コンピュータ実験２　レポート１

24cb062h 菅原明

以下にコードがあります。 https://github.com/meowwowcat/report1.git

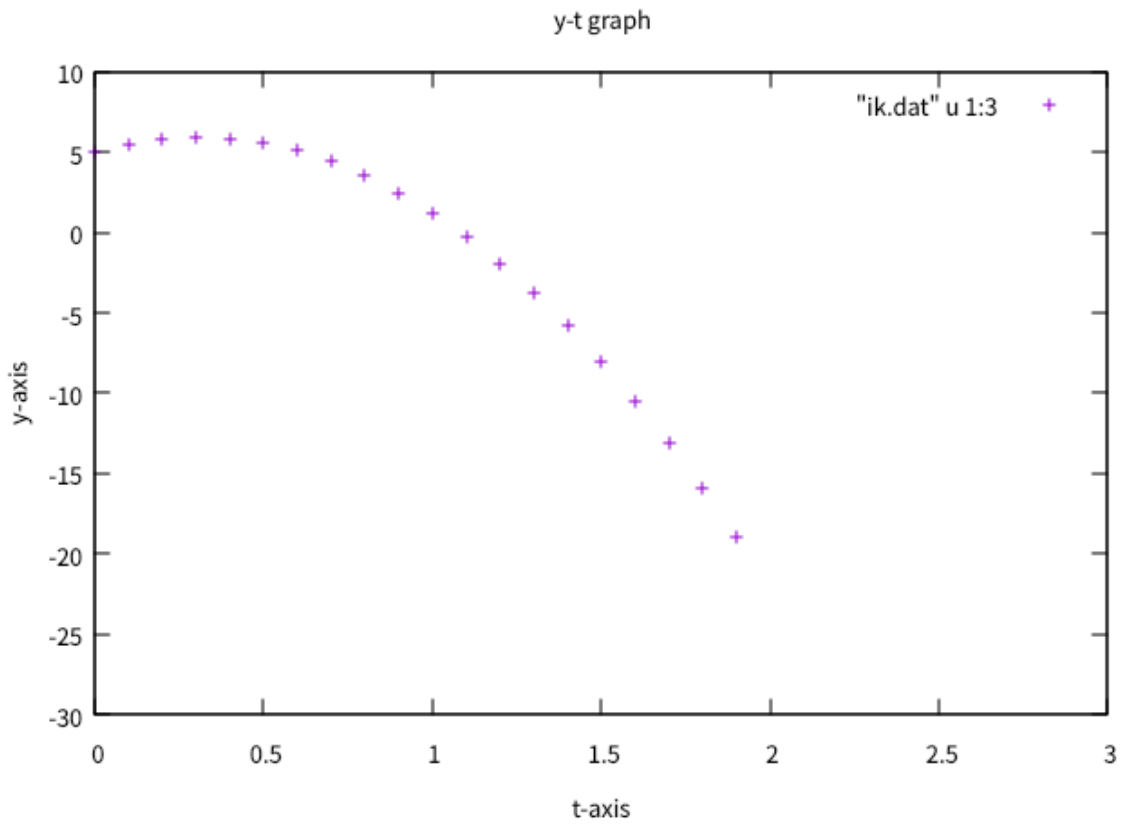
課題 13

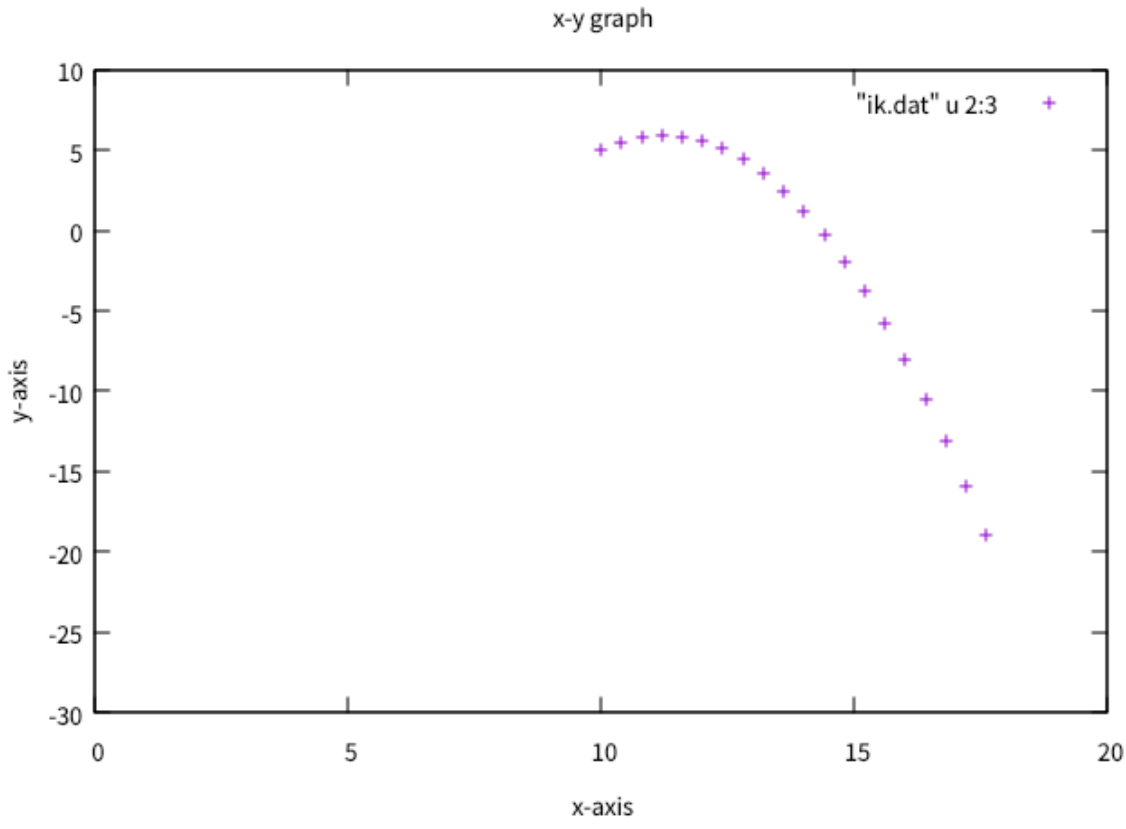
方法

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

#define g 9.8 /*gravitation acc  (m/s^2)*/
#define dt 0.1

int main(void){
        float x0,y0,vx0,vy0; /*初期条件(m,m/s)*/
        float x,y,t,vx,vy; /*位置(m), 速度8(m/s)時刻(s)*/
        float delx,dely,delvx,delvy;

        x0=10; y0=5;
        vx0=4; vy0=6; /*初期条件の値*/

        FILE*fp;   /*ファイルポインタの宣言*/
        fp=fopen("ik.dat","w");
        for(t=0;t<2.0;t=t+dt){
                        vx=vx0;
                        delvx=0;
                        vy=vy0-g*t;
                        delvy=vy-g*dt;
                        x=x0+vx*t;
                        delx=x+vx*dt;
                        y=y0+vy*t-1/2*g*t*t;
                        dely=y+vy*dt-g*t*dt;
                        printf("%8.3f,%8.3f,%8.3f\n",t,x,y);
                        fprintf(fp,"%8.3f,%8.3f,%8.3f\n",t,x,y);
                }
    fclose(fp);
    return 0;
}
```

結果

```
akira@akira:~/akira/clang/13-14-15-16/13$ cat ik.dat
   0.000,  10.000,   5.000
   0.100,  10.400,   5.502
   0.200,  10.800,   5.808
   0.300,  11.200,   5.918
   0.400,  11.600,   5.832
   0.500,  12.000,   5.550
   0.600,  12.400,   5.072
   0.700,  12.800,   4.398
   0.800,  13.200,   3.528
   0.900,  13.600,   2.462
   1.000,  14.000,   1.200
   1.100,  14.400,  -0.258
   1.200,  14.800,  -1.912
   1.300,  15.200,  -3.762
   1.400,  15.600,  -5.808
   1.500,  16.000,  -8.050
   1.600,  16.400, -10.488
   1.700,  16.800, -13.122
   1.800,  17.200, -15.952
   1.900,  17.600, -18.978
```


y-t graph
"ik.dat" u 1:3
y-axis
t-axis
10
5
0
-5
-10
-15
-20
-25
-30
0
0.5
1
1.5
2
2.5
3


x-y graph
"ik.dat" u 2:3
y-axis
x-axis
10
5
0
-5
-10
-15
-20
-25
-30
0
5
10
15
20

考察

放物運動のグラフが得られた。dt を$\lim_{dt \to 0}$とすれば完全な放物運動が得られそう。

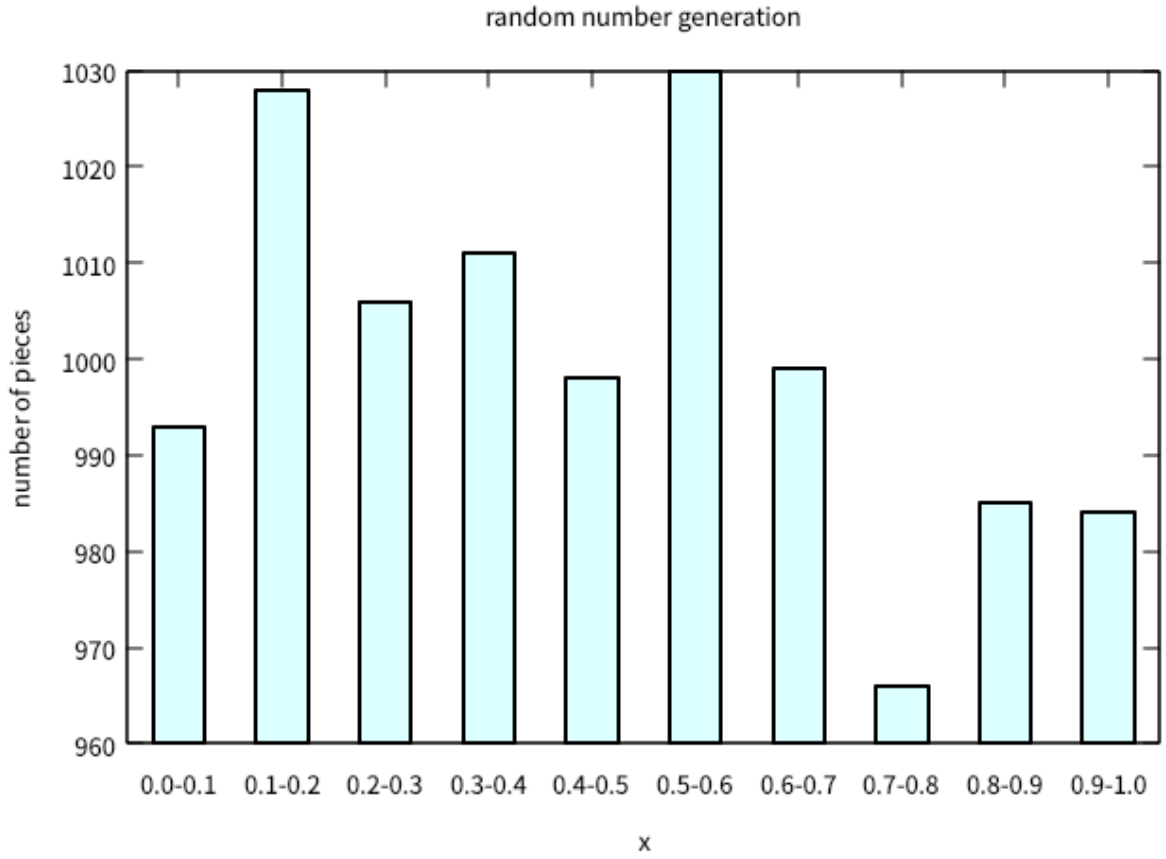
課題　14

方法

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

int main(void)
{
	FILE*fp;

	int i,rnd;
	float x;
	int hst[10]={0};


	for(i=0;i<10000;i++){	/*乱数10000個生成*/
			rnd=rand();
			x=(float)rnd/RAND_MAX;	/*生成したものを0.0~1.0に収める*/

			int zone=(int)(x*10);	/*分類*/
			if(zone>=0&&zone<10){
			hst[zone]++;
				}
		}

		fp=fopen("qa.dat","w");
		for(i=0;i<10;i++){
				printf("hst[&d]=%d\n",i,hst[i]);
				fprintf(fp,"%.1f-%.1f %d\n",i*0.1,(i+1)*0.1,hst[i]);
			}
			fclose(fp);

			return 0;

	}
```

結果

```
akira@akira:~/akira/clang/13-14-15-16/14$ cat qa.dat
0.0-0.1 993
0.1-0.2 1028
0.2-0.3 1006
0.3-0.4 1011
0.4-0.5 998
0.5-0.6 1030
0.6-0.7 999
0.7-0.8 966
0.8-0.9 985
0.9-1.0 984
```



考察

妥当な結果である．どの区間も同じような値を取っていることから，乱数生成ができている．

課題　15

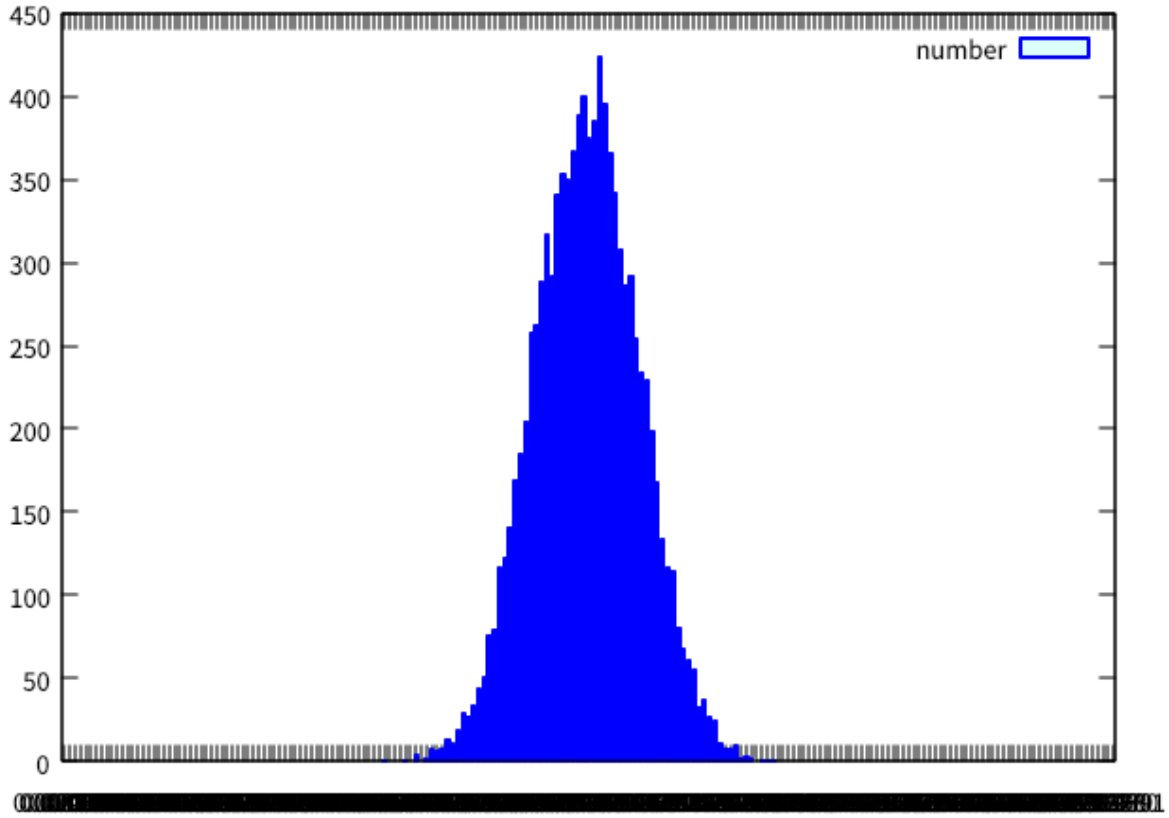
方法

幅を 0.01 にしてヒストグラムをつくた．

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

#define Max_x 2.0
#define Min_x 0.0
#define Max_y 1.0
#define mu 1.0
#define sigma 0.1

float func(float x);

int main(void)
{
                FILE*fp;
                int i,j,k,rank;
                float x,y_try,y_ref;
                int hst[201]={0};
                j=0;
                for(i=0;j<10000;i++){
                                x=(float)rand()/RAND_MAX*(Max_x-Min_x)+Min_x;
                                y_ref=func(x); /*棄却法*/
                                y_try=(float)rand()/RAND_MAX*Max_y;
                                if(y_try<=y_ref){
                                                j++;
                                                rank=(int)((x-Min_x)/0.01);
                                                if(rank >=0 && rank<201){  /*同じ幅の乱数まとめる*/
                                                        hst[rank]++;
                                                        }
                                        }
                        }
                        fp=fopen("er.dat","w");
                        for(k=0;k<201;k++){
                                        printf("hst[%d]=%d\n",k,hst[k]);
                                        fprintf(fp,"%.2f-%.2f %d\n",k*0.01,(k+1)*0.01,hst[k]);
                                }
                                fclose(fp);

                                return 0;

        }
float func(float x){
      float y;
      y=exp(-(x-mu)*(x-mu)/(2*sigma*sigma));
      return y;
        }

```

結果

er.dat ファイルはう url で見れます。



考察

1.0 を中心とする正規分布が得られた。結果は正しいだろう。

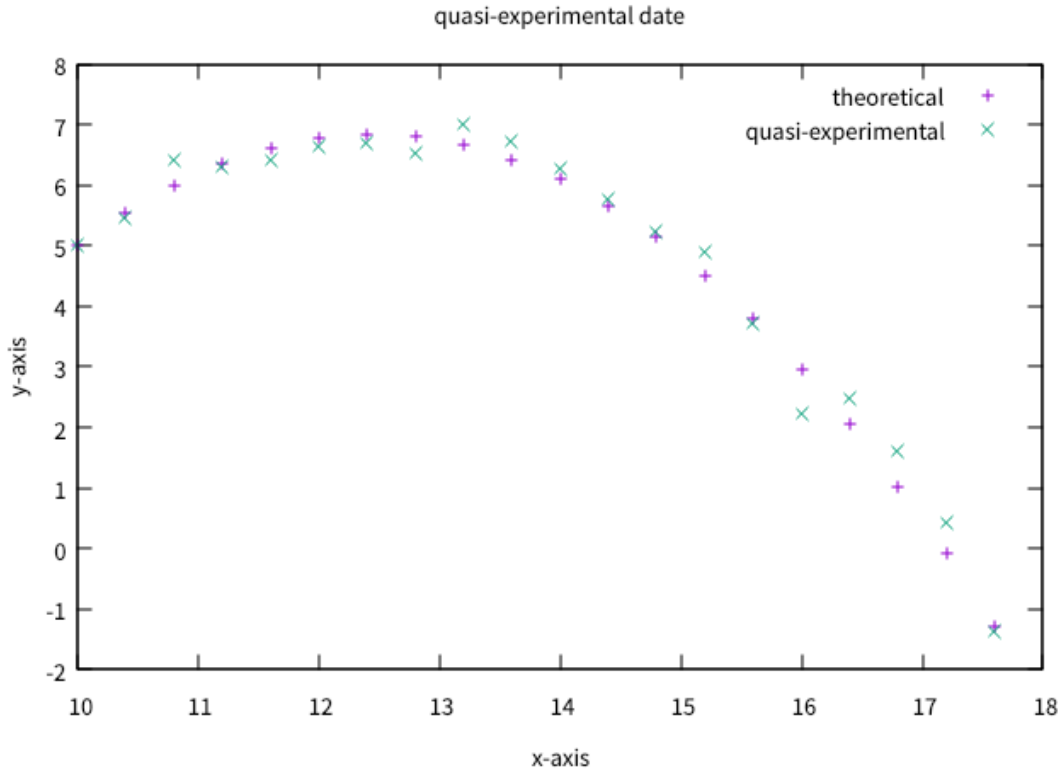
課題 16

方法

```c

#define Max_x 2.0
#define Min_x -2.0
#define Max_y 1.0
#define mu 0.0
#define sigma 0.5
#define g 9.8
#define dt 0.001
#define OUTPUT 100

float func(float x);
float gen_error();

int main()
{
   FILE*fp;
   fp=fopen("u.dat","w");
   int i,j,output_counter;
   float t,x,y,vx,vy,error;

   x=10; y=5;
   vx=4; vy=6;

   fprintf(fp,"%8.3f %8.3f %8.3f %8.3f 0.000\n",t,x,y,y);

   output_counter=0;

   for(t=0;t<=2;t+=dt){
                   error=gen_error();
                   printf("%8.3f %8.3f %8.3f %8.3f\n",t,x,y+error,error);
                   if(output_counter==OUTPUT){
                                   fprintf(fp,"%8.3f %8.3f %8.3f %8.3f %8.3f\n",t,x,y,y+error,error);
                                   output_counter=0;
                           }
                           output_counter++;

                           vy=vy-g*dt;
                           x=x+vx*dt;
                           y=y+vy*dt;
           }
           fclose(fp);
           return 0;
       }
float func(float x){
                float y;
                y=exp(-(x-mu)*(x-mu)/(2*sigma*sigma));
                return y;
        }

        float gen_error(){
                        float x_rnd,y_try,y_ref;
                        while(1){
                                x_rnd=Min_x+(float)rand()/RAND_MAX*(Max_x-Min_x);
                                y_try=(float)rand()/RAND_MAX*Max_y;
                                y_ref=func(x_rnd);
                                if(y_try<=y_ref){
                                                return x_rnd;
                                        }
                                }
                }
16.c
```

結果

```
  0.000   10.000    5.000    5.000 0.000
  0.100   10.400    5.551    5.454   -0.097
  0.200   10.800    6.003    6.400    0.397
  0.300   11.200    6.358    6.312   -0.045
  0.400   11.600    6.614    6.424   -0.190
  0.500   12.000    6.773    6.651   -0.121
  0.600   12.400    6.833    6.708   -0.125
  0.700   12.800    6.796    6.531   -0.265
  0.800   13.200    6.660    7.004    0.344
  0.900   13.600    6.427    6.735    0.309
  1.000   14.000    6.095    6.285    0.189
  1.100   14.400    5.666    5.764    0.098
  1.200   14.800    5.138    5.234    0.096
  1.300   15.200    4.513    4.909    0.397
  1.400   15.600    3.789    3.708   -0.081
  1.500   16.000    2.968    2.219   -0.749
  1.600   16.400    2.048    2.474    0.426
  1.700   16.800    1.031    1.620    0.590
  1.800   17.199   -0.085    0.422    0.507
  1.900   17.599   -1.298   -1.366   -0.067
```



考察

誤差付きのデータも理論値付近にいるので，妥当なデータが得られた．

サポート

小木曽くんにすべてのコードの解説をしてもらいました．