



# 数値解析特論I

---

- ・ C言語によるプログラミングー初歩
- ・ gnuplotによる可視化

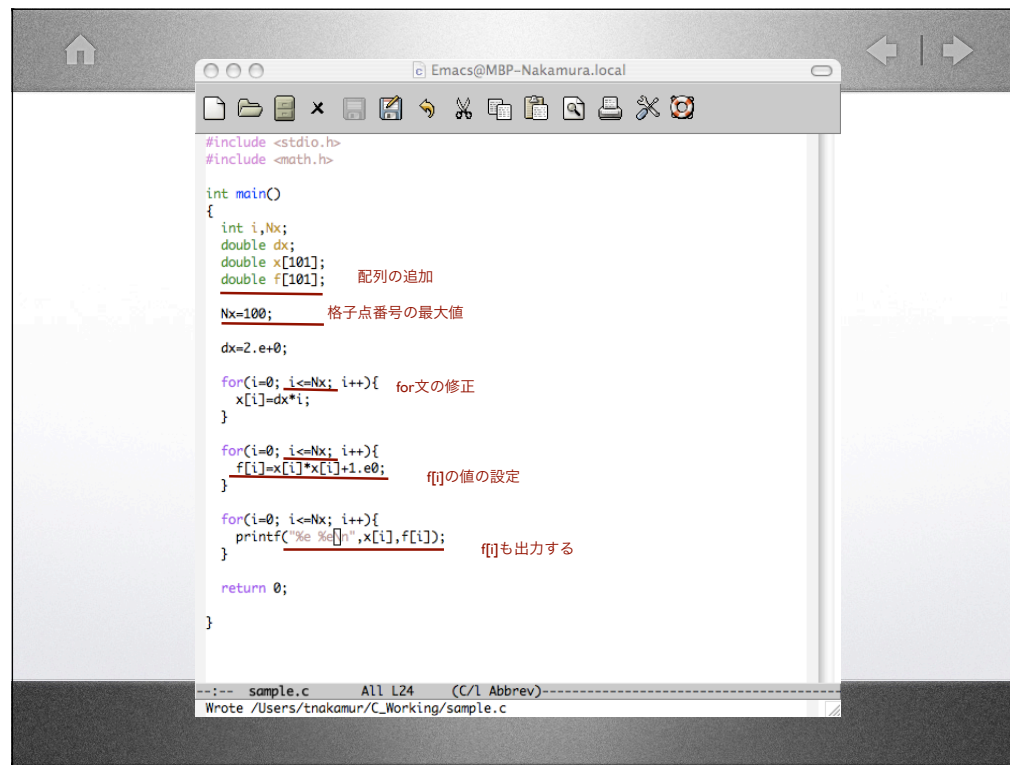


## 宿題



- 等間隔格子:  $x_i = \Delta x \times i$   $\Delta x = 2$   $i = 0, 1, 2, 3, \dots, 100$
- 関数  $f(x) = x^2 + 1 \Rightarrow f[i] = x[i] * x[i] + 1$
- 各格子点の座標と関数の値を以下のように出力するプログラムをx及びfの配列を用いて書きなさい。

0	x[0]	f[0]
1	x[1]	f[1]
2	x[2]	f[2]
3	x[3]	f[3]
...		



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

int main()
{
    int i, Nx;
    double dx;
    double x[101];
    double f[101];    配列の追加

    Nx=100;           格子点番号の最大値

    dx=2.e+0;

    for(i=0; i<=Nx; i++){    for文の修正
        x[i]=dx*i;
    }

    for(i=0; i<=Nx; i++){
        f[i]=x[i]*x[i]+1.e0;    f[i]の値の設定
    }

    for(i=0; i<=Nx; i++){
        printf("%e %e\n", x[i], f[i]);    f[i]も出力する
    }

    return 0;
}
```

--:-- sample.c All L24 (C/l Abbrev)--  
Wrote /Users/tnakamur/C\_Working/sample.c



A diagram of a horizontal beam with two downward-pointing arrows representing forces. The first arrow is positioned over the first of three circles on the left, and the second arrow is positioned further to the right, over an empty section of the beam.



## ファイルへの出力

- 出力するファイルの場所を格納する変数

`FILE *fp;`

- `fopen`
- `fclose`
- `fprintf`

```
Emacs@MBP-Nakamura.local
#include <stdio.h>
#include <math.h>

int main()
{
    int i, Nx;
    double dx;
    double x[101];
    double f[101];
    FILE *fp;

    Nx=100;
    dx=2.0e-01;

    for(i=0; i<=Nx; i++){
        x[i]=dx*i;
    }

    for(i=0; i<=Nx; i++){
        f[i]=x[i]*x[i]+1.0;
    }

    fp=fopen("output_file.dat", "w");

    for(i=0; i<=Nx; i++){
        fprintf(fp, "%e %e\n", x[i], f[i]);
    }

    fclose(fp);

    return 0;
}
```

--:-- sample.c All L31 (C/l Abbrev)-----  
Wrote /Users/tnakamura/C\_Working/sample.c



- fopen文

出力（入力）するファイルを指定する。

```
fp=fopen("output_file.dat", "w");
```

出力するファイル名

書き出すときの指定

- fprintf 文 ( printf 文とほぼ同じ書き方)

ファイルへ出力する。

```
fprintf(fp, "%e %e \n", x[i], f[i]);
```

出力するファイル

printfと同じ



- fclose文

fclose文で指定したファイルへの出力（入力）を完了する。（ファイルを閉じる）

```
fclose(fp);
```





## お約束

1. `fprintf` 文の前にファイルをopenする。
2. 書き込みが終了したらfcloseする。



- コンパイル+実行

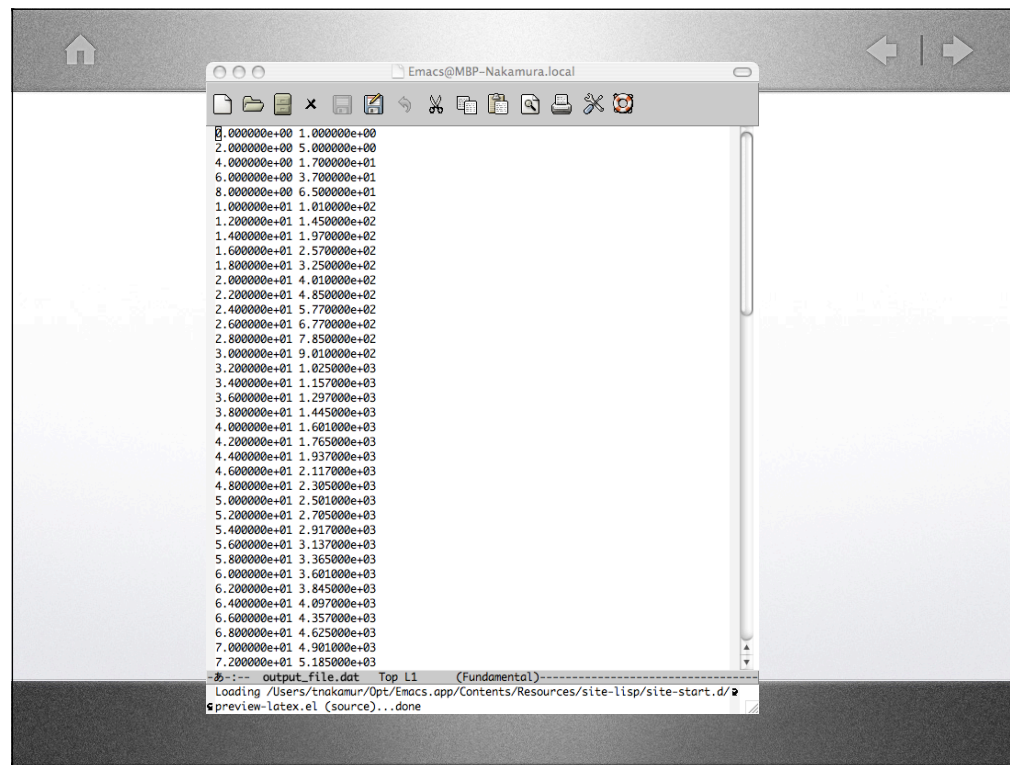
```
~~~~~  
MBP/Users/tnakamur/C_Working 18> gcc -o run sample.c -lm  
MBP/Users/tnakamur/C_Working 19> ./run
```

- lsでoutput\_file.datが作成されているか？

```
MBP/Users/tnakamur/C_Working 20> gcc -o run sample.c -lm  
MBP/Users/tnakamur/C_Working 21> ./run  
MBP/Users/tnakamur/C_Working 22> ls  
output_file.dat      sample.c  
run*                  sample.c~  
MBP/Users/tnakamur/C_Working 23>
```

- output\_file.datをemacsで開いてみる。

```
MBP/Users/tnakamur/C_Working 23> emacs output_file.dat
```





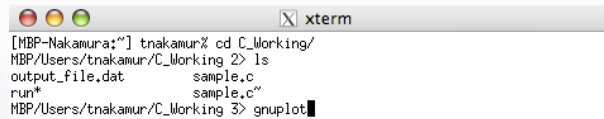
- Freeの高速・多機能グラフソフトの定番
- Windows版、Unix版、Mac版、etc 全部無料
- Internetからダウンロードして使用可能
- コンソールみたいに使用します。(コマンドを打ち込み描画させます)
- 1次元、2次元の各種グラフを書けます。



- gnuplotの起動

コンソール（ターミナル）で

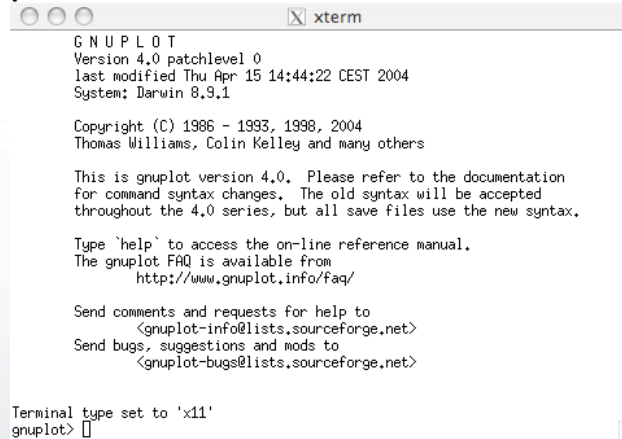
% gnuplot ↓



```
MBP-Nakamura:~$ cd C_Working/
MBP/Users/tnakamura/C_Working 2> ls
output_file.dat      sample.c
run*                  sample.c"
MBP/Users/tnakamura/C_Working 3> gnuplot
```



- gnuplotの入力画面



```
GNUPLOT
Version 4.0 patchlevel 0
last modified Thu Apr 15 14:44:22 CEST 2004
System: Darwin 8.9.1

Copyright (C) 1986 - 1993, 1998, 2004
Thomas Williams, Colin Kelley and many others

This is gnuplot version 4.0. Please refer to the documentation
for command syntax changes. The old syntax will be accepted
throughout the 4.0 series, but all save files use the new syntax.

Type 'help' to access the on-line reference manual.
The gnuplot FAQ is available from
  http://www.gnuplot.info/faq/

Send comments and requests for help to
  <gnuplot-info@lists.sourceforge.net>
Send bugs, suggestions and mods to
  <gnuplot-bugs@lists.sourceforge.net>

Terminal type set to 'x11'
gnuplot> 
```

”gnuplot>” のところに各種コマンドを打ち込む



- データファイルの描画

```
xterm
GNUPLOT
Version 4.0 patchlevel 0
last modified Thu Apr 15 14:44:22 CEST 2004
System: Darwin 8.9.1

Copyright (C) 1986 - 1993, 1998, 2004
Thomas Williams, Colin Kelley and many others

This is gnuplot version 4.0. Please refer to the documentation
for command syntax changes. The old syntax will be accepted
throughout the 4.0 series, but all save files use the new syntax.

Type 'help' to access the on-line reference manual.
The gnuplot FAQ is available from
http://www.gnuplot.info/faq/

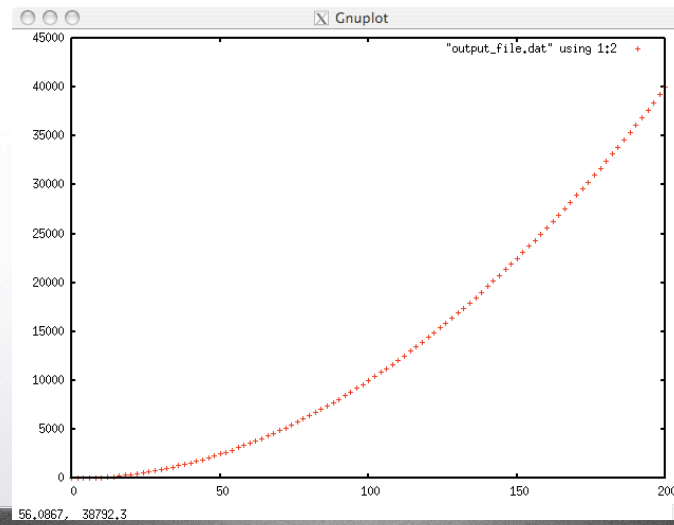
Send comments and requests for help to
<gnuplot-info@lists.sourceforge.net>
Send bugs, suggestions and mods to
<gnuplot-bugs@lists.sourceforge.net>

Terminal type set to 'x11'
gnuplot> plot "output_file.dat" using 1:2 with points
```

gnuplot>plot “ファイル名” using 1:2 with points



- データファイルの描画







- plot コマンドの意味 (X-Yのグラフを書く)

```
gnuplot> plot "output_file.dat" using 1:2 with points
```

ファイル名

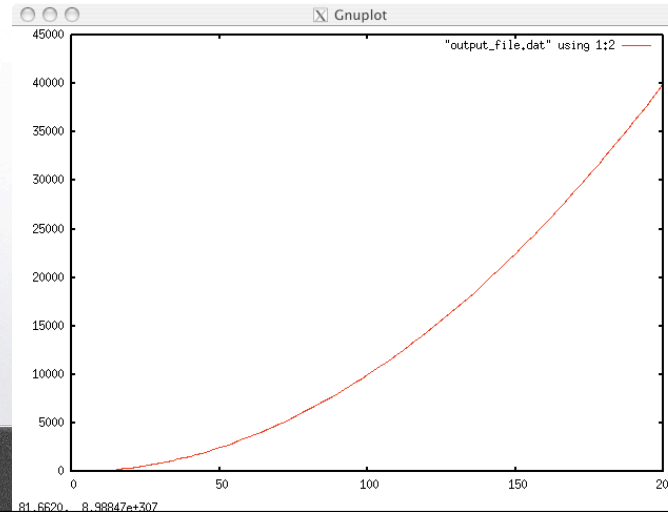
↑  
ファイルの1列目をX  
ファイルの2列目をY

↑  
点を描く



- plot コマンドでいろいろやってみる！

```
gnuplot> plot "output_file.dat" using 1:2 with lines
```

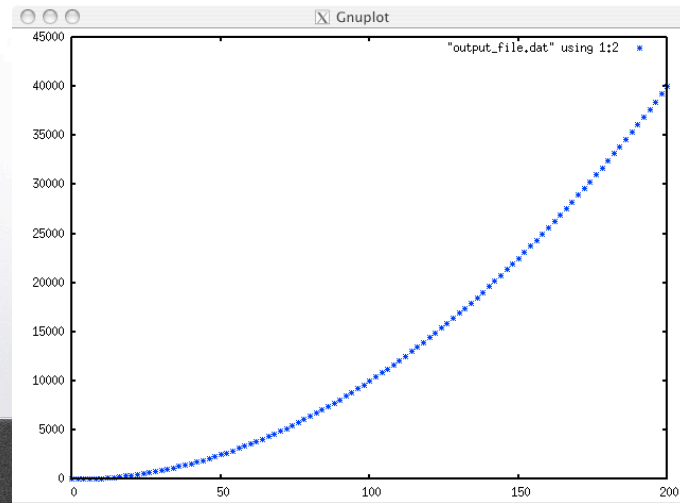


← 線を描く



- plot コマンドでいろいろやってみる！

```
gnuplot> plot "output_file.dat" using 1:2 with points 3
```

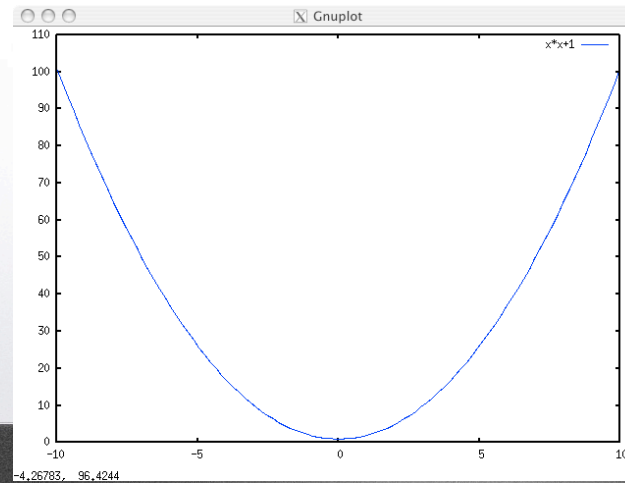


色を変更



- plot コマンドでいろいろやってみる！

gnuplot> plot  $x*x+1$  with lines 3



関数

$x*x+1$   
の形を書く



- plot コマンドでいろいろやってみる！

ちなみにいろんな関数が使えます

```
gnuplot> plot sin(x) with lines 3
```

```
gnuplot> plot cos(x) with lines 3
```

```
gnuplot> plot exp(x) with lines 3
```

```
gnuplot> plot log(x) with lines 3
```

```
gnuplot> plot x**(1/2) with lines 3
```

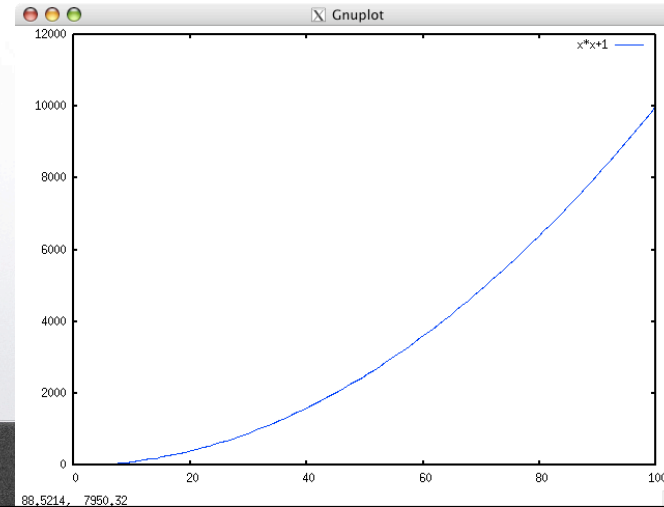
$x^{1/2} \rightarrow x \text{ の } 1/2 \text{ 乗}$



- x軸の範囲を変更する。

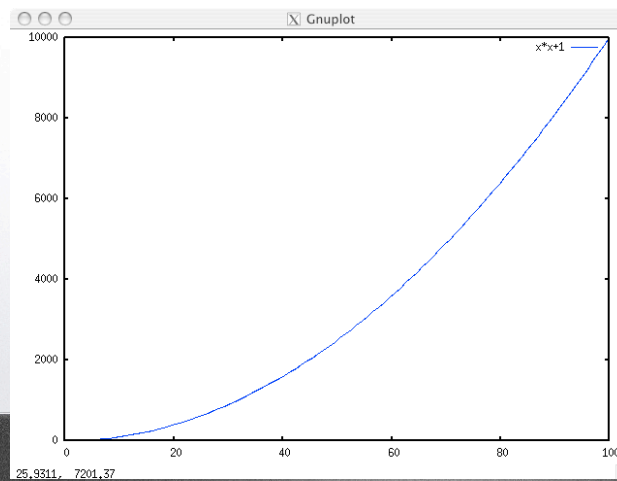
```
gnuplot > set xrange [0:100]
```

```
gnuplot > plot x*x+1 with lines 3
```





- y軸の範囲を変更する。  
gnuplot > set yrange [0:10000]  
gnuplot > plot x\*x+1 with lines 3



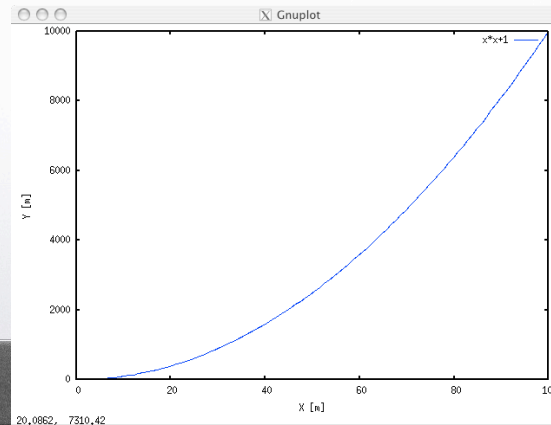


- x,y軸にラベルをつける。

```
gnuplot > set xlabel "X [m]"
```

```
gnuplot > set ylabel "Y [m]"
```

```
gnuplot > plot x*x+1 with lines 3
```

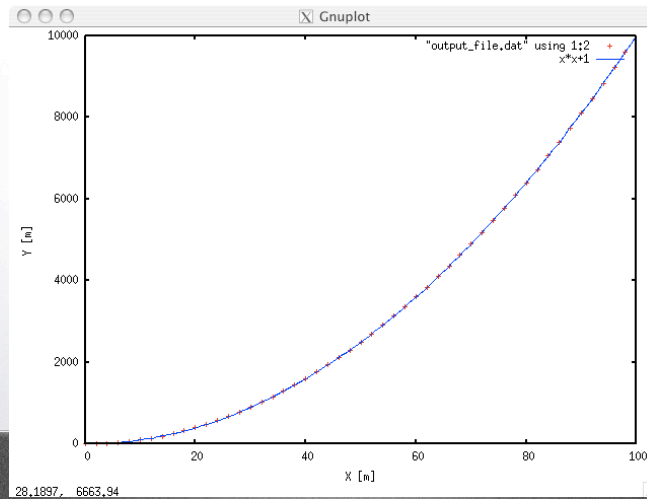






- 複数のグラフを重ねて書かせる。

gnuplot > plot "output\_file.dat" using 1:2 with points,  $x*x+1$  with line 3

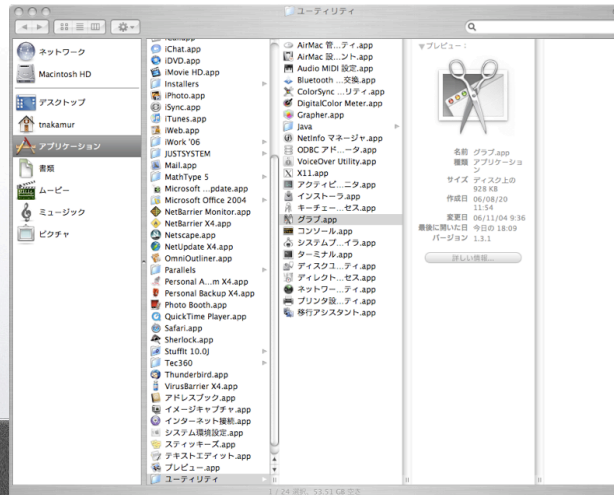




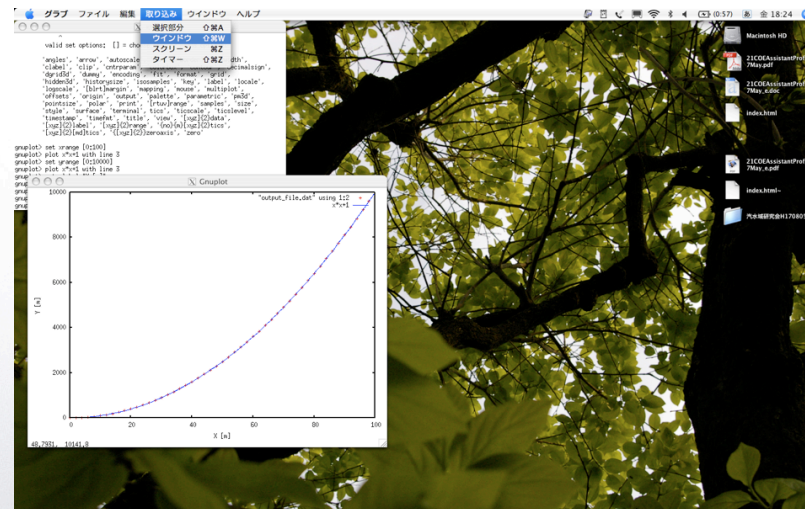
## gnuplotを使ってみる



- 画像の保存の仕方。  
gnuplotの操作は煩雑なので、Macに付いてくるソフト”グラブ”を使用する。（アプリケーション → ユーティリティ → がある。）



- 取り込み→ウィンドウ を選択



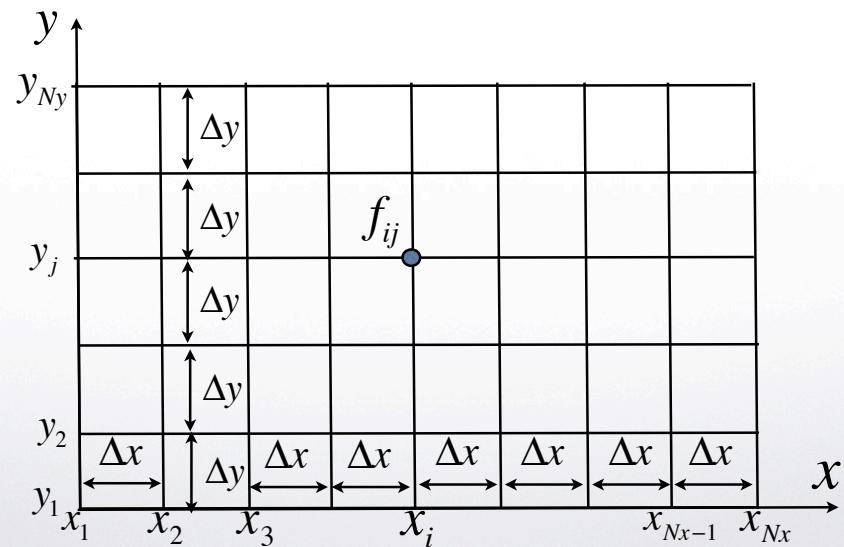
取り込んだ画像を保存する！



## 2次元にしてみよう



- 等間隔直交格子





## 2次元にしてみよう



- 等間隔直交格子

$$\begin{aligned}x_i &= \Delta x \times (i - 1) & i &= 1, 2, 3, \dots, Nx \\y_j &= \Delta y \times (j - 1) & j &= 1, 2, 3, \dots, Ny\end{aligned}$$

計算する値は格子の交点（計算格子点）上にある。

$$f_{ij} = f(x_i, y_j) \rightarrow \begin{array}{l} f_{11}, f_{12}, f_{13}, \dots, f_{1, Ny} \\ f_{21}, f_{22}, f_{23}, \dots, f_{2, Ny} \\ f_{31}, f_{32}, f_{33}, \dots, f_{3, Ny} \\ \dots \\ f_{Nx1}, f_{Nx2}, f_{Nx3}, \dots, f_{Nx, Ny} \end{array} \quad \begin{array}{l} \text{を保存する変} \\ \text{数が必要} \end{array}$$



## 2次元にしてみよう



- 2次元配列を使用する。

$$N_x = 100, N_y = 50, \Delta x = 1, \Delta y = 2$$

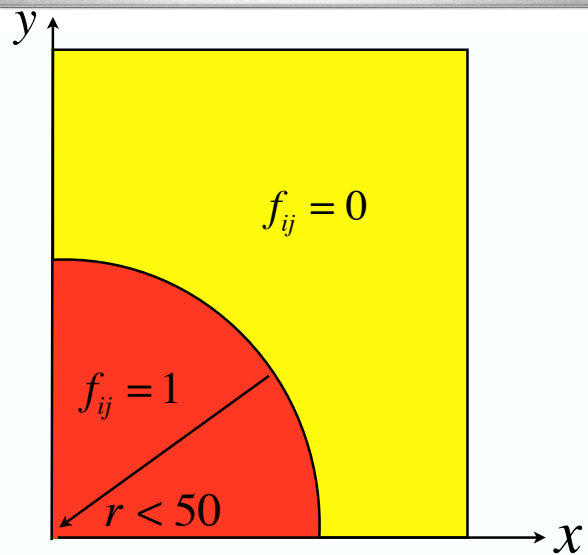
として、

$$f_{ij} = f(x_i, y_j) = \begin{cases} 1 & \text{for } \sqrt{x_i^2 + y_j^2} < 50 \\ 0 & \text{otherwise} \end{cases}$$

となるように計算して出力してみる。



## 2次元にしてみよう



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

int main()
{
    int i,j,Nx,Ny;
    double dx,dy,rr;
    double x[51],y[101];
    double f[51][101]; 2次元配列
    FILE *fp;

    Nx=50;
    Ny=100;
    dx=1.e+0;
    dy=2.e+0;
    for(i=1; i<=Nx; i++){x[i]=dx*(i-1);}
    for(j=1; j<=Ny; j++){y[j]=dy*(j-1);}

    for(i=1; i<=Nx; i++){
        for(j=1; j<=Ny; j++){
            rr=sqrt(x[i]*x[i]+y[j]*y[j]); 原点からの距離
            if(rr<50.e+0){
                f[i][j]=1.e+0;
            }else{
                f[i][j]=0.e+0;
            }
        }
    }

    fp=fopen("output_file.dat","w");
    for(i=1; i<=Nx; i++){
        for(j=1; j<=Ny; j++){
            fprintf(fp,"%e %e\n",x[i],y[j],f[i][j]);
        }
        fprintf(fp,"\n");
    }
    fclose(fp);

    return 0;
}
```



double f[51][101];

2次元配列

f[ 0][0],f[ 0][1],f[ 0][2],...,f[ 0][100]  
f[ 1][0],f[ 1][1],f[ 1][2],...,f[ 1][100]

....

f[49][0],f[49][1],f[49][2],...,f[49][100]  
f[50][0],f[50][1],f[50][2],...,f[50][100]

が使用できる。

## 2重for文

```
for(i=1; i<=Nx; i++){  
    for(j=1; j<=Ny; j++){  
        計算A  
    }  
}
```



i=1,j=1としてAを計算  
i=1,j=2としてAを計算  
i=1,j=3としてAを計算  
...  
i=1,j=NyとしてAを計算

i=2,j=1としてAを計算  
i=2,j=2としてAを計算  
i=2,j=3としてAを計算  
...  
i=2,j=NyとしてAを計算

...  
...  
...

i=Nx,j=1としてAを計算  
...  
i=Nx,j=2としてAを計算  
i=Nx,j=3としてAを計算  
i=Nx,j=NyとしてAを計算




# if 文



- 条件によって計算式を変える！

`rr=sqrt(x[i]*x[i]+y[j]*y[j]);`  `sqrt(a)` : aの平方根の関数

```
if(rr<50.e+0){  
    f[i][j]=1.e+0;  
}else{  
    f[i][j]=0.e+0;  
}
```

—————  if - else 文

rr(距離)が50未満であれば、

`f[i][j]=1.e0`

それ以外は

`f[i][j]=0.e0`

を代入する。



# if 文



- if 文の構造

```
if(条件式){  
    計算A  
}else{  
    計算B  
}
```

```
if(条件式){  
    計算A  
}
```

else文は省略可能。  
この場合条件式を満たさ  
ない場合には、何も実行  
されない。



## if 文



- 条件式

<code>if(A &lt; B){...}</code>	AがBより小さいなら
<code>if(A &gt; B){...}</code>	AがBより大きいなら
<code>if(A &lt;= B){...}</code>	AがB以下なら (A=Bも含む)
<code>if(A &gt;= B){...}</code>	AがB以上なら (A=Bも含む)
<code>if(A == B){...}</code>	AがBと等しいなら (A=B)
<code>if(A != B){...}</code>	AがBと等しくないなら ( $A \neq B$ )



## if 文



- 条件式の組み合わせ(And, Or)

```
if((A < B) && (C < D)){...}
```

AがBより小さい **かつ** CがDより小さい

```
if((A < B) || (C < D)){...}
```

AがBより小さい **あるいは** CがDより小さい



- 2次元データ  
(output\_file.dat)

```
x[ 1] y[ 1] f[ 1][ 1]  
x[ 1] y[ 2] f[ 1][ 2]  
x[ 1] y[ 3] f[ 1][ 3]  
...  
x[ 1] y[Ny] f[ 1][Ny]  
  
x[ 2] y[ 1] f[ 2][ 1]  
x[ 2] y[ 2] f[ 2][ 2]  
...  
x[ 2] y[Ny] f[ 2][Ny]  
  
x[ 3] y[ 1] f[ 3][ 1]  
x[ 3] y[ 2] f[ 3][ 2]  
...  
x[ 3] y[Ny] f[ 3][Ny]
```

← 改行

← 改行



- 2次元データファイルの描画

```
xterm
Version 4.0 patchlevel 0
last modified Thu Apr 15 14:44:22 CEST 2004
System: Darwin 8.9.1

Copyright (C) 1986 - 1993, 1998, 2004
Thomas Williams, Colin Kelley and many others

This is gnuplot version 4.0. Please refer to the documentation
for command syntax changes. The old syntax will be accepted
throughout the 4.0 series, but all save files use the new syntax.

Type `help` to access the on-line reference manual.
The gnuplot FAQ is available from
  http://www.gnuplot.info/faq/

Send comments and requests for help to
  <gnuplot-info@lists.sourceforge.net>
Send bugs, suggestions and mods to
  <gnuplot-bugs@lists.sourceforge.net>

Terminal type set to 'x11'
gnuplot> splot "output_file.dat" using 1:2:3 with line
gnuplot> 
```

gnuplot>splot “ファイル名” using 1:2:3 with lines



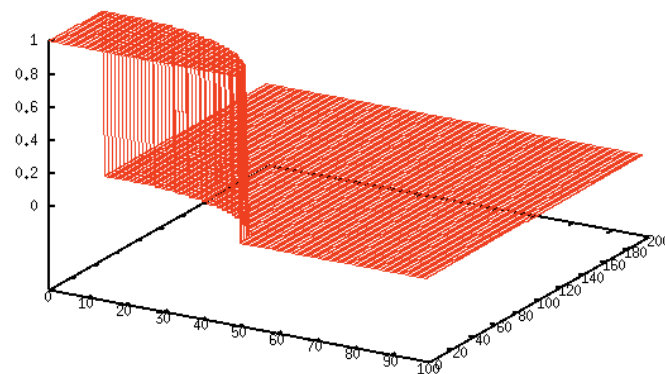


# gnuplotを使ってみる



Gnuplot

"output\_file.dat" using 1:2:3



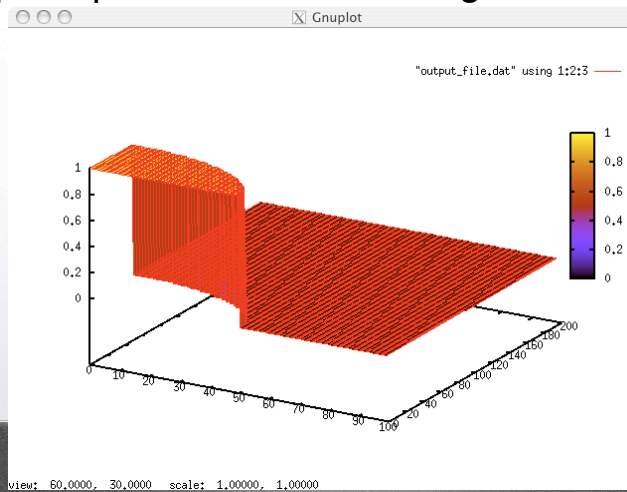
view: 60,0000, 30,0000 scale: 1,00000, 1,00000



- カラーコンター

```
gnuplot>set pm3d
```

```
gnuplot>splot "ファイル名" using 1:2:3 with lines
```



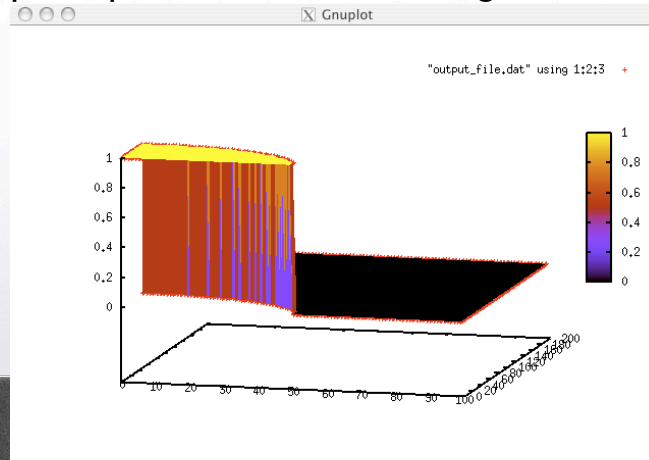


- 遮蔽処理

```
gnuplot>set pm3d
```

```
gnuplot>set hidden
```

```
gnuplot>splot “ファイル名” using 1:2:3 with lines
```





# 時間があまったら

- 積分の計算を考えてみよう！

区間 $[0,100]$ で  $I = \int_0^{100} x^2 + 1 dx$

を計算する方法を考えて、プログラムしてみる。

ヒント：計算格子点を考え、

$$x_i = \Delta x \times i \quad \Delta x = 2 \quad i = 0, 1, 2, 3, \dots, 100$$

補間をしてから積分する。