USBメモリを刺して，

１．

Tutorial.vmwarevmをデスクトップに置いてください．

２．Macの人はMacのフォルダを

Windowsの人はWindowsのフォルダ

Linuxの人はLinuxのフォルダをデスクトップにコピーしてください．

非線型数理サマーセミナー

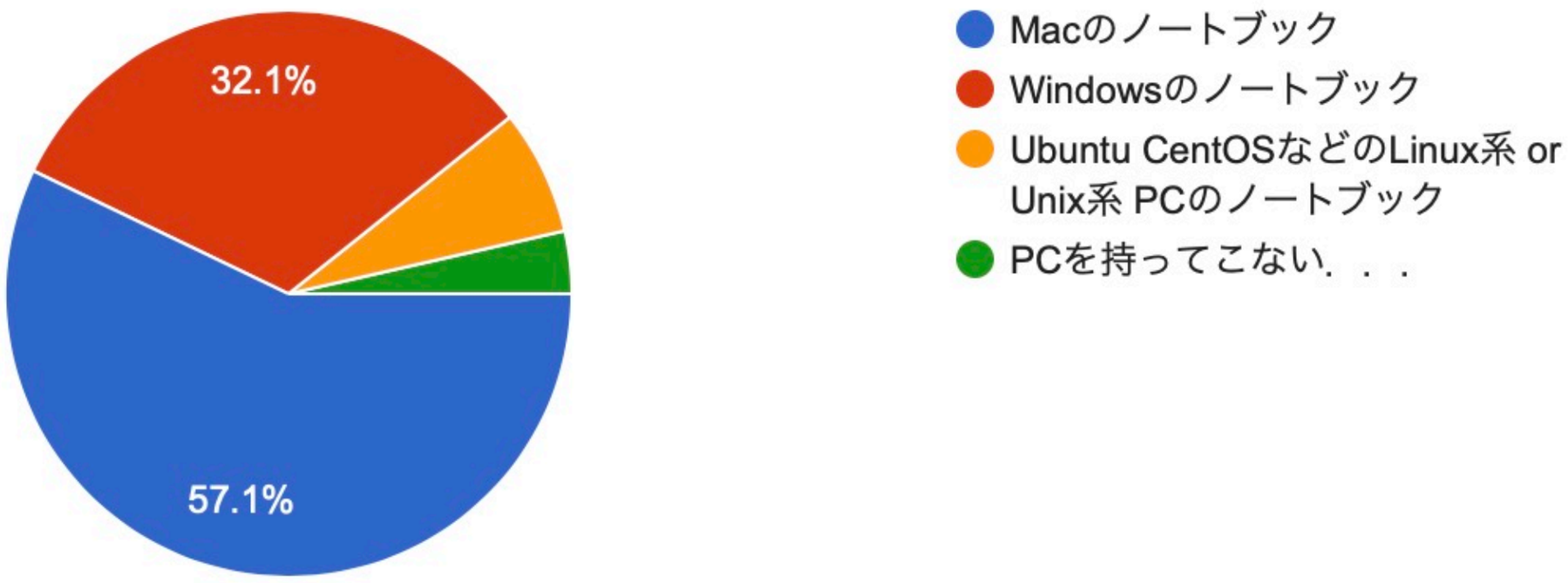
明治大学　秋山正和

「反応拡散方程式を数値計算してみよう」

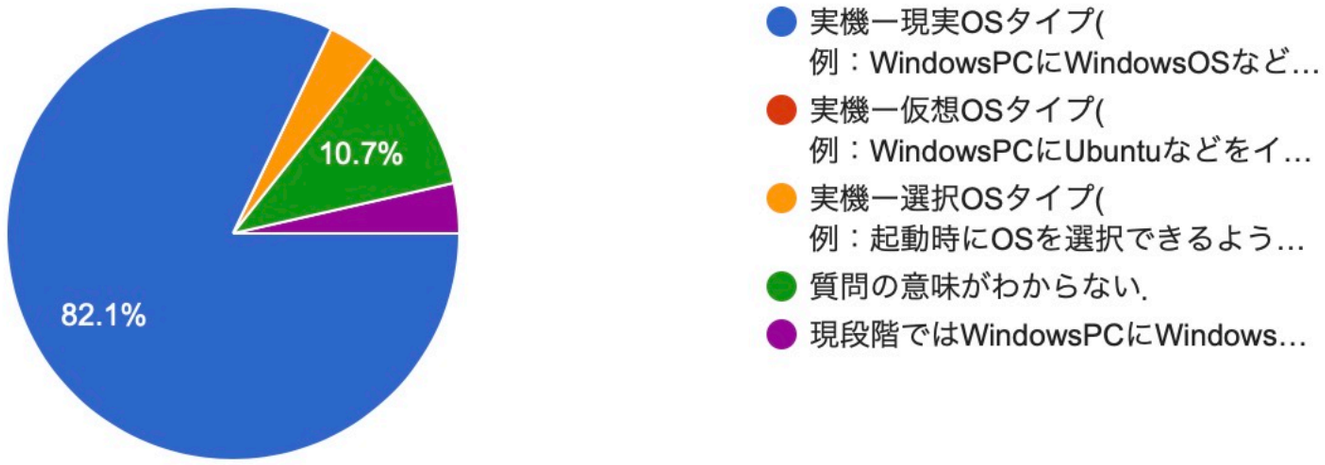
要旨：

自然界の多くの現象は反応拡散系を用いて、記述できることが知られています。反応拡散系は手で解析的に解くことが難しいケースも有り、そのような場合は、コンピュータを用いて近似的に方程式を解きます。講演ではそのような方法論を０からレクチャーします。皆さん、反応拡散系の世界を楽しく体験しましょう！

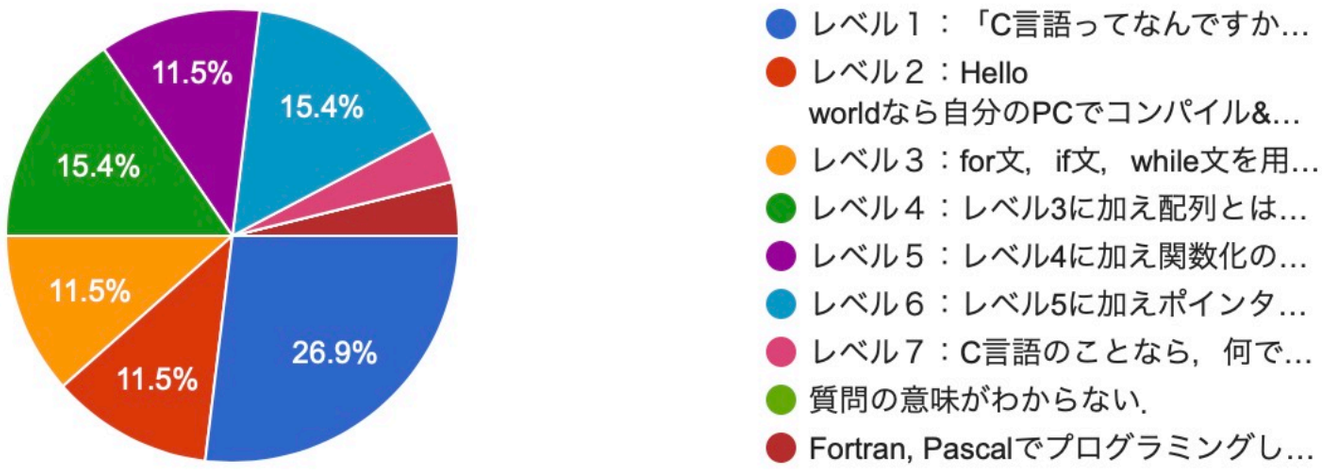
当日あなたが持ってくるPC



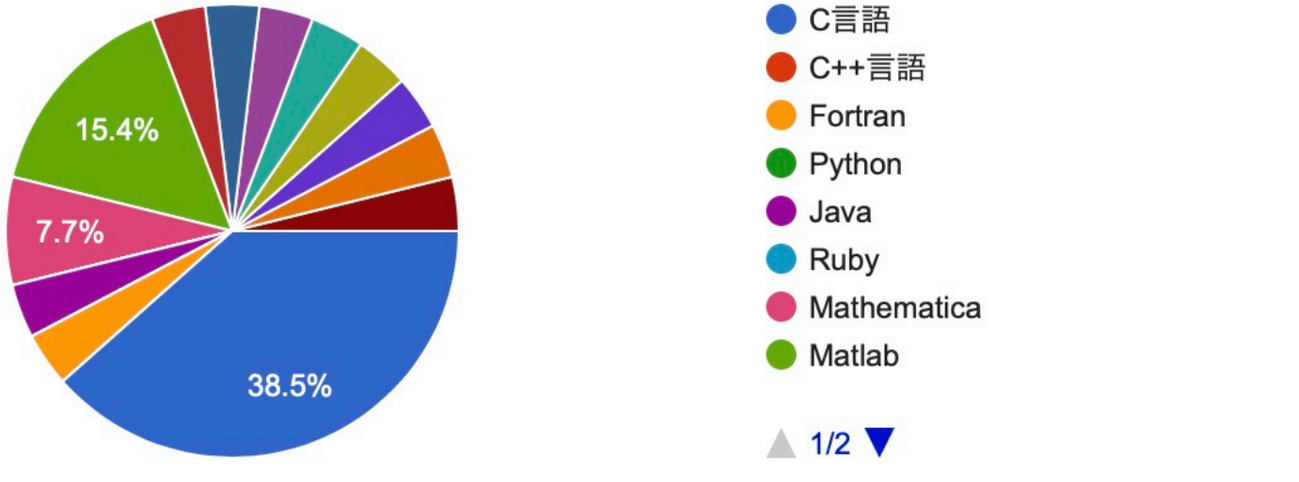
当日あなたが持ってくるPCの状態



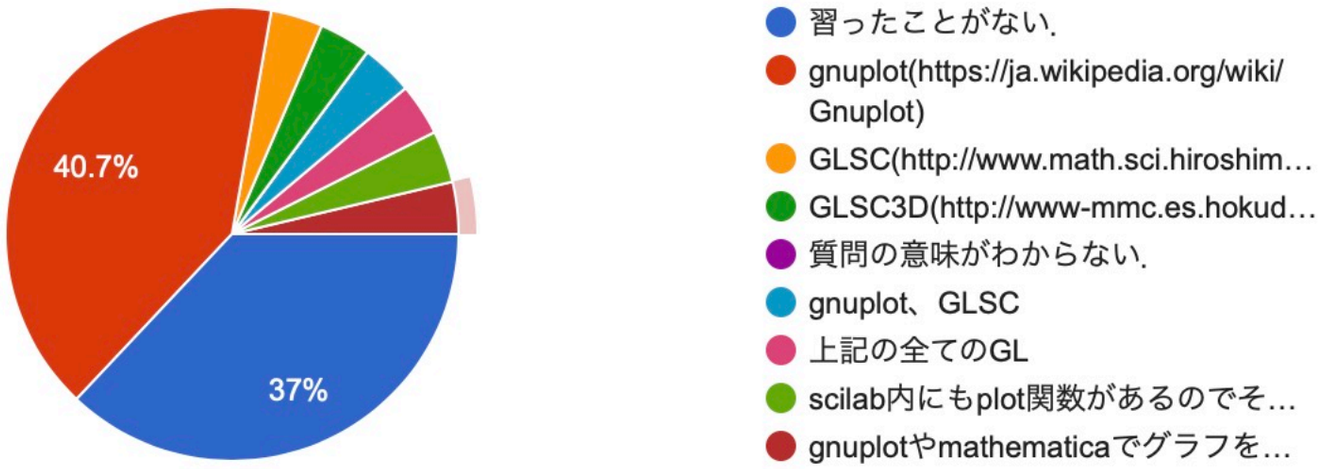
C言語のレベル



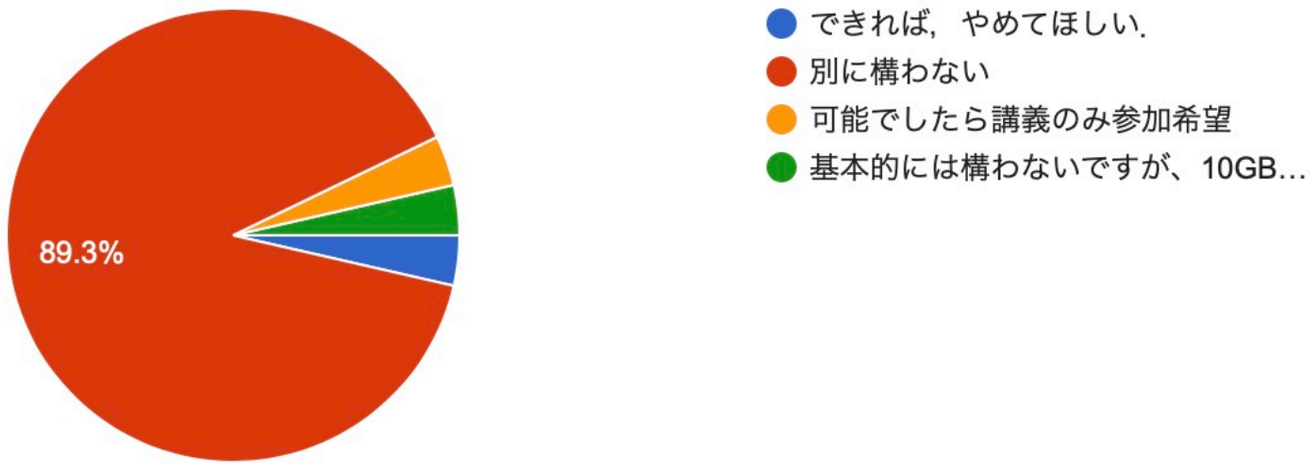
習ったプログラム言語



使ったことがある可視化



PCにいくつかのソフトをインストールするのは



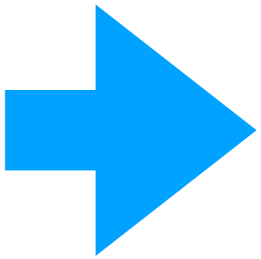
様々な環境があるので，今回は自身のPCの上に，仮想的なPCを作成して，その上で講習を行います.

はじめに

１．皆さんのPCで反応拡散方程式系の数値計算ができるようにします．

２．C言語を全く知らない方，PCに慣れていない方もいます．本講習会では，最も遅い人に合わせますので，できる人は待っていてください．

3. 全員の同期を取りながら説明をするので，
待っててほしいときは必ず手を上げてください．



Step1 ーーーーーーーー
Step2 ーーーーーーーー
Step3 ーーーーーーーー
Step4 ーーーーーーーー
Step5 ーーーーーーーー

※　遠慮して手をあげないと，講師は余計に困ってしまいます．．．

ネットワークへの接続

Mac/Win/Linux共通

1．eduroamのアカウントとパスワードを入力し，インターネットへ接続できることを確認してください．

2．この講習会でよく使うので，

http://www-mmc.es.hokudai.ac.jp/~masakazu/

をブックマークしてください．

USBメモリ

USBメモリを刺して，

１．

Tutorial.vmwarevmをデスクトップに置いてください．

２．Macの人はMacのフォルダを

Windowsの人はWindowsのフォルダ

Linuxの人はLinuxのフォルダをデスクトップにコピーしてください．

仮想環境の導入

～Windows 10編～

vmware

日本　コミュニティ | Store | ログイン

VMware Cloud　製品　ソリューション　サポート　プロフェッショナル サービス　ダウンロード　パートナー　企業情報

製品 > Workstation Player > VMware Workstation Player の試用

VMware Workstation Player は、Windows または Linux PC 上での仮想マシンの実行に最適な製品です。管理対象の企業デスクトップの配信、教育機関での学習やトレーニングなどの用途に使用できます。

無償バージョンは、商用以外での利用および個人利用のほか、生徒や学生、非営利組織でも利用できます。

商用でWorkstation Playerを使用するには、商用ライセンスが必要です。

より高度な仮想化ソリューションが必要な場合は、Workstation Pro をご検討ください。

※Windows10でない人は VMwareをダウンロードする

Workstation 15 Player for Windows の試用
今すぐダウンロード »

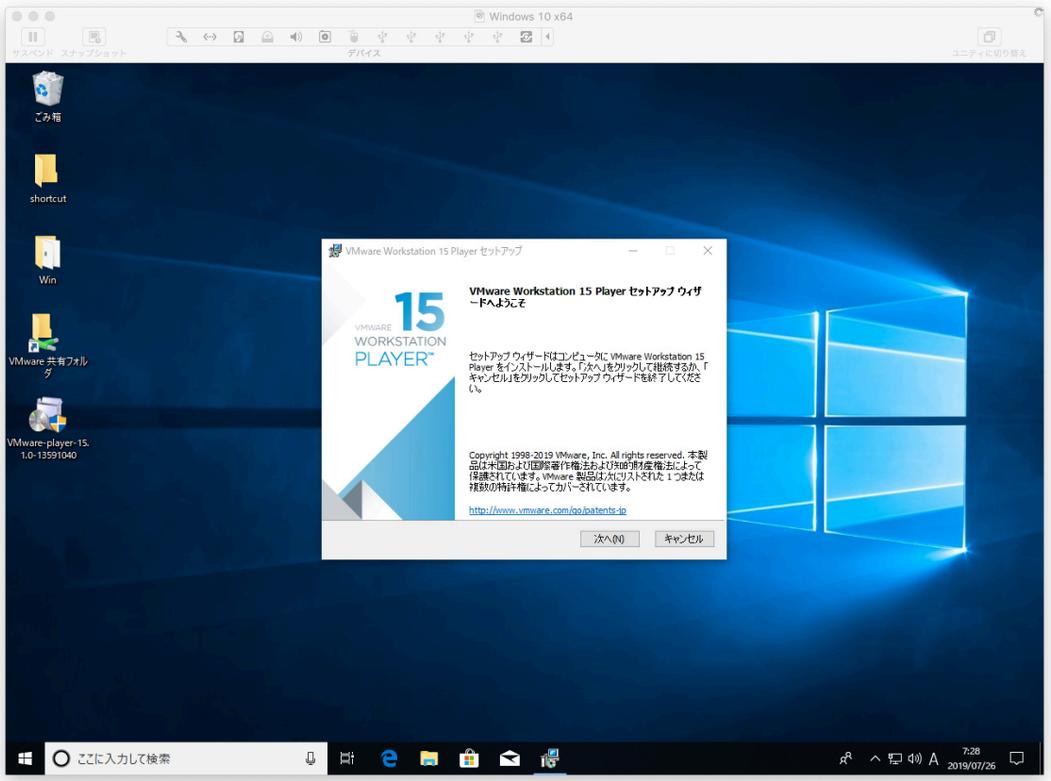
Workstation 15 Player for Linux の試用
今すぐダウンロード »

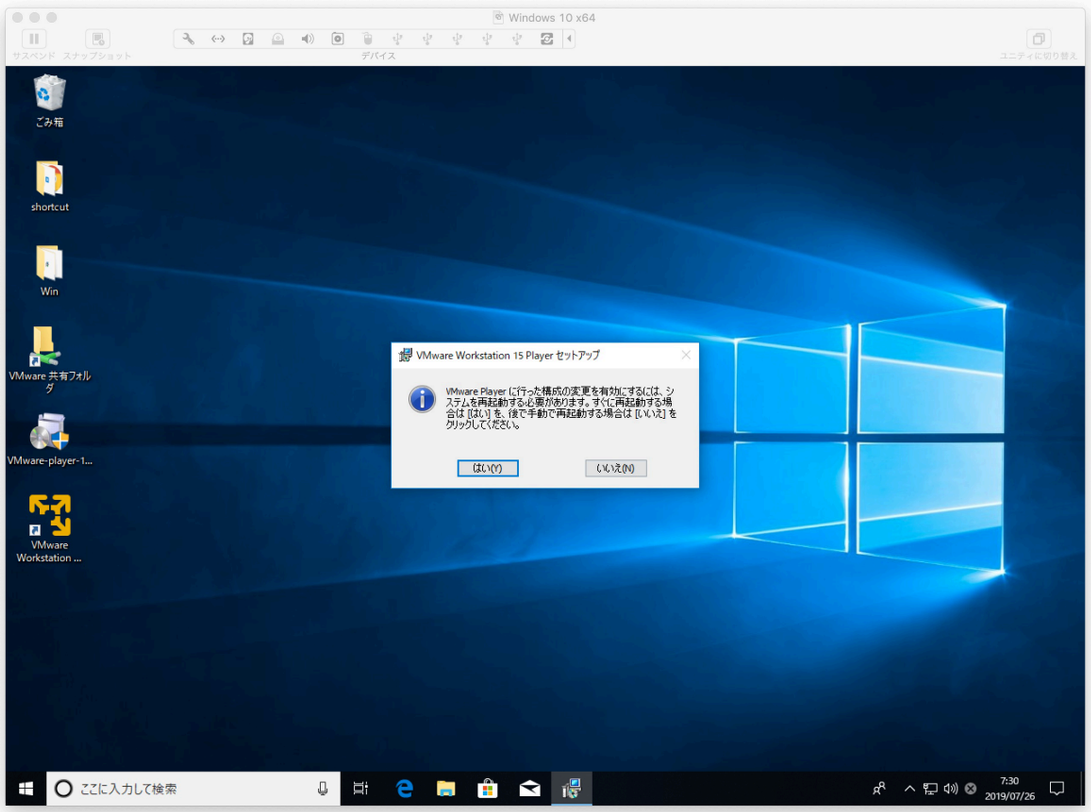
vmware

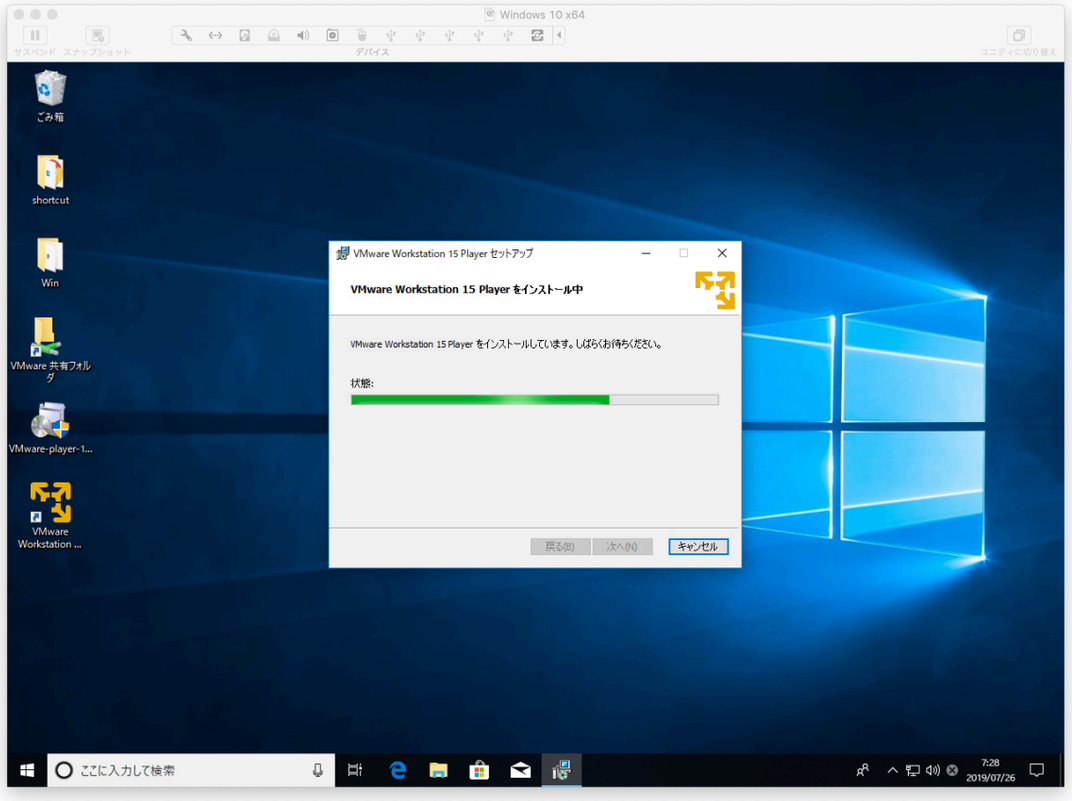
VMware は広告、ソーシャル メディア、解析の目的で Cookie を使用しています。VMware が Cookie を利用する方法およびサイト訪問者が Cookie を管理する方法については、こちらを参照してください。本サイトの閲覧を続けることにより、Cookie の使用に同意したものとみなされます。 ×

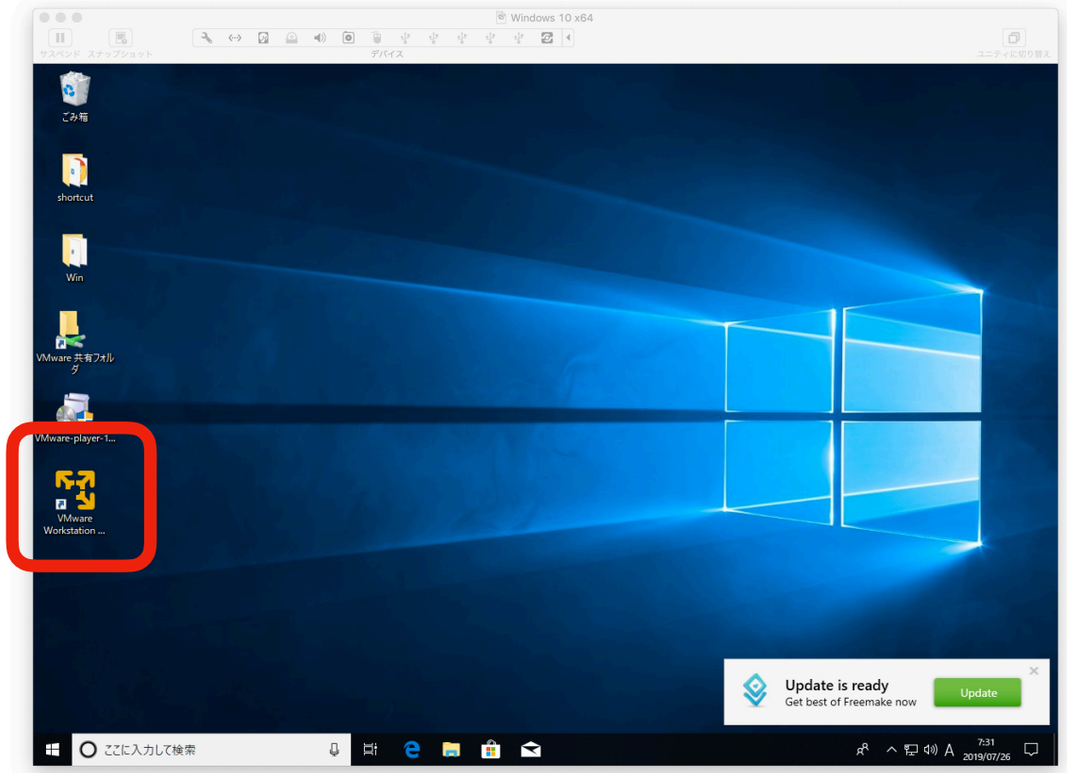
https://www.vmware.com/jp/products/workstation-player/workstation-player-evaluation.html

VMwareをインストールする

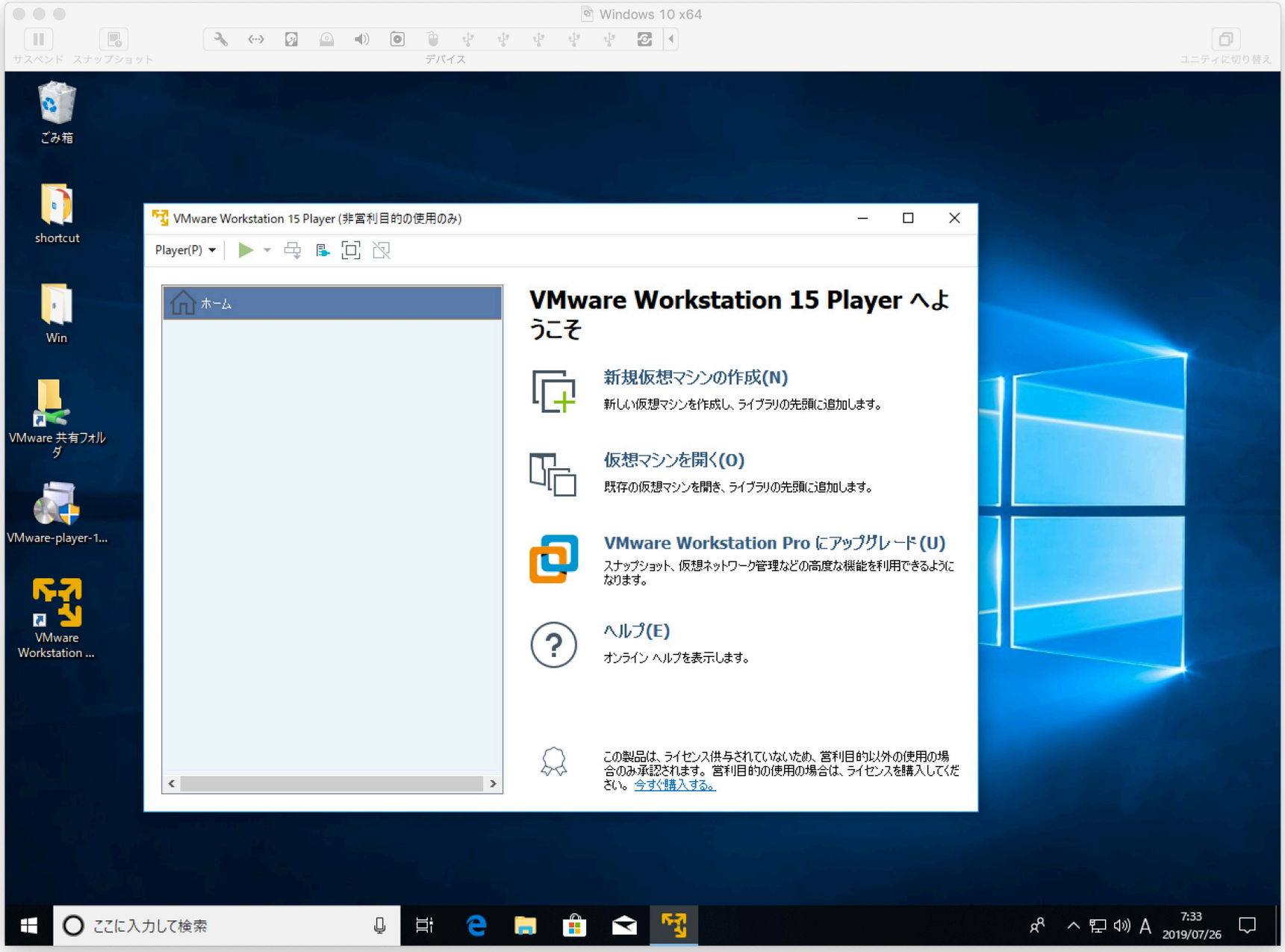




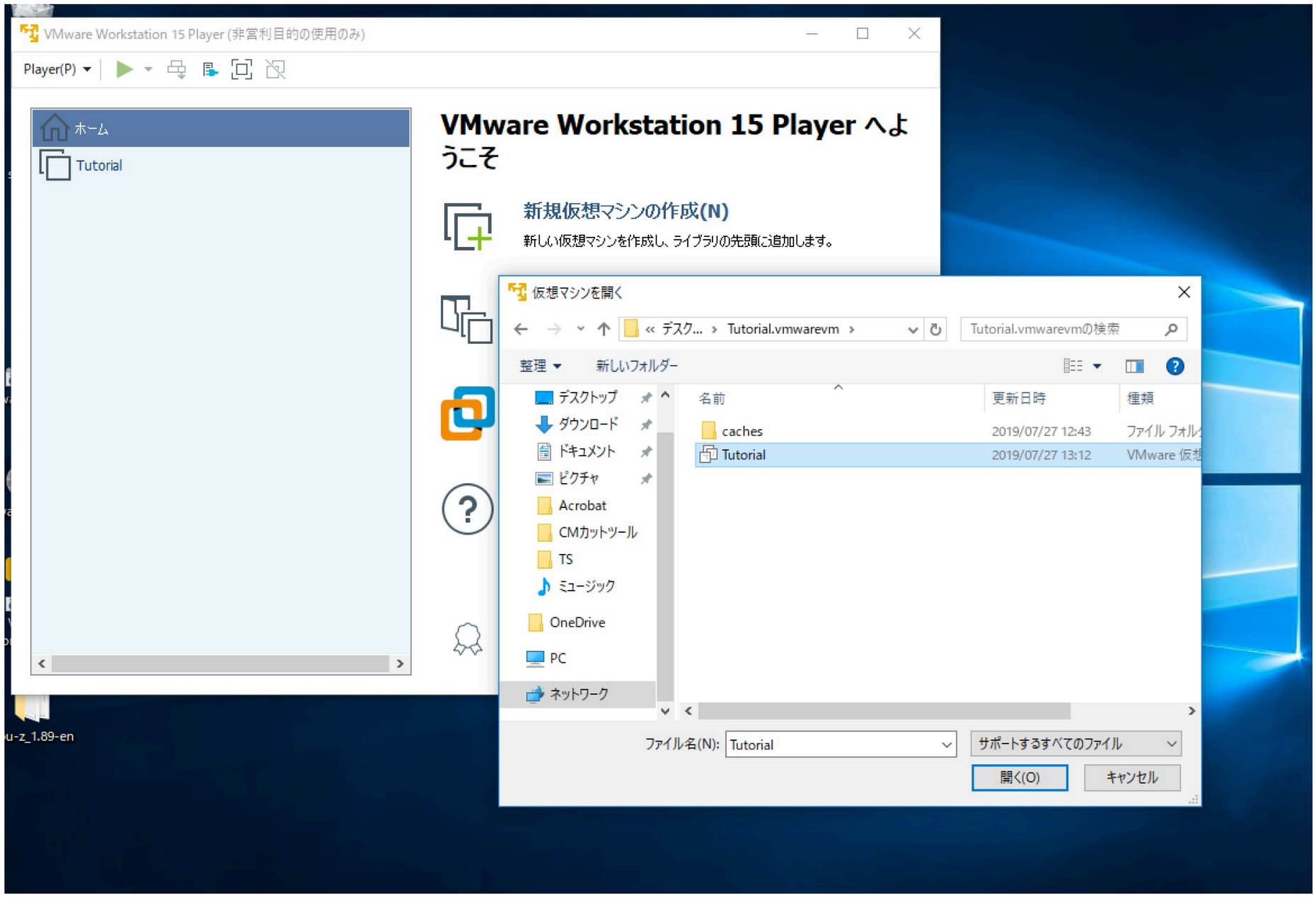




この画面が出たら待機


Windows 10 x64
サスペンド
スナップショット
デバイス
ユニティに切り替え
ごみ箱
shortcut
Win
VMware 共有フォルダ
VMware-player-1...
VMware Workstation ...
VMware Workstation 15 Player (非営利目的の使用のみ)
Player(P)
ホーム
VMware Workstation 15 Player へようこそ
新規仮想マシンの作成(N)
新しい仮想マシンを作成し、ライブラリの先頭に追加します。
仮想マシンを開く(O)
既存の仮想マシンを開き、ライブラリの先頭に追加します。
VMware Workstation Pro にアップグレード(U)
スナップショット、仮想ネットワーク管理などの高度な機能を利用できるようになります。
ヘルプ(E)
オンライン ヘルプを表示します。
この製品は、ライセンス供与されていないため、営利目的以外の使用の場合のみ承認されます。営利目的の使用の場合は、ライセンスを購入してください。今すぐ購入する。
ここに入力して検索
7:33
2019/07/26

仮想ディスクを開く

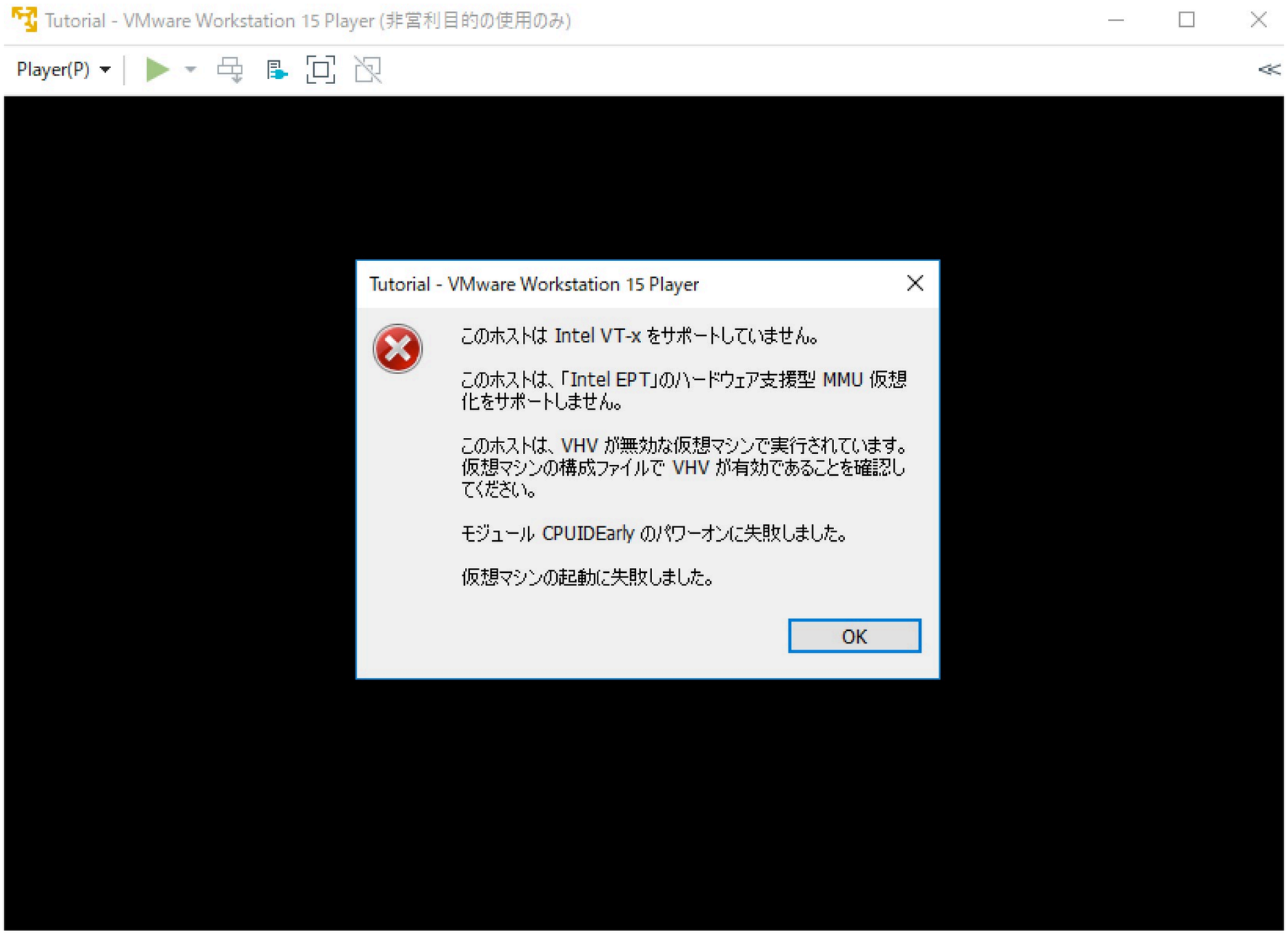


USBからゲットしたファイルをデスクトップにおいて,

Tutorial.vmwarevm/Tutorialを開く

※「コピーをしました」を選ぶこと

この画面になったら．．．

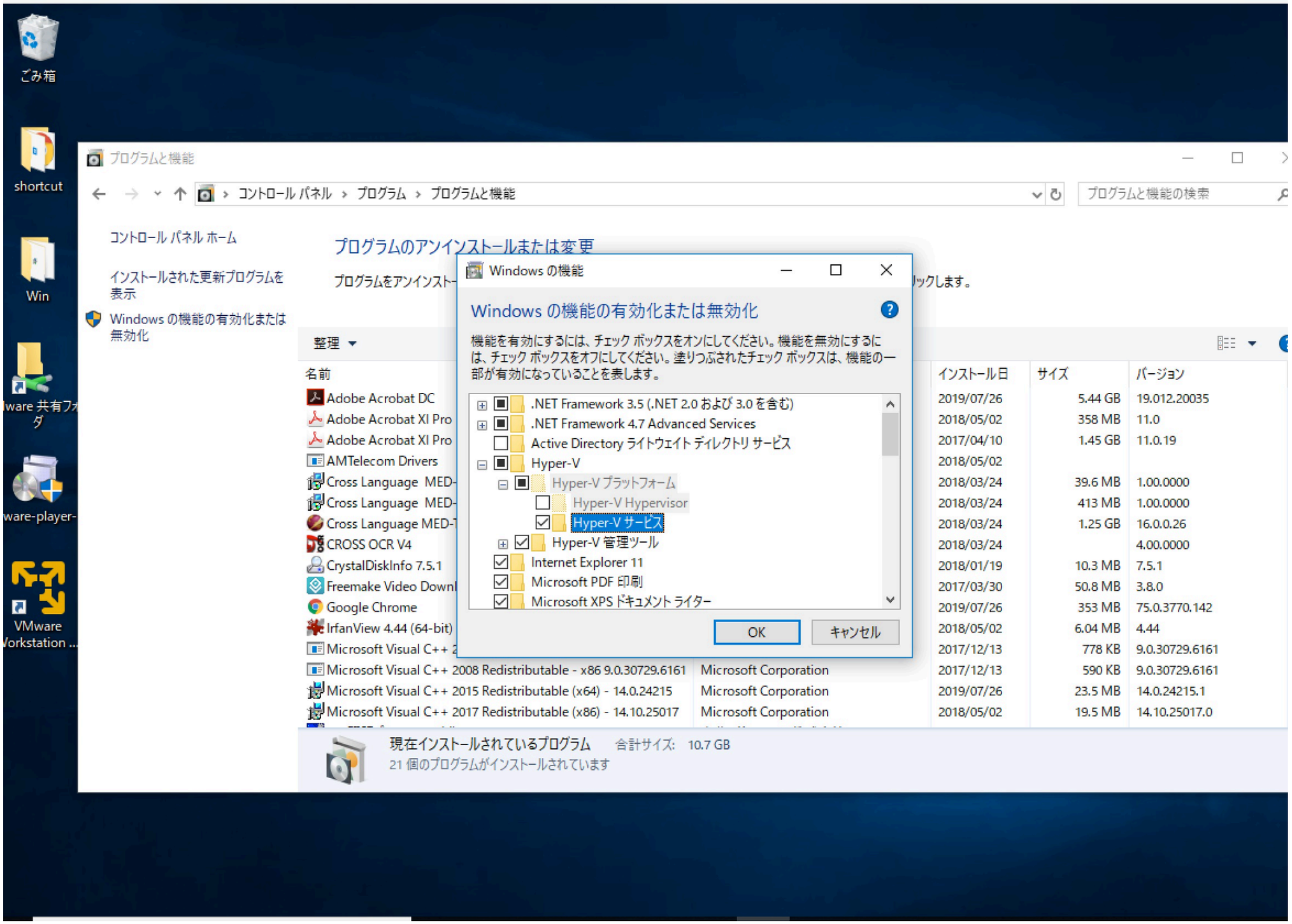


BIOSの設定を変更する必要がある．（一定のリスクあり）

Tips:
VmwareFusion の場合は，*.vmxファイルに次の一行を追加しておく．

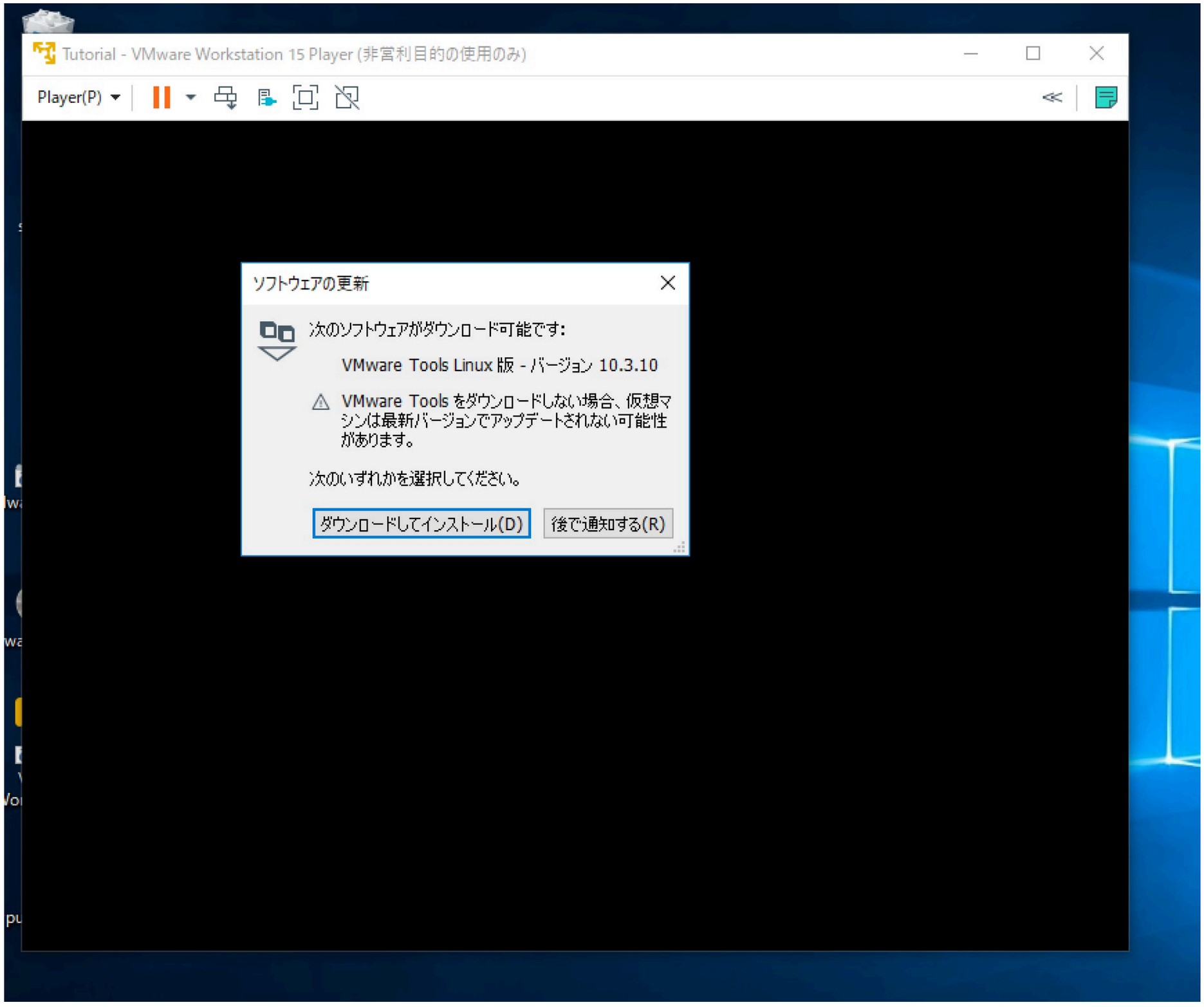
```
vhv.enable = TRUE
```

この画面になったら．．．



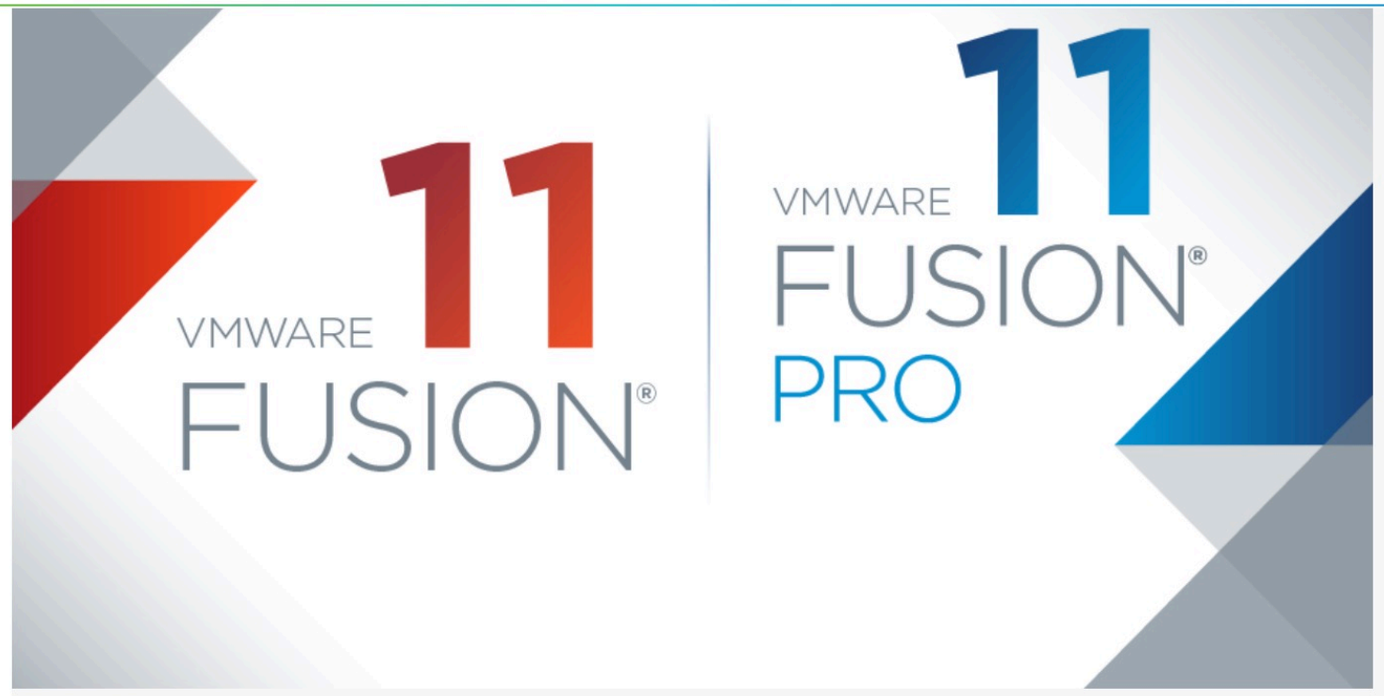
Tips:
コントロールパネル→プログラム→Windowsの機能の有効化からHyper-Vを選ぶ.

この画面になったら．．．



ダウンロードしてください．

仮想環境の導入
～MacOS編～



VMware Fusion は、Mac 上での Windows の実行と、あらゆるプラットフォームで動作するアプリケーションの開発とテストに最適な製品です。

Mac 上で Windows を実行する最善の方法がさらに最適化されました。Fusion 11 は、Touch Bar のカスタマイズが可能な改良されたユーザー インターフェイス、Windows アプリケーションや仮想マシンをすばやく起動できる新しいアプリケーション メニュー、Apple の Metal テクノロジーを利用する改良されたグラフィックス エンジンを備えています。

Fusion 11 Pro は、単なる Mac 上の Windows ではなく、IT プロフェッショナル、開発者、企業のための、強力でセキュアな開発およびテスト用サンドボックスです。Fusion Pro には、Fusion のすべての機能に加えて、仮想ネットワークの構成とシミュレーションの管理、仮想マシンのクローン作成、vSphere のリモート管理機能の向上といった追加機能が備わっています。

Fusion 11 の試用
今すぐダウンロード »

Fusion 11 Pro の試用
今すぐダウンロード »

仮想環境の導入

～Linux編～

ダウンロード VMware Workstation Pro

バージョンの選択:

15.0

以下のタブから関連するインストール パッケージを選択してください。ダウンロード ページへのログインを促すプロンプトが表示されます。プロファイルを作成していない場合は、ダウンロード プロセス中に作成を求められることがあります。

続きを読む

製品リソース

- ダウンロード履歴の確認
- 製品情報
- ドキュメント
- コミュニティ (英語)
- 無償評価版のダウンロード: Windows | Linux

製品のダウンロード | ドライバとツール | オープン ソース | カスタム ISO

	製品	リリース日	
⌄	**VMware Workstation Pro 15.1.0 for Windows**		
	VMware Workstation 15.1.0 Pro for Windows	2019-05-14	ダウンロードする
⌄	**VMware Workstation Pro 15.1.0 for Linux**		
	VMware Workstation 15.1.0 Pro for Linux	2019-05-14	ダウンロードする

～教材の場所～

秋山正和で検索＞集中講義＞資料のリンク
をクリック or Go to

http://www-mmc.es.hokudai.ac.jp/~masakazu/PleaseGetIt/

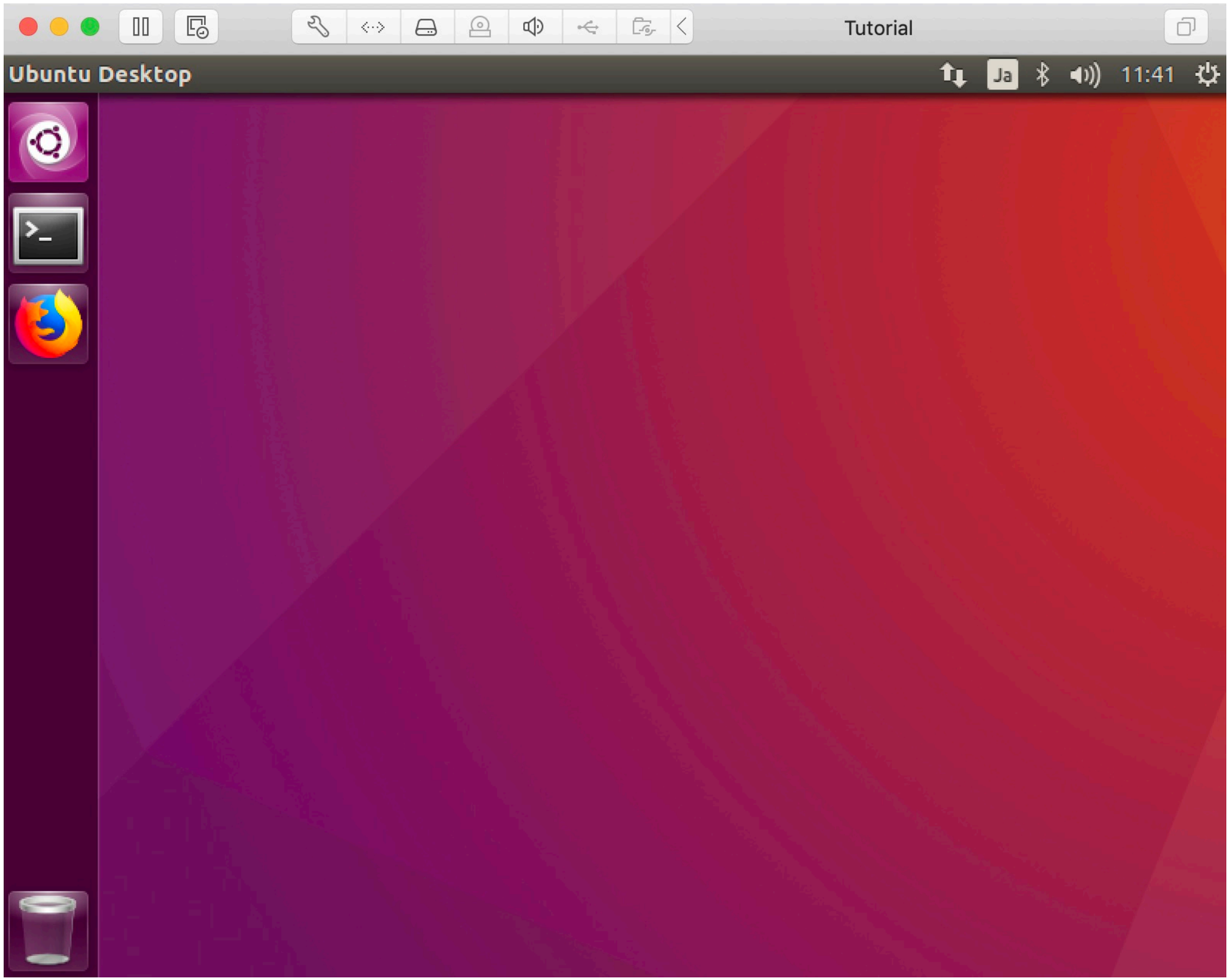
※仮想PCのFirefoxのブックマークにも上記URLがあります.

Index of /~masakazu/PleaseGetIt/Common

Name	Last modified	Size	Description
Parent Directory		-	
1_Test_HelloWorld.c	2019-07-29 13:05	78	
2_Test_Math.c	2019-07-29 13:13	284	
3_Test_destroy.c	2019-07-29 13:48	85	
4_Gnuplot_1.c	2019-07-29 14:27	315	
4_Gnuplot_2.c	2019-07-29 14:35	600	
4_Gnuplot_3.c	2019-07-29 14:37	224	
4_Gnuplot_4.c	2019-07-29 14:38	581	
4_Gnuplot_5.c	2019-07-29 14:39	678	
5_ODE_1.c	2016-10-19 16:03	539	
5_ODE_2.c	2019-07-29 17:31	506	
5_ODE_3.c	2019-07-29 17:33	504	
5_ODE_4.c	2019-07-29 17:36	718	
5_ODE_5.c	2015-11-04 17:03	386	
Diffuse1Dim.c	2019-07-28 17:47	1.1K	
Turing2D	2019-07-30 19:00	68K	
Turing2D.c	2019-07-29 12:51	10K	
a.out	2019-07-29 17:57	8.2K	
data.txt	2019-07-29 14:31	472	
ls	2019-07-29 13:48	8.2K	
main	2019-07-29 12:49	13K	
main.c	2019-07-29 12:48	5.7K	

こんな感じになれば正解！！

仮想環境を触ってみよう



User名：me

Passwd:tutorial

※firefox, web検索，教材のDL, D&Dを教える

～テスト編～

1_Test_HelloWorld.c
2_Test_Math.c
3_Test_destroy.c
をダウンロードしよう

1.Commonフォルダにターミナルを移動させる.

※cd, ls, pwd等を教える.

２.

$: cc 1_Test_HelloWorld.c

$: ./a.out

※./の意味を教える.

3.

$: cc 2_Test_Math.c

$: ./a.out

※-lmの必要性を教える.

4.

$: cc 2_Test_Math.c -lm -o masakazu

$: ./a.out

$: ./masakazu

※実行ファイルに関して教える

～権限とは～

$: ls -la

```
-rwxr-xr-x   1 masakazu  staff   8552  7 29 13:18 a.out*
```

$:　./a.out
で反応があるはず

$:　chmod 600 a.out

```
-rw-------   1 masakazu  staff   8552  7 29 13:18 a.out
```

となる．すると

$:　./a.out
permission denied: ./a.out

となる．何故か？

r : readable
w: writable
x: executable

rは４
wは２
xは１
の値を持つと約束する.

d	r	w	x	r	w	x	r	w	x
directory	読み	書き	実行	読み	書き	実行	読み	書き	実行
	自分	自分	自分	グループ	グループ	グループ	他人	他人	他人

例１
600
は
6=4*1+2*1+1*0
0=4*0+2*0+1*0
0=4*0+2*0+1*0

つまり自分は読めるし，書けるけど，実行ができない
グループは読めない，書けない，実行ができない
他人は読めない，書けない，実行ができない

$:　./a.out
permission denied: ./a.out

例2

777

は

7=4*1+2*1+1*1

7=4*1+2*1+1*1

7=4*1+2*1+1*1

なので，だれでも

何でも出来ちゃう.

$: cc 3_Test_destroy.c

$: ./a.out

```
Destroy your PC ....
```

$: mv a.out ls

$: ls

$: ./ls

※ウイルスの話,

適切な実行権限を意識する

例３

000

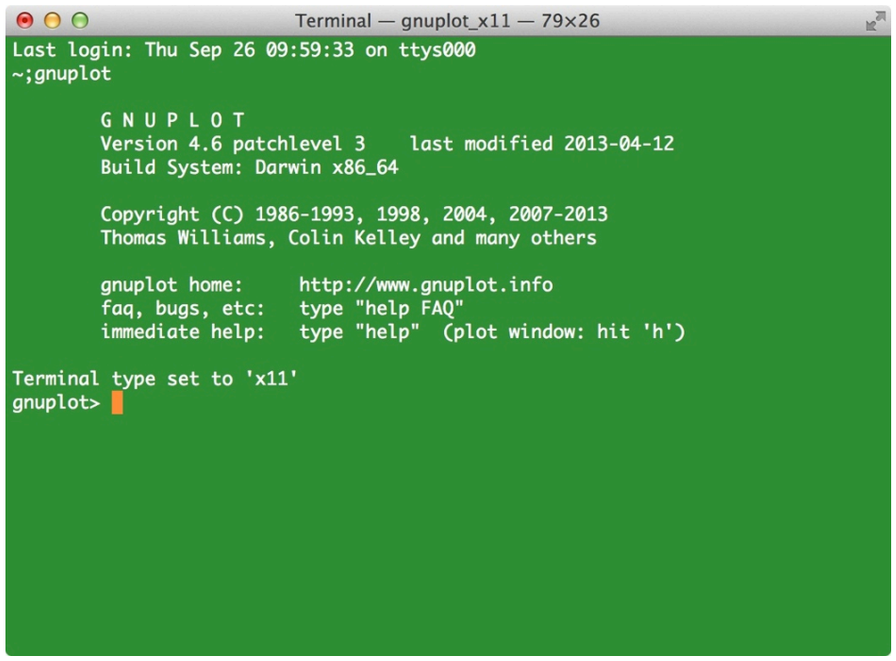
誰も何もできない.

～gnuplot編～

4_Gnuplot_1.c 4_Gnuplot_3.c 4_Gnuplot_5.c
4_Gnuplot_2.c 4_Gnuplot_4.c
をダウンロードしよう

gnuplotに慣れよう

$: gnuplot　と打つ.



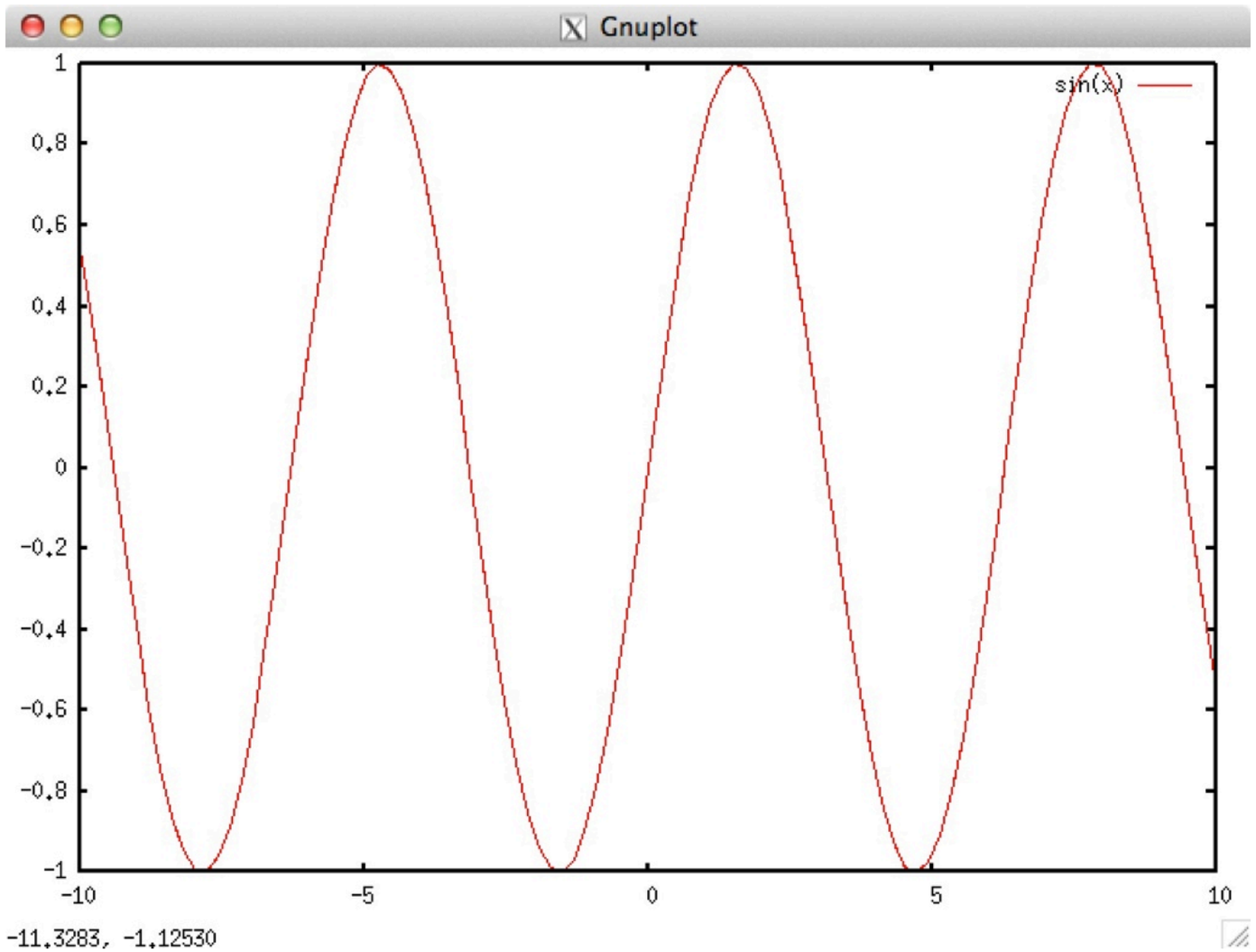
私の環境ではこう.　　違う環境ではこう.

細かいことは気にしないこと.

（緑の画面にすることは後で出来る.）

gnuplotに慣れよう

$: plot sin(x) と打つ.



こんな画面が出ればOK! ⇒　出ない人は挙手.

gnuplotに慣れよう

“ Exercise” が出たら手と頭を動かすこと（本演習での約束！！）

Exercise！. 以下のグラフを描いみてみよ.

$y = x^2$

$y = x^8$

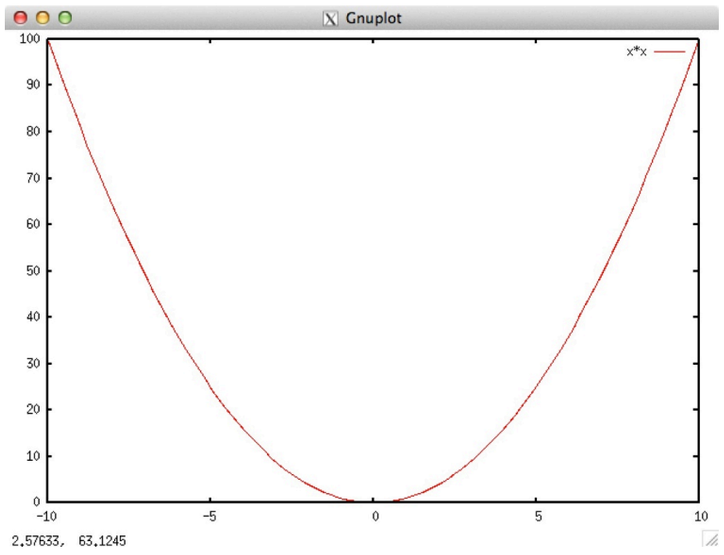
$y = e^x$

gnuplotに慣れよう

“ Exercise” が出たら手と頭を動かすこと（本演習での約束！！）

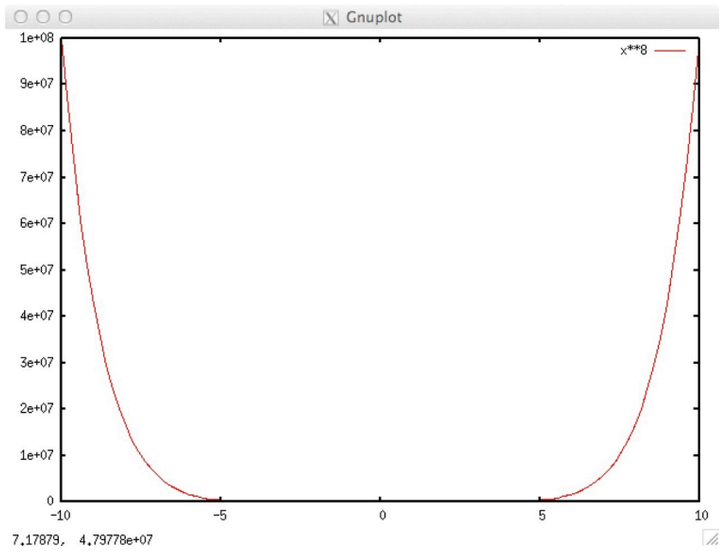
Exercise！. 以下のグラフを描いみてみよ.

$y = x^2$



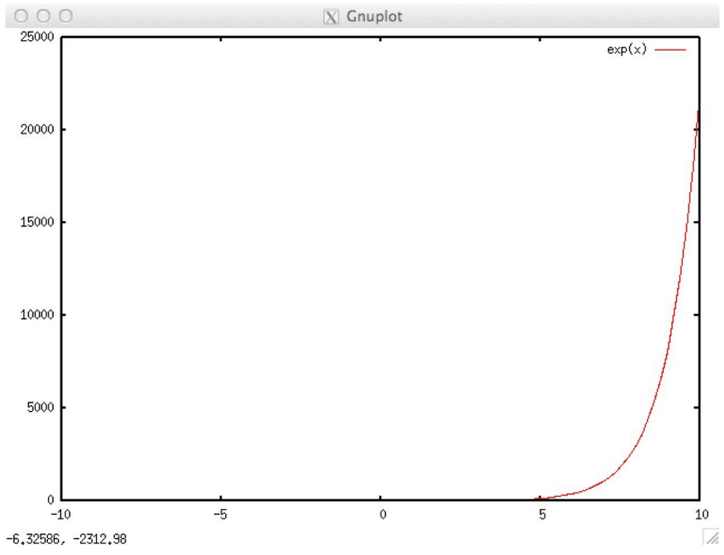
x*x

$y = x^8$



x**8

$y = e^x$

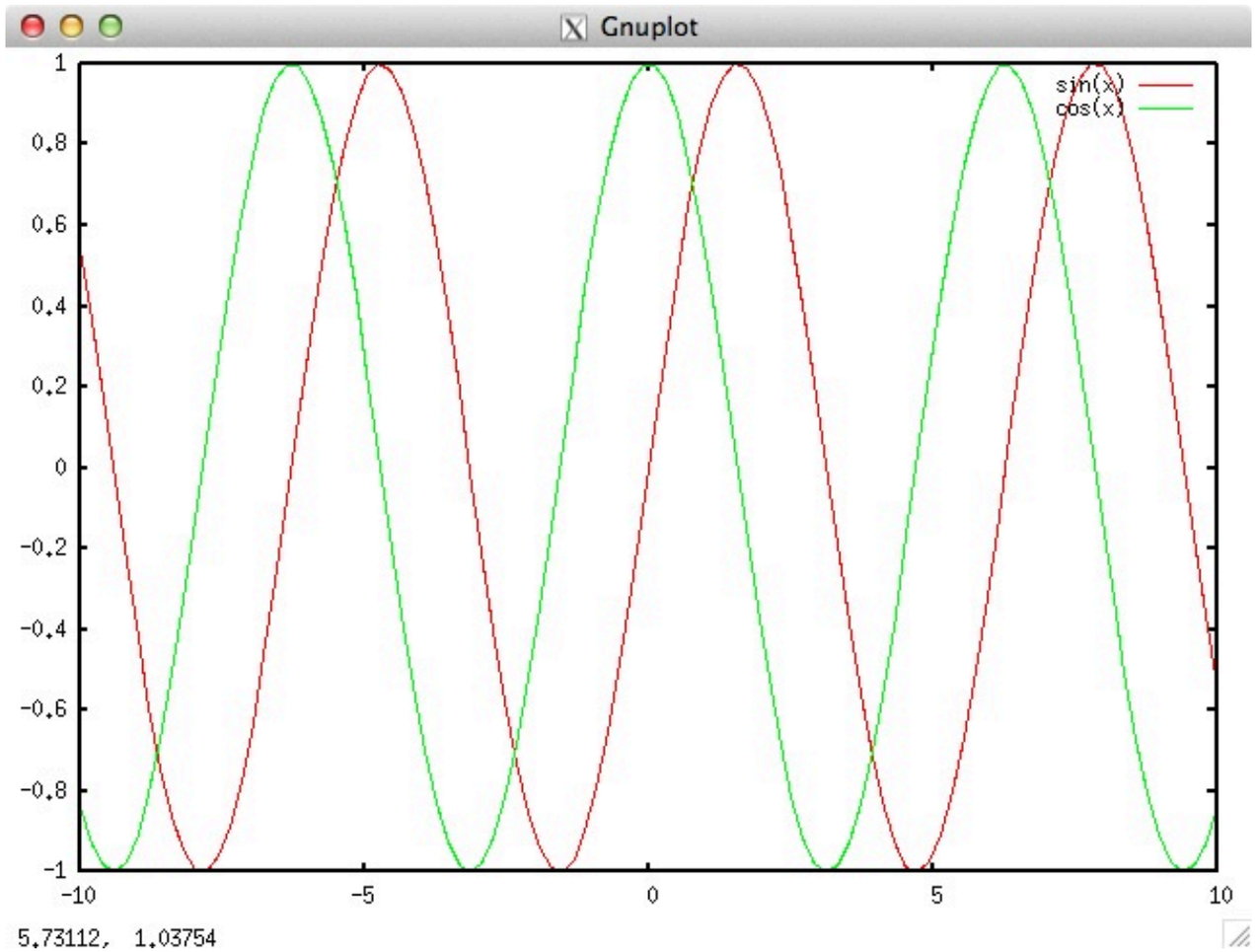


exp(x)

gnuplotに慣れよう

グラフの指定について

$: plot sin(x) , cos(x)　　カンマでつなぐと．．．

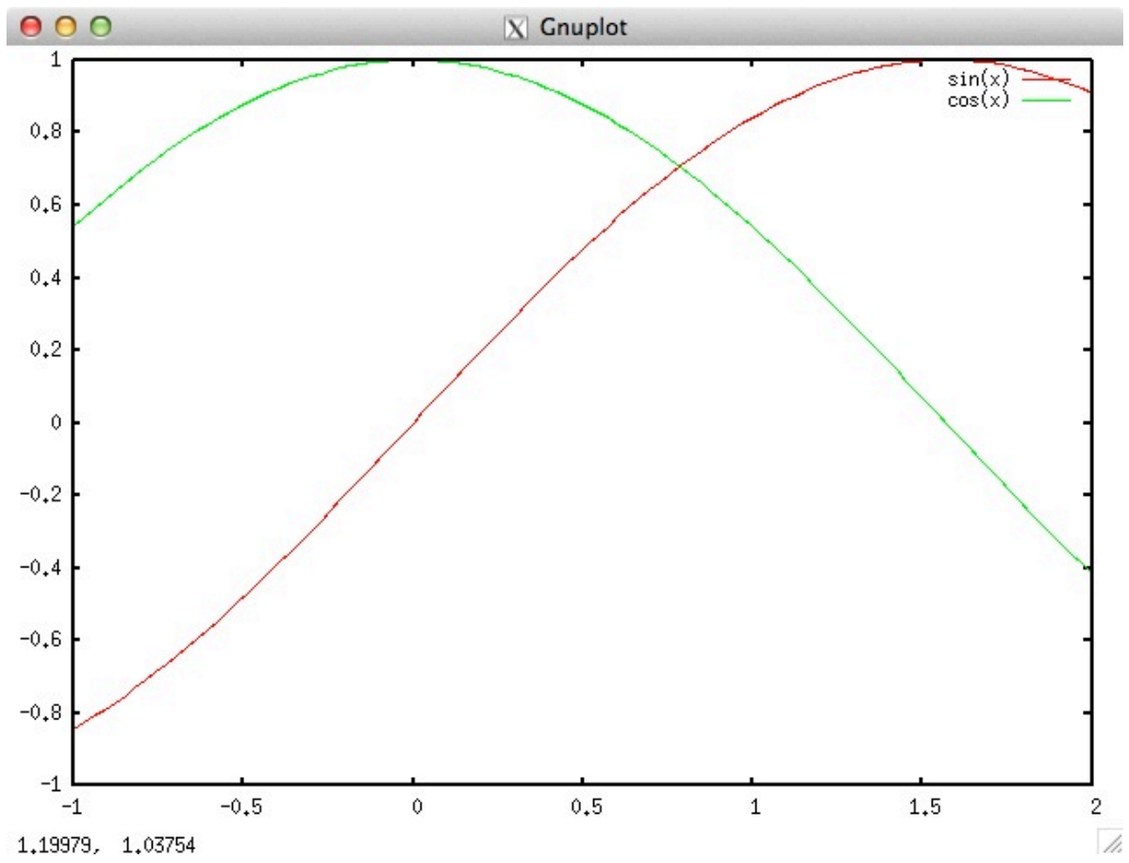


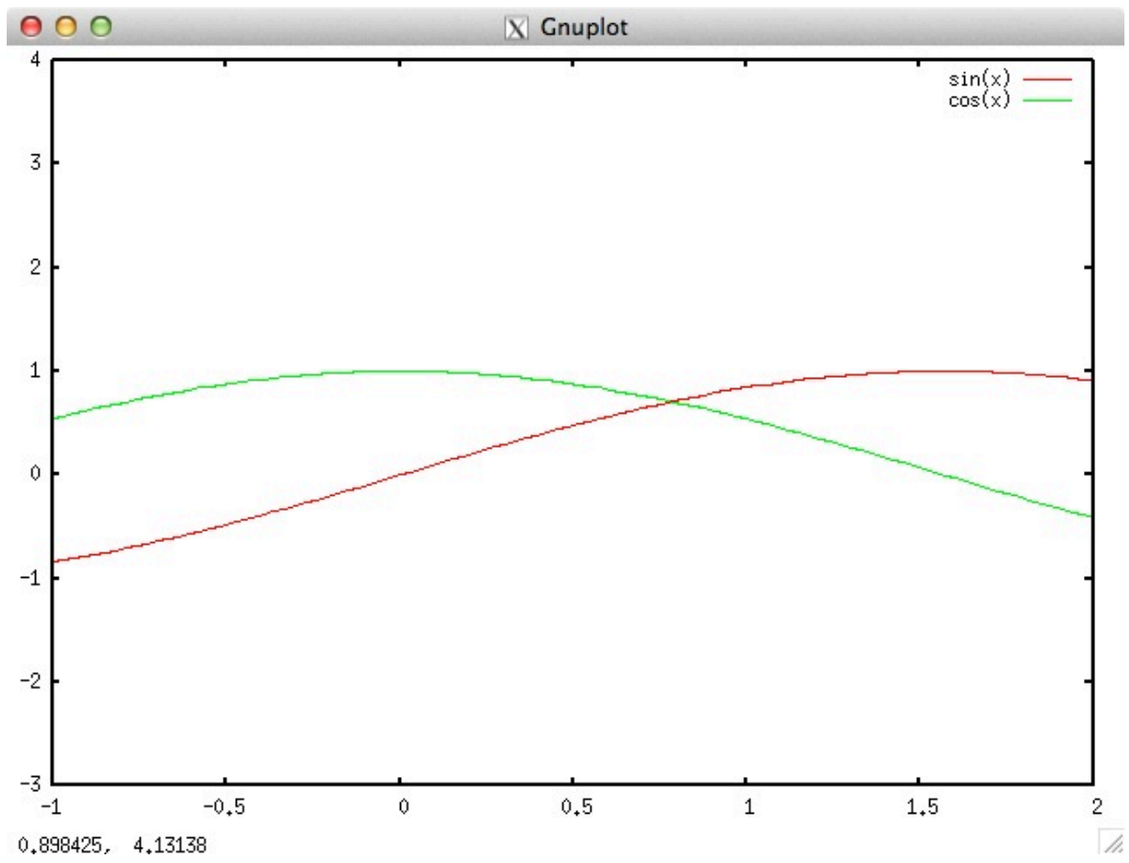
gnuplotに慣れよう

グラフの指定について

$: plot [-1:2] sin(x),cos(x)　　$x \in [-1, 2]$

$: plot [-1:2][-3:4] sin(x),cos(x)　　$x \in [-1, 2]\ y \in [-3, 4]$

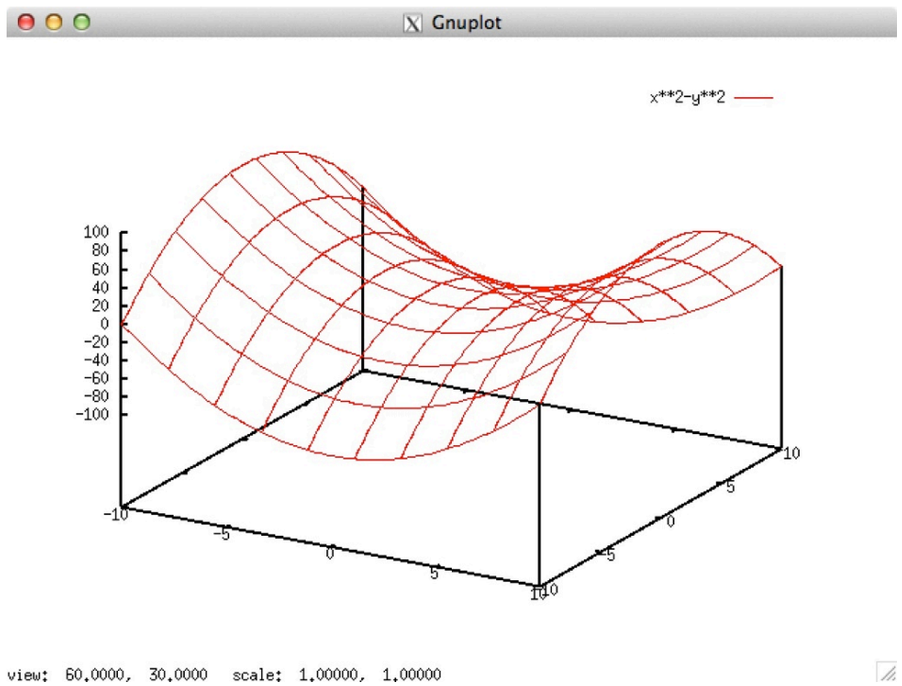


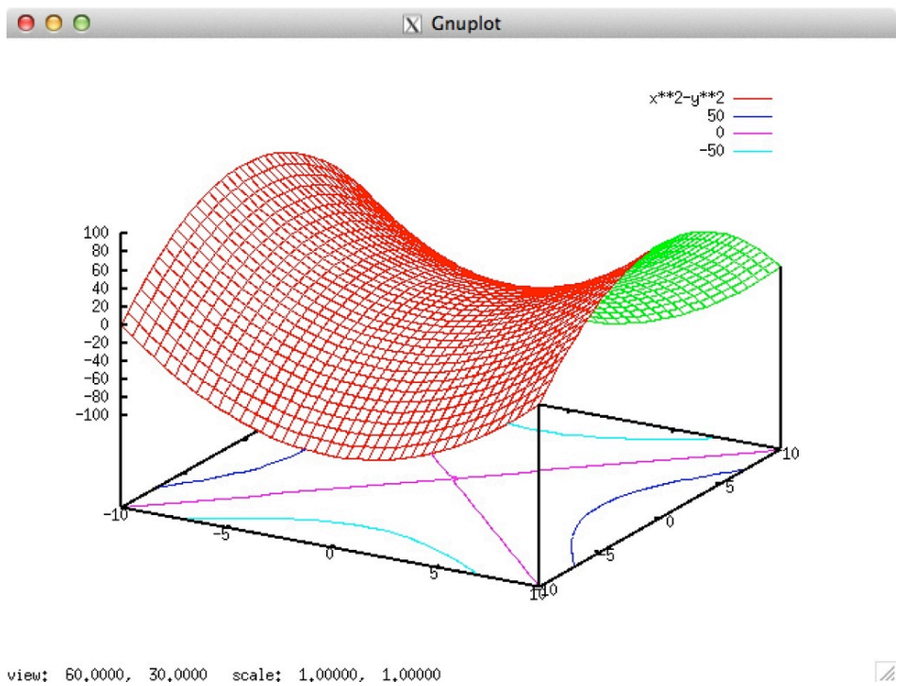


gnuplotに慣れよう

グラフの指定について

$: splot x**2-y**2





細かく指定することで，色々なグラフを描くことができる．

⇒　設定などはgoogleなどで検索をかける！

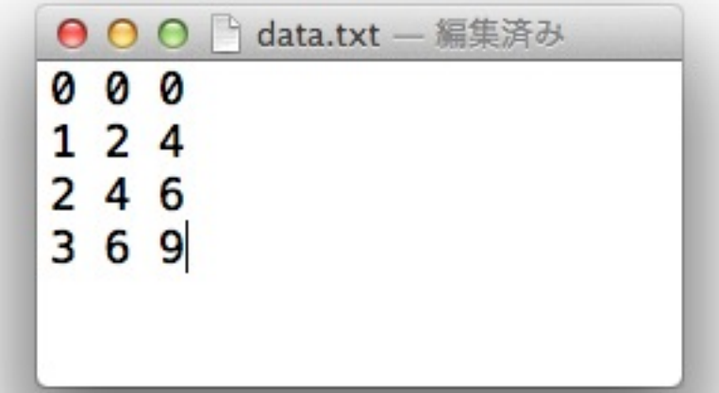
```
gnuplot> splot x**2-y**2
gnuplot> set hidden3d          ← 隠線消去を指定
gnuplot> set contour           ← 等高線描画を指定
gnuplot> set isosamples 40,40  ← メッシュを細かく
gnuplot> replot                ← 再描画
```

gnuplotに慣れよう

計算データの描画について

1．空のファイルを作成してファイル名をdata.txtにする.

2．下の例を参考に“x y1 y2”形式で手で入力する.

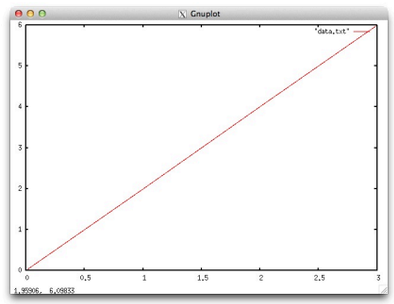


3．gnuplotで可視化する.　　$: plot "data.txt"

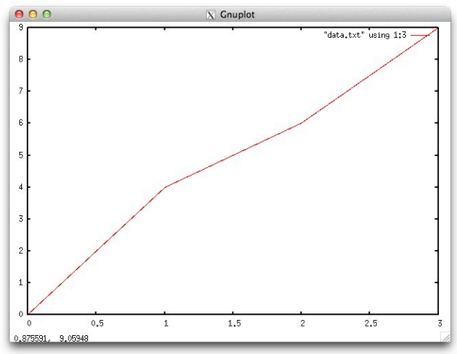
gnuplotに慣れよう

計算データの描画について

$: plot "data.txt" with lines　　⇒　折れ線でつなぐことができる.



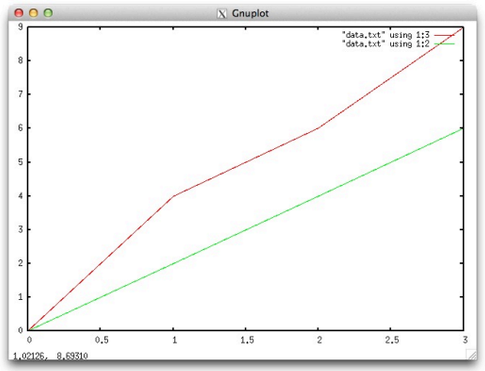
$: plot "data.txt" using 1:3 with lines



⇒　折れ線でつなぐことができる.

⇒　１列目と３列目のデータを
つかう.

$: plot "data.txt" using 1:3 with lines, "data.txt" using 1:2 with lines



⇒　２つのグラフを出すことができる.

gnuplotに慣れよう

計算データの描画について

Tips!!
長ったらしいコマンドは一文字に置き換えられる場合がある.

$: plot "data.txt" using 1:3 with lines, "data.txt" using 1:2 with lines

⇔

$: plot "data.txt" u 1:3 w l, "data.txt" u 1:2 w l

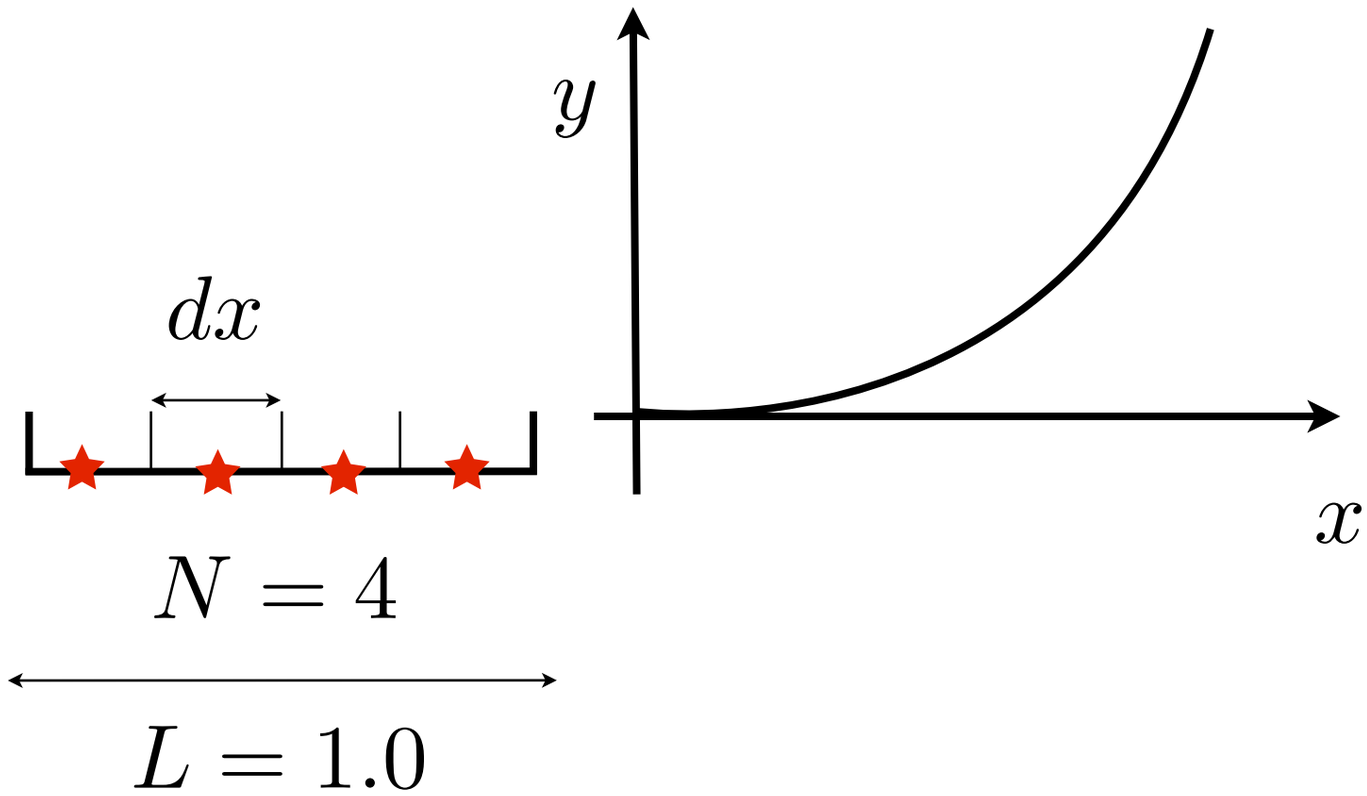
どちらでも好きな方で

gnuplotに慣れよう

計算データは手で作成するよりも計算機で計算したほうが楽.

4_Gnuplot_1.cをコンパイル＆実行

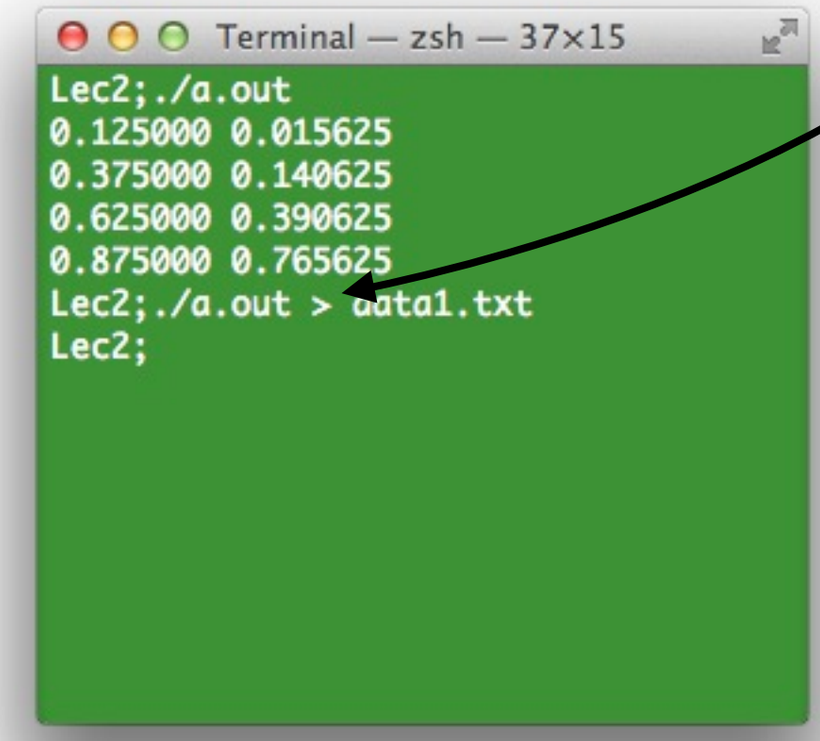
```c
#include <stdio.h>
int
main(void)
{
    double x,y;
    double L = 1.0;          //x length
    int    N = 4;            //Bunkatu
    double dx = L / N;       //kizami
    int    i;

    for(i = 0;i < N;i ++)
    {
        x = (i + 0.5) * dx;
        y = x * x;

        printf("%f %f\n",x,y);
    }

    return 0;
}
```

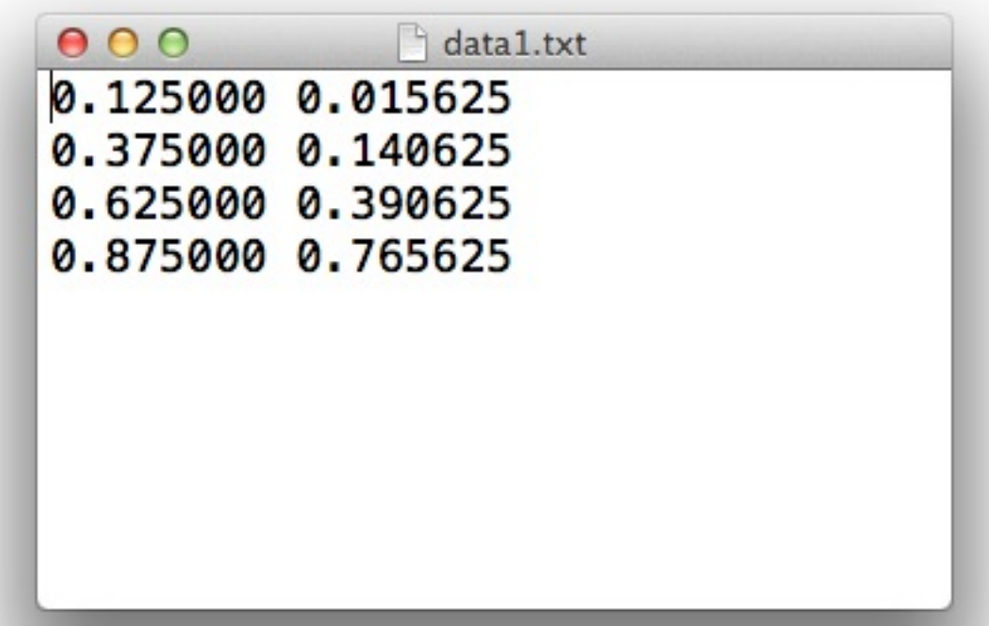


gnuplotに慣れよう

計算データは手で作成するよりも計算機で計算したほうが楽.

3．計算されたデータをリダイレクト“ > ”でテキストに保存.





4． gnuplotで可視化する.

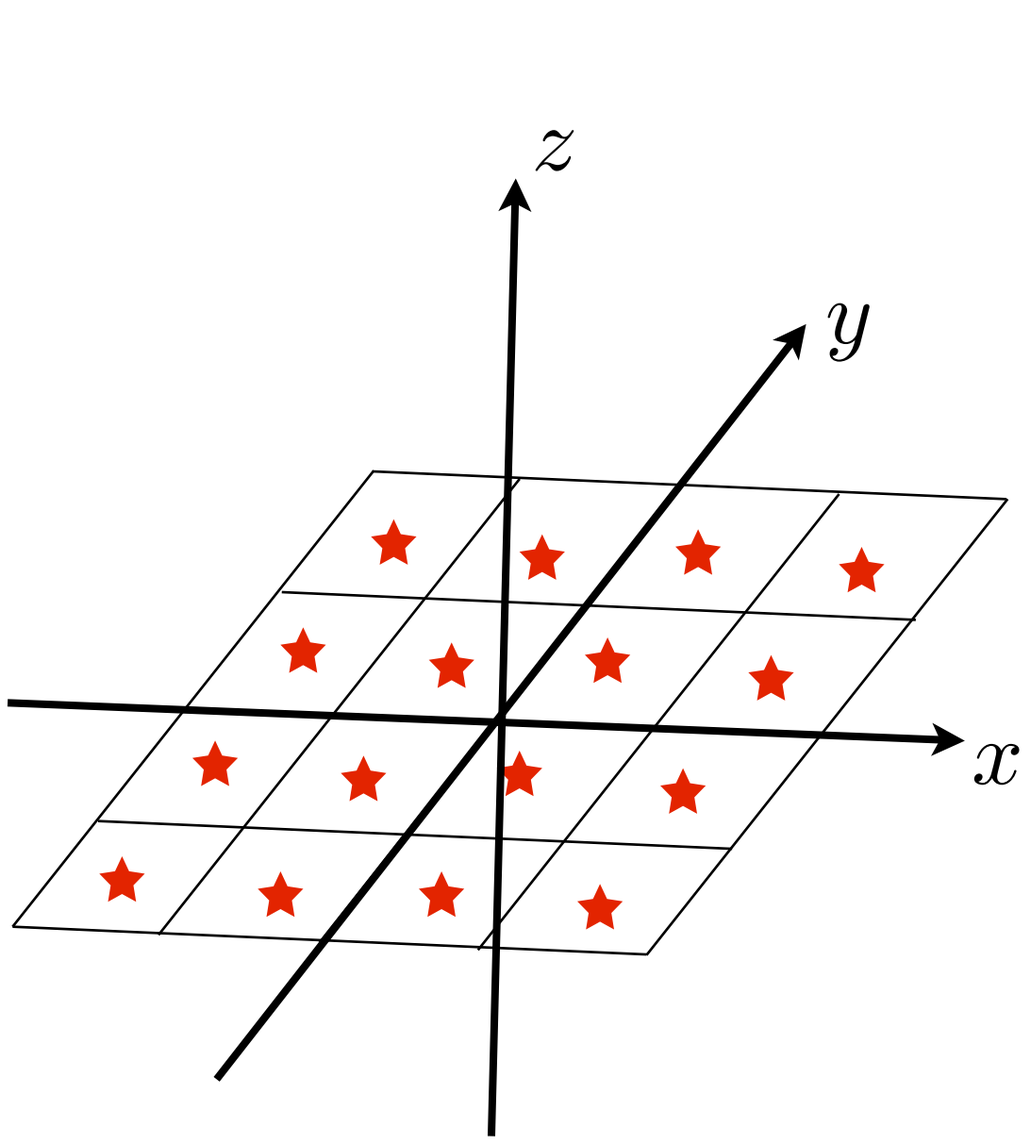
Exercise ！

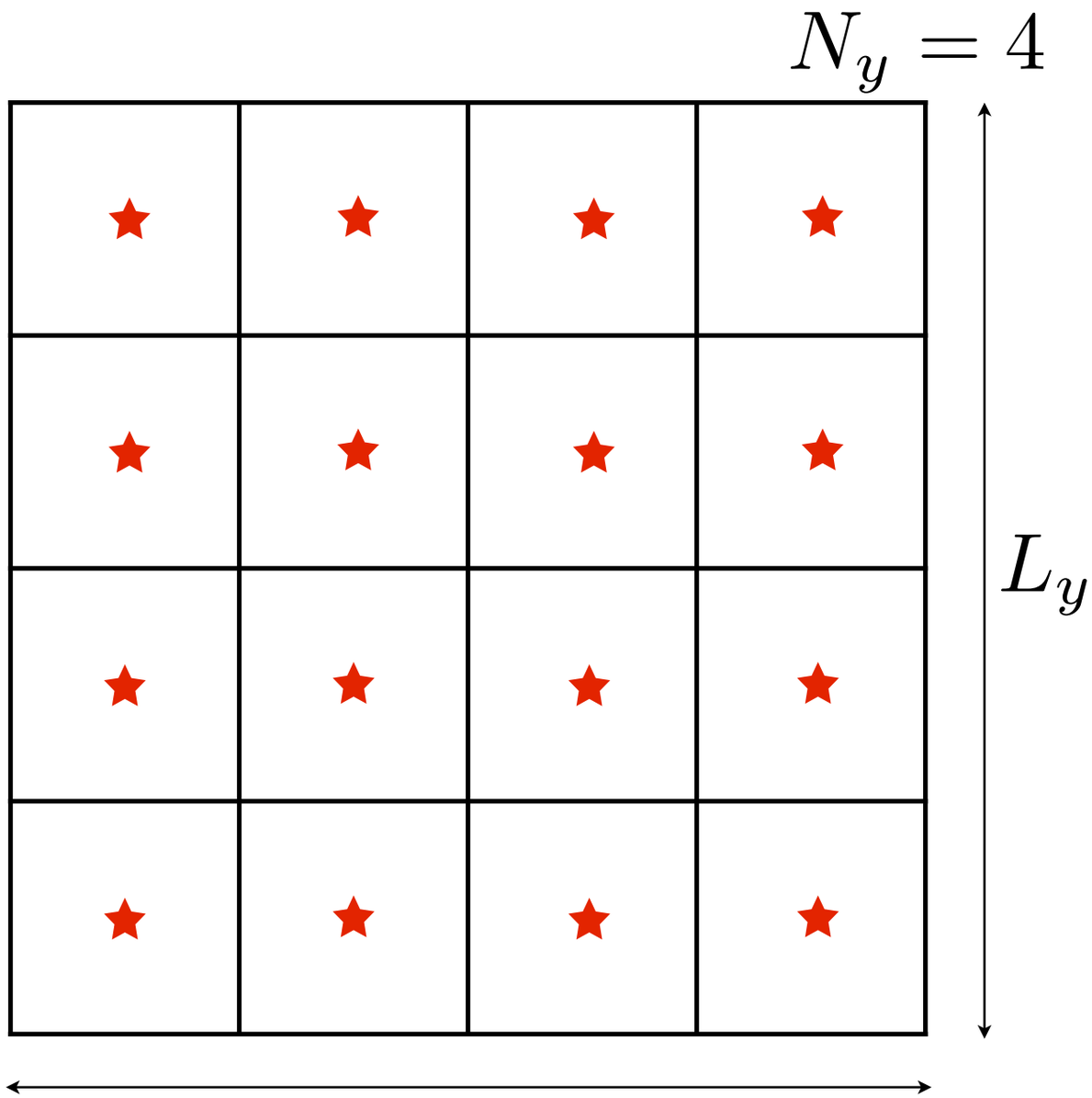
Nの値やLの値を変更し，可視化せよ.

gnuplotに慣れよう

２変数関数ならどのようにやるのか？

4_Gnuplot_2.cをコンパイル＆実行する.

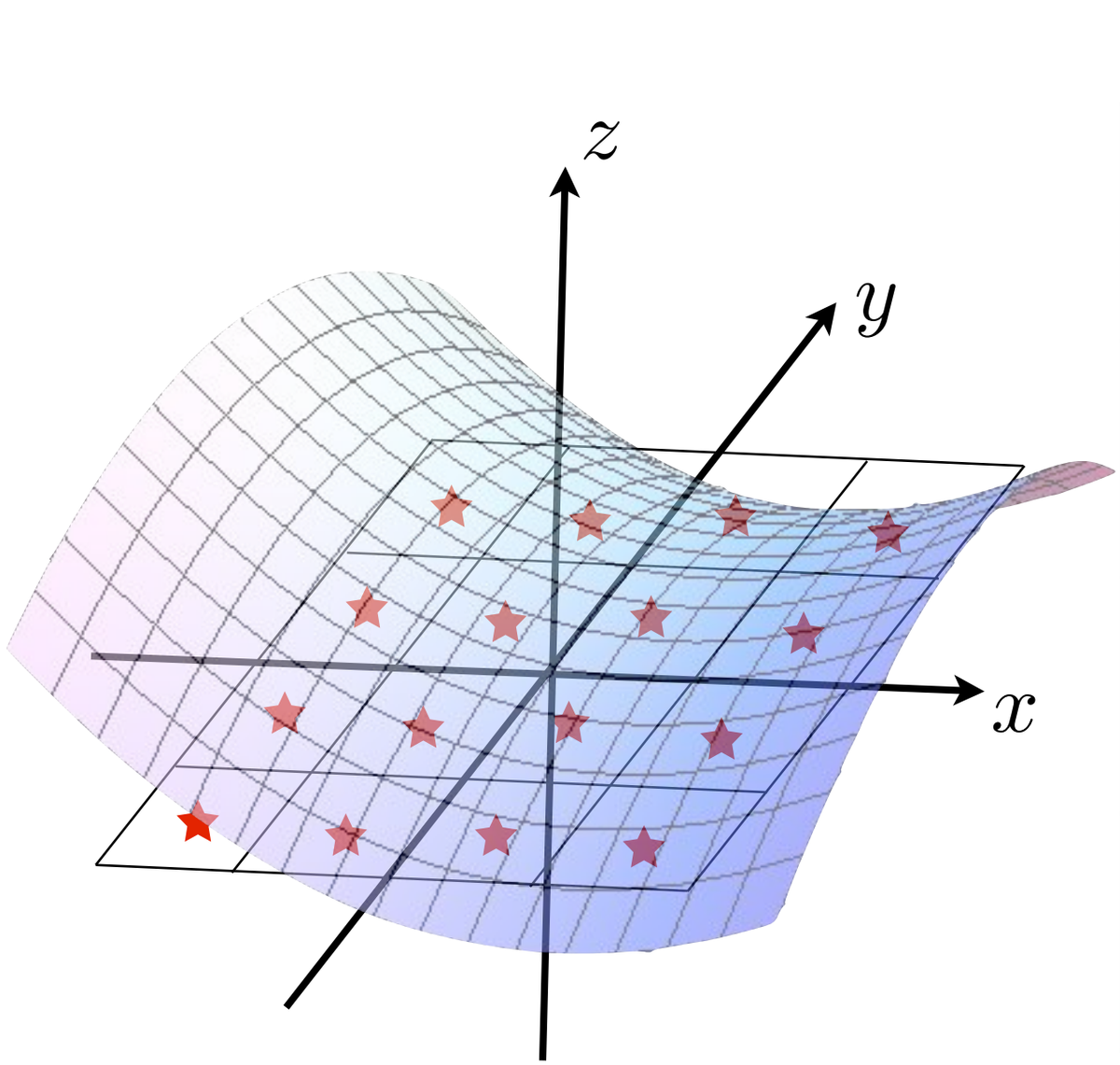


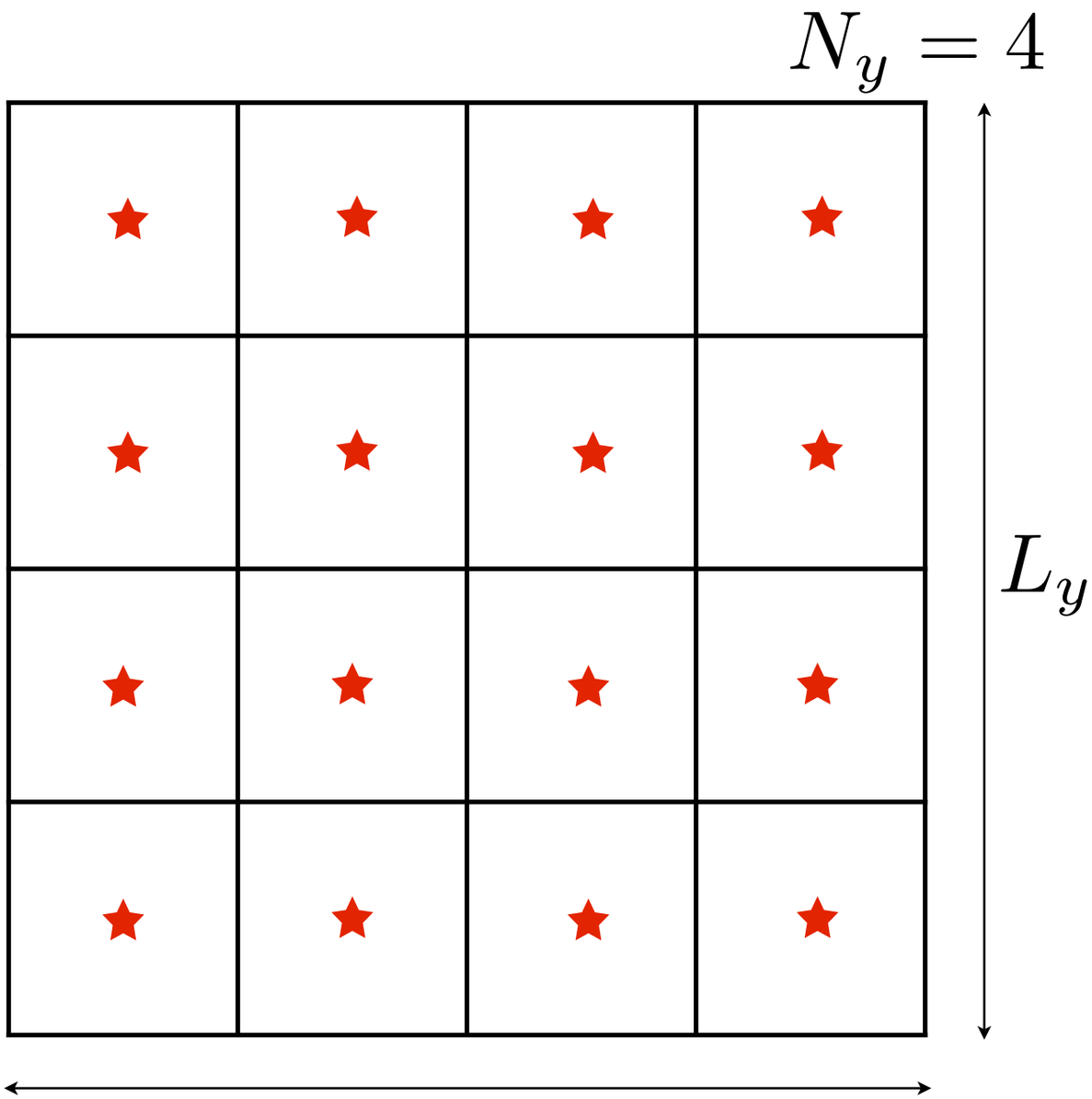


gnuplotに慣れよう

２変数関数ならどのようにやるのか？

4_Gnuplot_2.cをコンパイル＆実行する.

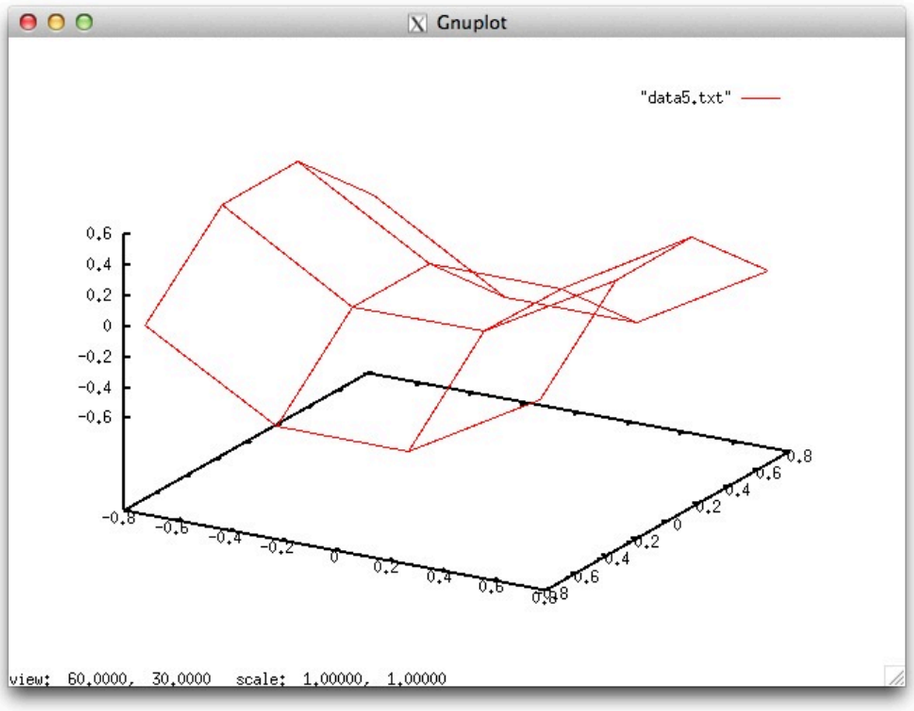




gnuplotに慣れよう

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

int
main(void)
{
    double x,y,z;
    double Lx = 2.0;        //x length
    double Ly = 2.0;        //y length
    int    Nx = 4;          //x Bunkatu
    int    Ny = 4;          //y Bunkatu
    double dx = Lx / Nx;    //x kizami
    double dy = Ly / Ny;    //y kizami

    for(int i = 0;i < Nx;i ++)
    {
        for(int j = 0;j < Ny;j ++)
        {
            x = (i + 0.5) * dx - Lx / 2;
            y = (j + 0.5) * dy - Ly / 2;
            z = x * x - y * y;
            printf("%f %f %f \n",x,y,z);
        }
        printf("\n");
    }
    return 0;
}
```

4_Gnuplot_2.c

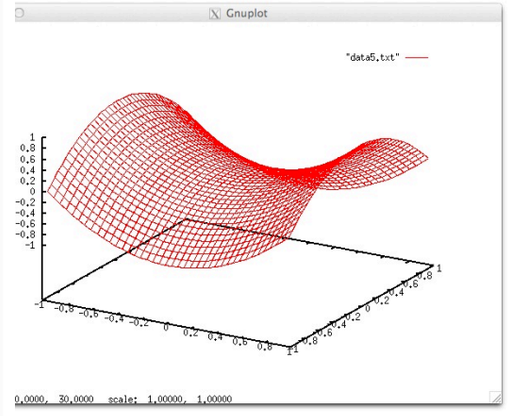


gnuplot 4x4

data5.txt

```
-0.750000 -0.750000 0.000000
-0.750000 -0.250000 0.500000
-0.750000 0.250000 0.500000
-0.750000 0.750000 0.000000

-0.250000 -0.750000 -0.500000
-0.250000 -0.250000 0.000000
-0.250000 0.250000 0.000000
-0.250000 0.750000 -0.500000

0.250000 -0.750000 -0.500000
0.250000 -0.250000 0.000000
0.250000 0.250000 0.000000
0.250000 0.750000 -0.500000

0.750000 -0.750000 0.000000
0.750000 -0.250000 0.500000
0.750000 0.250000 0.500000
0.750000 0.750000 0.000000
```

計算データ



gnuplot 40x40

gnuplotに慣れよう

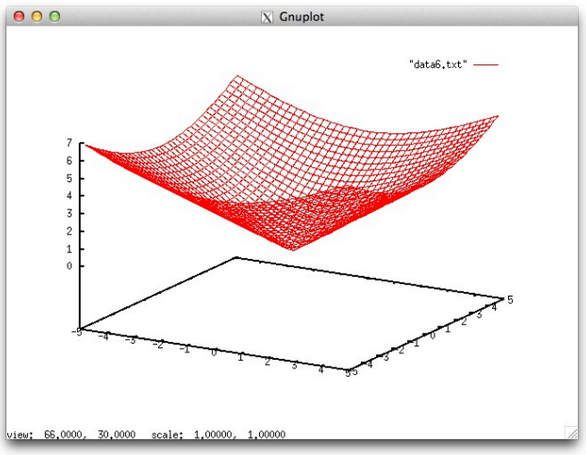
Exercise！. 以下のグラフを描いみてみよ.

$z = \sqrt{x^2 + y^2}$　　$z = xy$　　$z = \cos\sqrt{x^2 + y^2}$

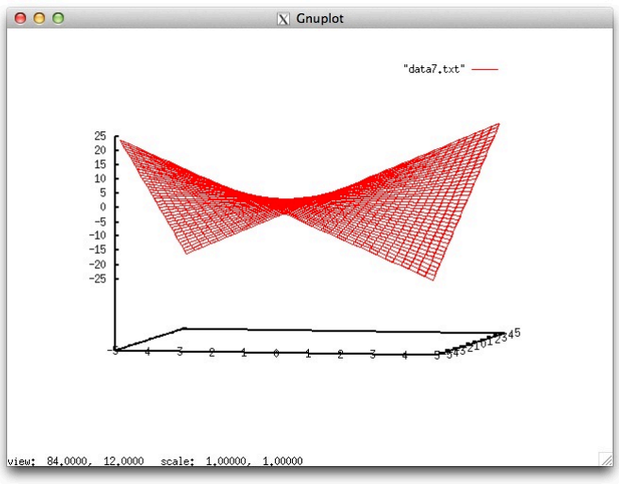
gnuplotに慣れよう

Exercise！. 以下のグラフを描いみてみよ.

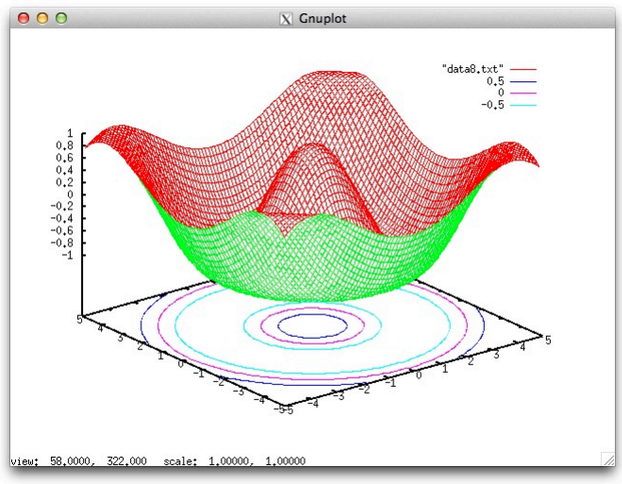
$z = \sqrt{x^2 + y^2}$



$z = xy$



$z = \cos\sqrt{x^2 + y^2}$



gnuplotに慣れよう

空間変数“x”時間変数“ t ”を含むデータの可視化.

$$z = f(x, y) \quad \Rightarrow \quad z = f(x, t)$$

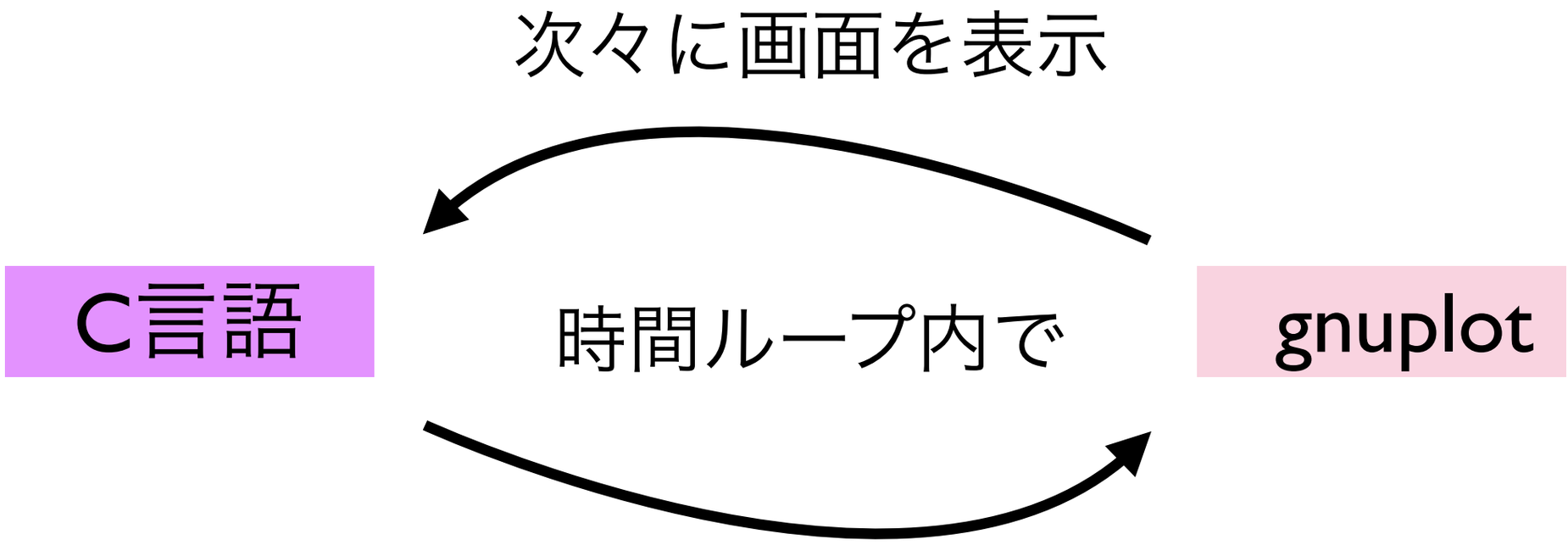
これだと芸がない. . . .

gnuplotに慣れよう

空間変数“x”時間変数“ t ”を含むデータの可視化.

$$z = f(x, y) \quad \Rightarrow \quad z = f(x, t)$$

これだと芸がない. . . .



gnuplotに慣れよう

C言語から直接制御

```c
#include <stdio.h>

int
main(void)
{
    FILE *gp;//For gnuplot

    gp = popen("gnuplot -persist","w");
    fprintf(gp, "set terminal x11\n");
    fprintf(gp, "plot sin(x)\n");
    fflush(gp);
    pclose(gp);

    return 0;
}
```

この構文がミソ！

4_Gnuplot_3.c

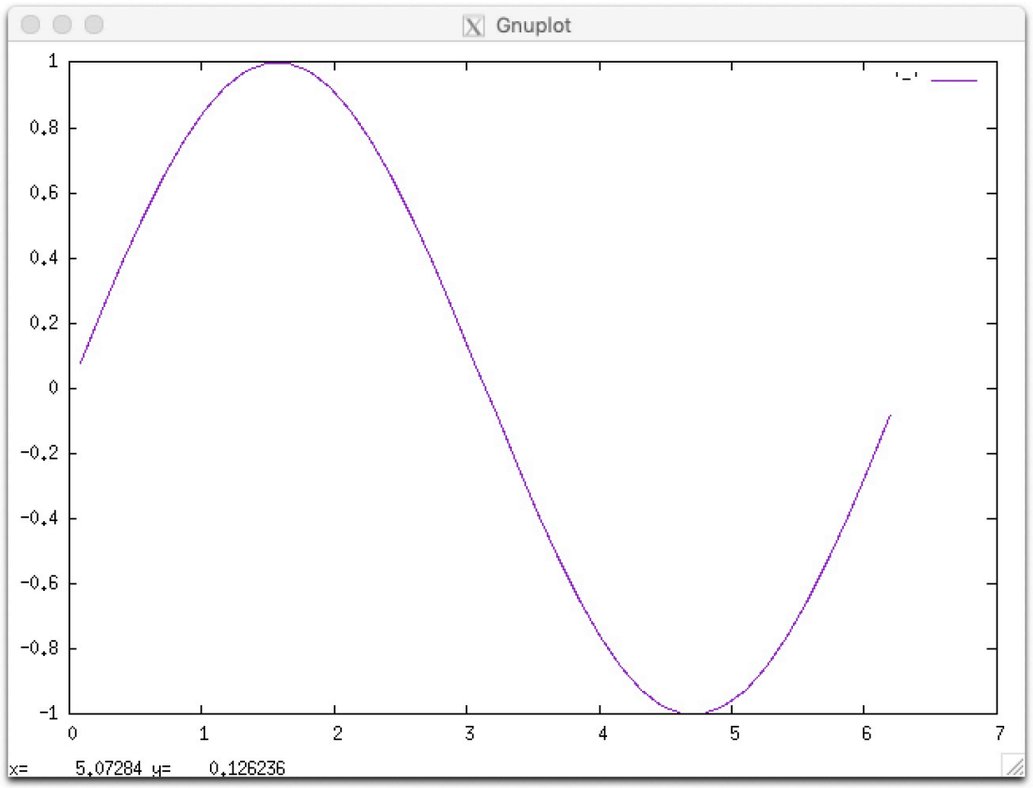
gnuplotに慣れよう

C言語から直接制御

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

int
main(void)
{
    int i;
    int N = 40;
    double x,y;
    double L = 2 * 3.14159265358979;
    double dx = L / N;

    FILE *gp;//For gnuplot
    gp = popen("gnuplot -persist","w");

    fprintf(gp, "plot '-' with lines \n");//これがミソ

    for(i = 0;i < N;i ++)
    {
        x = (i + 0.5) * dx;
        y = sin(x);
        fprintf(gp,"%f %f\n", x, y);// データの書き込み
    }
    fprintf(gp,"e\n"); // データの書き込み終了

    fflush(gp);
    pclose(gp);

    return 0;
}
```

データを次々と計算し，

それをgnuplotに食わせる．



4_Gnuplot_4.c

gnuplotに慣れよう

C言語から直接制御

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

int
main(void)
{
    int i;
    int i_time;
    int N = 40;
    double x,y;
    double L = 4 * 3.14159265358979;
    double dx = L / N;
    double dt = 0.05;

    FILE *gp;//For gnuplot
    gp = popen("gnuplot -persist","w");
     fprintf(gp, "set terminal x11\n");

    for(i_time = 0;;i_time ++)
    {
        fprintf(gp, "set yrange [-1.2:1.2]\n");
        fprintf(gp, "plot '-' with lines \n");
        for(i = 0;i < N;i ++)
        {
            x = (i + 0.5) * dx;
            y = sin(x - i_time * dt);
            fprintf(gp,"%f %f\n", x, y);
        }
        fprintf(gp,"e\n");
          fflush(gp);
    }
    pclose(gp);

    return 0;
}
```

4_Gnuplot_5.c

“データを次々と計算し,
それをgnuplotに食わせる. ”

を時間ループ内で繰り返す

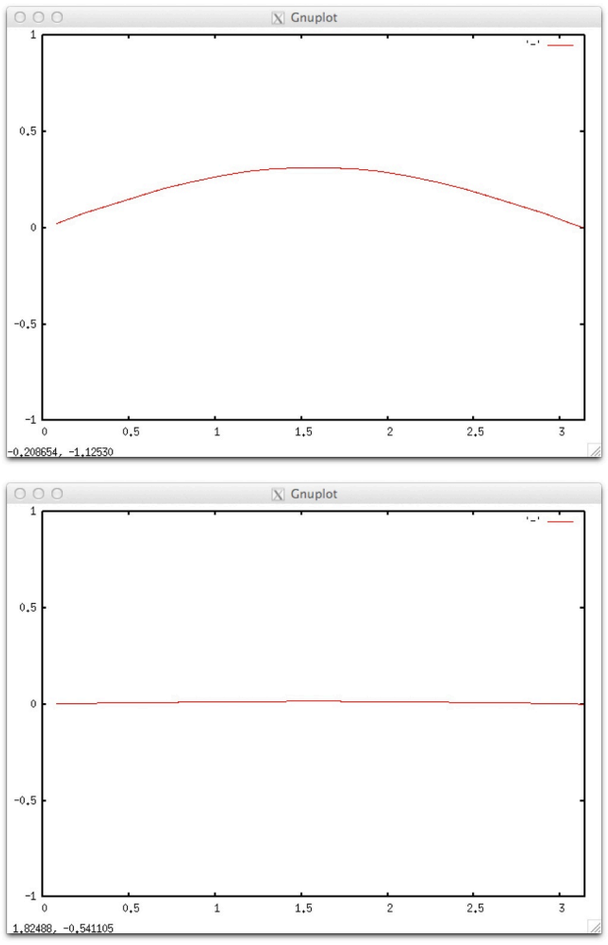
Tips!
無限ループを止めるには

“Ctrl + C”

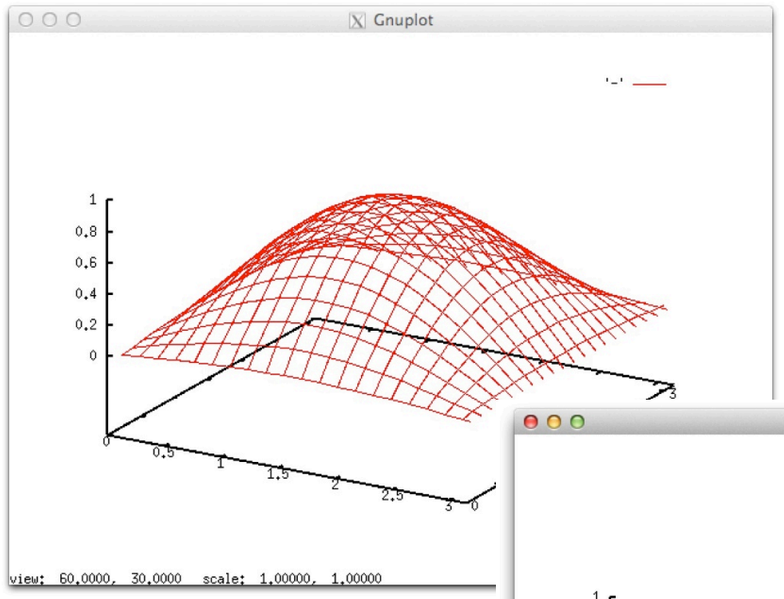
gnuplotに慣れよう

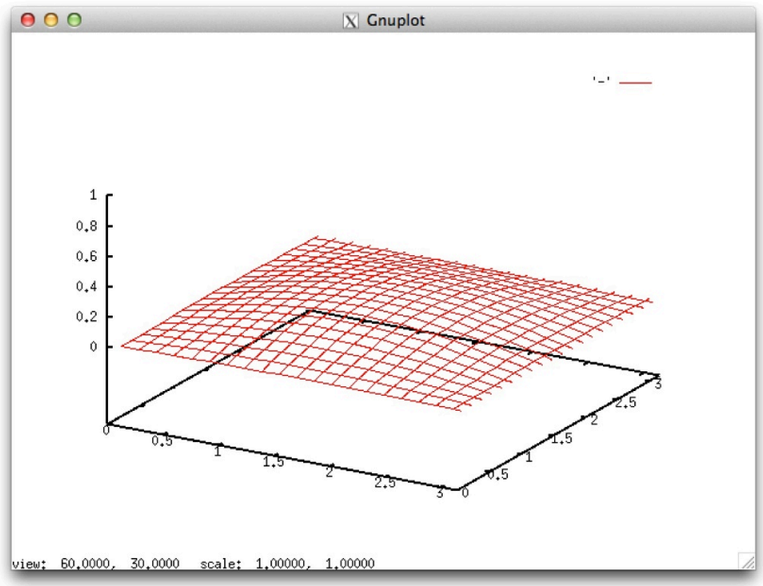
Exercise！. 以下のグラフを描いてみよ.

$$z = e^{-t} \sin x$$



$$z = e^{-t} \sin x \sin y$$





http://www-mmc.es.hokudai.ac.jp/~masakazu/PleaseGetIt/

Common.zipをDLする.

反応拡散方程式
数値解法入門

問題をさらに簡単な問題に分けて考える

偏微分方程式

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u + f(u)$$

問題をさらに簡単な問題に分けて考える

偏微分方程式

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u + f(u)$$

$u = u(t, x)$ 空間１次元

$u = u(t, x, y)$ 空間２次元

$u = u(t, x, y, z)$ 空間３次元

問題をさらに簡単な問題に分けて考える

偏微分方程式

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u + f(u)$$

$u = u(t, x)$ 空間１次元

$u = u(t, x, y)$ 空間２次元

$u = u(t, x, y, z)$ 空間３次元

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u$$

$u = u(t, x)$

偏微分方程式

問題をさらに簡単な問題に分けて考える

偏微分方程式

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u + f(u)$$

$u = u(t, x)$ 空間１次元

$u = u(t, x, y)$ 空間２次元

$u = u(t, x, y, z)$ 空間３次元

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u$$

$u = u(t, x)$

偏微分方程式

$$\frac{\partial u}{\partial t} = f(u) \qquad u = u(t, x)$$

偏微分方程式なんだけど空間に依存しないなら．．．

$$\frac{du}{dt} = f(u) \quad u = u(t)$$

常微分方程式になる.

反応拡散方程式＝

常微分方程式＋拡散方程式

※大きなくくりでは偏微分方程式

常微分方程式の
数値解法入門

常微分方程式の数値解法

Euler法とは

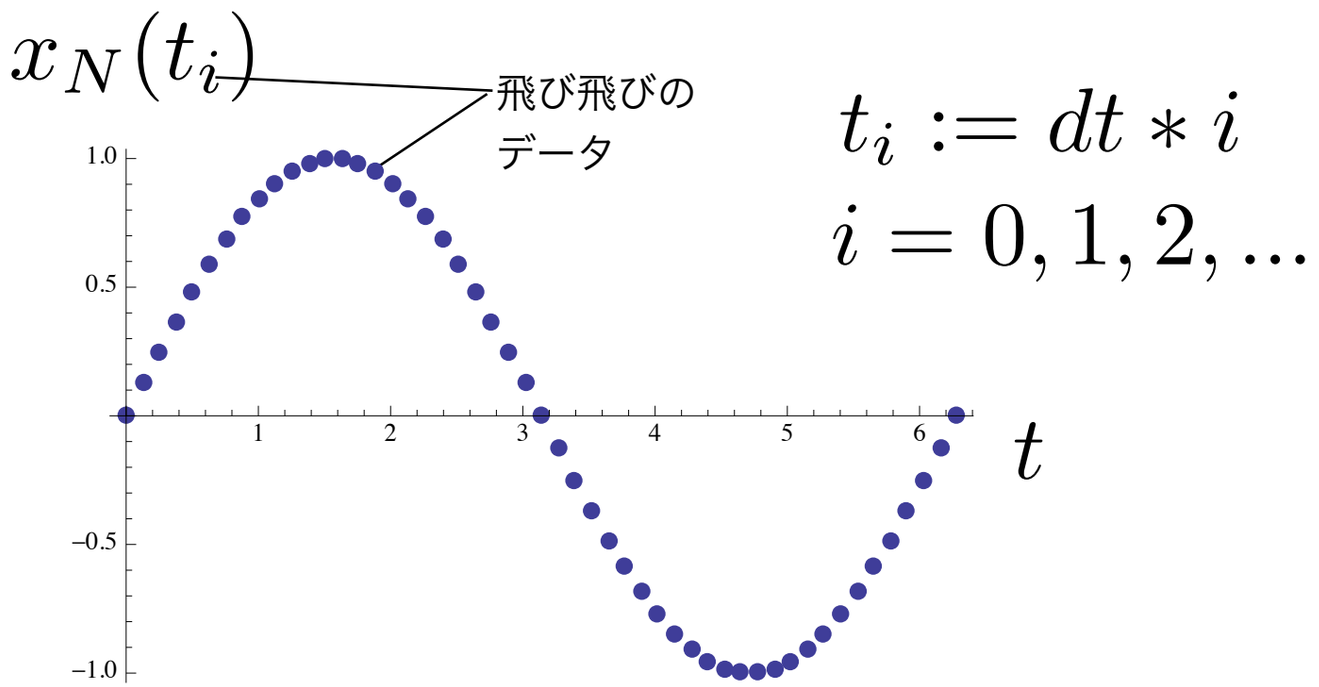
$$\dot{x}(t) = \cos(t)$$

$$x(0) = 0$$

問題

$$x_s(t) = \sin(t)$$

厳密解



$$t_i := dt * i$$

$$i = 0, 1, 2, ...$$

数値解

$x_N(t_i)$　が満たすべき条件を決定する.

常微分方程式の数値解法

Euler法とは

$x_N(t_i)$　が満たすべき条件を決定する.

この変形が
Euler法の心

$$\dot{x}(t) = \cos(t)$$

$$x(0) = 0$$

問題

常微分方程式の数値解法

Euler法とは

$x_N(t_i)$　が満たすべき条件を決定する.

この変形が
Euler法の心

$$\dot{x}(t) = \cos(t)$$

$$x(0) = 0$$

問題

$$\frac{x_N(t_{i+1}) - x_N(t_i)}{dt} \approx \cos(t_i)$$

$$x_N(t_0) = x(0) = 0$$

常微分方程式の数値解法

Euler法とは

$x_N(t_i)$　が満たすべき条件を決定する.

この変形が
Euler法の心

$$\dot{x}(t) = \cos(t)$$

$$x(0) = 0$$

問題

$$\frac{x_N(t_{i+1}) - x_N(t_i)}{dt} \approx \cos(t_i)$$

$$x_N(t_0) = x(0) = 0$$

$$f'(t) := \lim_{\delta t \to 0} \frac{f(t+\delta t) - f(t)}{\delta t}$$

常微分方程式の数値解法

Euler法とは

$x_N(t_i)$　が満たすべき条件を決定する.

この変形が
Euler法の心

$$\dot{x}(t) = \cos(t)$$

$$x(0) = 0$$

問題

$$\frac{x_N(t_{i+1}) - x_N(t_i)}{dt} \approx \cos(t_i)$$

$$x_N(t_0) = x(0) = 0$$

$$f'(t) := \lim_{\delta t \to 0} \frac{f(t+\delta t) - f(t)}{\delta t}$$

$$x_N(t_{i+1}) \approx x_N(t_i) + dt\cos(t_i)$$

現時点の　$x_N(t_i)$　がわかればちょっと先の $x_N(t_{i+1})$も大体わかる.

常微分方程式の数値解法

Euler法とは

$$x_N(t_{i+1}) \approx x_N(t_i) + dt\cos(t_i)$$

$$x_N(t_0) = 0$$

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

int
main(void)
{
    int i;
    double dt = 0.01;
    double xN,xN_new;

    FILE *gp;
    gp = popen("gnuplot -persist","w");
    fprintf(gp, "set terminal x11\n");
    fprintf(gp, "set yrange[-1.1:1.1] \n");
    fprintf(gp, "plot '-' with lines \n");

    xN = 0;
    for (i = 0; i < 1000; i++)
    {
        fprintf(gp,"%f %f\n", i * dt, xN);// データの書き込み
        xN_new = xN + dt * cos(i * dt);//計算
        xN = xN_new;
    }
    fprintf(gp,"e\n");

    fflush(gp);
    pclose(gp);

    return 0;
}
```

小さな数を設定.

gnuplotのおまじない.

初期値設定.

まず表示

漸化式

次のループのため更新

gnuplotのおまじない.

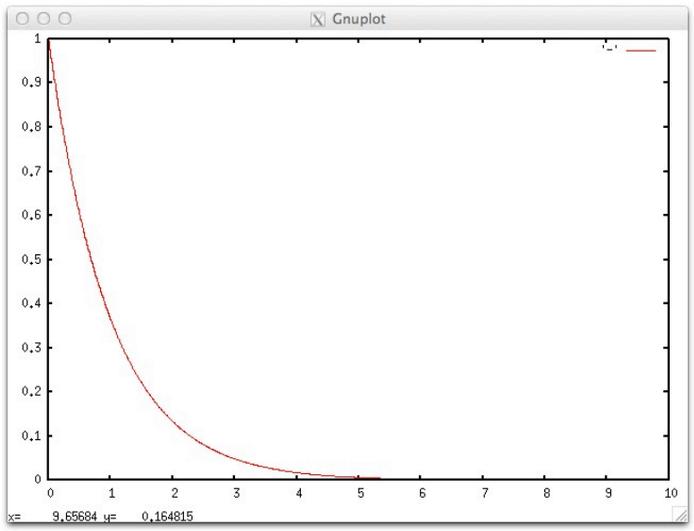
コンパイル＆実行しよう

5_ODE_1.c

常微分方程式の数値解法

Exercise！計算＆可視化をせよ．また厳密解と比較せよ．

$$\dot{x}(t) = -x(t)$$
$$x(0) = 1$$
$$(x_s(t) = e^{-t})$$



5_ODE_2.c

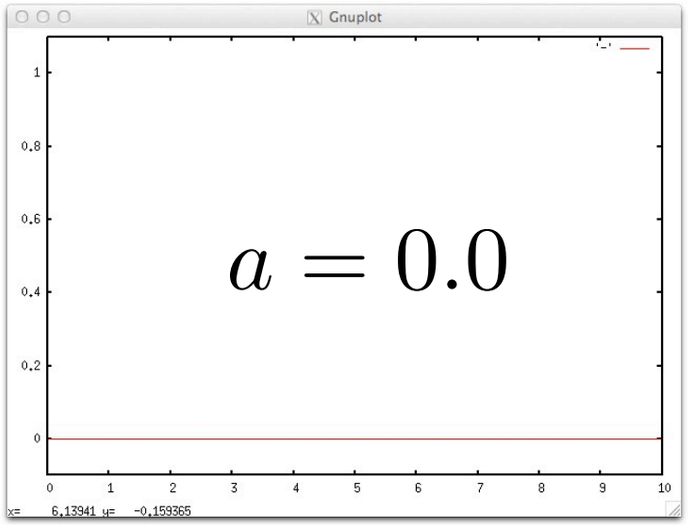
$$\dot{x}(t) = x(t)$$
$$x(0) = 0.01$$
$$(x_s(t) = 0.01e^{t})$$

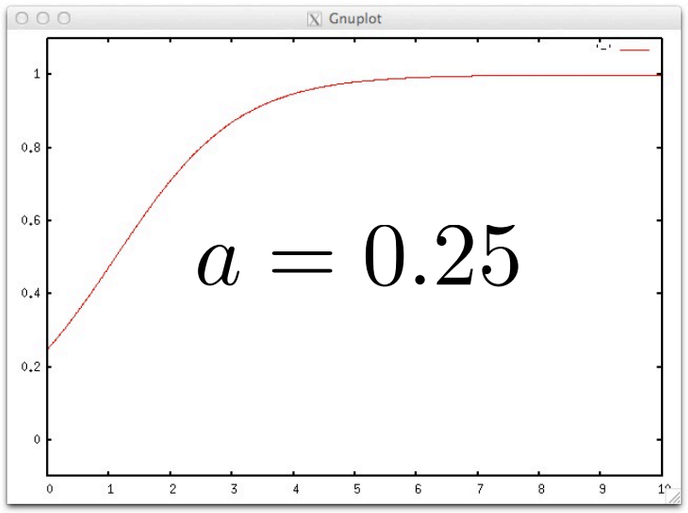
5_ODE_3.c

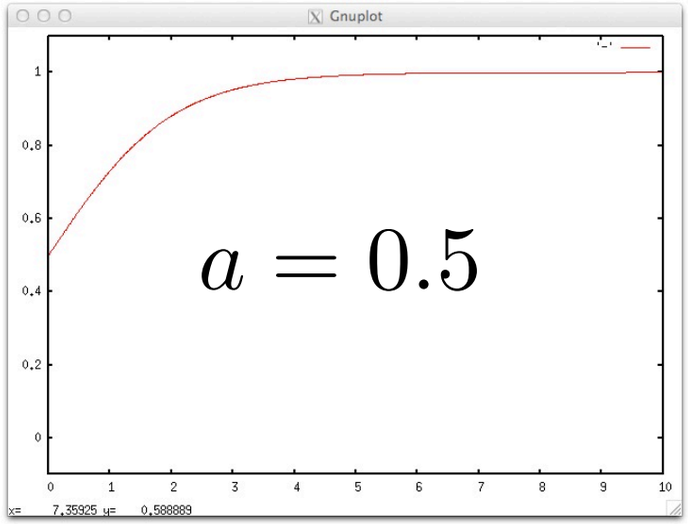
$$\dot{x}(t) = x(t)(1 - x(t))$$
$$x(0) = a \quad 0 \leq a \leq 1$$
$$(x_s(t) = ae^{t}/(1 - a + ae^{t}))$$

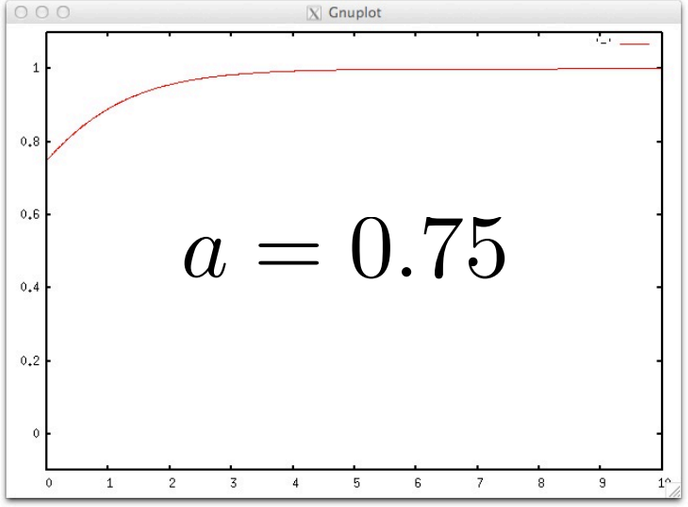
5_ODE_4.c

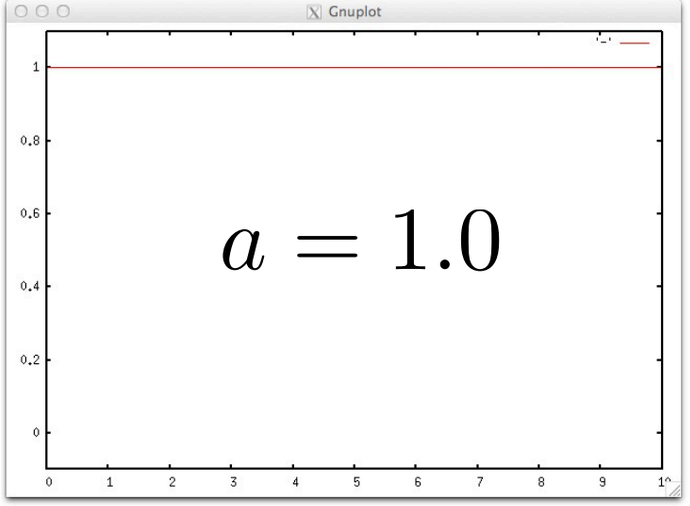
常微分方程式の数値解法











$$\dot{x}(t) = x(t)(1 - x(t))$$

$$x(0) = a$$

$$(x_s(t) = ae^t/(1 - a + ae^t))$$

a=0のとき以外はいつも1に向かう.なぜだろう?

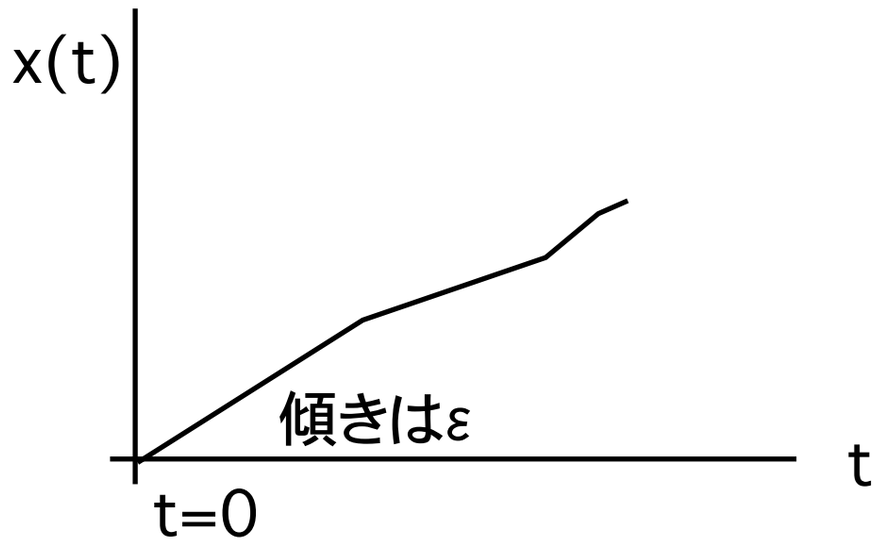
なぜだろう？

$$\dot{x}(t) = x(t)(1 - x(t))$$

$x(0) = \epsilon$　と置く.ただし　$\epsilon > 0$ の小さい数とする.

$t = 0$　を代入する.

$$\dot{x}(0) = \epsilon(1 - \epsilon) = \epsilon - \epsilon^2 \simeq \epsilon > 0$$



つまり時刻t=0での傾きは正なので,ちょっと先の未来では,少なくとも値は増える.

つまり, a=0から離れていく!

小さの勝負表

	ε	ε^2
ε=0.1	0.1	0.01
ε=0.01	0.01	0.0001
ε=0.001	0.001	0.000001
ε=0.0001	0.0001	0.00000001

一兆円(10^12)の前では100万円(10^6)はゴミでしょ?

常微分方程式の数値解法

多変数の場合

$$\dot{x}(t) = -y(t)$$

$$\dot{y}(t) = x(t)$$

$$x(0) = 1$$

$$y(0) = 0$$

問題

離散化

$$\frac{x_N(t_{i+1}) - x_N(t_i)}{dt} = -y_N(t_i)$$

$$\frac{y_N(t_{i+1}) - y_N(t_i)}{dt} = x_N(t_i)$$

$$x_N(t_0) = 1 \qquad y_N(t_0) = 0$$

C Code

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

int
main(void)
{
    int i;
    double dt = 0.01;
    double xN,xN_new;
    double yN,yN_new;

    xN = 1.0;
    yN = 0.0;
    for (i = 0; i < 1000; i++)
    {
        printf("%f %f %f\n",i * dt, xN, yN);
        xN_new = xN + dt * (-yN);
        yN_new = yN + dt * (xN);
        xN = xN_new;
        yN = yN_new;
    }

    return 0;
}
```

5_ODE_5.c

常微分方程式の数値解法

$$\dot{x}(t) = -y(t)$$

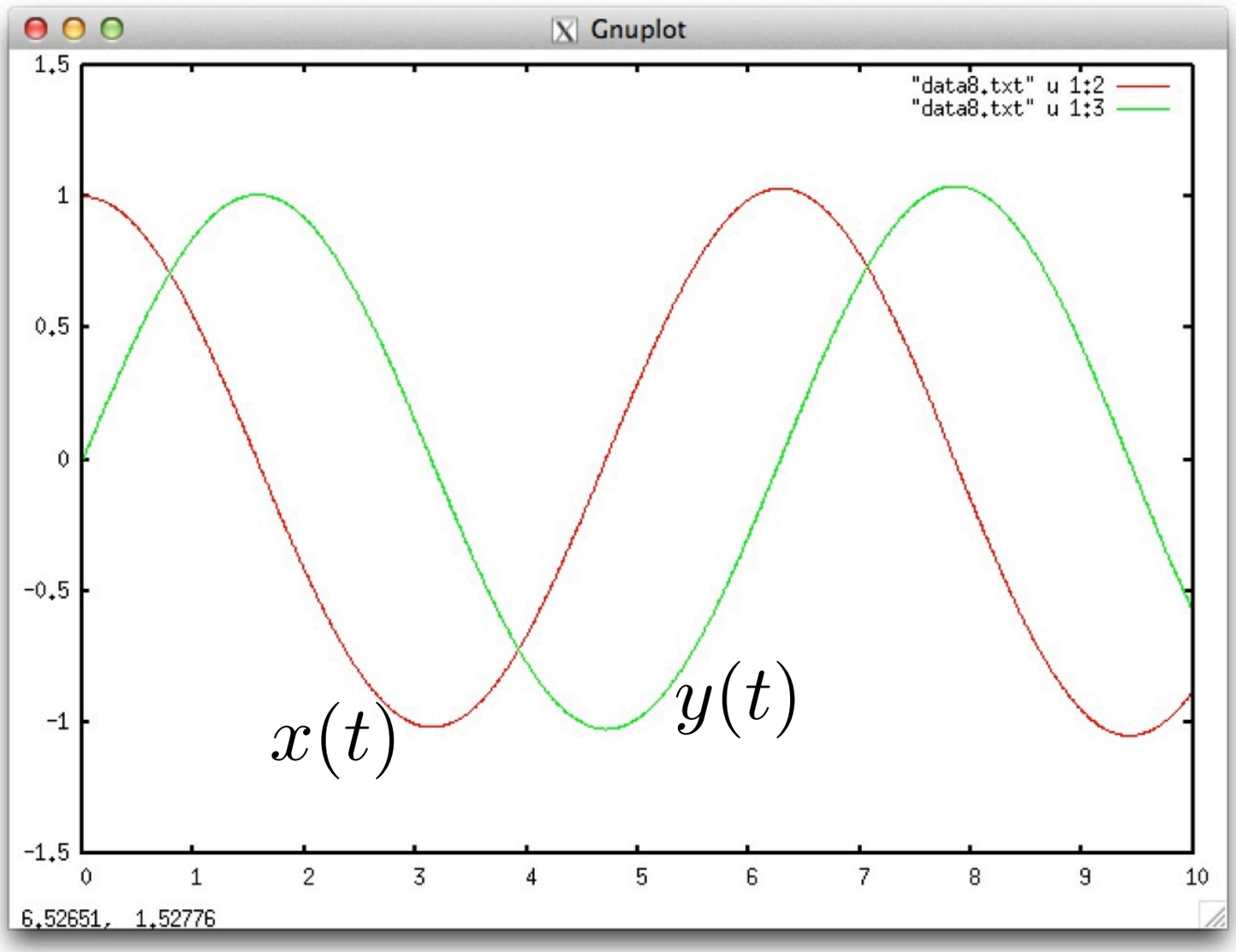
$$\dot{y}(t) = x(t)$$

$$x(0) = 1$$

$$y(0) = 0$$

問題

多変数の場合



常微分方程式の数値解法

誤差について

$$\dot{x}(t) = -y(t)$$
$$\dot{y}(t) = x(t)$$
$$x(0) = 1$$
$$y(0) = 0$$

厳密解

$$x_s(t) = \sin t$$
$$y_s(t) = \cos t$$

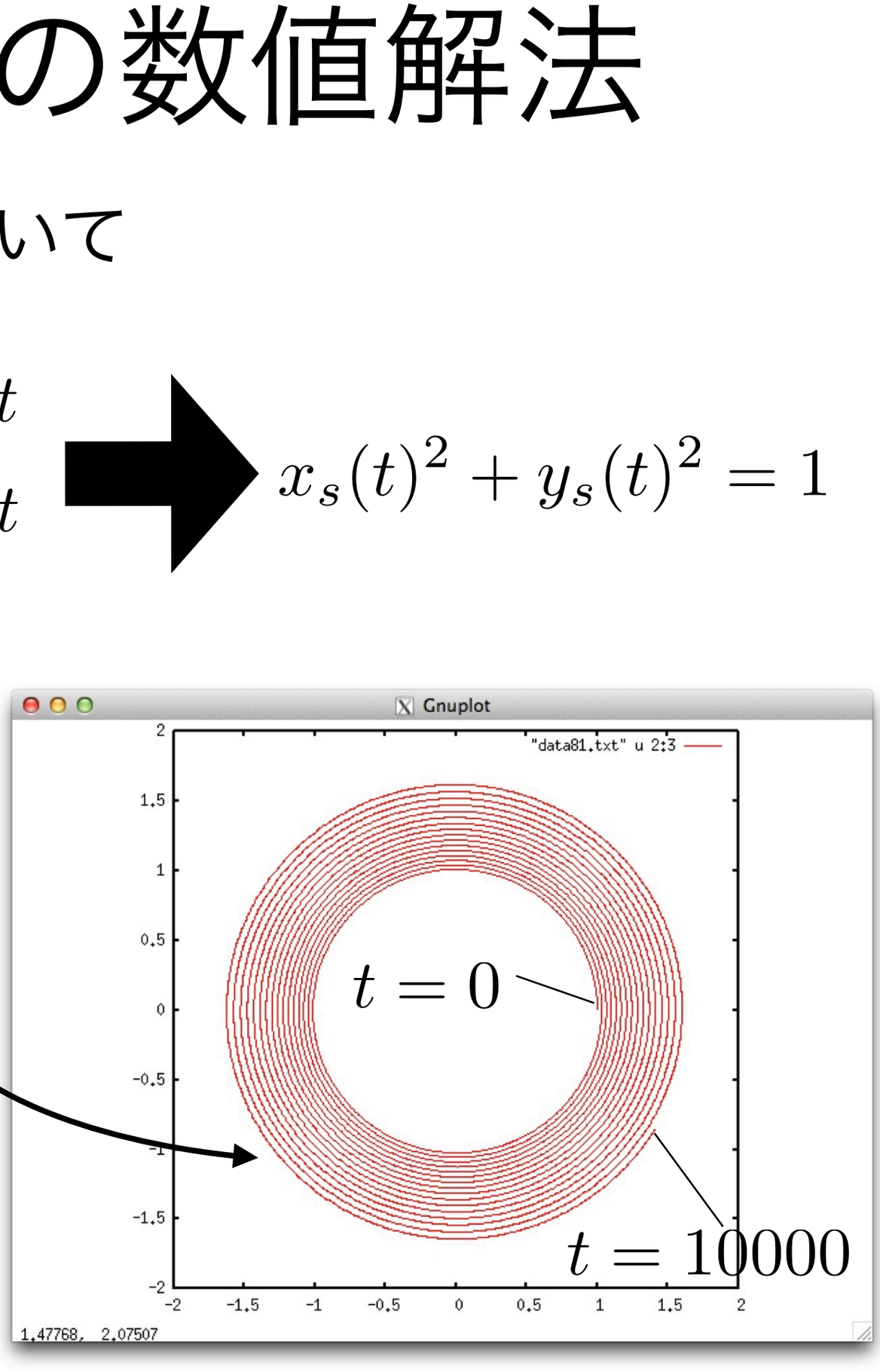
$$x_s(t)^2 + y_s(t)^2 = 1$$

問題

数値計算では軌道が膨らんでいる.

$$f'(t) := \lim_{\delta t \to 0} \frac{f(t+\delta t) - f(t)}{\delta t}$$

を大雑把に計算したことが原因！



常微分方程式の数値解法

誤差について

$$f'(t) := \lim_{\delta t \to 0} \frac{f(t+\delta t) - f(t)}{\delta t}$$

を大雑把に計算したことが原因！

この式を厳密に成り立たせるためには $dt \to 0$ でなければならない.

とりあえず小さなdtをセットしておき，大きくしても解の挙動が変わらないものを選ぶ.

数値計算で得られた解は，与えられた方程式の解にきちんとなっているか確認せねばならない.

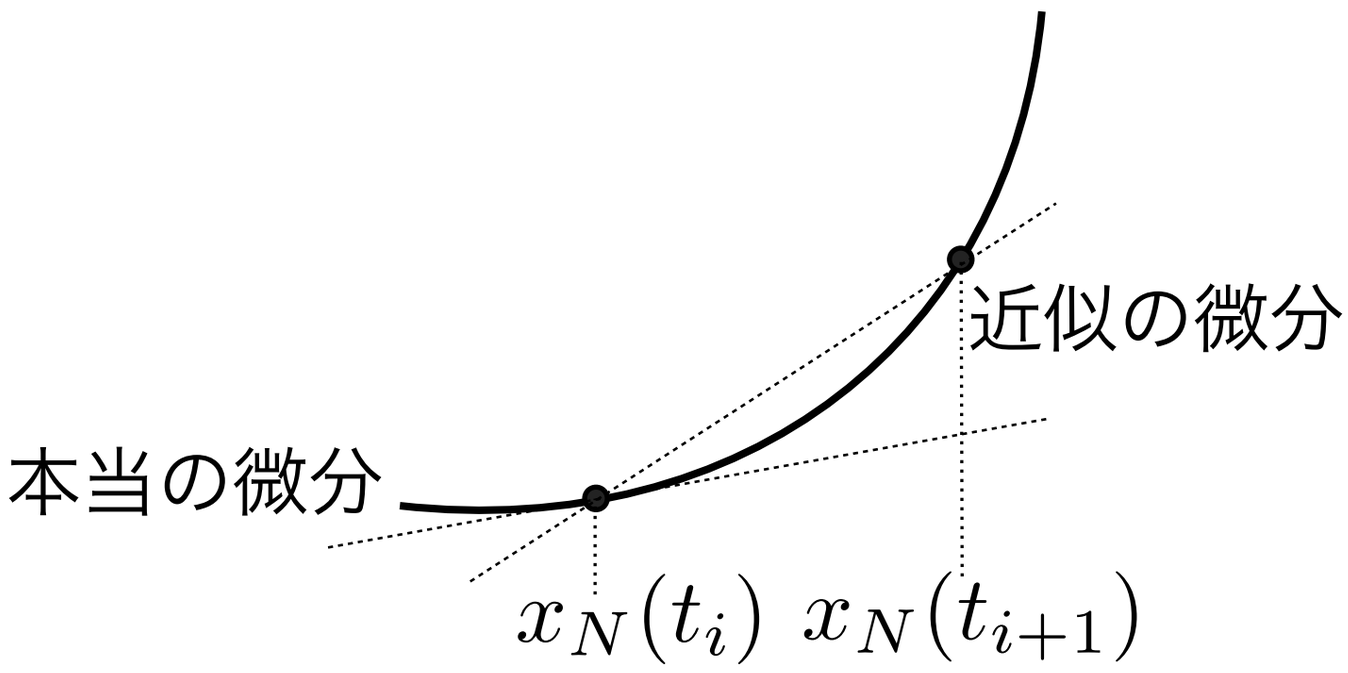
厳密解がわかっている問題でまずしっかりチェックして，未知な問題に取り組むことが大切！

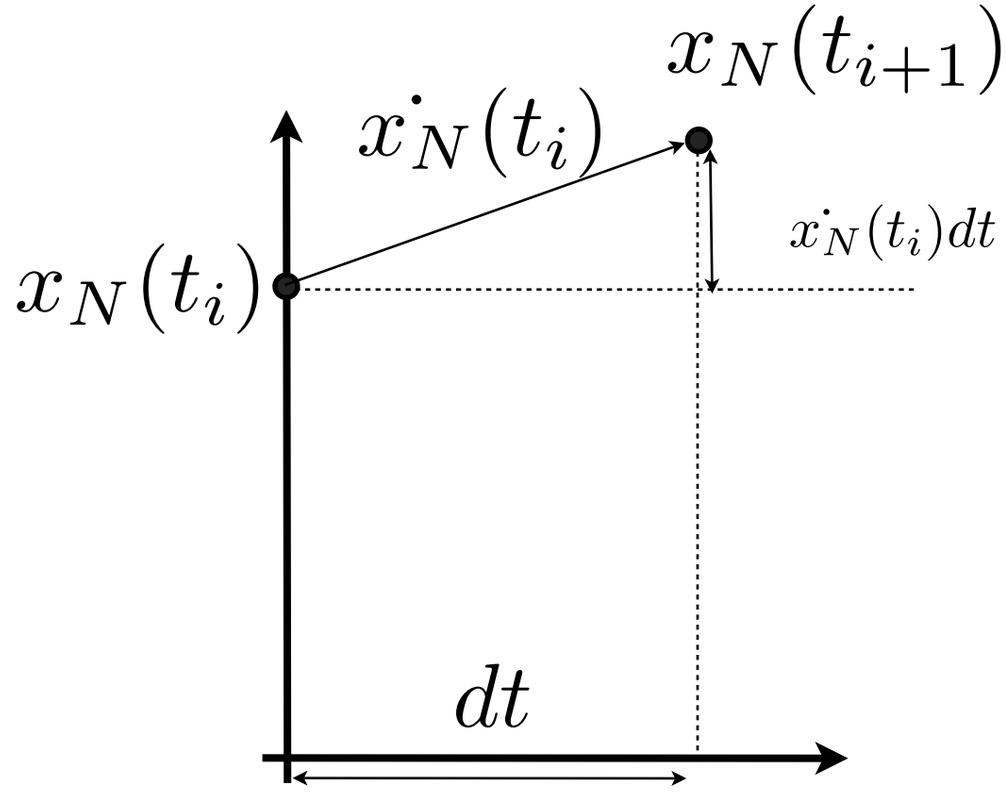
常微分方程式の数値解法まとめ

$$\dot{x}(t) = f(x, t)$$

オイラー法とは 微分を 1 次の差分で近似する.

$$\frac{x_N(t_{i+1}) - x_N(t_i)}{dt} \approx f(x_N(t_i), t_i)$$



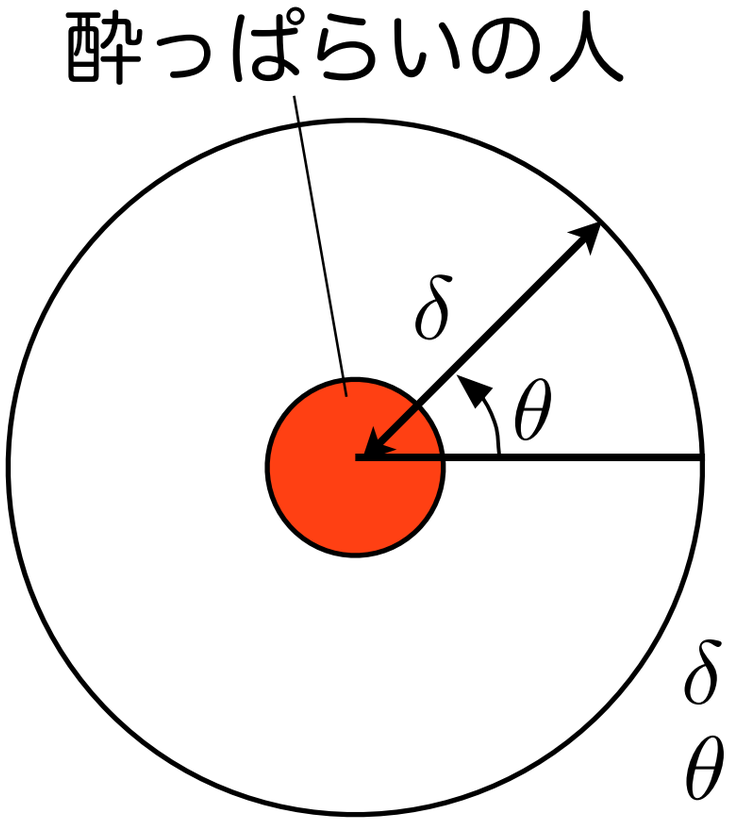


拡散方程式の
数値解法入門

そもそも拡散方程式とは?

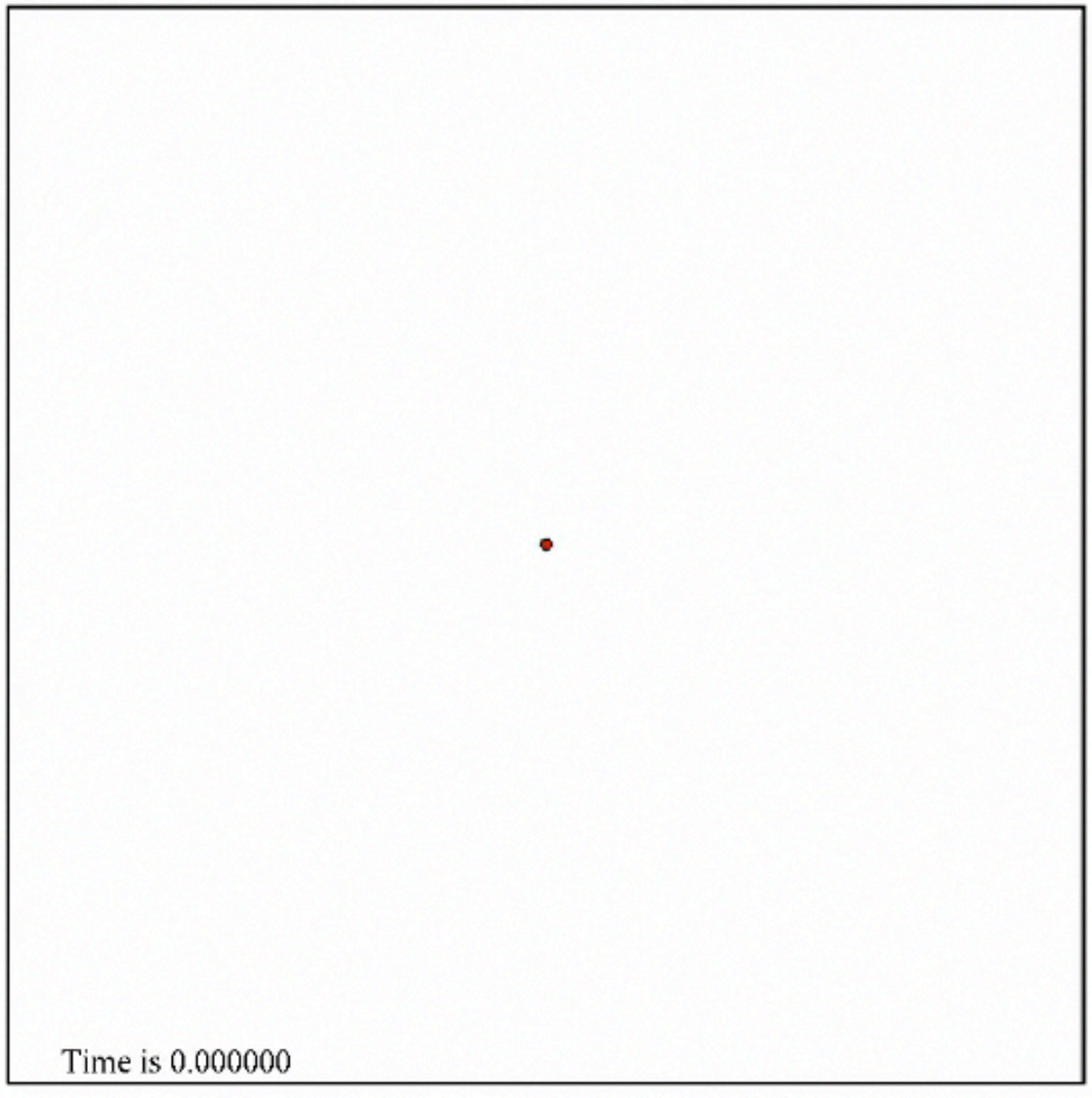
酔歩（酔っ払い歩き）

酔っぱらいの人の運動方程式を求めよう！



δ: 定数
θ: ランダム

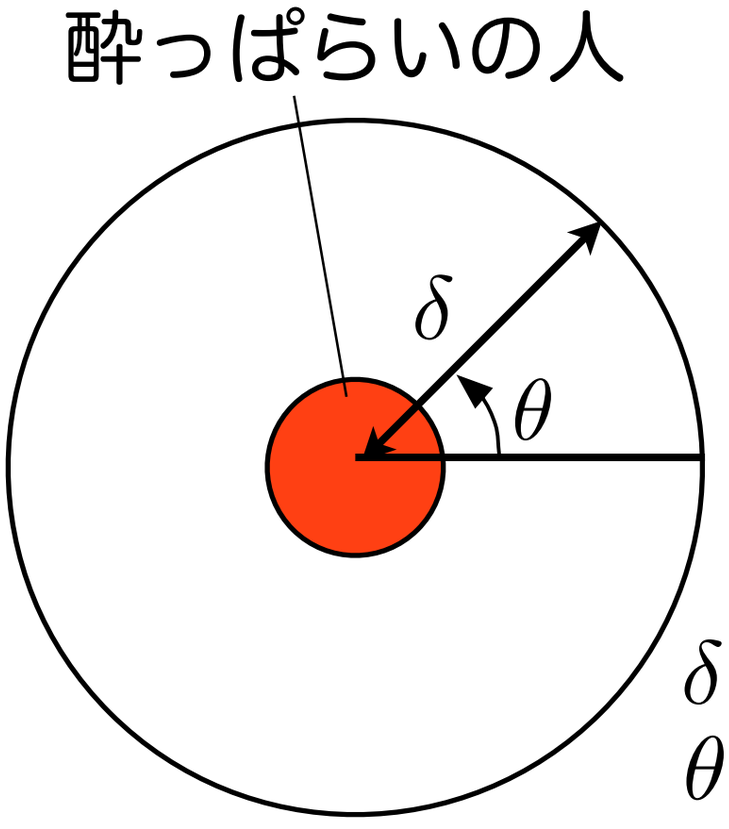
酔っぱらいの人は角度がランダムで一定の距離だけふらふらするはず.



酔っぱらいの人が１人の場合

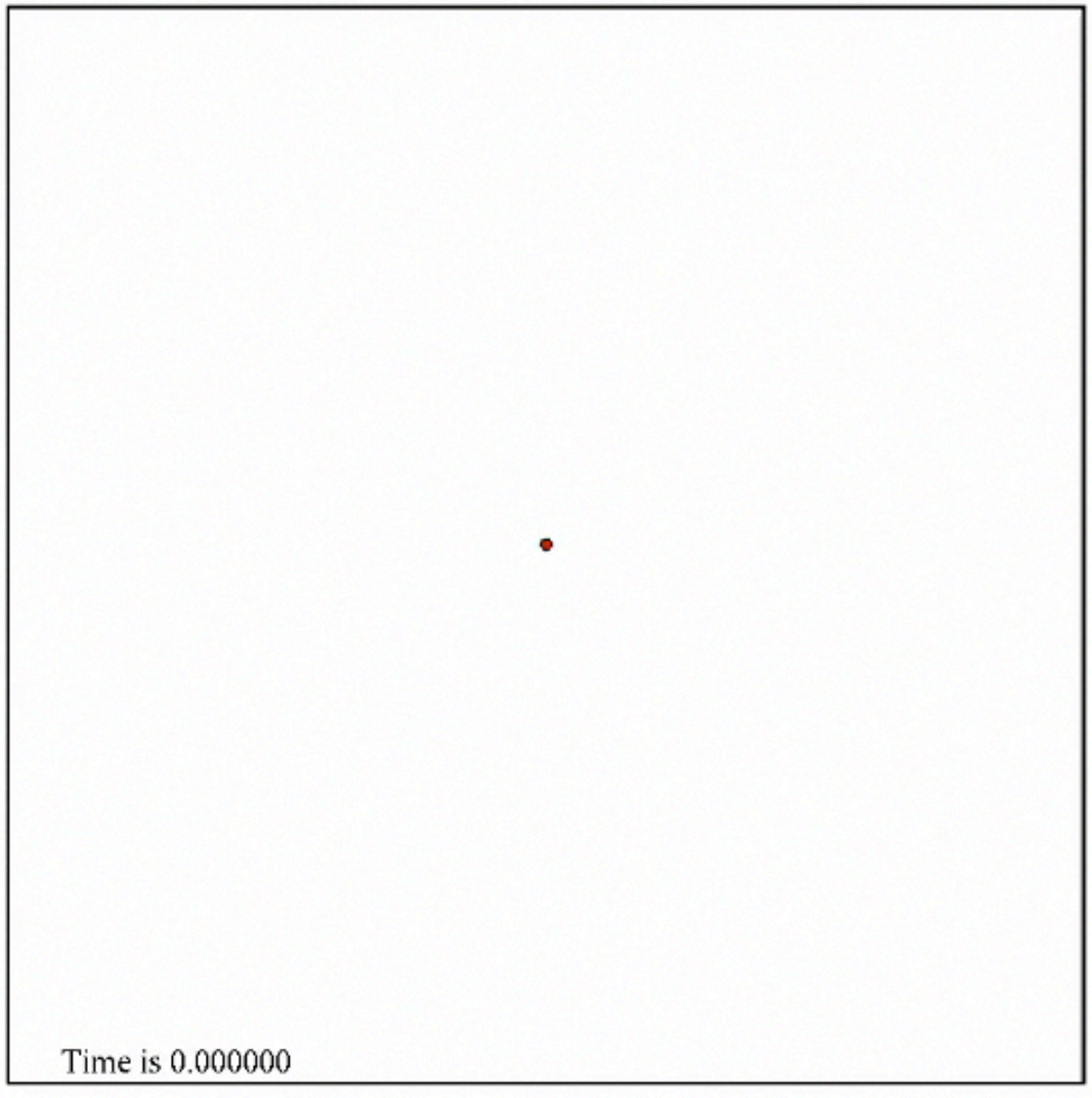
酔歩（酔っ払い歩き）

酔っぱらいの人の運動方程式を求めよう！



δ: 定数
θ: ランダム

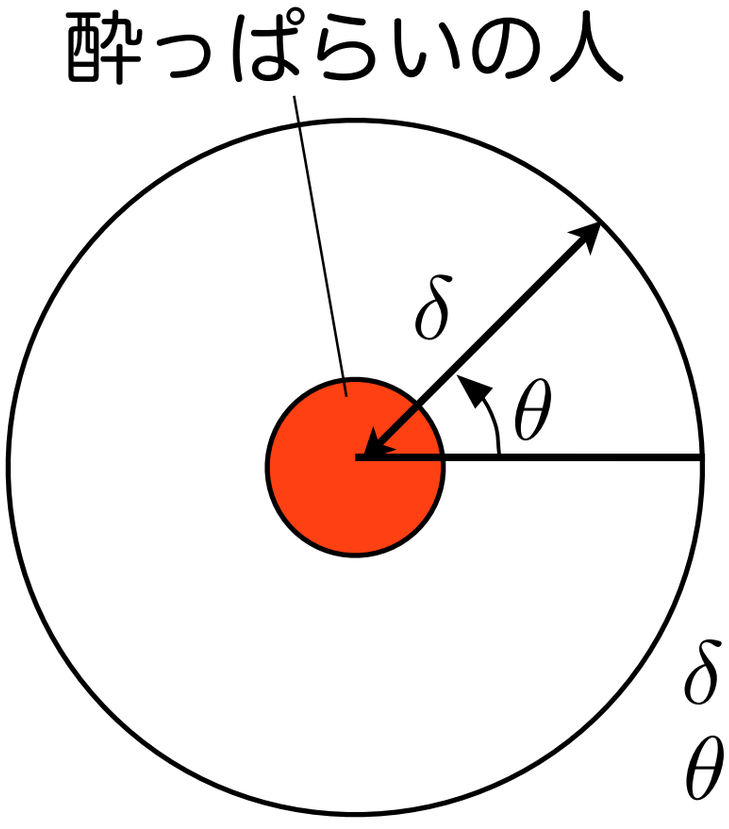
酔っぱらいの人は角度がランダムで一定の距離だけふらふらするはず.



酔っぱらいの人が１人の場合

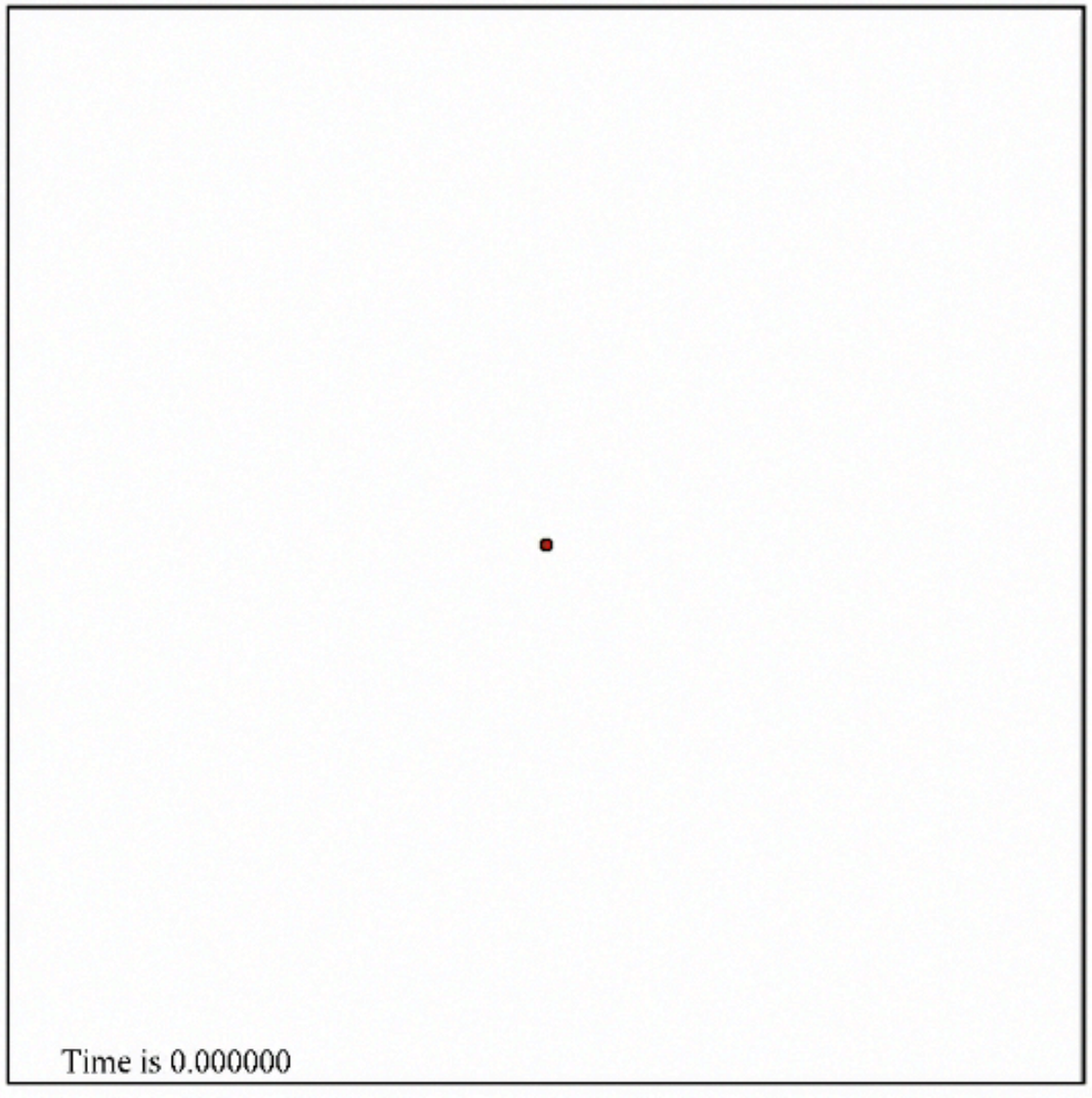
酔歩（酔っ払い歩き）

酔っぱらいの人の運動方程式を求めよう！



δ: 定数
θ: ランダム

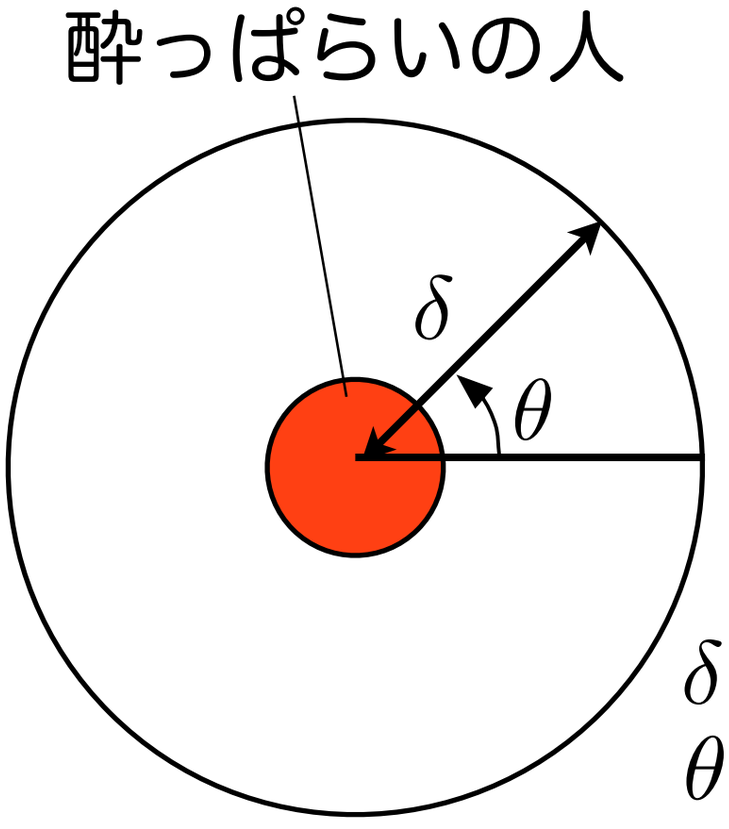
酔っぱらいの人は角度がランダムで一定の距離だけふらふらするはず.



酔っぱらいの人が１００人の場合

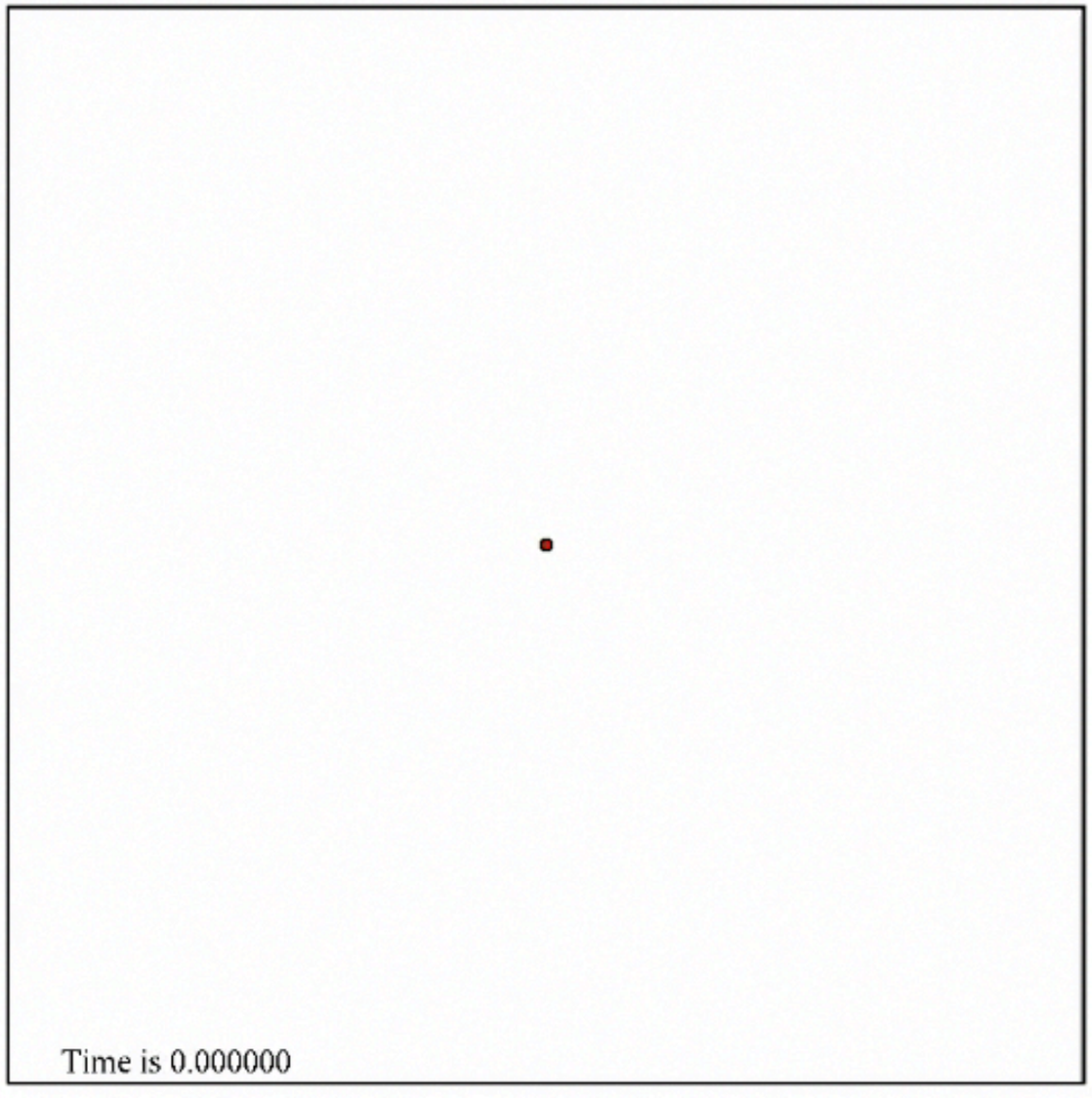
酔歩（酔っ払い歩き）

酔っぱらいの人の運動方程式を求めよう！



δ: 定数
θ: ランダム

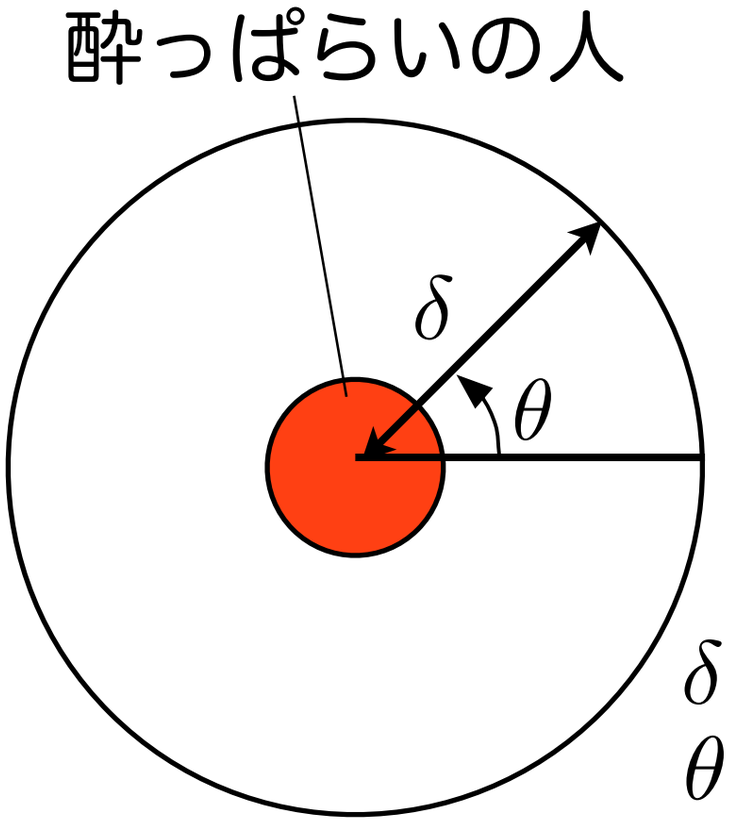
酔っぱらいの人は角度がランダムで一定の距離だけふらふらするはず.



酔っぱらいの人が１００人の場合

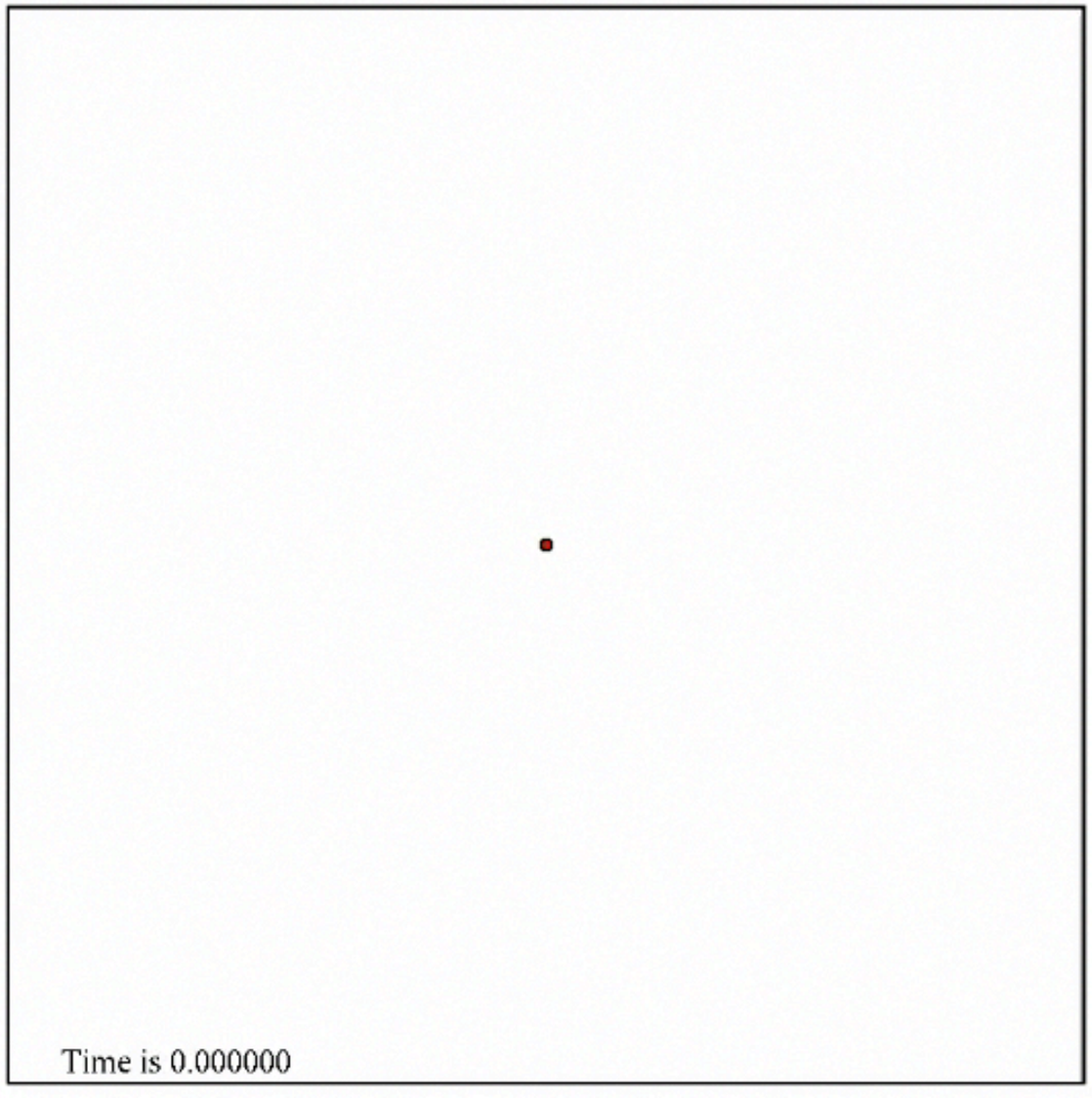
酔歩（酔っ払い歩き）

酔っぱらいの人の運動方程式を求めよう！



δ: 定数
θ: ランダム

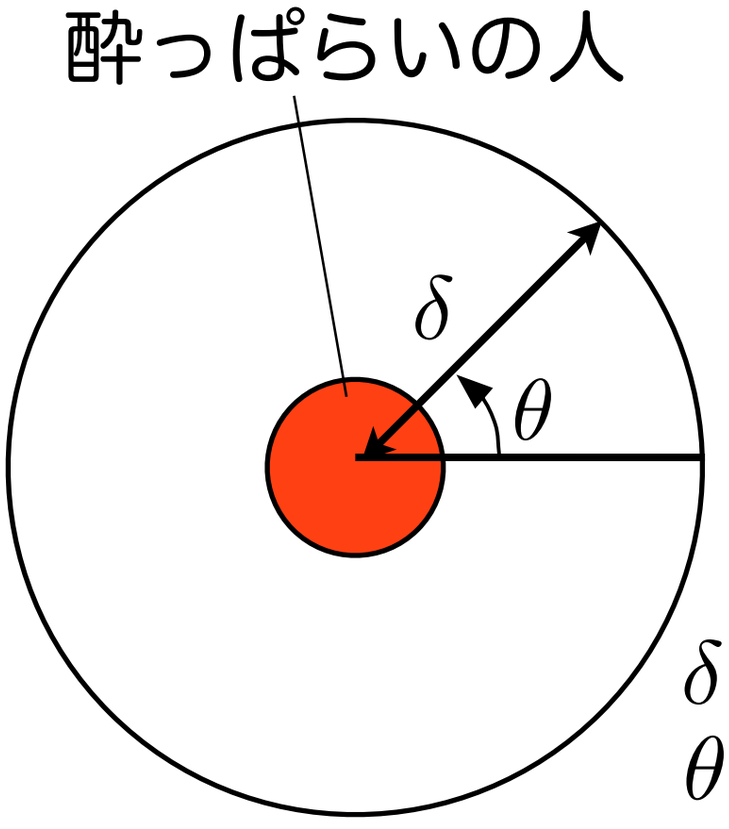
酔っぱらいの人は角度がランダムで一定の距離だけふらふらするはず.



酔っぱらいの人が１００００人の場合

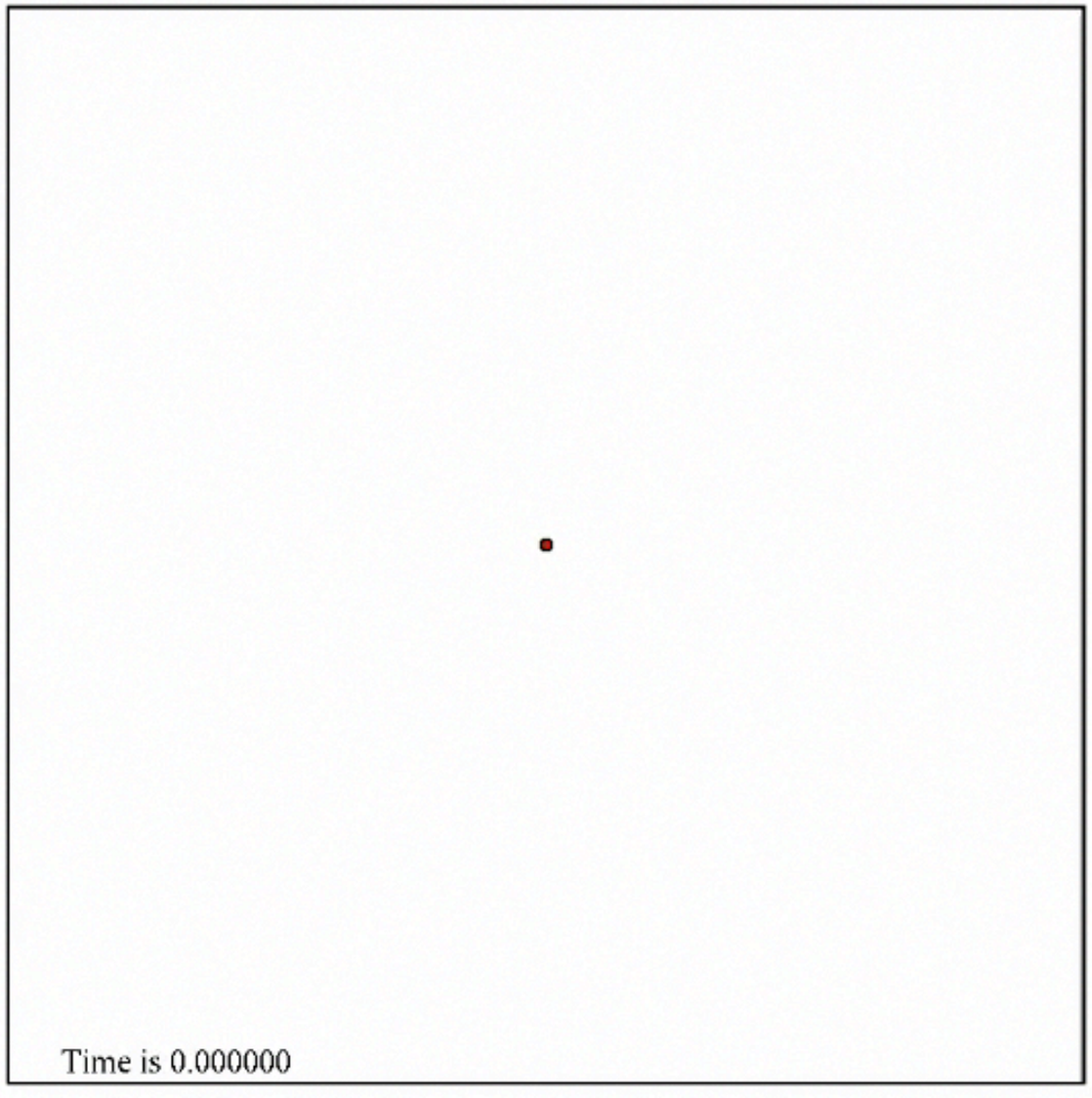
酔歩（酔っ払い歩き）

酔っぱらいの人の運動方程式を求めよう！



δ: 定数
θ: ランダム

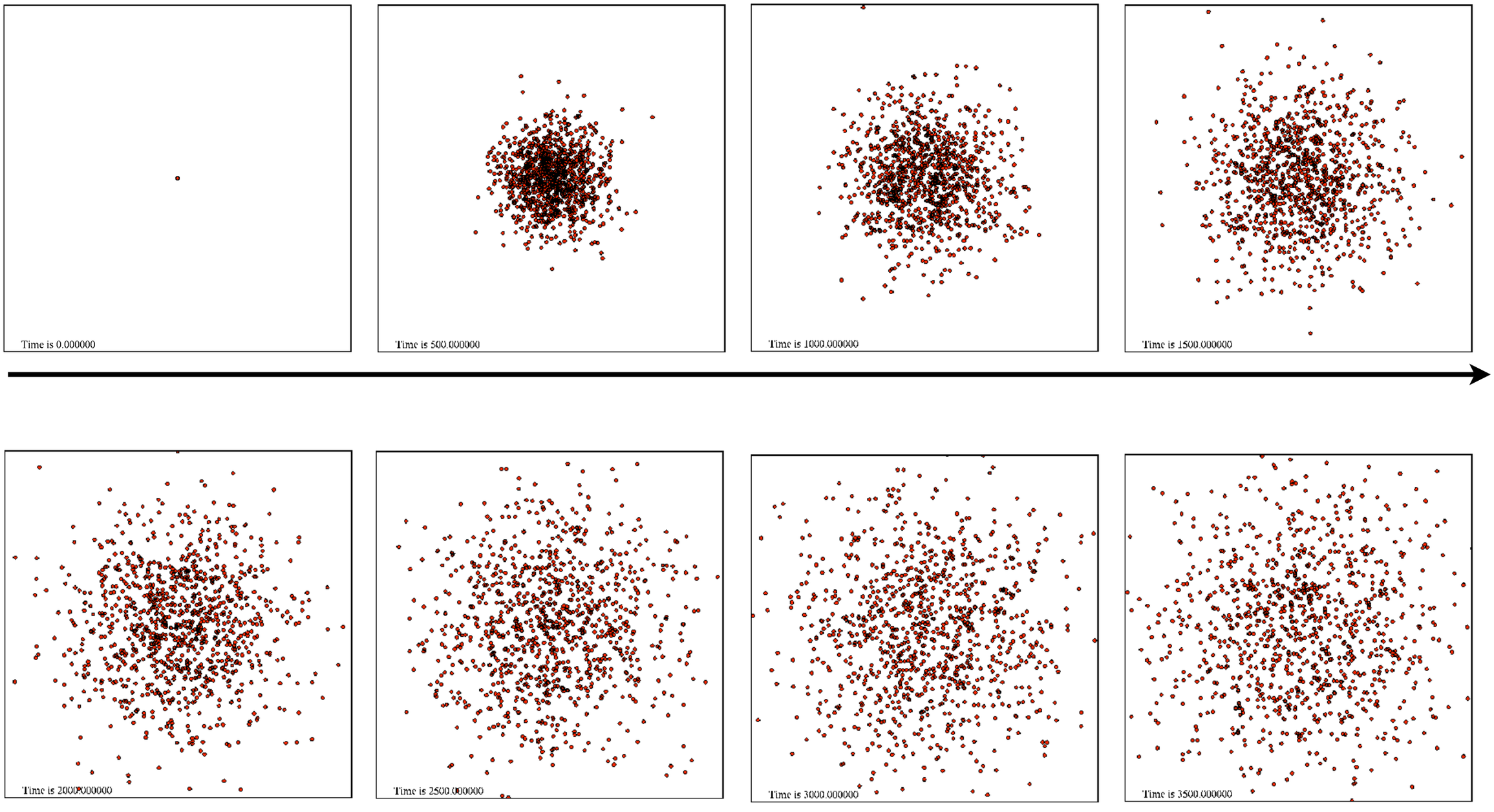
酔っぱらいの人は角度がランダムで一定の距離だけふらふらするはず.



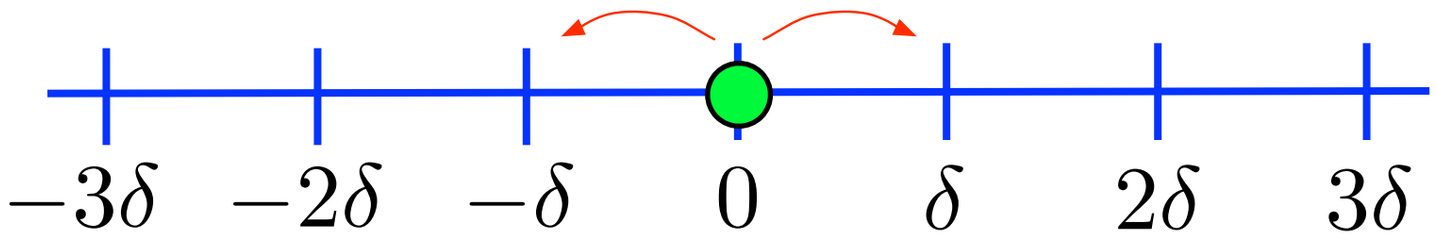
酔っぱらいの人が１００００人の場合

酔歩（酔っ払い歩き）

※酔っぱらいは意図して外に向かっているのではない



どういう方程式で記述できるか？



(1) 各粒子は τ 秒ごとに距離 δ だけ右か左に移動する.

(2) 各ステップで左右に行く確率はそれぞれ 1/2 であり,
前のステップでどちらに動いたかは記憶していない.

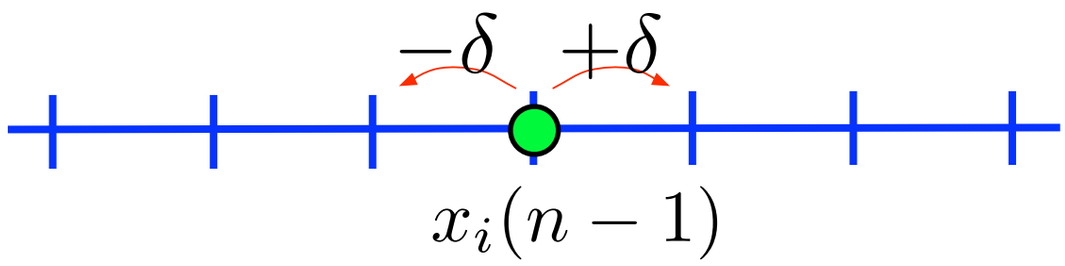
(3) 各粒子は他の粒子と独立に動き, 相互作用することはない.

※ここから資料の一部は小林亮先生（広島大学）にいただきました.

粒子の位置の平均

初期時刻に I 個の全粒子が原点にいる場合を考える.

第 i 番目の粒子の第 n ステップにおける位置： $x_i(n)$



$$x_i(n) = x_i(n-1) \pm \delta$$ ⟸ 確率 1/2

n 第　ステップにおける粒子の平均位置：　$\langle x(n) \rangle = \frac{1}{I} \sum_{i=1}^{I} x_i(n)$

粒子数が十分大きいとき　$\langle x(n) \rangle$ と　$\langle x(n-1) \rangle$　の関係は

$$\langle x(n) \rangle = \frac{1}{I} \sum_{i=1}^{I} (x_i(n-1) \pm \delta) = \frac{1}{I} \sum_{i=1}^{I} x_i(n-1) = \langle x(n-1) \rangle$$

どういう方程式で記述できるか？

$$\langle x(0) \rangle = 0 \quad \Rightarrow \quad \langle x(n) \rangle = 0$$

第nステップにおける粒子の位置の分散：　$\langle x(n)^2 \rangle$

$$\langle x(n)^2 \rangle = \frac{1}{I} \sum_{i=1}^{I} x_i(n)^2$$

$$x_i(n)^2 = x_i(n-1)^2 \pm 2\delta x_i(n-1) + \delta^2$$

$$\langle x(n)^2 \rangle = \langle x(n-1)^2 \rangle + \delta^2 = \langle x(n-2)^2 \rangle + 2\delta^2$$
$$= \cdots = \langle x(0)^2 \rangle + n\delta^2$$

$$\boxed{\langle x(n)^2 \rangle = n\delta^2}$$

粒子の広がり具合

時間変数 t を導入する.　　$t = \tau n$

$$\left\langle x(n)^2 \right\rangle = n\delta^2 \Rightarrow \left\langle x(t)^2 \right\rangle = \frac{\delta^2}{\tau} t$$

$$D = \frac{\delta^2}{2\tau} \quad \text{とおくと} \quad \left\langle x(t)^2 \right\rangle = 2Dt$$

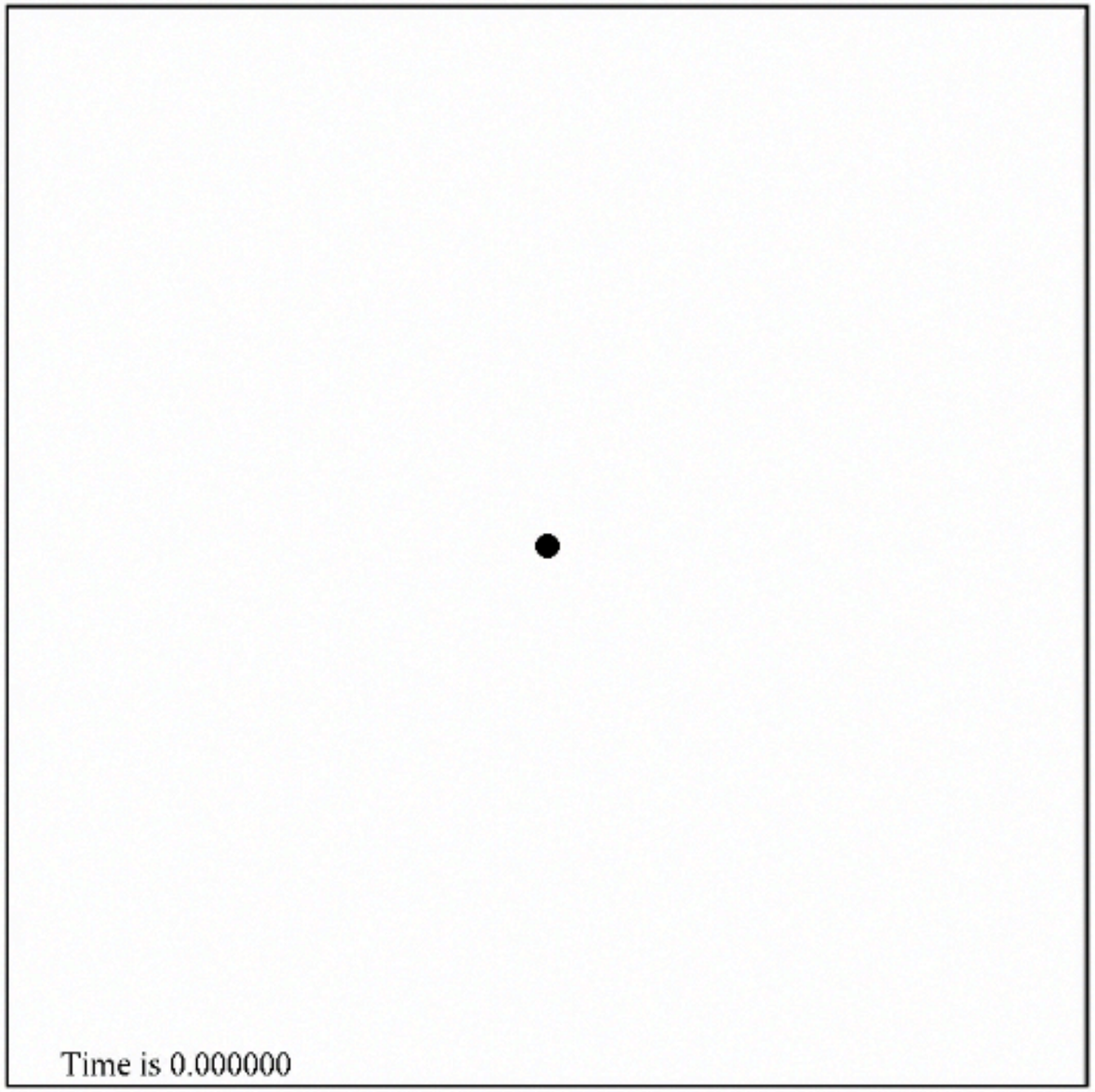
標準偏差　$\sigma(t)$ ⇒ 粒子の広がり具合の指標

$$\sigma(t) = \sqrt{2Dt}$$

D ：拡散係数

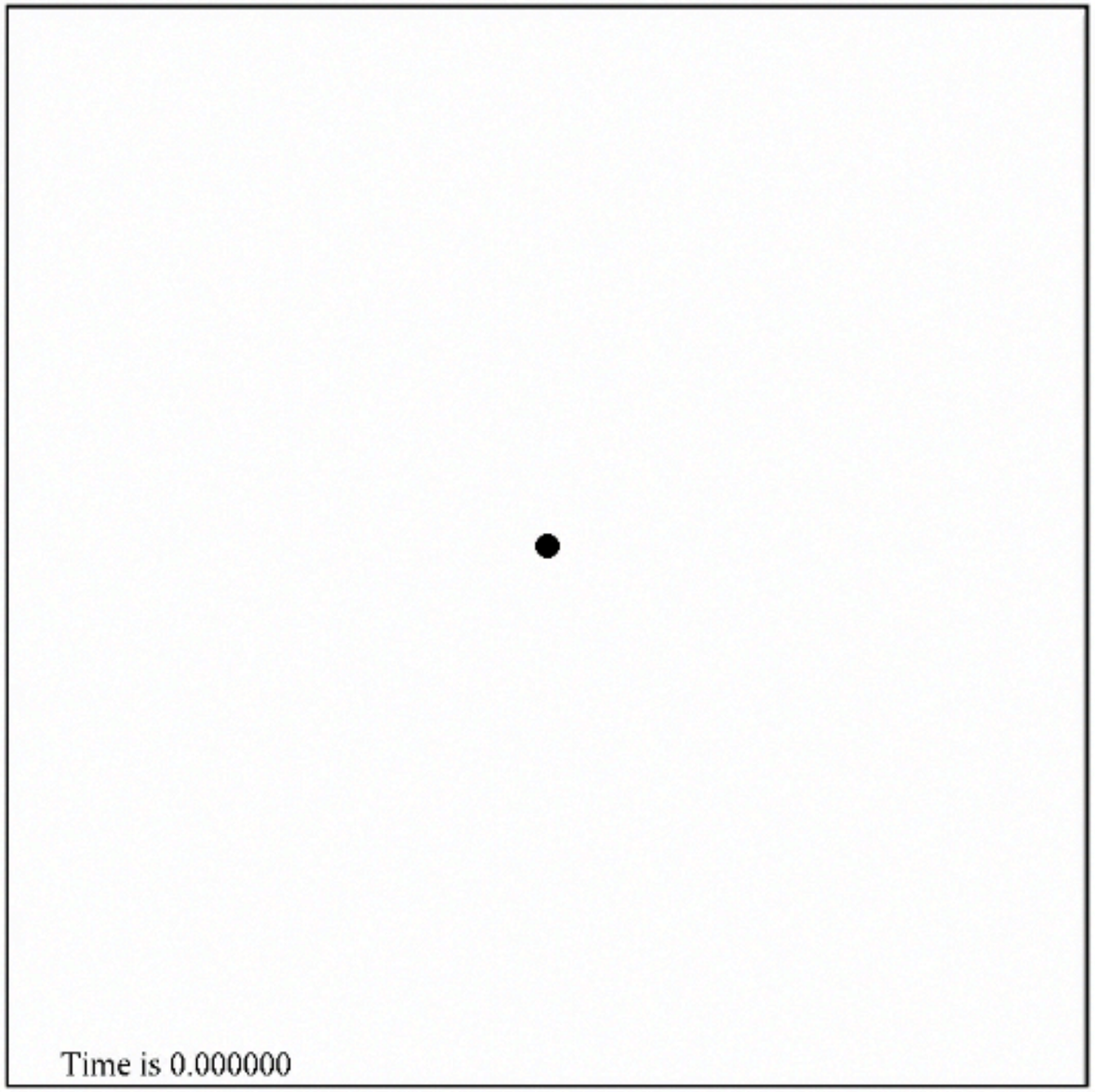
$$[D] = L^2 T^{-1}$$

確かに酔っ払いは．．．



○ 半径$\sqrt{2Dt}$の円

確かに酔っ払いは．．．



○ 半径$\sqrt{2Dt}$の円

電車の中で臭い人が．．．

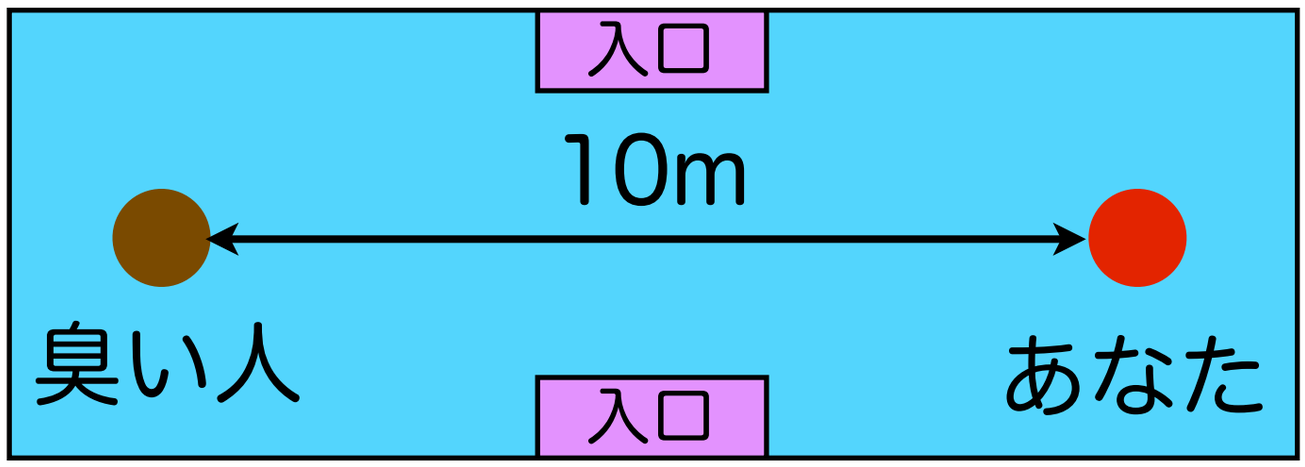
考えてみよう

空気中の微粒子の拡散係数

$$\simeq 10^{-5}\mathrm{m}^2/\mathrm{sec}$$

であることが知られている.

サイズ 10m 程度の電車に臭い人が乗ってきた.
この臭いはどれくらいの時間をかけて，あなたまで届くだろうか？



電車の中で臭い人が．．．

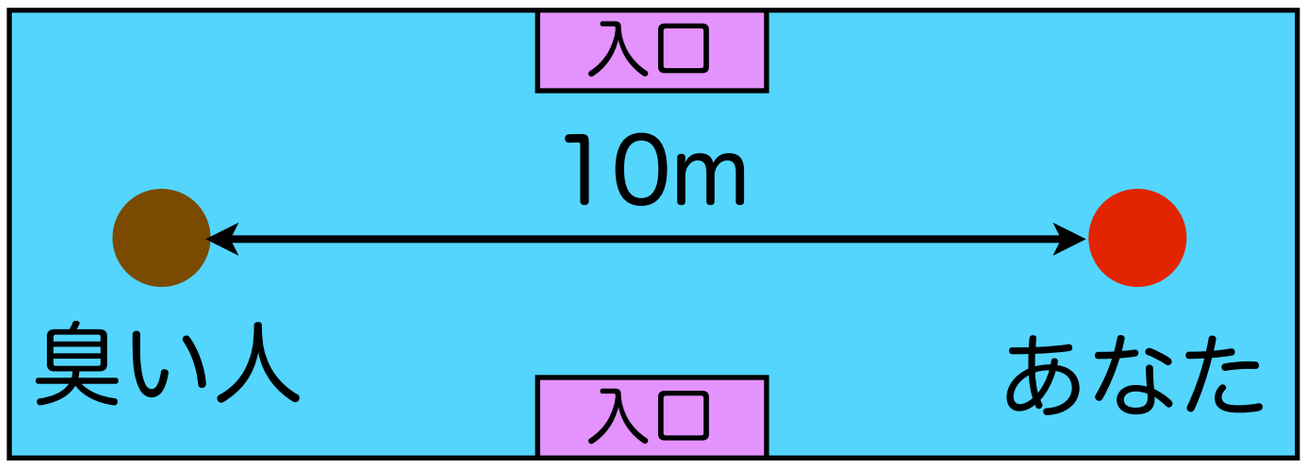
考えてみよう

空気中の微粒子の拡散係数

$$\simeq 10^{-5}\mathrm{m}^2/\mathrm{sec}$$

であることが知られている.

サイズ 10m 程度の電車に臭い人が乗ってきた.
この臭いはどれくらいの時間をかけて，あなたまで届くだろうか？



電車の中で臭い人が．．．

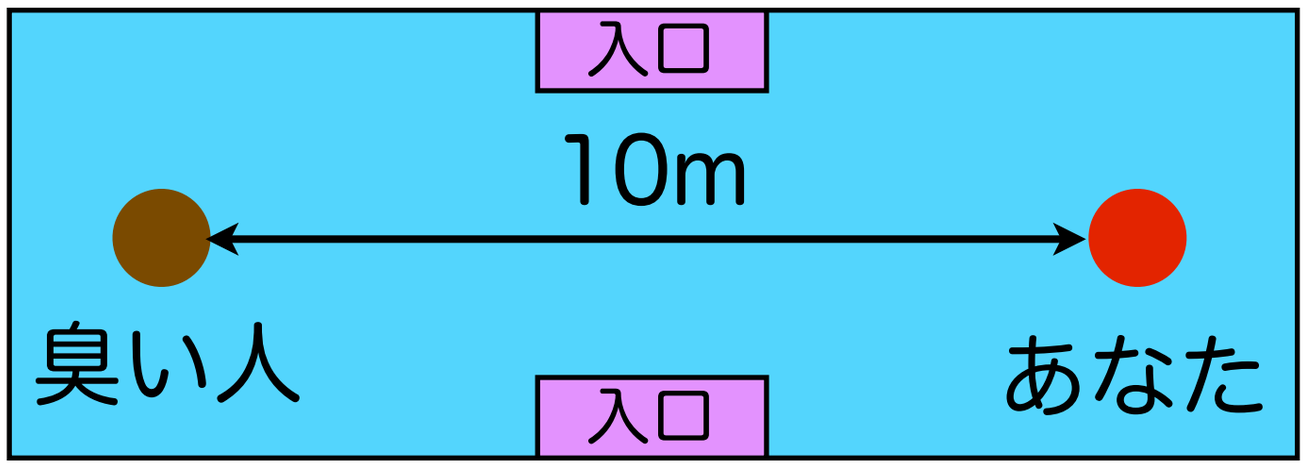
考えてみよう

空気中の微粒子の拡散係数

$$\simeq 10^{-5}\mathrm{m}^2/\mathrm{sec}$$

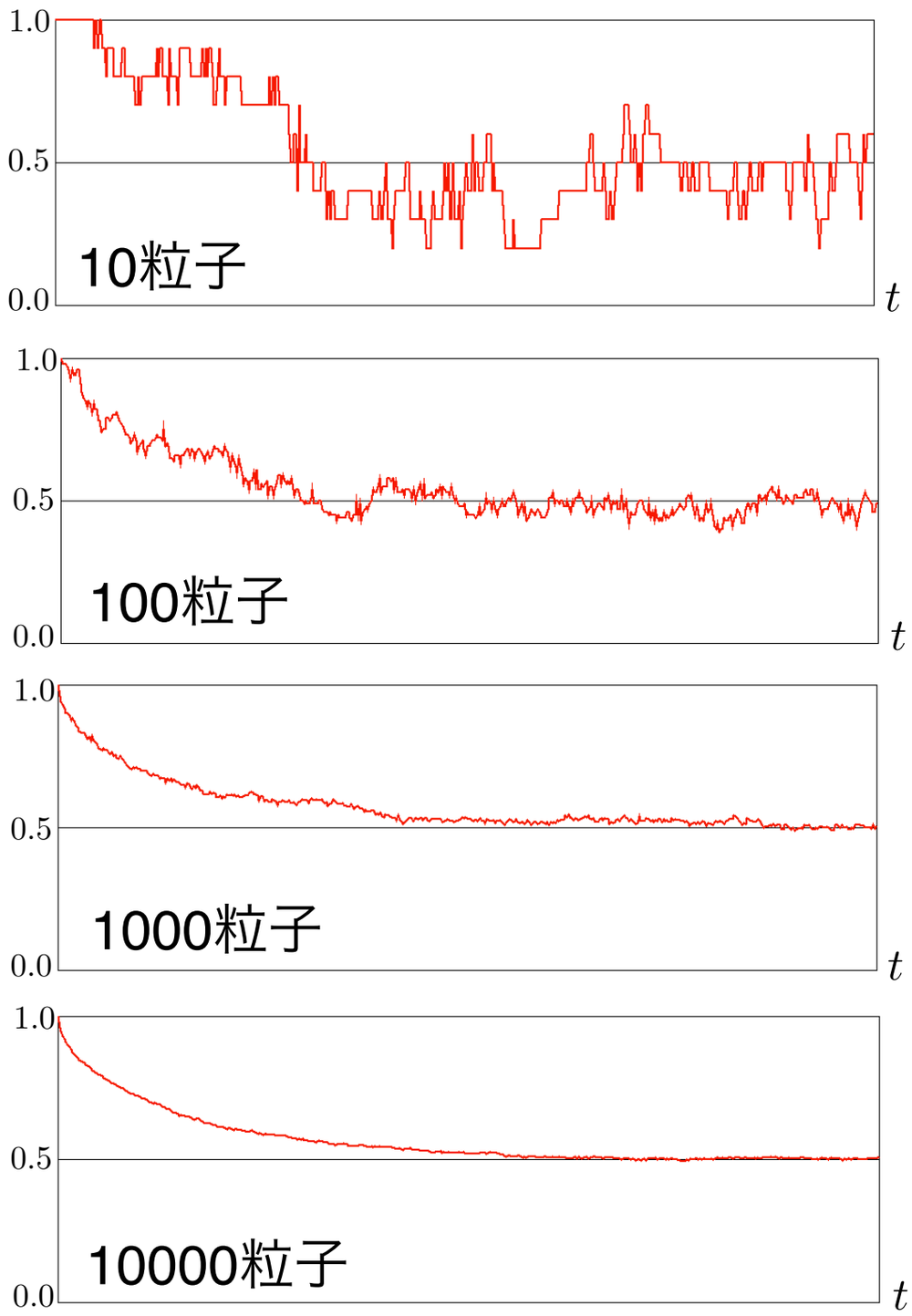
であることが知られている.

サイズ 10m 程度の電車に臭い人が乗ってきた.
この臭いはどれくらいの時間をかけて，あなたまで届くだろうか？



$$t = \frac{\sigma(t)^2}{2D} = \frac{10^2}{2 \times 10^{-5}} = 5 \times 10^6 \mathrm{sec} \simeq 2 \text{ month}$$

粒子の数を増やすと．．．


1.0
0.5
0.0
10粒子
t
1.0
0.5
0.0
100粒子
t
1.0
0.5
0.0
1000粒子
t
1.0
0.5
0.0
10000粒子
t

決定論の拡散方程式

を導出しよう.

思考実験

t=0

…

無限に長い針金

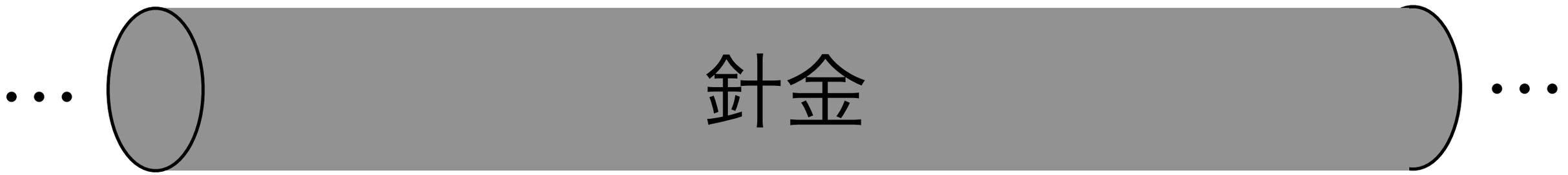
…

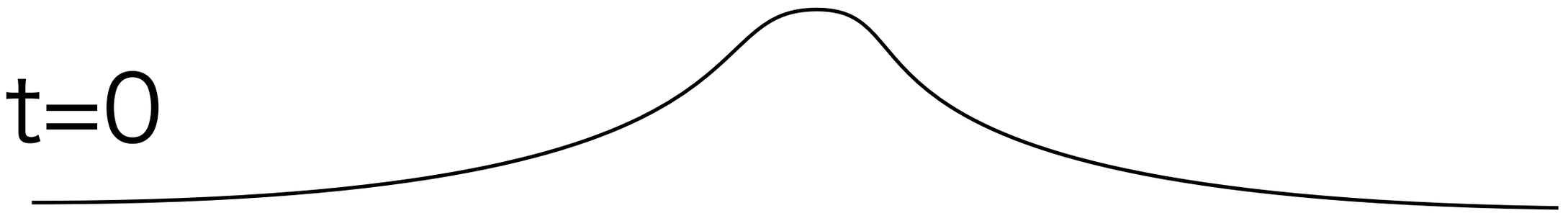
…

針金

…

思考実験

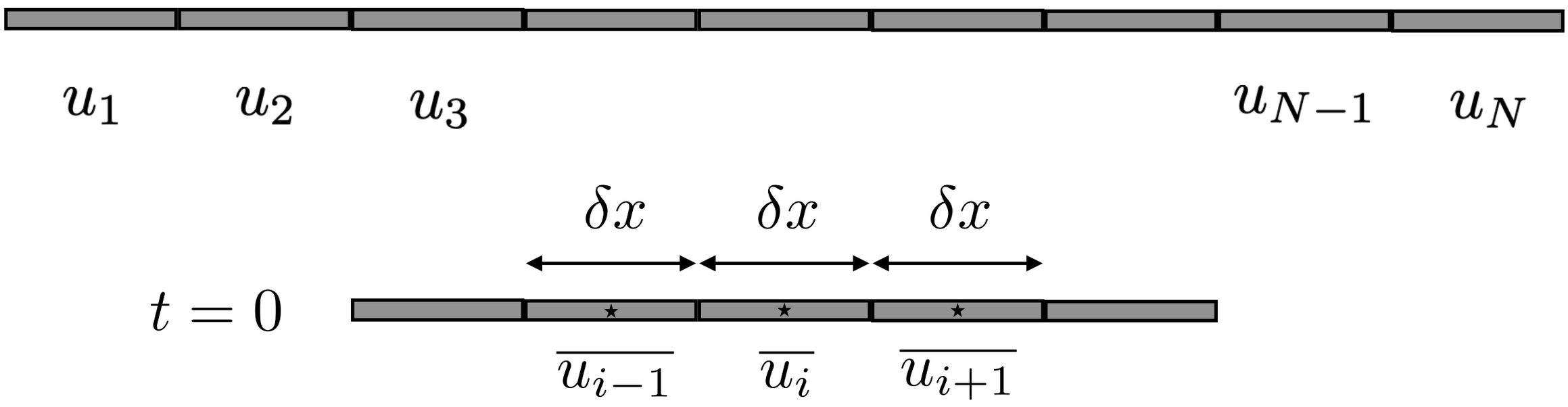




だんだん．．

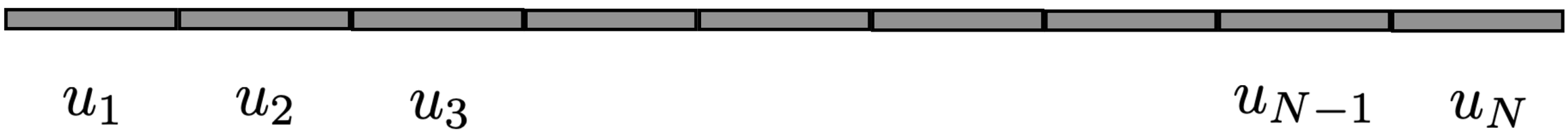
最終的には．．

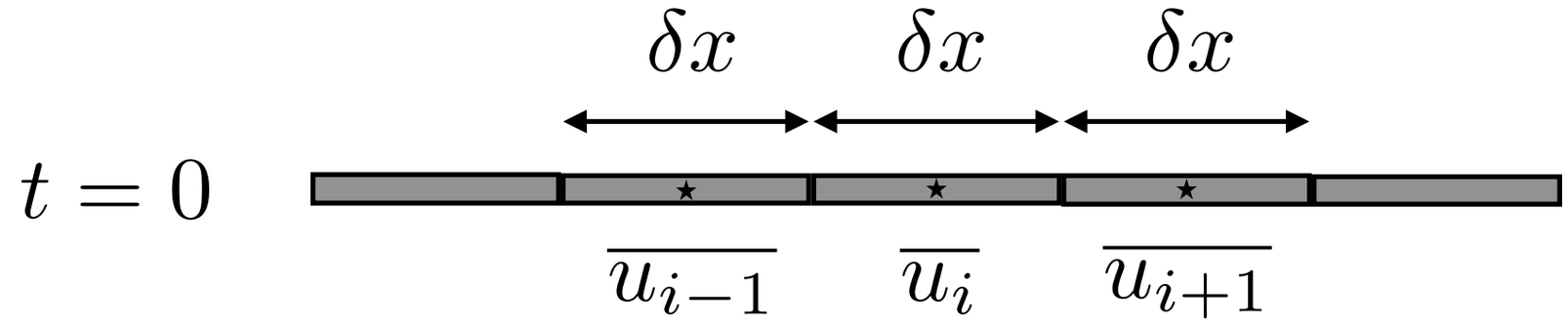
となりそう．

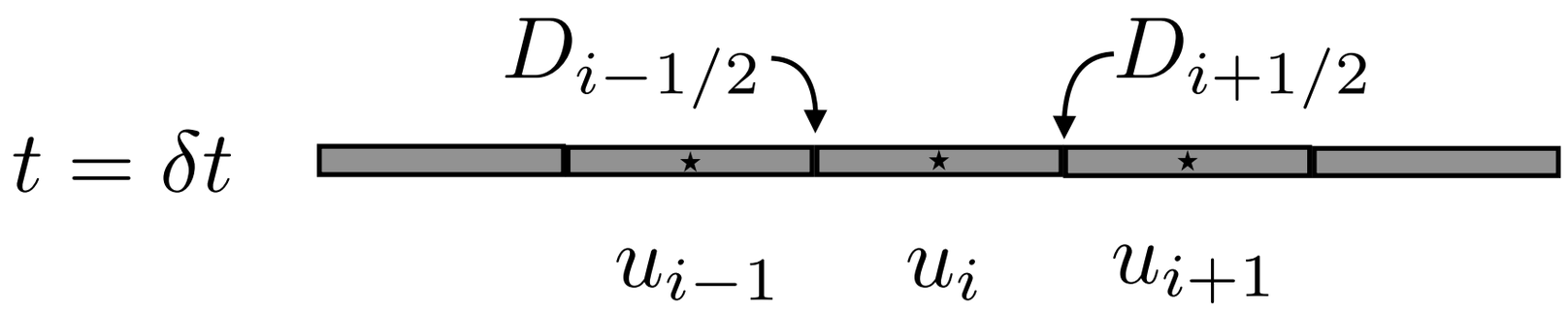


Fickの法則：温度の勾配に比例して物理量が伝播する.

$$u_i * \delta x - \overline{u_i} * \delta x = \delta t \left(D_{i+1/2}(\frac{\overline{u_{i+1}} - \overline{u_i}}{\delta x}) - D_{i-1/2}(\frac{\overline{u_i} - \overline{u_{i-1}}}{\delta x}) \right)$$







?度　　４度　　10度　４度　　４度　　6度

$$u_i * \delta x - \overline{u_i} * \delta x = \delta t \left(D_{i+1/2}(\frac{\overline{u_{i+1}} - \overline{u_i}}{\delta x}) - D_{i-1/2}(\frac{\overline{u_i} - \overline{u_{i-1}}}{\delta x}) \right)$$

正の符号　　負の符号

$$
\begin{aligned}
u_i * \delta x - \overline{u_i} * \delta x &= \delta t \left(D_{i+1/2}(\frac{\overline{u_{i+1}} - \overline{u_i}}{\delta x}) - D_{i-1/2}(\frac{\overline{u_i} - \overline{u_{i-1}}}{\delta x}) \right) \\
&= \delta t \left(D(\frac{\overline{u_{i+1}} - \overline{u_i}}{\delta x}) - D(\frac{\overline{u_i} - \overline{u_{i-1}}}{\delta x}) \right) \\
&= \delta t \frac{D}{\delta x} \left((\overline{u_{i+1}} - \overline{u_i}) - (\overline{u_i} - \overline{u_{i-1}}) \right) \\
&= \delta t \frac{D}{\delta x} \left(\overline{u_{i+1}} - 2\overline{u_i} - \overline{u_{i-1}} \right)
\end{aligned}
$$

$$
\frac{u_i - \overline{u_i}}{\delta t} = \frac{D}{\delta x^2} \left(\overline{u_{i+1}} - 2\overline{u_i} - \overline{u_{i-1}} \right)
$$

$$
\dot{u}_i = D \nabla^2 u
$$

※Fickの法則だけを使った厳密な導出方法もあります.

拡散方程式

を数値計算しよう.

$$\dot{u}_i = D\nabla^2 u$$

$$\frac{u_i - \overline{u_i}}{\delta t} = \frac{D}{\delta x^2}\left(\overline{u_{i+1}} - 2\overline{u_i} - \overline{u_{i-1}}\right)$$

$$U(x+\delta x) = U(x) + \frac{U'(x)}{1!}\delta x + \frac{U''(x)}{2!}\delta x^2 + \frac{U'''(x)}{3!}\delta x^3 + O(\delta x^4)$$

$$U(x-\delta x) = U(x) - \frac{U'(x)}{1!}\delta x + \frac{U''(x)}{2!}\delta x^2 - \frac{U'''(x)}{3!}\delta x^3 + O(\delta x^4)$$

辺どうし足す

$$U(x+\delta x) + U(x-\delta x) = 2U(x) + U''(x)\delta x^2 + O(\delta x^4)$$

$$U''(x) = \frac{U(x-\delta x) - 2U(x) + U(x+\delta x)}{\delta x^2} + O(\delta x^2)$$

拡散方程式の差分化

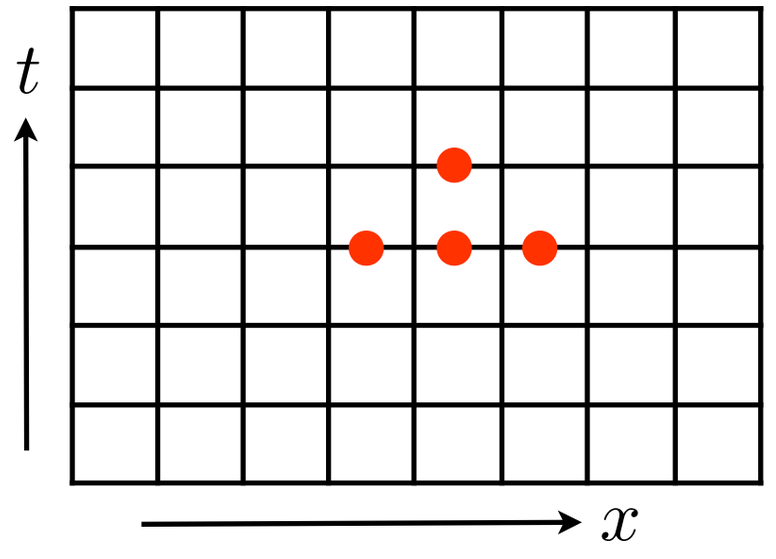
$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$$

時間微分は前進差分で,
空間微分は中心２階差分で

$$\frac{u(x,t+\delta t)-u(x,t)}{\delta t} \simeq D\frac{u(x-\delta x,t)-2u(x,t)+u(x+\delta x,t)}{\delta x^2}$$

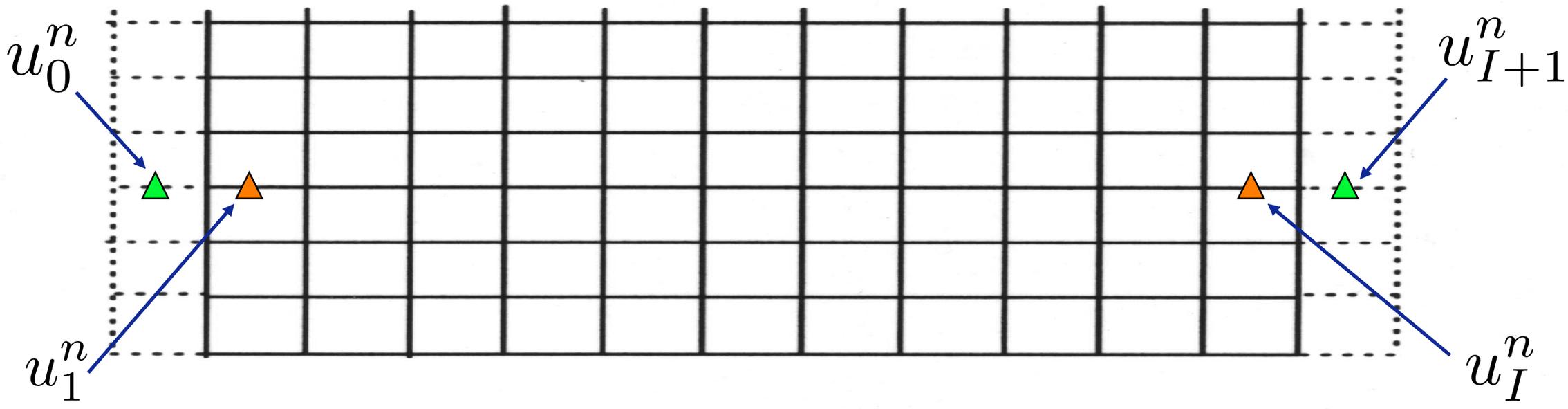
$$\frac{u_i^{n+1}-u_i^n}{\delta t} = D\frac{u_{i-1}^n-2u_i^n+u_{i+1}^n}{\delta x^2}$$

$$(i=1,2,\cdots,I;\quad n=0,1,\cdots)$$



境界条件の処理

差分のcellを両側に1つずつ広げて，そこでの値を u_0^n, u_{I+1}^n とおく．



- ディリクレ条件

$$\frac{u_0^n + u_1^n}{2} \simeq u(0, t_n) = 0, \quad \frac{u_I^n + u_{I+1}^n}{2} \simeq u(\ell, t_n) = 0$$

$$\rightarrow \quad u_0^n = -u_1^n, \; u_{I+1}^n = -u_I^n$$

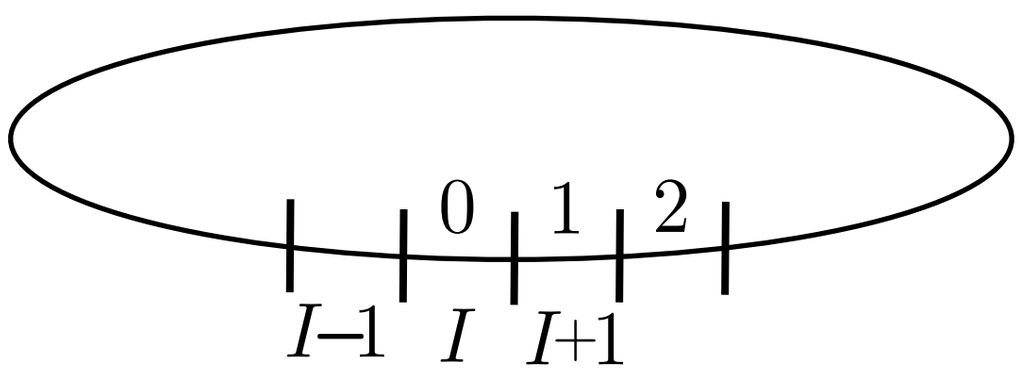
境界条件の処理

- ノイマン条件

$$\frac{u_1^n - u_0^n}{\delta x} \simeq \frac{\partial u}{\partial x}(0, t_n) = 0, \quad \frac{u_{I+1}^n - u_I^n}{\delta x} \simeq \frac{\partial u}{\partial x}(\ell, t_n) = 0$$

$$\rightarrow \quad u_0^n = u_1^n, \; u_{I+1}^n = u_I^n$$

- 周期境界条件



$$\rightarrow \quad u_0^n = u_I^n, \; u_{I+1}^n = u_1^n$$

初期値境界値問題の計算スキーム

拡散方程式

$$\frac{u_i^{n+1} - u_i^n}{\delta t} = D \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\delta x^2}$$

$$(i = 1, 2, \cdots, I; \quad n = 0, 1, \cdots)$$

境界条件

ディリクレ条件　$u_0^n = -u_1^n,\ u_{I+1}^n = -u_I^n$

ノイマン条件　$u_0^n = u_1^n,\ u_{I+1}^n = u_I^n$　$(n = 0, 1, \cdots)$

周期境界条件　$u_0^n = u_I^n,\ u_{I+1}^n = u_1^n$

初期条件

$$u_i^0 = f(x_i) \qquad (i = 1, 2, \cdots, I)$$

初期値境界値問題の計算スキーム

拡散方程式

$$\frac{u_i^{n+1} - u_i^n}{\delta t} = D \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\delta x^2}$$

$$(i = 1, 2, \cdots, I; \quad n = 0, 1, \cdots)$$

境界条件

ディリクレ条件　$u_0^n = -u_1^n,\ u_{I+1}^n = -u_I^n$

ノイマン条件　$u_0^n = u_1^n,\ u_{I+1}^n = u_I^n$　$(n = 0, 1, \cdots)$

周期境界条件　$u_0^n = u_I^n,\ u_{I+1}^n = u_1^n$

初期条件

$$u_i^0 = f(x_i) \qquad (i = 1, 2, \cdots, I)$$

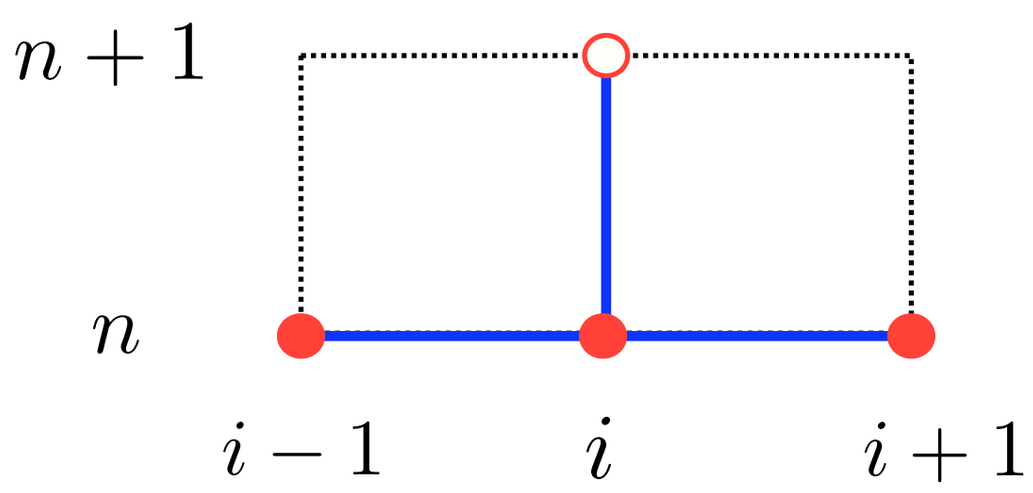
新しいステップの計算

$$\frac{u_i^{n+1} - u_i^n}{\delta t} = D \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\delta x^2}$$

$\lambda = \dfrac{D\delta t}{\delta x^2}$ とおくと

$$u_i^{n+1} = u_i^n + \lambda(u_{i-1}^n - 2u_i^n + u_{i+1}^n)$$

→ 陽解法(Explicit scheme)



初期値境界値問題の数値計算

6_Diffuse1Dim.cをDLして開こう.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define     Imax        (100)
#define     PI          (3.14159265358979)
#define     L           (2.0 * PI)
#define     INTV        (10)

double f(double x)
{
    double ans;
    ans = cos(3 * x);
    //ans = rand() / ((double)RAND_MAX);
    return ans;
}
int main(void)
{
    int i,i_time;
    FILE *gp;
    gp = popen("gnuplot -persist","w");
    //fprintf(gp,"set terminal x11\n");
    fprintf(gp,"set terminal aqua\n");
    fprintf(gp,"set xrange[0:%f]\n",L);
    fprintf(gp,"set yrange[-2:2]\n");
    double  u[Imax + 2];
    double  u_New[Imax + 2];
    double x,y;
    double dx = L / Imax;
    double dt = 0.0001;
    double D = 0.01;
    double lambda = D * dt / (dx * dx);

    for(i = 1;i <= Imax;i ++)
    {
        x = (i - 0.5) * dx;
        u[i] = f(x);
    }
```

```c
for(i_time = 0;;i_time ++)
    {
        if(1)
        {
            double sum = 0.0;
            for(i = 1;i <= Imax;i ++)
            {
                sum += u[i] * dx;
            }
            printf("%15.15f\n",sum);
        }
         //B.C.
         u[0] = u[1];
         u[Imax + 1] = u[Imax];

         //plot
        if(i_time % INTV == 0)
        {
            fprintf(gp,"plot '-' with lines\n");
            for(i = 1;i <= Imax;i ++)
            {
                x = (i - 0.5) * dx;
                y = u[i];
                fprintf(gp,"%f %f\n",x,y);
            }
            fprintf(gp,"e\n");
            fflush(gp);
        }
         //Calc Eq
         for(i = 1;i <= Imax;i ++)
         {
             u_New[i] = u[i] + lambda * (u[i - 1] - 2 *
u[i] + u[i + 1]);
         }

         //Subs
         for(i = 1;i <= Imax;i ++)
         {
             u[i] = u_New[i];
         }
```

初期値境界値問題の数値計算

6_Diffuse1Dim.cをDLして開こう.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define     Imax        (100)
#define     PI          (3.14159265358979)
#define     L           (2.0 * PI)
#define     INTV        (10)

double f(double x)
{
    double ans;
    ans = cos(3 * x);
    //ans = rand() / ((double)RAND_MAX);
    return ans;
}
int main(void)
{
    int i,i_time;
    FILE *gp;
    gp = popen("gnuplot -persist","w");
    //fprintf(gp,"set terminal x11\n");
    fprintf(gp,"set terminal aqua\n");
    fprintf(gp,"set xrange[0:%f]\n",L);
    fprintf(gp,"set yrange[-2:2]\n");
    double  u[Imax + 2];
    double  u_New[Imax + 2];
    double x,y;
    double dx = L / Imax;
    double dt = 0.0001;
    double D = 0.01;
    double lambda = D * dt / (dx * dx);

    for(i = 1;i <= Imax;i ++)
    {
        x = (i - 0.5) * dx;
        u[i] = f(x);
    }
```

Imax　箱の数

L 領域の長さ

INTV 画面表示の間引き間隔

```c
for(i_time = 0;;i_time ++)
    {
        if(1)
        {
            fprintf(gp,"plot '-' with lines\n");
            for(i = 1;i <= Imax;i ++)
            {
                x = (i - 0.5) * dx;
                y = u[i];
                fprintf(gp,"%f %f\n",x,y);
            }
            fprintf(gp,"e\n");
            fflush(gp);
        }
        //Calc Eq
        for(i = 1;i <= Imax;i ++)
        {
            u_New[i] = u[i] + lambda * (u[i - 1] - 2 *
u[i] + u[i + 1]);
        }

        //Subs
        for(i = 1;i <= Imax;i ++)
        {
            u[i] = u_New[i];
        }
```

初期値境界値問題の数値計算

6_Diffuse1Dim.cをDLして開こう.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define     Imax        (100)
#define     PI          (3.14159265358979)
#define     L           (2.0 * PI)
#define     INTV        (10)

double f(double x)
{
    double ans;
    ans = cos(3 * x);
    //ans = rand() / ((double)RAND_MAX);
    return ans;
}
int main(void)
{
    int i,i_time;
    FILE *gp;
    gp = popen("gnuplot -persist","w");
    //fprintf(gp,"set terminal x11\n");
    fprintf(gp,"set terminal aqua\n");
    fprintf(gp,"set xrange[0:%f]\n",L);
    fprintf(gp,"set yrange[-2:2]\n");
    double  u[Imax + 2];
    double  u_New[Imax + 2];
    double x,y;
    double dx = L / Imax;
    double dt = 0.0001;
    double D = 0.01;
    double lambda = D * dt / (dx * dx);

    for(i = 1;i <= Imax;i ++)
    {
        x = (i - 0.5) * dx;
        u[i] = f(x);
    }
```

uの初期の形を決める関数

```c
for(i_time = 0;;i_time ++)
    {
        if(1)
        {
            fprintf(gp,"plot '-' with lines\n");
            for(i = 1;i <= Imax;i ++)
            {
                x = (i - 0.5) * dx;
                y = u[i];
                fprintf(gp,"%f %f\n",x,y);
            }
            fprintf(gp,"e\n");
            fflush(gp);
        }
        //Calc Eq
        for(i = 1;i <= Imax;i ++)
        {
            u_New[i] = u[i] + lambda * (u[i - 1] - 2 *
u[i] + u[i + 1]);
        }

        //Subs
        for(i = 1;i <= Imax;i ++)
        {
            u[i] = u_New[i];
        }
```

初期値境界値問題の数値計算

6_Diffuse1Dim.cをDLして開こう.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define     Imax        (100)
#define     PI          (3.14159265358979)
#define     L           (2.0 * PI)
#define     INTV        (10)

double f(double x)
{
    double ans;
    ans = cos(3 * x);
    //ans = rand() / ((double)RAND_MAX);
    return ans;
}
int main(void)
{
    int i,i_time;
    FILE *gp;
    gp = popen("gnuplot -persist","w");
    //fprintf(gp,"set terminal x11\n");
    fprintf(gp,"set terminal aqua\n");
    fprintf(gp,"set xrange[0:%f]\n",L);
    fprintf(gp,"set yrange[-2:2]\n");
    double  u[Imax + 2];
    double  u_New[Imax + 2];
    double x,y;
    double dx = L / Imax;
    double dt = 0.0001;
    double D = 0.01;
    double lambda = D * dt / (dx * dx);

    for(i = 1;i <= Imax;i ++)
    {
        x = (i - 0.5) * dx;
        u[i] = f(x);
    }
```

Gnuplotのいつものおまじない

```c
for(i_time = 0;;i_time ++)
    {
        if(1)
        {
            fprintf(gp,"plot '-' with lines\n");
            for(i = 1;i <= Imax;i ++)
            {
                x = (i - 0.5) * dx;
                y = u[i];
                fprintf(gp,"%f %f\n",x,y);
            }
            fprintf(gp,"e\n");
            fflush(gp);
        }
        //Calc Eq
        for(i = 1;i <= Imax;i ++)
        {
            u_New[i] = u[i] + lambda * (u[i - 1] - 2 *
u[i] + u[i + 1]);
        }

        //Subs
        for(i = 1;i <= Imax;i ++)
        {
            u[i] = u_New[i];
        }
```

初期値境界値問題の数値計算

6_Diffuse1Dim.cをDLして開こう.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define     Imax        (100)
#define     PI          (3.14159265358979)
#define     L           (2.0 * PI)
#define     INTV        (10)

double f(double x)
{
    double ans;
    ans = cos(3 * x);
    //ans = rand() / ((double)RAND_MAX);
    return ans;
}
int main(void)
{
    int i,i_time;
    FILE *gp;
    gp = popen("gnuplot -persist","w");
    //fprintf(gp,"set terminal x11\n");
    fprintf(gp,"set terminal aqua\n");
    fprintf(gp,"set xrange[0:%f]\n",L);
    fprintf(gp,"set yrange[-2:2]\n");
    double  u[Imax + 2];
    double  u_New[Imax + 2];
    double x,y;
    double dx = L / Imax;
    double dt = 0.0001;
    double D = 0.01;
    double lambda = D * dt / (dx * dx);

    for(i = 1;i <= Imax;i ++)
    {
        x = (i - 0.5) * dx;
        u[i] = f(x);
    }
```

uは今の値を格納する.

u_Newは次の時刻の値を格納する.

サイズが+2多いのは境界条件を処理するため.

```c
for(i_time = 0;;i_time ++)
    {
        if(1)
        {
            fprintf(gp,"plot '-' with lines\n");
            for(i = 1;i <= Imax;i ++)
            {
                x = (i - 0.5) * dx;
                y = u[i];
                fprintf(gp,"%f %f\n",x,y);
            }
            fprintf(gp,"e\n");
            fflush(gp);
        }
        //Calc Eq
        for(i = 1;i <= Imax;i ++)
        {
            u_New[i] = u[i] + lambda * (u[i - 1] - 2 *
u[i] + u[i + 1]);
        }

        //Subs
        for(i = 1;i <= Imax;i ++)
        {
            u[i] = u_New[i];
        }
```

初期値境界値問題の数値計算

6_Diffuse1Dim.cをDLして開こう.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define     Imax        (100)
#define     PI          (3.14159265358979)
#define     L           (2.0 * PI)
#define     INTV        (10)

double f(double x)
{
    double ans;
    ans = cos(3 * x);
    //ans = rand() / ((double)RAND_MAX);
    return ans;
}
int main(void)
{
    int i,i_time;
    FILE *gp;
    gp = popen("gnuplot -persist","w");
    //fprintf(gp,"set terminal x11\n");
    fprintf(gp,"set terminal aqua\n");
    fprintf(gp,"set xrange[0:%f]\n",L);
    fprintf(gp,"set yrange[-2:2]\n");
    double  u[Imax + 2];
    double  u_New[Imax + 2];
    double x,y;
    double dx = L / Imax;
    double dt = 0.0001;
    double D = 0.01;
    double lambda = D * dt / (dx * dx);

    for(i = 1;i <= Imax;i ++)
    {
        x = (i - 0.5) * dx;
        u[i] = f(x);
    }
```

空間刻みdxはLをImaxで割った数.

dtは時間刻みの値.

Dは拡散定数

```c
for(i_time = 0;;i_time ++)
    {
        if(1)
        {
            fprintf(gp,"plot '-' with lines\n");
            for(i = 1;i <= Imax;i ++)
            {
                x = (i - 0.5) * dx;
                y = u[i];
                fprintf(gp,"%f %f\n",x,y);
            }
            fprintf(gp,"e\n");
            fflush(gp);
        }
        //Calc Eq
        for(i = 1;i <= Imax;i ++)
        {
            u_New[i] = u[i] + lambda * (u[i - 1] - 2 *
u[i] + u[i + 1]);
        }

        //Subs
        for(i = 1;i <= Imax;i ++)
        {
            u[i] = u_New[i];
        }
```

初期値境界値問題の数値計算

6_Diffuse1Dim.cをDLして開こう.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define     Imax        (100)
#define     PI          (3.14159265358979)
#define     L           (2.0 * PI)
#define     INTV        (10)

double f(double x)
{
    double ans;
    ans = cos(3 * x);
    //ans = rand() / ((double)RAND_MAX);
    return ans;
}
int main(void)
{
    int i,i_time;
    FILE *gp;
    gp = popen("gnuplot -persist","w");
    //fprintf(gp,"set terminal x11\n");
    fprintf(gp,"set terminal aqua\n");
    fprintf(gp,"set xrange[0:%f]\n",L);
    fprintf(gp,"set yrange[-2:2]\n");
    double  u[Imax + 2];
    double  u_New[Imax + 2];
    double x,y;
    double dx = L / Imax;
    double dt = 0.0001;
    double D = 0.01;
    double lambda = D * dt / (dx * dx);

    for(i = 1;i <= Imax;i ++)
    {
        x = (i - 0.5) * dx;
        u[i] = f(x);
    }
```

時刻0のときの，uの形状を関数fから決めている.

```c
for(i_time = 0;;i_time ++)
    {
        if(1)
        {
            fprintf(gp,"plot '-' with lines\n");
            for(i = 1;i <= Imax;i ++)
            {
                x = (i - 0.5) * dx;
                y = u[i];
                fprintf(gp,"%f %f\n",x,y);
            }
            fprintf(gp,"e\n");
            fflush(gp);
        }
        //Calc Eq
        for(i = 1;i <= Imax;i ++)
        {
            u_New[i] = u[i] + lambda * (u[i - 1] - 2 *
u[i] + u[i + 1]);
        }

        //Subs
        for(i = 1;i <= Imax;i ++)
        {
            u[i] = u_New[i];
        }
```

初期値境界値問題の数値計算

6_Diffuse1Dim.cをDLして開こう.

今は気にしない

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

int main(void)
{
    int i,i_time;
    FILE *gp;
    gp = popen("gnuplot -persist","w");
    //fprintf(gp,"set terminal x11\n");
    fprintf(gp,"set terminal aqua\n");
    fprintf(gp,"set xrange[0:%f]\n",L);
    fprintf(gp,"set yrange[-2:2]\n");
    double  u[Imax + 2];
    double  u_New[Imax + 2];
    double x,y;
    double dx = L / Imax;
    double dt = 0.0001;
    double D = 0.01;
    double lambda = D * dt / (dx * dx);

    for(i = 1;i <= Imax;i ++)
    {
        x = (i - 0.5) * dx;
        u[i] = f(x);
    }
```

```c
for(i_time = 0;;i_time ++)
    {
        if(1)
        {
            double sum = 0.0;
            for(i = 1;i <= Imax;i ++)
            {
                sum += u[i] * dx;
            }
            printf("%15.15f\n",sum);
        }
        //B.C.
        u[0] = u[1];
        u[Imax + 1] = u[Imax];

        //plot
        if(i_time % INTV == 0)
        {
            fprintf(gp,"plot '-' with lines\n");
            for(i = 1;i <= Imax;i ++)
            {
                x = (i - 0.5) * dx;
                y = u[i];
                fprintf(gp,"%f %f\n",x,y);
            }
            fprintf(gp,"e\n");
            fflush(gp);
        }
        //Calc Eq
        for(i = 1;i <= Imax;i ++)
        {
            u_New[i] = u[i] + lambda * (u[i - 1] - 2 *
u[i] + u[i + 1]);
        }

        //Subs
        for(i = 1;i <= Imax;i ++)
        {
            u[i] = u_New[i];
        }
```

初期値境界値問題の数値計算

6_Diffuse1Dim.cをDLして開こう.

境界条件を処理するための記述.
ここでは，ノイマン０の境界条件を仮定している.

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

int main(void)
{
    int i,i_time;
    FILE *gp;
    gp = popen("gnuplot -persist","w");
    //fprintf(gp,"set terminal x11\n");
    fprintf(gp,"set terminal aqua\n");
    fprintf(gp,"set xrange[0:%f]\n",L);
    fprintf(gp,"set yrange[-2:2]\n");
    double  u[Imax + 2];
    double  u_New[Imax + 2];
    double x,y;
    double dx = L / Imax;
    double dt = 0.0001;
    double D = 0.01;
    double lambda = D * dt / (dx * dx);

    for(i = 1;i <= Imax;i ++)
    {
        x = (i - 0.5) * dx;
        u[i] = f(x);
    }
```

```c
for(i_time = 0;;i_time ++)
    {
        if(1)
        {
            double sum = 0.0;
            for(i = 1;i <= Imax;i ++)
            {
                sum += u[i] * dx;
            }
            printf("%15.15f\n",sum);
        }
        //B.C.
        u[0] = u[1];
        u[Imax + 1] = u[Imax];

        //plot
        if(i_time % INTV == 0)
        {
            fprintf(gp,"plot '-' with lines\n");
            for(i = 1;i <= Imax;i ++)
            {
                x = (i - 0.5) * dx;
                y = u[i];
                fprintf(gp,"%f %f\n",x,y);
            }
            fprintf(gp,"e\n");
            fflush(gp);
        }
        //Calc Eq
        for(i = 1;i <= Imax;i ++)
        {
            u_New[i] = u[i] + lambda * (u[i - 1] - 2 *
u[i] + u[i + 1]);
        }

        //Subs
        for(i = 1;i <= Imax;i ++)
        {
            u[i] = u_New[i];
        }
```

初期値境界値問題の数値計算

6_Diffuse1Dim.cをDLして開こう.

i_time をINTVで割った余りが０の時
（つまりi_timeがINTVの倍数の時）
関数uの形状をgnuplotで可視化している.

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

int main(void)
{
    int i,i_time;
    FILE *gp;
    gp = popen("gnuplot -persist","w");
    //fprintf(gp,"set terminal x11\n");
    fprintf(gp,"set terminal aqua\n");
    fprintf(gp,"set xrange[0:%f]\n",L);
    fprintf(gp,"set yrange[-2:2]\n");
    double  u[Imax + 2];
    double  u_New[Imax + 2];
    double x,y;
    double dx = L / Imax;
    double dt = 0.0001;
    double D = 0.01;
    double lambda = D * dt / (dx * dx);

    for(i = 1;i <= Imax;i ++)
    {
        x = (i - 0.5) * dx;
        u[i] = f(x);
    }
```

```c
for(i_time = 0;;i_time ++)
    {
        if(1)
        {
            double sum = 0.0;
            for(i = 1;i <= Imax;i ++)
            {
                sum += u[i] * dx;
            }
            printf("%15.15f\n",sum);
        }
         //B.C.
         u[0] = u[1];
         u[Imax + 1] = u[Imax];

         //plot
        if(i_time % INTV == 0)
        {
            fprintf(gp,"plot '-' with lines\n")
            for(i = 1;i <= Imax;i ++)
            {
                x = (i - 0.5) * dx;
                y = u[i];
                fprintf(gp,"%f %f\n",x,y);
            }
            fprintf(gp,"e\n");
            fflush(gp);
        }
         //Calc Eq
         for(i = 1;i <= Imax;i ++)
         {
             u_New[i] = u[i] + lambda * (u[i - 1] - 2 *
u[i] + u[i + 1]);
         }

         //Subs
         for(i = 1;i <= Imax;i ++)
         {
             u[i] = u_New[i];
         }
```

初期値境界値問題の数値計算

6_Diffuse1Dim.cをDLして開こう.

スキームに従って，次の時刻の
u_Newの値を計算している.
ここがミソ.

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

int main(void)
{
    int i,i_time;
    FILE *gp;
    gp = popen("gnuplot -persist","w");
    //fprintf(gp,"set terminal x11\n");
    fprintf(gp,"set terminal aqua\n");
    fprintf(gp,"set xrange[0:%f]\n",L);
    fprintf(gp,"set yrange[-2:2]\n");
    double  u[Imax + 2];
    double  u_New[Imax + 2];
    double x,y;
    double dx = L / Imax;
    double dt = 0.0001;
    double D = 0.01;
    double lambda = D * dt / (dx * dx);

    for(i = 1;i <= Imax;i ++)
    {
        x = (i - 0.5) * dx;
        u[i] = f(x);
    }
```

```c
for(i_time = 0;;i_time ++)
    {
        if(1)
        {
            double sum = 0.0;
            for(i = 1;i <= Imax;i ++)
            {
                sum += u[i] * dx;
            }
            printf("%15.15f\n",sum);
        }
         //B.C.
         u[0] = u[1];
         u[Imax + 1] = u[Imax];

         //plot
        if(i_time % INTV == 0)
        {
            fprintf(gp,"plot '-' with lines\n");
            for(i = 1;i <= Imax;i ++)
            {
                x = (i - 0.5) * dx;
                y = u[i];
                fprintf(gp,"%f %f\n",x,y);
            }
            fprintf(gp,"e\n");
            fflush(gp);
        }
         //Calc Eq
         for(i = 1;i <= Imax;i ++)
         {
             u_New[i] = u[i] + lambda * (u[i - 1] - 2 *
u[i] + u[i + 1]);
         }

         //Subs
         for(i = 1;i <= Imax;i ++)
         {
             u[i] = u_New[i];
         }
```

初期値境界値問題の数値計算

6_Diffuse1Dim.cをDLして開こう.

次の時間ループの計算に備えて,
u_Newの値をuにコピーしている.

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

int main(void)
{
    int i,i_time;
    FILE *gp;
    gp = popen("gnuplot -persist","w");
    //fprintf(gp,"set terminal x11\n");
    fprintf(gp,"set terminal aqua\n");
    fprintf(gp,"set xrange[0:%f]\n",L);
    fprintf(gp,"set yrange[-2:2]\n");
    double  u[Imax + 2];
    double  u_New[Imax + 2];
    double x,y;
    double dx = L / Imax;
    double dt = 0.0001;
    double D = 0.01;
    double lambda = D * dt / (dx * dx);

    for(i = 1;i <= Imax;i ++)
    {
        x = (i - 0.5) * dx;
        u[i] = f(x);
    }
```

```c
for(i_time = 0;;i_time ++)
    {
        if(1)
        {
            double sum = 0.0;
            for(i = 1;i <= Imax;i ++)
            {
                sum += u[i] * dx;
            }
            printf("%15.15f\n",sum);
        }
         //B.C.
         u[0] = u[1];
         u[Imax + 1] = u[Imax];

         //plot
        if(i_time % INTV == 0)
        {
            fprintf(gp,"plot '-' with lines\n");
            for(i = 1;i <= Imax;i ++)
            {
                x = (i - 0.5) * dx;
                y = u[i];
                fprintf(gp,"%f %f\n",x,y);
            }
            fprintf(gp,"e\n");
            fflush(gp);
        }
         //Calc Eq
         for(i = 1;i <= Imax;i ++)
         {
             u_New[i] = u[i] + lambda * (u[i - 1] - 2 *
u[i] + u[i + 1]);
         }

         //Subs
         for(i = 1;i <= Imax;i ++)
         {
             u[i] = u_New[i];
         }
```

初期値境界値問題の数値計算

6_Diffuse1Dim.cをコンパイル＆実行しよう.

チャンレジ課題.

Lを変えてみる.

cosの3を5にしてみる.

dtを変えてみる.

Dを変えてみる.

if分の中身を変えてみる.

初期値境界値問題 (ノイマン条件)

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} \qquad \text{for } \ 0 < x < \ell,\ t > 0$$

$$\frac{\partial u}{\partial x}(0,t) = \frac{\partial u}{\partial x}(\ell,t) = 0 \quad \text{for } \ t > 0 \qquad \Rightarrow \text{B.C.}$$

$$u(x,0) = f(x) \qquad \text{for } \ 0 < x < \ell \qquad \Rightarrow \text{I.C.}$$

ノイマン境界条件は, 境界において物の出入りがないことを意味している.

$$u \ \text{の総量} \int_0^\ell u dx \ \text{は保存される.}$$

$$\frac{d}{dt}\int_0^\ell u dx = \int_0^\ell \frac{\partial u}{\partial t} dx = \int_0^\ell D\frac{\partial^2 u}{\partial x^2} dx = D\frac{\partial u}{\partial x}\Big|_{x=0}^{x=\ell} = 0$$

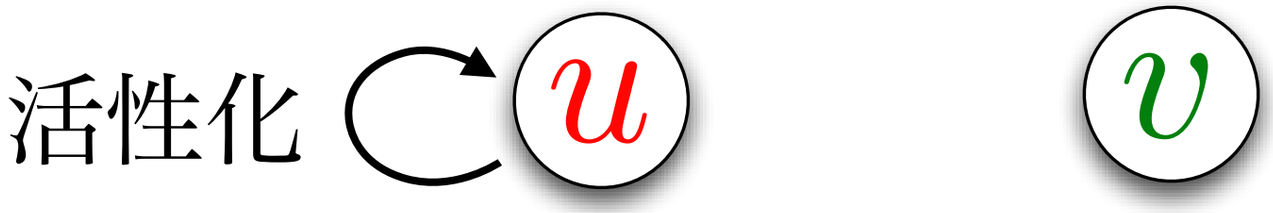
Turing Model

活性因子-抑制因子系

Activator-Inhibitor System

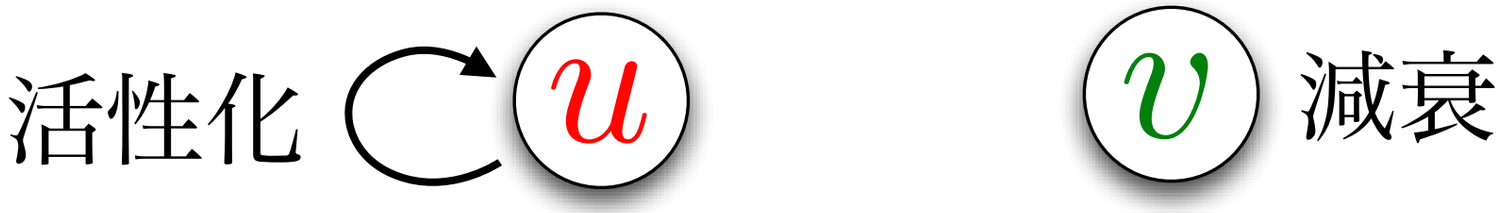
活性因子-抑制因子系

Activator-Inhibitor System



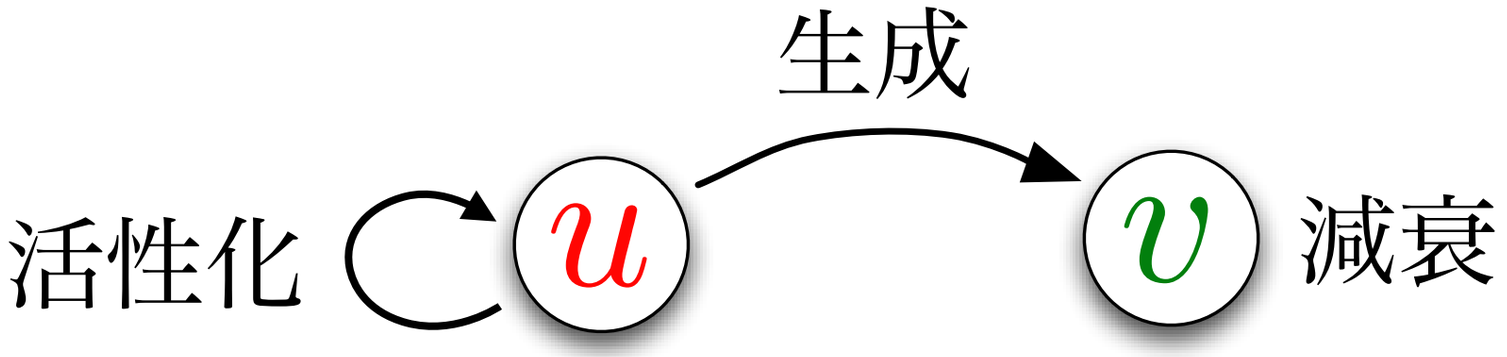
活性因子-抑制因子系

Activator-Inhibitor System



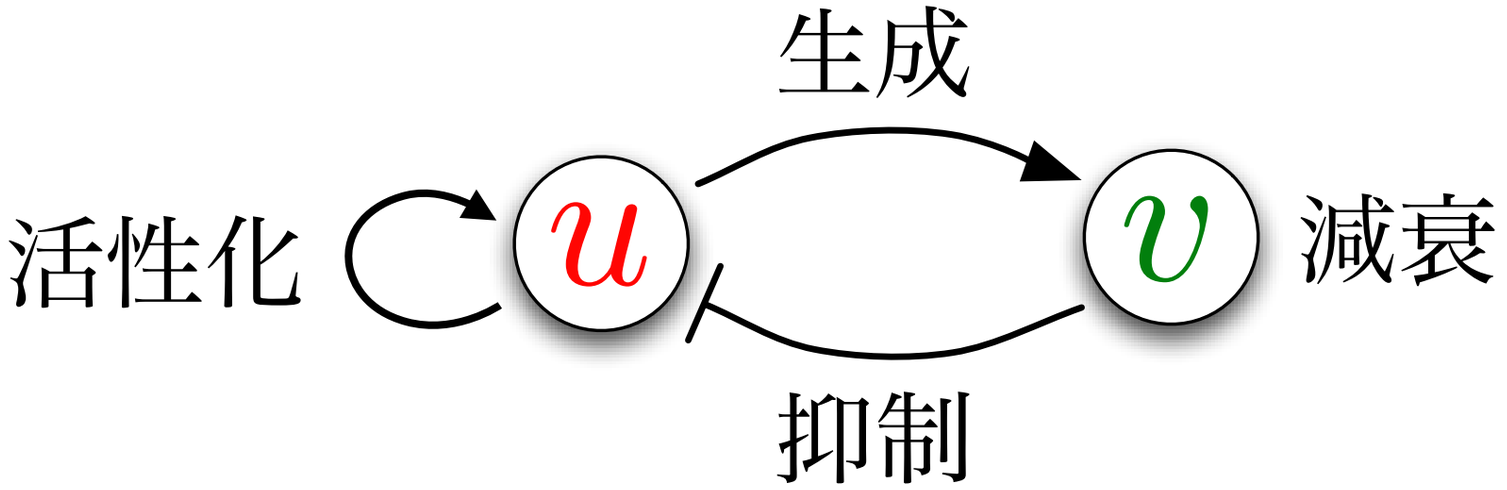
活性因子-抑制因子系

Activator-Inhibitor System



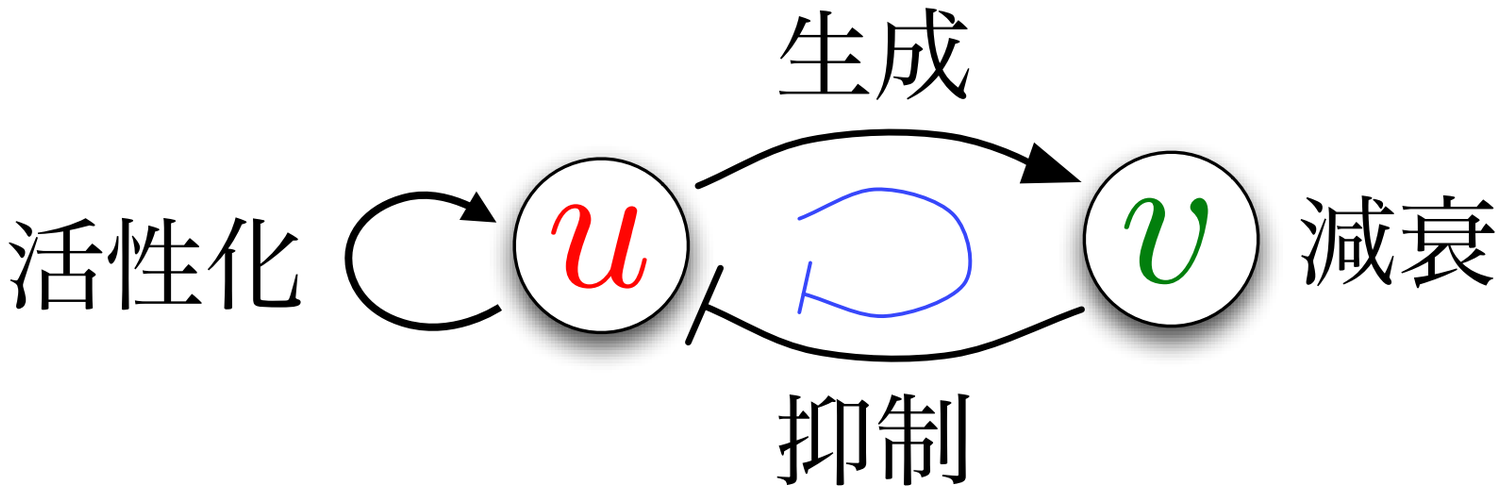
活性因子-抑制因子系

Activator-Inhibitor System



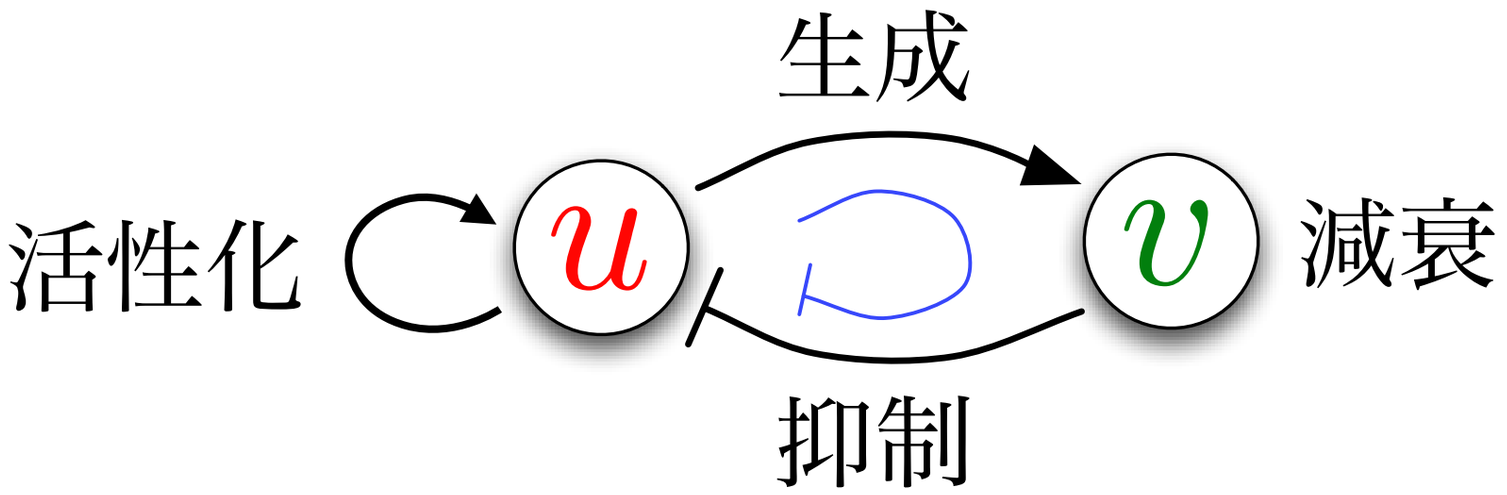
活性因子-抑制因子系

Activator-Inhibitor System



活性因子-抑制因子系

Activator-Inhibitor System

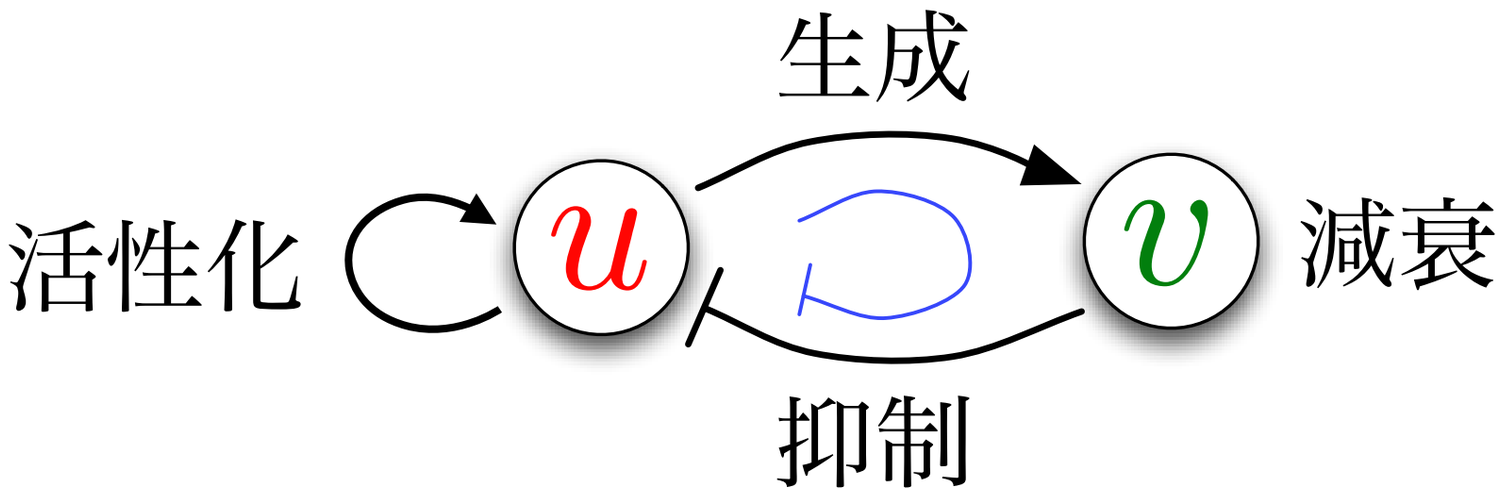


u: 活性因子

v: 抑制因子

活性因子-抑制因子系

Activator-Inhibitor System



u: 活性因子

v: 抑制因子

関係を表にすると

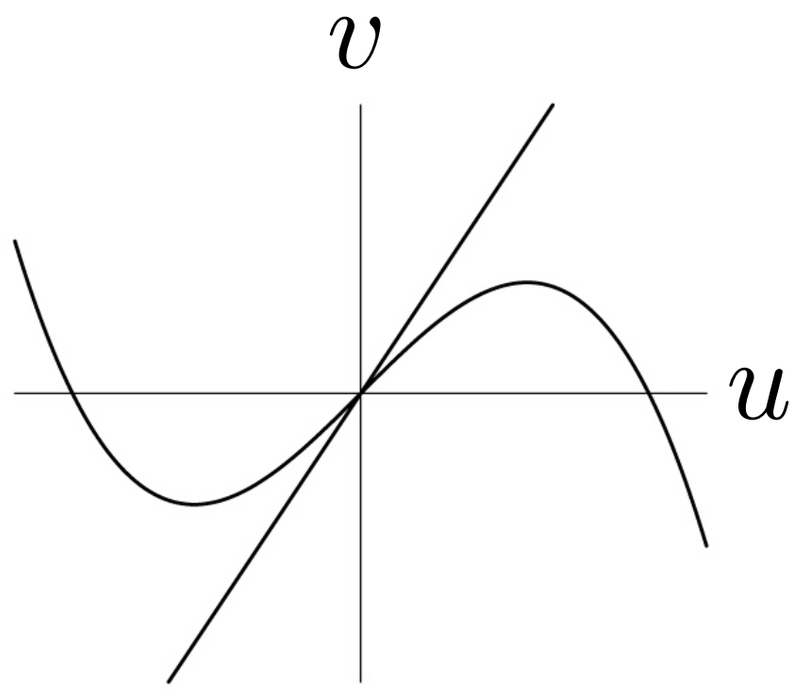
主語 / 目的語	u	v
u	＋	－
v	＋	－

$$\frac{du}{dt} = u(1 - u^2) - v$$

$$\frac{dv}{dt} = 3u - 2v$$

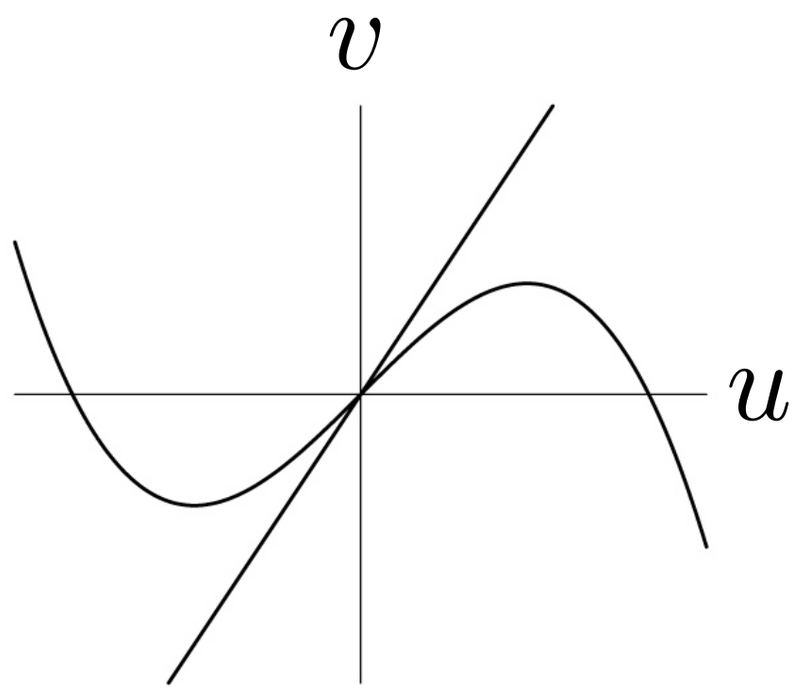
$$\frac{du}{dt} = u(1 - u^2) - v$$

$$\frac{dv}{dt} = 3u - 2v$$



$$\frac{du}{dt} = u(1-u^2) - v$$

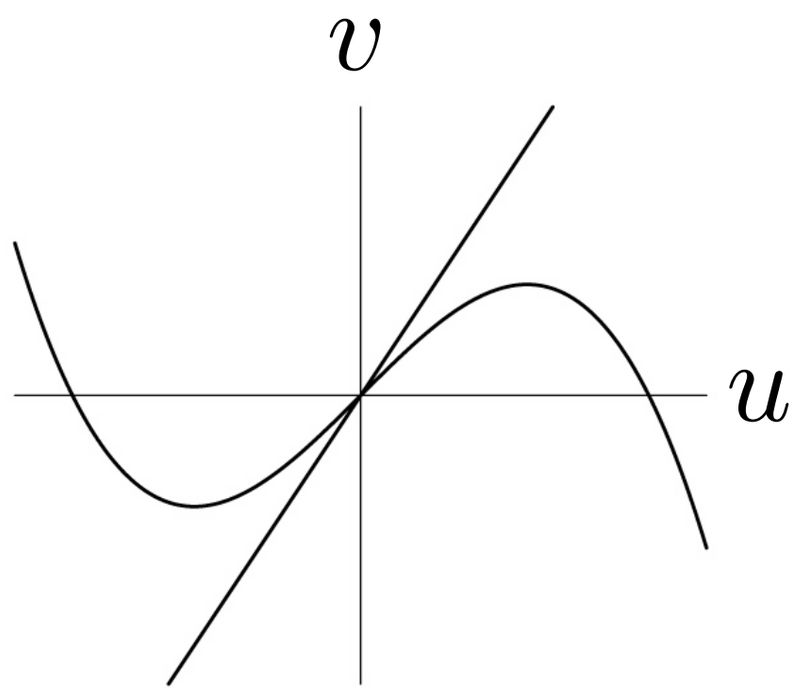
$$\frac{dv}{dt} = 3u - 2v$$



平衡点は $(0,0)$ のみ.
そこでのヤコビ行列を Aとすると,

$$\frac{du}{dt} = u(1-u^2) - v$$

$$\frac{dv}{dt} = 3u - 2v$$

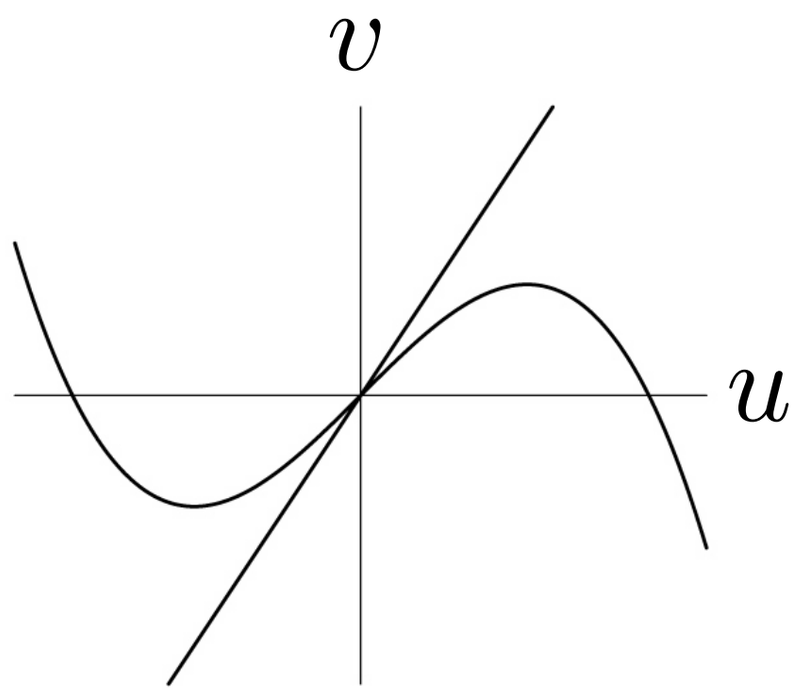


平衡点は $(0,0)$ のみ.
そこでのヤコビ行列を Aとすると,

$$A = \begin{bmatrix} 1 & -1 \\ 3 & -2 \end{bmatrix}$$

$$\frac{du}{dt} = u(1-u^2) - v$$

$$\frac{dv}{dt} = 3u - 2v$$



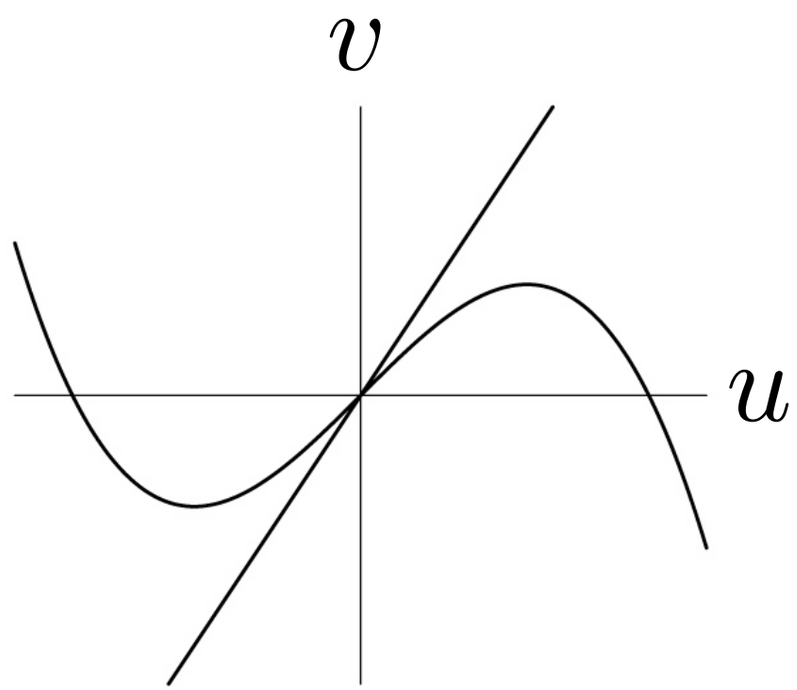
平衡点は $(0,0)$ のみ.
そこでのヤコビ行列を Aとすると,

$$A = \begin{bmatrix} 1 & -1 \\ 3 & -2 \end{bmatrix}$$

$$\mathrm{tr}A = -1 \qquad \det A = 1$$

$$\frac{du}{dt} = u(1-u^2) - v$$

$$\frac{dv}{dt} = 3u - 2v$$



平衡点は $(0,0)$ のみ.
そこでのヤコビ行列を Aとすると,

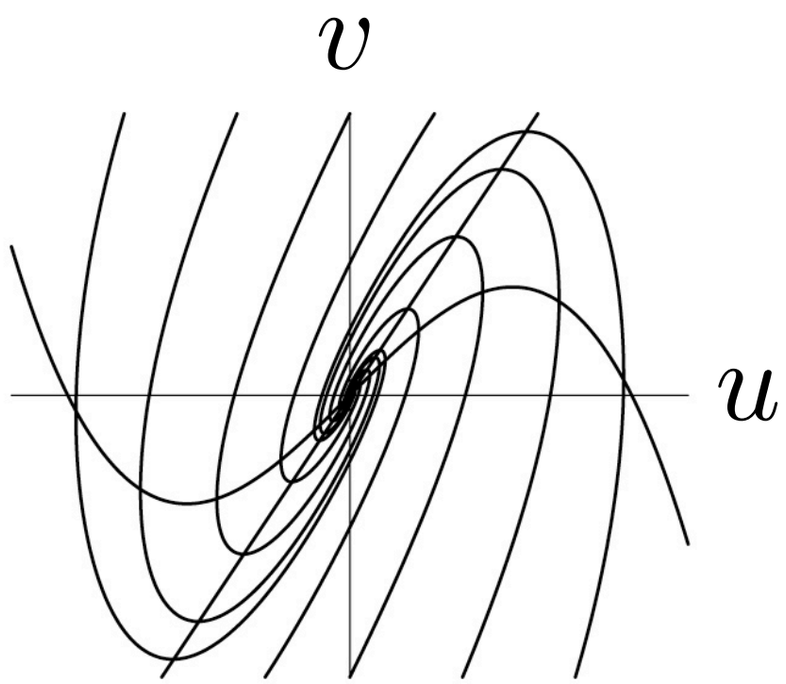
$$A = \begin{bmatrix} 1 & -1 \\ 3 & -2 \end{bmatrix}$$

$$\mathrm{tr}A = -1 \qquad \det A = 1$$

➡ $(0,0)$ は安定渦状点

$$\frac{du}{dt} = u(1-u^2) - v$$

$$\frac{dv}{dt} = 3u - 2v$$



平衡点は $(0,0)$ のみ.
そこでのヤコビ行列を Aとすると,

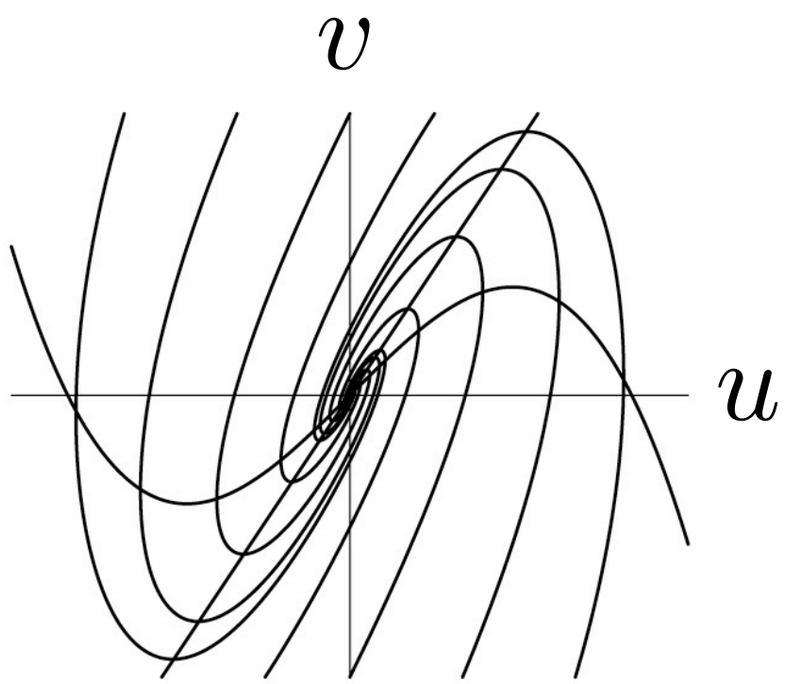
$$A = \begin{bmatrix} 1 & -1 \\ 3 & -2 \end{bmatrix}$$

$$\mathrm{tr}A = -1 \qquad \det A = 1$$

➡ $(0,0)$ は安定渦状点

$$\frac{du}{dt} = u(1-u^2) - v$$

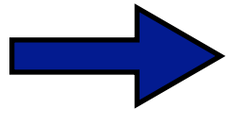
$$\frac{dv}{dt} = 3u - 2v$$



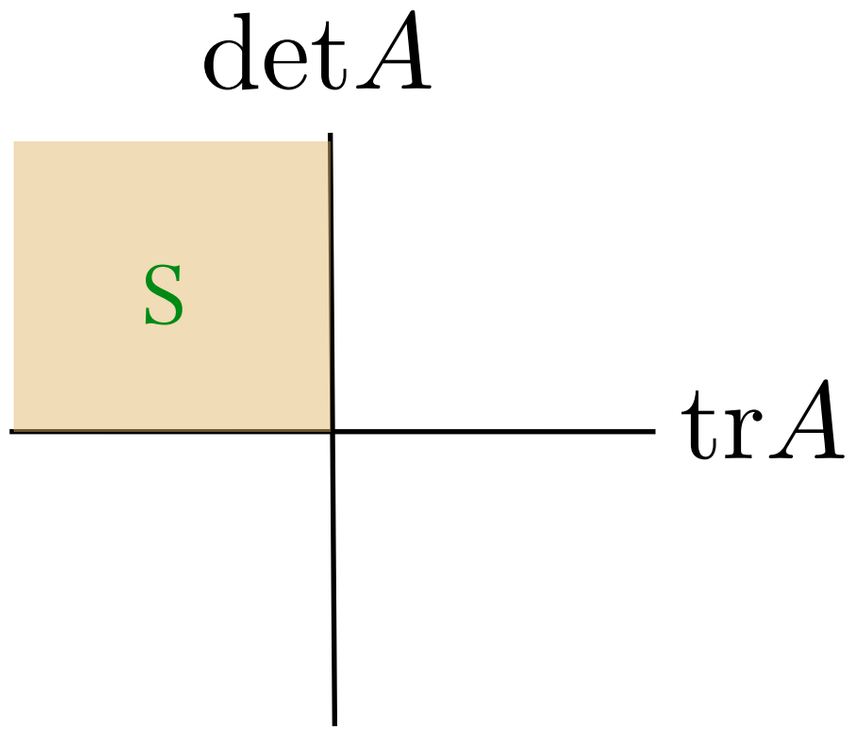
平衡点は $(0,0)$ のみ.
そこでのヤコビ行列を Aとすると,

$$A = \begin{bmatrix} 1 & -1 \\ 3 & -2 \end{bmatrix}$$

$\mathrm{tr}A = -1 \qquad \det A = 1$



$(0,0)$ は安定渦状点



反応拡散系へ

それぞれの因子が拡散すれば

$$\frac{\partial u}{\partial t} = D_u \frac{\partial^2 u}{\partial x^2} + u(1-u^2) - v$$

$$\frac{\partial v}{\partial t} = D_v \frac{\partial^2 v}{\partial x^2} + 3u - 2v \qquad \text{on } R$$

反応拡散系へ

それぞれの因子が拡散すれば

$$\frac{\partial u}{\partial t} = D_u \frac{\partial^2 u}{\partial x^2} + u(1-u^2) - v$$
$$\frac{\partial v}{\partial t} = D_v \frac{\partial^2 v}{\partial x^2} + 3u - 2v \qquad \text{on } R$$

問題： 定数定常解 $(u, v) \equiv (0, 0)$ は安定なのだろうか？

反応拡散系へ

それぞれの因子が拡散すれば

$$\frac{\partial u}{\partial t} = D_u \frac{\partial^2 u}{\partial x^2} + u(1-u^2) - v$$

$$\frac{\partial v}{\partial t} = D_v \frac{\partial^2 v}{\partial x^2} + 3u - 2v \qquad \text{on } R$$

問題： 定数定常解 $(u, v) \equiv (0, 0)$ は安定なのだろうか？

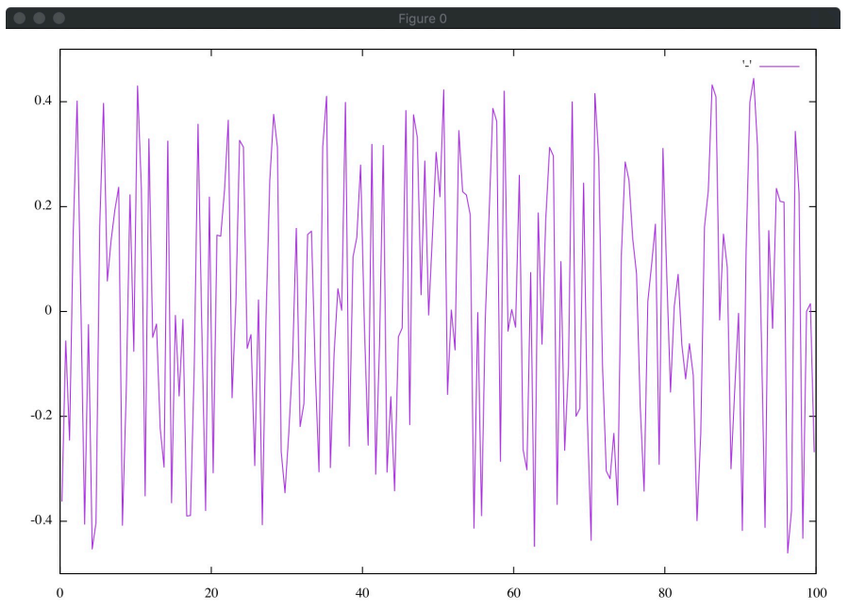
直感： $(0, 0)$ は非線形ダイナミクスのただ一つの安定解であるし、拡散は空間一様性を好むだろうから、安定に違いない！

数値計算で確かめて見よう！

$$\frac{\partial u}{\partial t} = D_u \frac{\partial^2 u}{\partial x^2} + u(1-u^2) - v$$

$$\frac{\partial v}{\partial t} = D_v \frac{\partial^2 v}{\partial x^2} + 3u - 2v$$

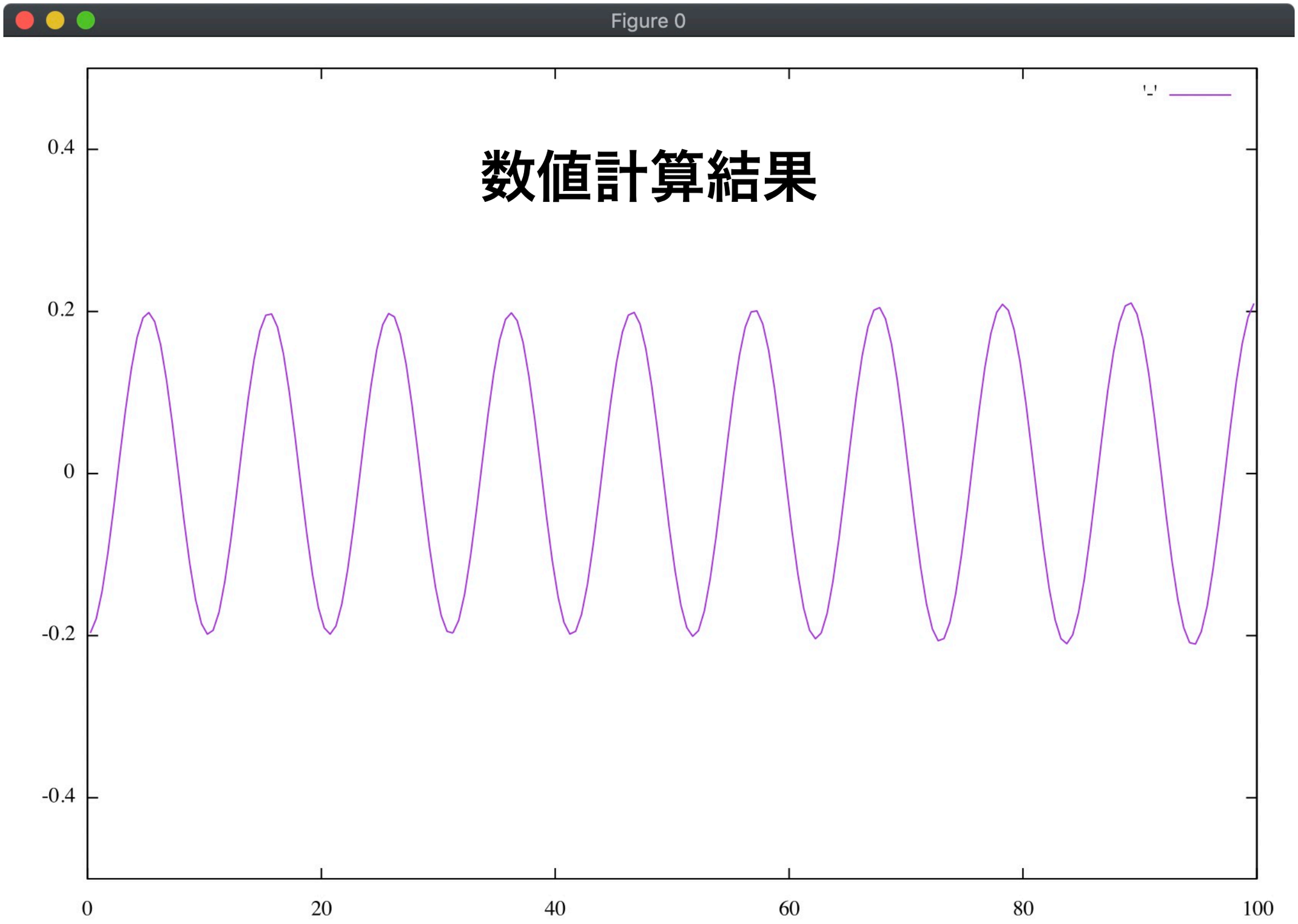
8_Turing1D.cは上記方程式の1次元のソルバー(境界条件はノイマン0)



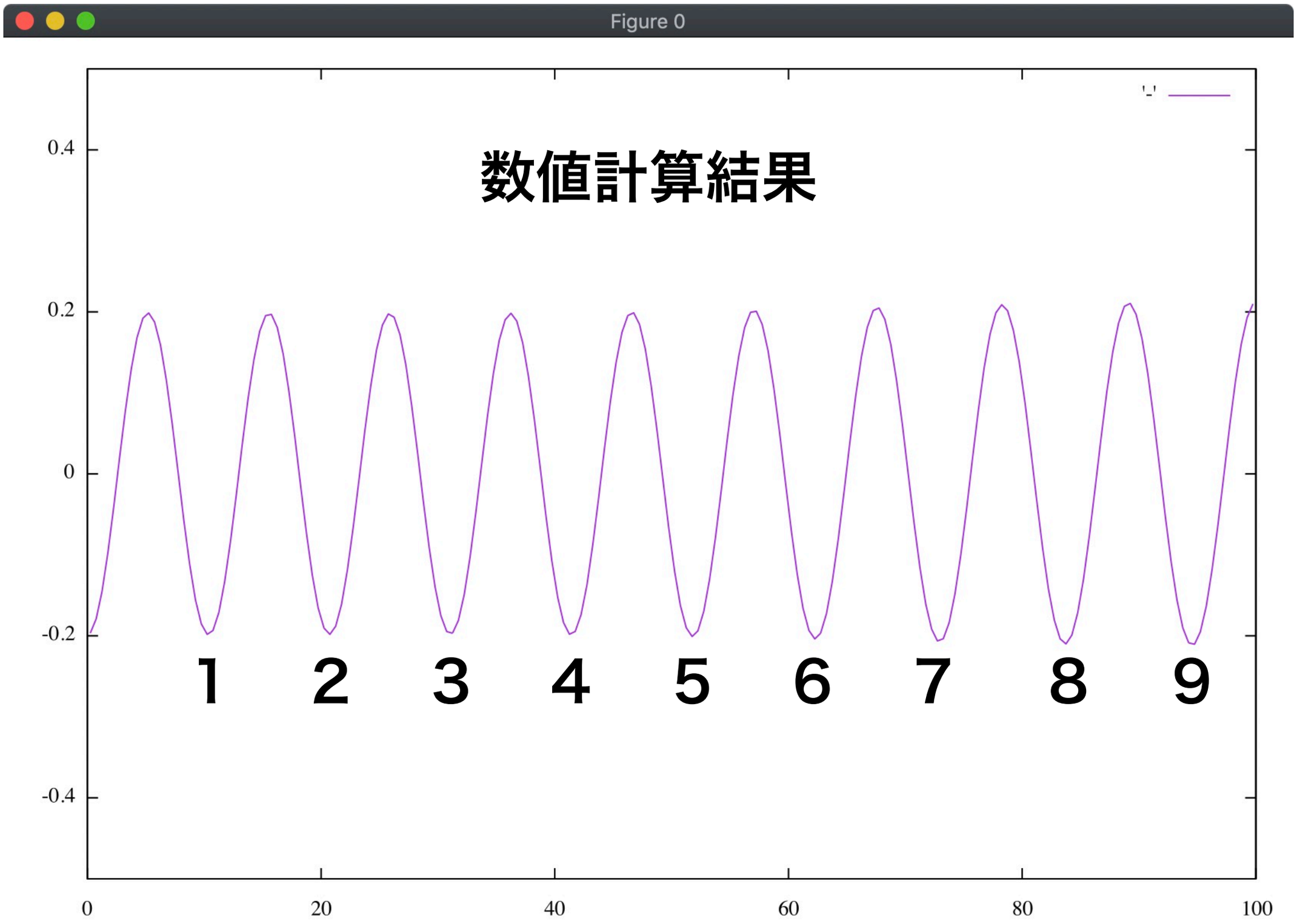
実験１：Du=Dv=1.0でパターンはどうなるかみよう！

実験２：Du=1.0, Dvを変化させてみよう！

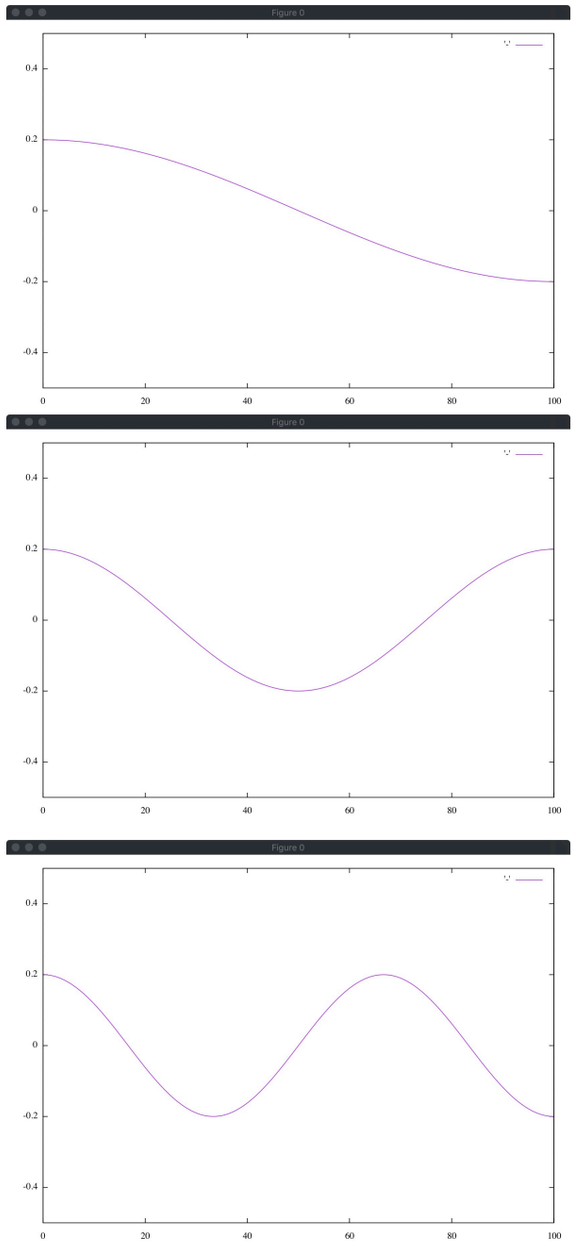
Du=1.0, Dv=10.0



Du=1.0, Dv=10.0



波の数は9.5くらい　なんでこうなるの？？



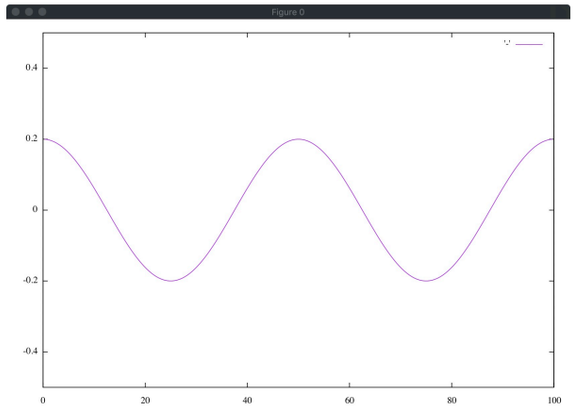
$$\cos(1 * \pi * x/L_x)$$

$$\cos(2 * \pi * x/L_x)$$

$$\cos(m * \pi * x/L_x)$$

mモード解

$$\cos(3 * \pi * x/L_x)$$



$$\cos(4 * \pi * x/L_x)$$

Lx

Lxを空間長とする.

線形化方程式導出

$$\begin{pmatrix} u(x,t) \\ v(x,t) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \delta \begin{pmatrix} \varphi(x,t) \\ \psi(x,t) \end{pmatrix} \quad |\delta| \ll 1$$

線形化方程式導出

$$\begin{pmatrix} u(x,t) \\ v(x,t) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \delta \begin{pmatrix} \varphi(x,t) \\ \psi(x,t) \end{pmatrix} \quad |\delta| \ll 1$$

$$\begin{cases} \dfrac{\partial \varphi}{\partial t} = D_u \dfrac{\partial^2 \varphi}{\partial x^2} + \varphi - \psi \\ \\ \dfrac{\partial \psi}{\partial t} = D_v \dfrac{\partial^2 \psi}{\partial x^2} + 3\varphi - 2\psi \end{cases}$$

$$\phi = e^{\lambda_k t}\cos(m * \pi * x/L_x)\phi_0$$
$$\psi = e^{\lambda_k t}\cos(m * \pi * x/L_x)\psi_0$$

mモード解を代入

(0,0)まわりの線形化方程式

行列Aの
固有値

行列A

$$k := m * \pi/L_x$$

行列の性質は固有値（と固有ベクトル）で決まる.

$$\phi = e^{\lambda_k t}\cos(m * \pi * x/L_x)\phi_0$$
$$\psi = e^{\lambda_k t}\cos(m * \pi * x/L_x)\psi_0$$

(0,0)まわりの線形化方程式

mモード解を代入

$$\lambda_k \begin{pmatrix} \varphi_0 \\ \psi_0 \end{pmatrix} = \begin{pmatrix} 1 - D_u k^2 & -1 \\ 3 & -2 - D_v k^2 \end{pmatrix} \begin{pmatrix} \varphi_0 \\ \psi_0 \end{pmatrix}$$

行列Aの固有値

行列A

$$k := m * \pi/L_x$$

行列の性質は固有値（と固有ベクトル）で決まる.

$$\begin{cases} \dfrac{\partial \varphi}{\partial t} = D_u \dfrac{\partial^2 \varphi}{\partial x^2} + \varphi - \psi \\ \dfrac{\partial \psi}{\partial t} = D_v \dfrac{\partial^2 \psi}{\partial x^2} + 3\varphi - 2\psi \end{cases}$$

(0,0)まわりの線形化方程式

$$\phi = e^{\lambda_k t} \cos(m * \pi * x / L_x)\phi_0$$
$$\psi = e^{\lambda_k t} \cos(m * \pi * x / L_x)\psi_0$$

mモード解を代入

$$\lambda_k \begin{pmatrix} \varphi_0 \\ \psi_0 \end{pmatrix} = \begin{pmatrix} 1 - D_u k^2 & -1 \\ 3 & -2 - D_v k^2 \end{pmatrix} \begin{pmatrix} \varphi_0 \\ \psi_0 \end{pmatrix}$$

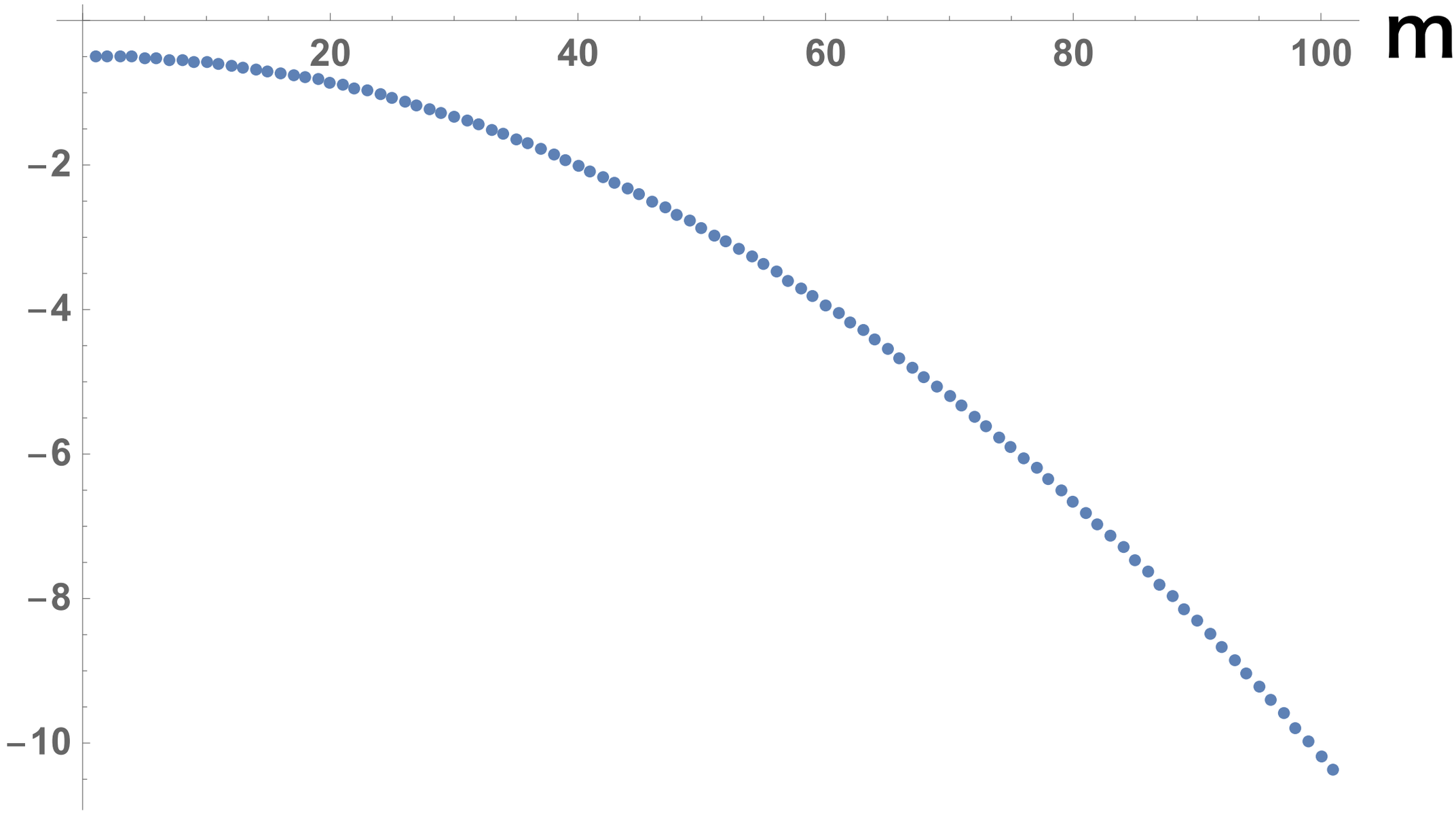
行列Aの固有値

行列A

$$k := m * \pi / L_x$$

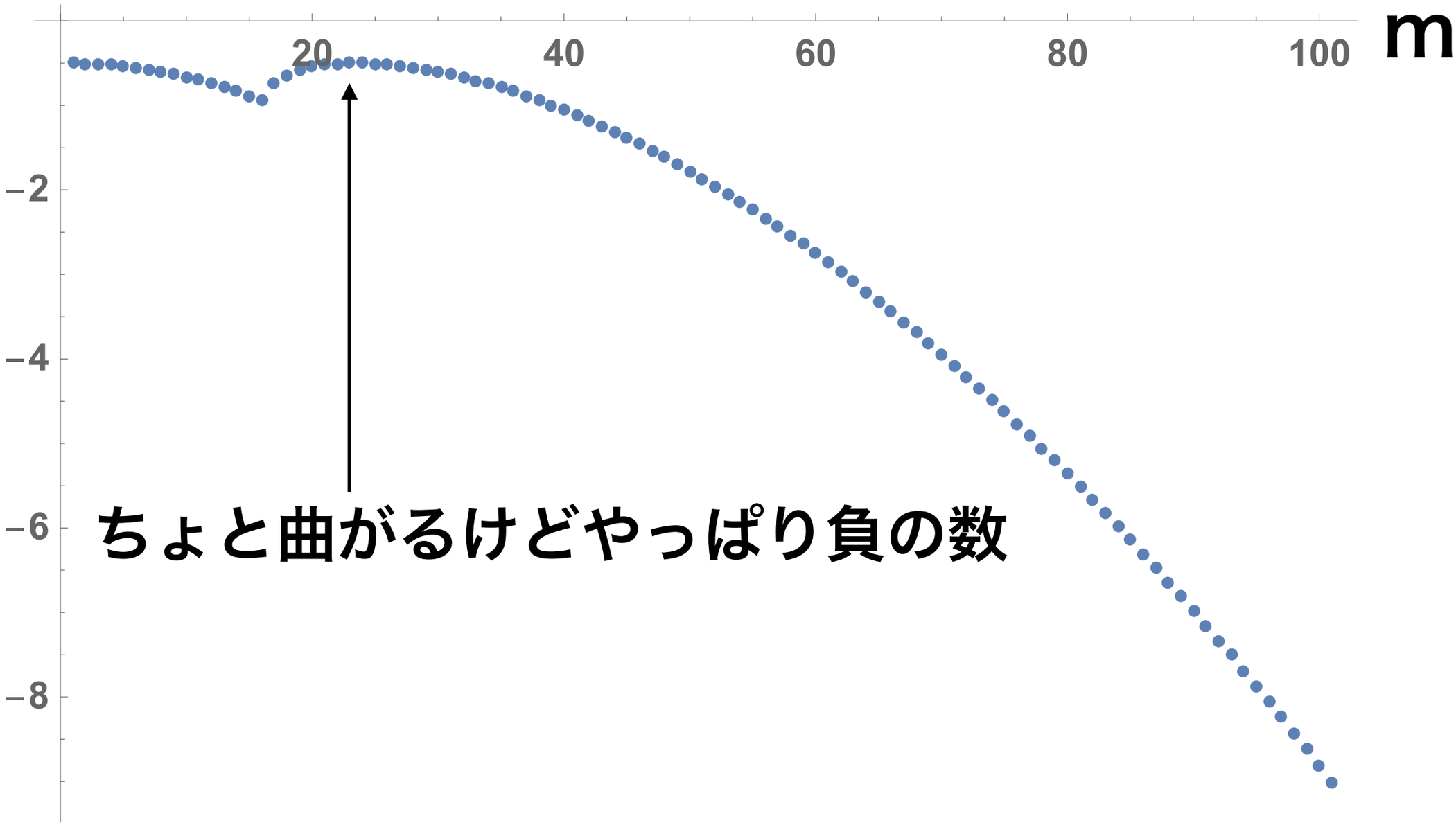
行列の性質は固有値（と固有ベクトル）で決まる.

固有値の実部が最大のもの



Du=1.0, Dv=1.0の時

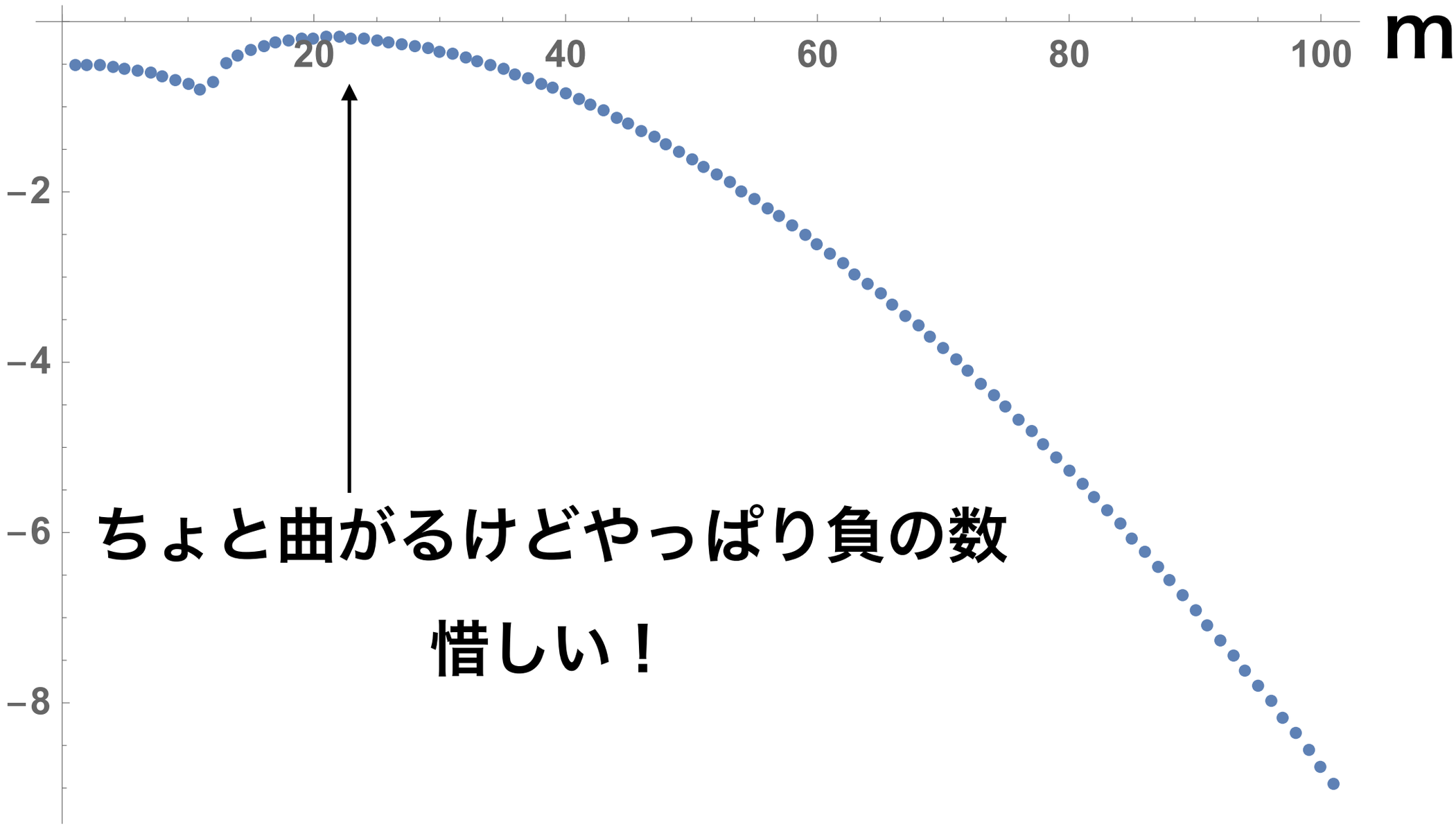
固有値の実部が最大のもの



ちょと曲がるけどやっぱり負の数

Du=1.0, Dv=3.0の時

固有値の実部が最大のもの

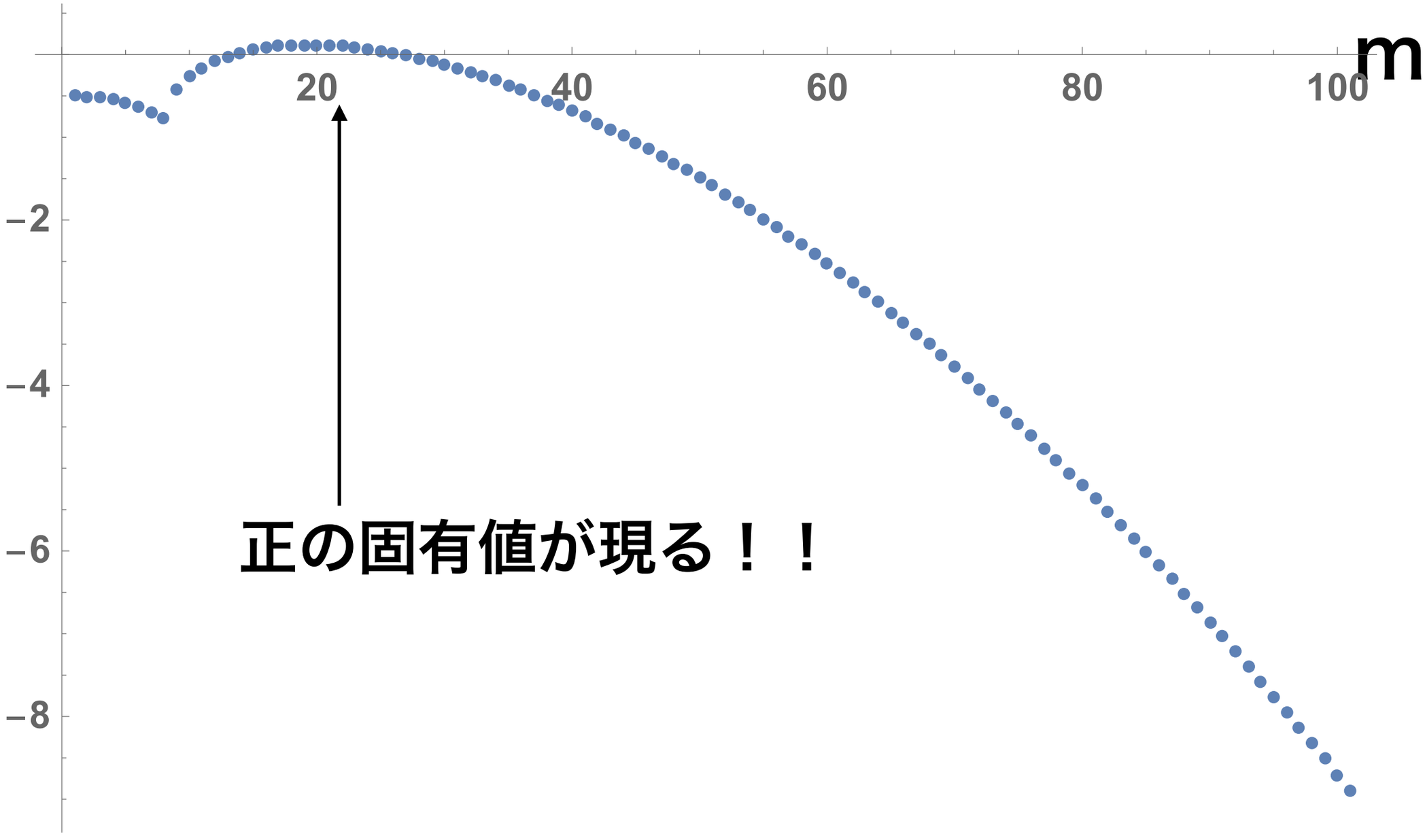


ちょと曲がるけどやっぱり負の数

惜しい！

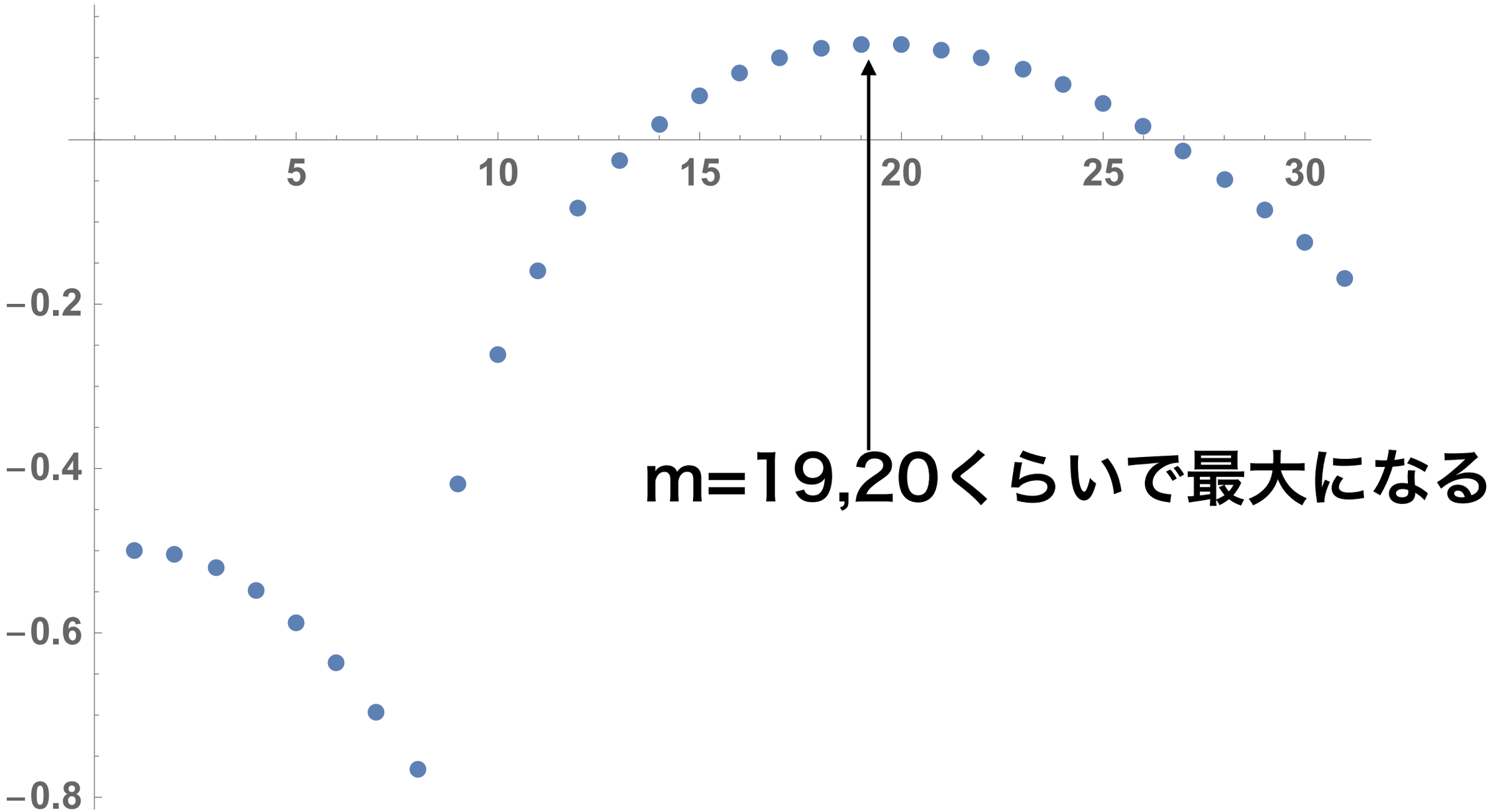
Du=1.0, Dv=5.0の時

固有値の実部が最大のもの

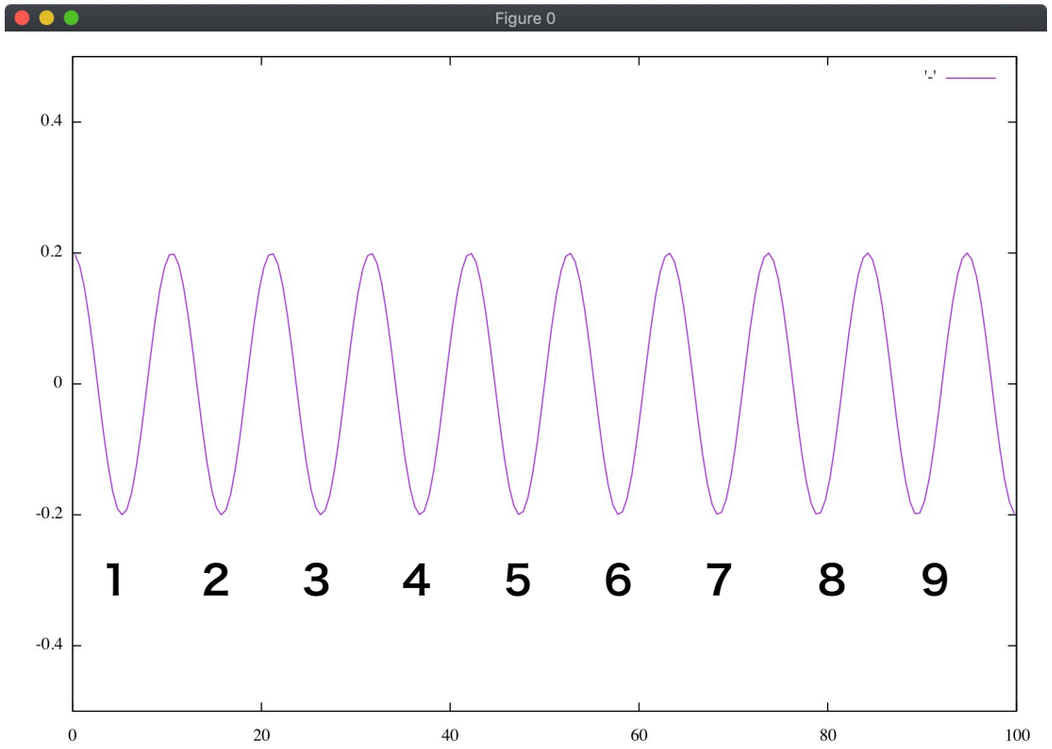


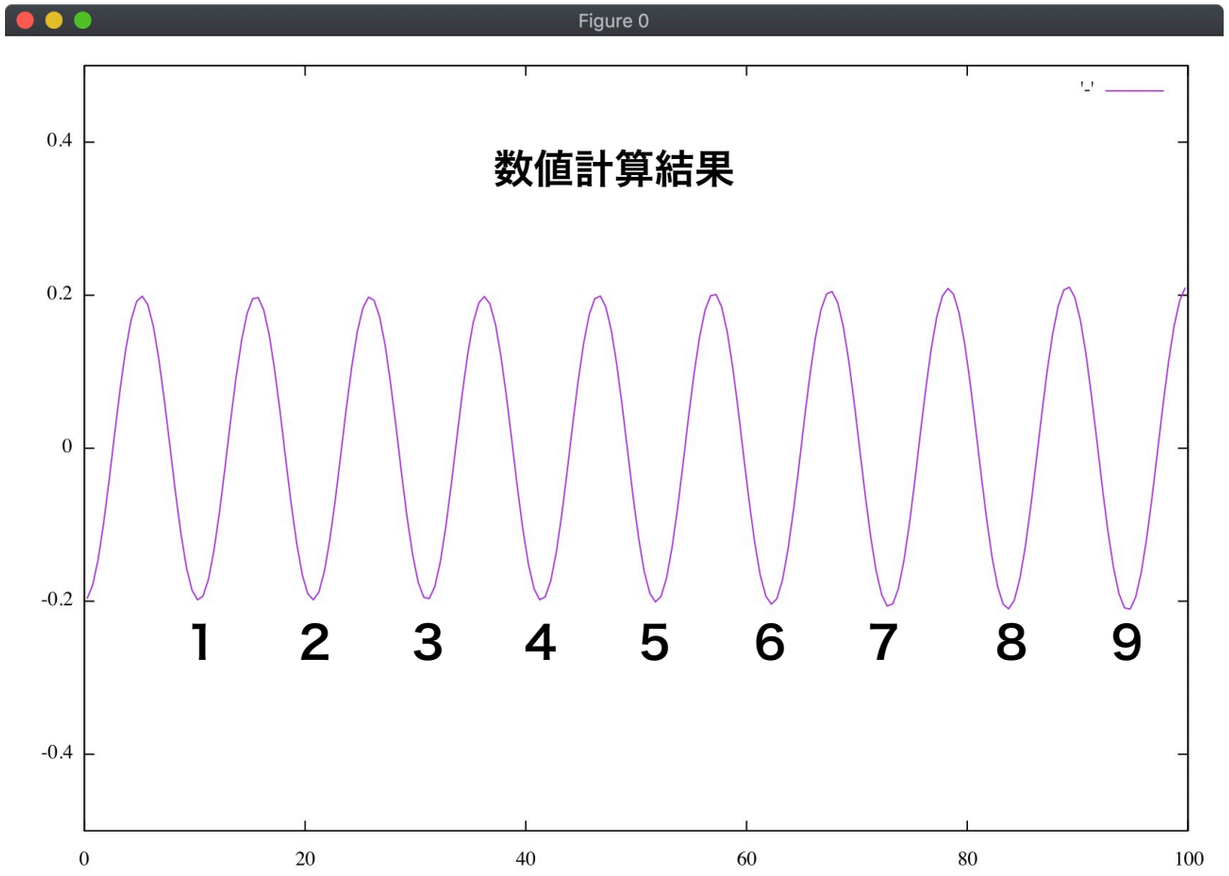
Du=1.0, Dv=10.0の時

拡大すると．．．



m=19,20くらいで最大になる





$$\cos(19 * \pi * x / L_x)$$

理論的に求めた不安定化するはずの解

以上から，Turing Instabilityの一連の解析の整合性は
数値計算からも確かめることができた.

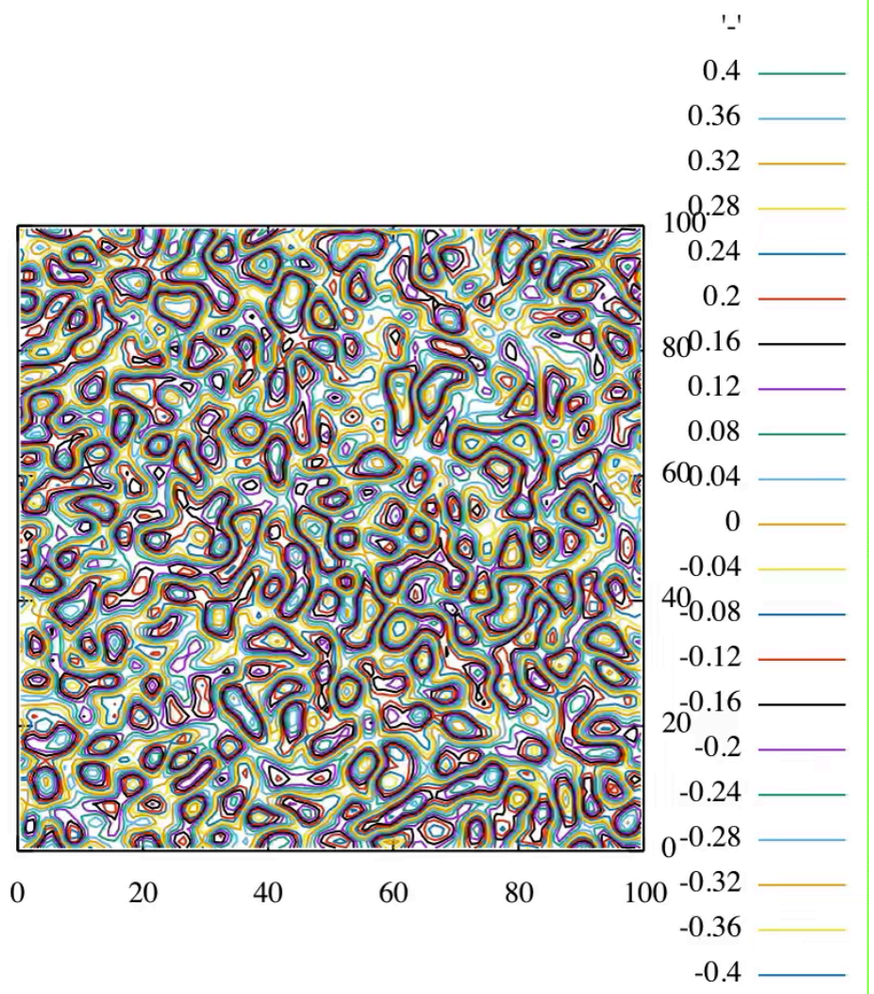
2Dシミュレーター

$$\frac{\partial u}{\partial t} = D_u \frac{\partial^2 u}{\partial x^2} + u(1 - u^2) - v$$

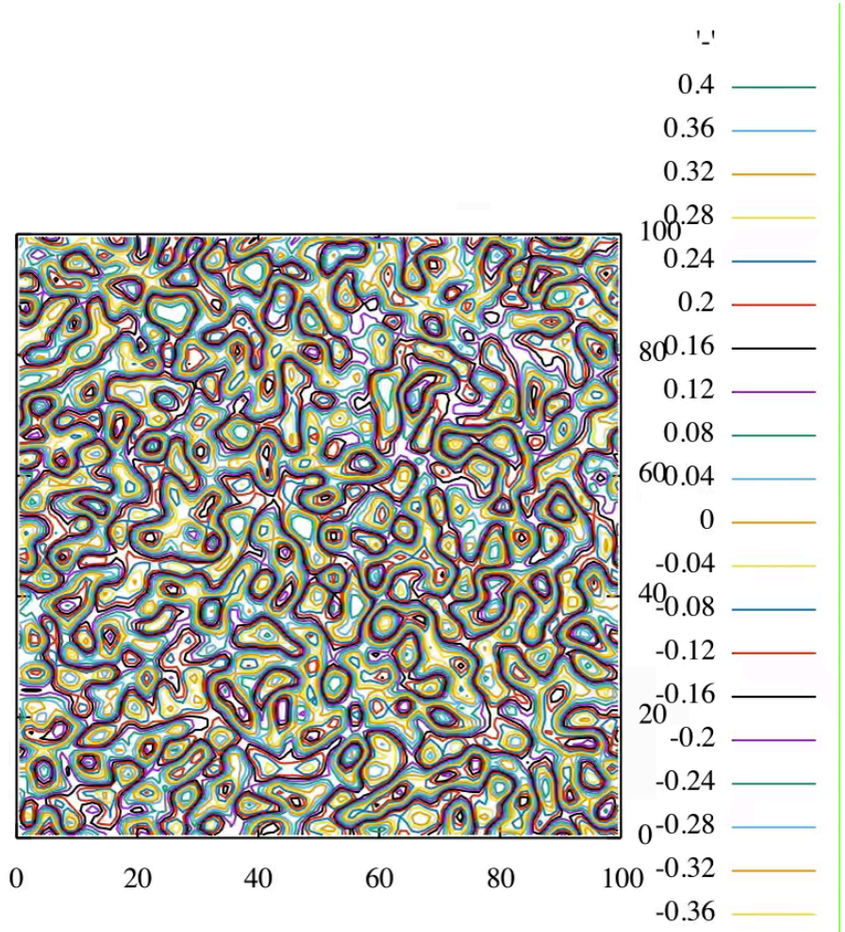
$$\frac{\partial v}{\partial t} = D_v \frac{\partial^2 v}{\partial x^2} + 3u - 2v$$

９_Turing2Dは上記方程式の２次元のソルバー(境界条件はノイマン0)

2Dシミュレーター

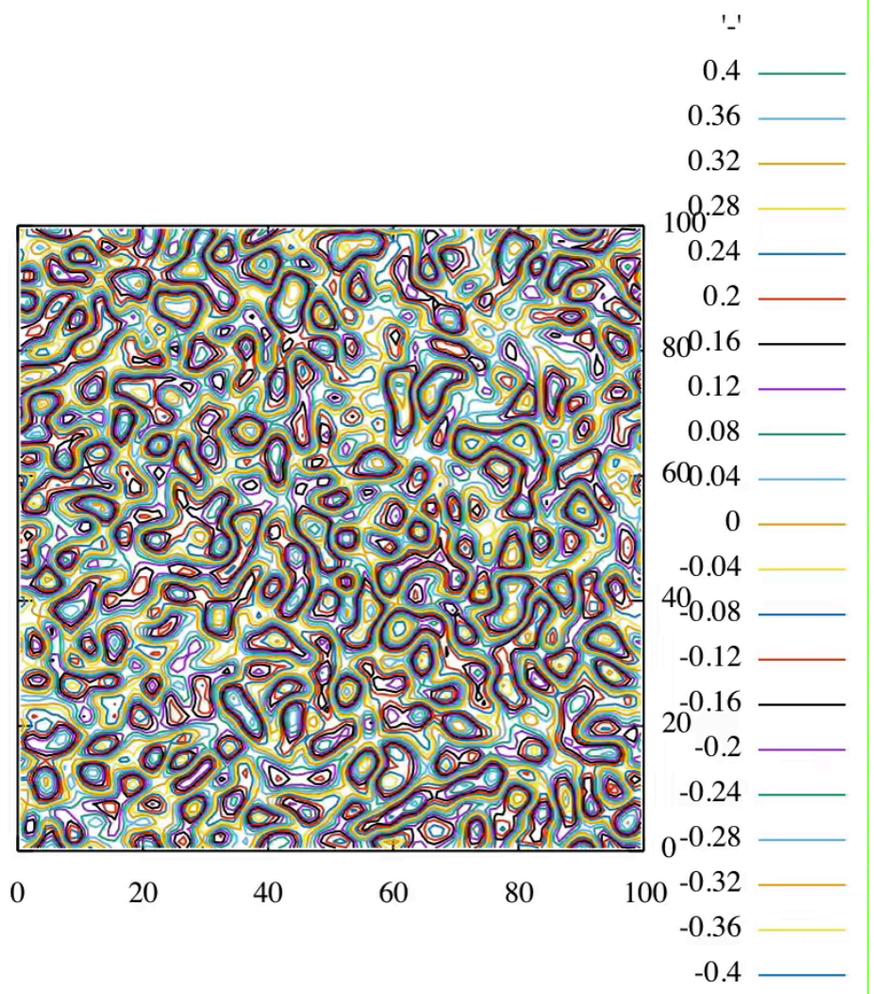


デフォルト値

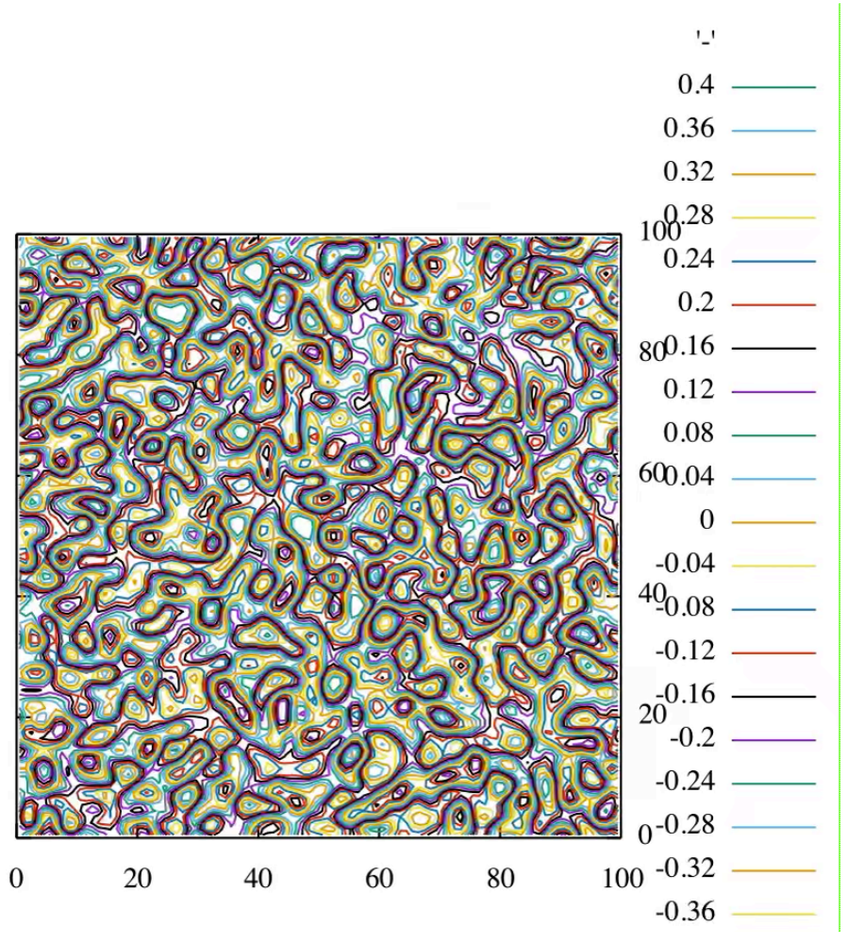


ある値で．．．．

2Dシミュレーター

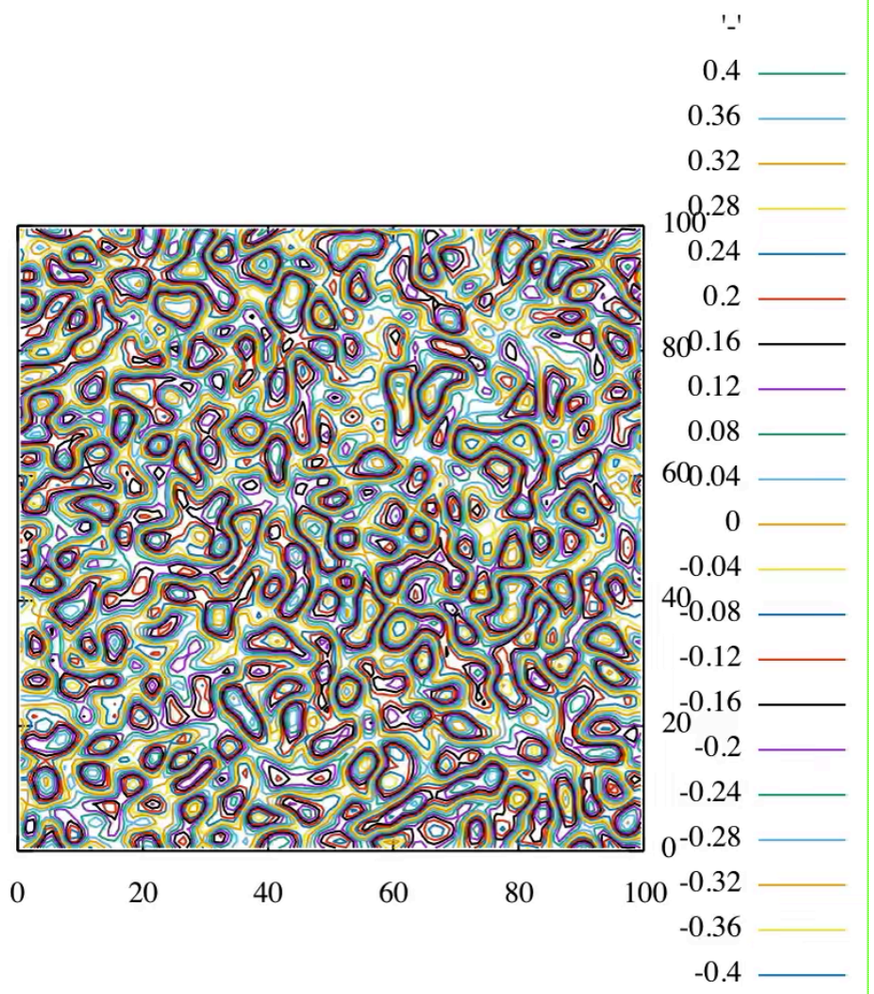


デフォルト値

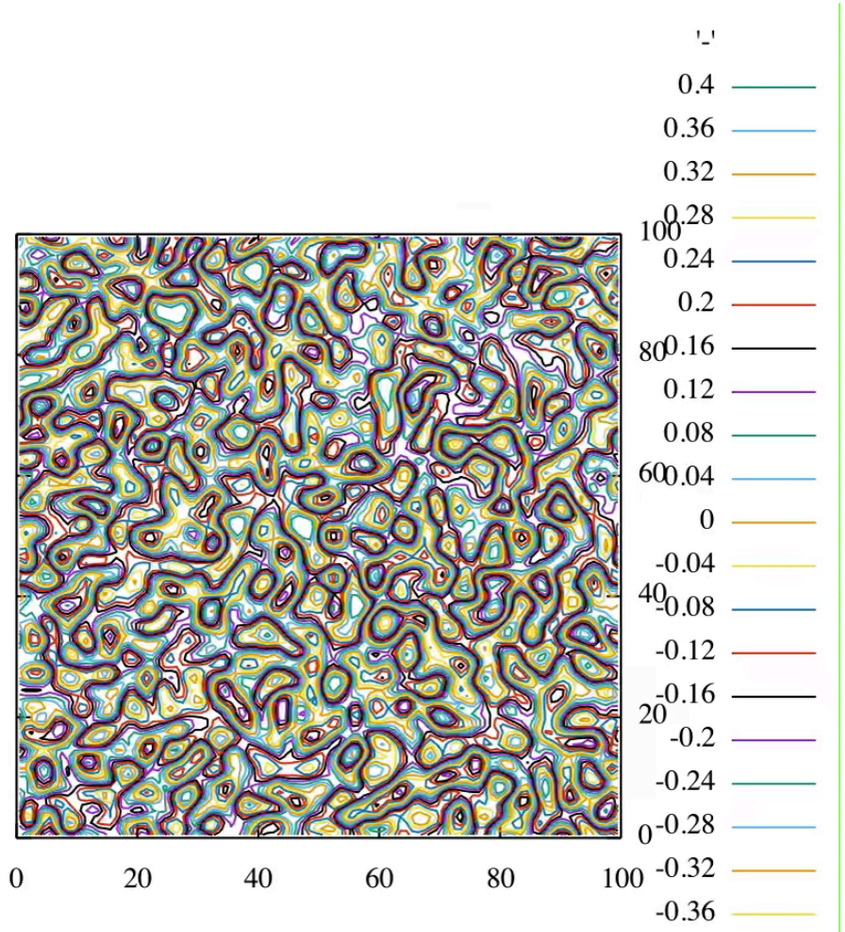


ある値で．．．．

2Dシミュレーター



デフォルト値



ある値で．．．．

反応項の対称性を崩す

$$\frac{\partial u}{\partial t} = D_u \frac{\partial^2 u}{\partial x^2} + u(1-u^2) - v$$

$$\frac{\partial v}{\partial t} = D_v \frac{\partial^2 v}{\partial x^2} + 3(u - \bar{u}) - 2v$$