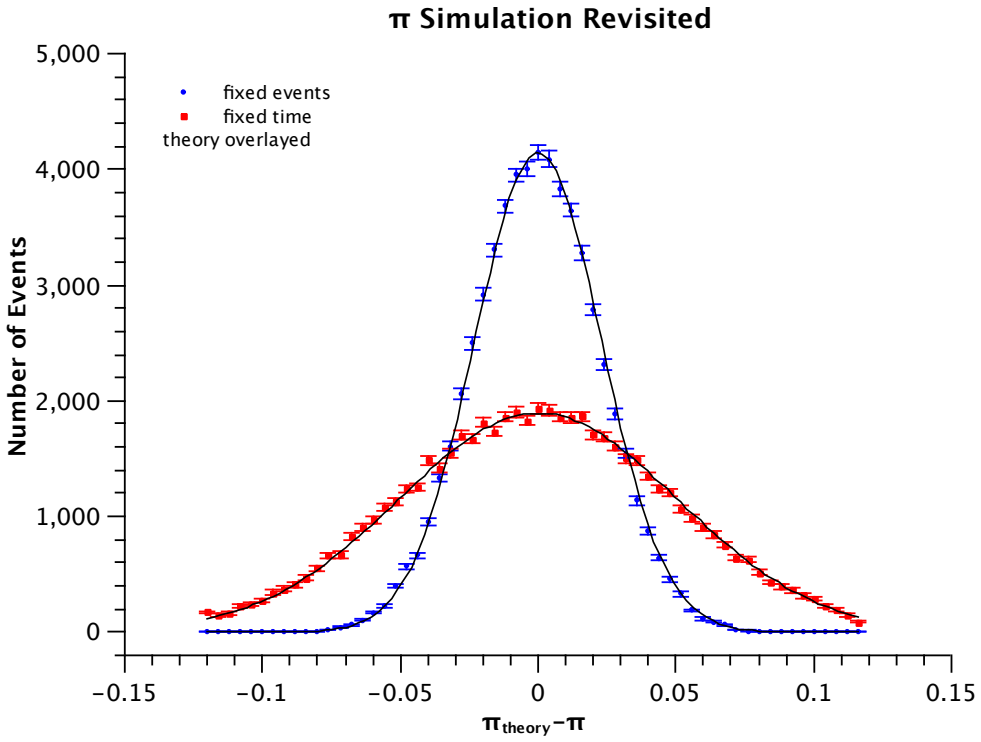
For 2) the time is fixed so the error is poissonian $\sigma_\pi = \frac{4\sqrt{\frac{\pi}{4}(5000)}}{5000} = 0.0501$

To the left are simulation of the two cases for 100,000 trials. Results:

3.141±0.023

3.142±0.050

(the code used for the simulation is on the next slide)



```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

double gauss(double mean,double sigma);

int main(){
  int i,j,k,accept;
  double xh;
  double x,y,dy;
  double sn,sx,sx2;
  double mean,sigma,inside;
  int ih;
  int hist[120]={0};
  double val1,val2,A,B,C;
  int entot;
  double ntot=5000.;
  double xmean,xsigma;
  FILE *fp;
  fp=fopen("binomial.txt","w");
  srand(123);
  for(i=0;i<100000;i++){
    if(i%1000==0)printf(" i %d \n",i);
  inside=0.0;
  mean=ntot*1.0;
  sigma=sqrt(ntot*1.0);
  entot=gauss(mean,sigma);
  entot=mean;     // comment out for poisson, leave in for binomial
  accept=0;
  while(accept<entot){
    x=(double)rand()/(double)RAND_MAX;
    y=(double)rand()/(double)RAND_MAX;
    accept++;
    if(x*x+y*y<1.0){
      inside=inside+1.0;
    }
  }
   sn=sn+1.0;
   sx=sx+4.0*inside/mean;
   sx2=sx2+(4.0*inside/mean)*(4.0*inside/mean);
   xh=(4.0*inside/mean-3.1415927)/0.002+60;
   ih=xh;
   if(ih>=0&&ih<120)hist[ih]++;
   }
   xmean=sx/sn;
   xsigma=sx2/sn-xmean*xmean;
   xsigma=sqrt(xsigma);
   fprintf(fp," mean %f sigma %f \n",xmean,xsigma);
   for(k=0;k<120;k++)fprintf(fp," %f %d \n",(k-60)*0.002,hist[k]);
   fclose(fp);
}

double gauss(double mean,double sigma){
  int i;
  double result;
  result=0.0;
  for(i=0;i<12;i++)result=result+(double)rand()/(double)RAND_MAX
  result=result*sigma;
  result=result+mean;
  return result;
}
```