# 複素微分方程式の解法

## 【１】複素数の係数を持つ１階の微分方程式

複素数を

$$z$$

として、微分方程式は、

$$\frac{dz}{dt}=f(z,t)$$

である。特に、

$$f(z,t)=(-0.1+0.5i)z \Leftarrow \text{実際には、}t\text{が含まれていないので、}f(z)=(-0.1+0.5i)z$$

とする。

## 【２】Runge-Kutta（ルンゲ・クッタ）法

解くべき差分方程式は、オイラー法で

$$z_{n+1}-z_n=\overbrace{f(z_n,t_n)}^{k_1}\times\Delta t$$

である。**【第５回 数値シミュレーション：１階の微分方程式（シラバス８・９回目）】ー【４】C言語ﾌﾟﾛｸﾞﾗﾑ：Runge-Kutta 法**では、記号 $k_1$ を用いて

$$z_{n+1}-z_n=\frac{k_1+2k_2+2k_3+k_4}{6}\times\Delta t$$

と変更される。ここで、

$$k_1=f(z_n,t_n)$$

$$k_2=f(z_n+\frac{k_1\times\Delta t}{2},t_n+\frac{\Delta t}{2})$$

$$k_3=f(z_n+\frac{k_2\times\Delta t}{2},t_n+\frac{\Delta t}{2})$$

$$k_4=f(z_n+k_3\times\Delta t,t_n+\Delta t)$$

である。

## 【３】複素数操作プログラムの準備

新しいプロジェクト

- bibun（微分）

を作成し、以下の２つの複素数を操作するプログラムをプロジェクトに追加する。。

１）以下のプログラムを、[fukusosuu.h]として保存：

```
typedef struct tagCOMPLEX{
   double r;
   double i;
} COMPLEX;

COMPLEX add(COMPLEX z1, COMPLEX z2);
COMPLEX subtract(COMPLEX z1, COMPLEX z2);
COMPLEX multiply(COMPLEX z1, COMPLEX z2);
COMPLEX divide(COMPLEX z1, COMPLEX z2);
COMPLEX conjugate(COMPLEX z);
double absolute(COMPLEX z);
COMPLEX multiply_number(double x, COMPLEX z);
```

```
COMPLEX cexp(double re, double im);
```

２）以下のプログラムを、[**fukusosuu.c**]として保存：

**該当箇所を【第７回 複素数の使用法（シラバス１２回目）】のプログラムからコピー可能**

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

#include "fukusosuu.h"

// z1+z2
COMPLEX add(COMPLEX z1, COMPLEX z2)
{
    COMPLEX z;

    z.r = z1.r+z2.r;
    z.i = z1.i+z2.i;

    return z;
}

// z1-z2
COMPLEX subtract(COMPLEX z1, COMPLEX z2)
{
    COMPLEX z;

    z.r = z1.r-z2.r;
    z.i = z1.i-z2.i;

    return z;
}

// z1*z2
COMPLEX multiply(COMPLEX z1, COMPLEX z2)
{
    COMPLEX z;

    z.r = z1.r*z2.r - z1.i*z2.i;
    z.i = z1.r*z2.i + z1.i*z2.r;

    return z;
}

// z1/z2
COMPLEX divide(COMPLEX z1, COMPLEX z2)
{
    COMPLEX z;
    COMPLEX bunsi;

    if((z2.r == 0) && (z2.i == 0)){
        COMPLEX zero = {0, 0};

        printf("エラー：０で割っています¥n");

        return zero;
    }
```

```
    bunsi.r = z1.r*z2.r + z1.i*z2.i;
    bunsi.i = z1.r*z2.i - z1.i*z2.r;

    z.r = bunsi.r/(z2.r*z2.r+z2.i*z2.i);
    z.i = bunsi.i/(z2.r*z2.r+z2.i*z2.i);

    return z;
}

// z^*
COMPLEX conjugate(COMPLEX z)
{
    z.i = -z.i;

    return z;
}

// |z|
double absolute(COMPLEX z)
{
    return sqrt(z.r*z.r + z.i*z.i);
}

// x*z
COMPLEX multiply_number(double x, COMPLEX z)
{
    z.r = x*z.r;
    z.i = x*z.i;

    return z;
}
```

## 【４】複素数操作関数の使用法

複素数の操作を

$$z_{n+1} - z_n = \frac{k_1 + 2k_2 + 2k_3 + k_4}{6} \times \Delta t$$

を例に取り解説する。プログラム内では、

- 時刻 $t_n$ → 時刻 $t_{n+1}\left(= t_n + \Delta t\right)$

に進む時

$$z_{n+1} = z_n + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6} \times \Delta t$$

と計算して、

- 時刻 $t_{n+1}\left(= t_n + \Delta t\right)$ での $z_{n+1}$

を求める。ここで、

$$k_2 = f(z_n + \frac{k_1 \times \Delta t}{2}, t_n + \frac{\Delta t}{2}) = f(z_n + \frac{\Delta t}{2} k_1, t_n + \frac{\Delta t}{2})$$

について、プログラムしてみよう:

**引数は、複素数と実数**

$$f(\overbrace{z_n+\frac{k_1\times\Delta t}{2}}^{\text{複素数}},\overbrace{t_n+\frac{\Delta t}{2}}^{\text{実数}}) \Rightarrow \text{f(COMPLEX z, double t)}$$

**計算結果は複素数**

**COMPLEX** f(COMPLEX z, double t)

**複素数 $z_n+\frac{k_1\times\Delta t}{2}=z_n+\frac{\Delta t}{2}k_1$ の作り方**

$$\left.\begin{array}{l}\overbrace{z_n}^{\text{COMPLEX}}+\overbrace{\frac{\Delta t}{2}}^{\text{double}}\overbrace{k_1}^{\text{COMPLEX}}\\ z_n\to \text{z},\ k_1\to \text{k1},\ \Delta t\to \text{step}\end{array}\right\}\Rightarrow\left\{\begin{array}{l}\overbrace{\frac{\Delta t}{2}}^{\text{step}}k_1=\overbrace{\text{実数}}^{\frac{\text{step}}{2}}\times\overbrace{\text{複素数}}^{\text{k1}}=\text{w}\to \text{w}=\boxed{\text{multyply\_number}\left(\frac{\text{step}}{2},\text{k1}\right)}\\ \overbrace{z_n}^{\text{z}}+\frac{\overbrace{k_1\Delta t}^{\text{w}}}{2}\to\boxed{\text{add(z, w)}}\end{array}\right.$$

ルンゲ･クッタ法のプログラム部分 $\boxed{z_n+\frac{k_1+2k_2+2k_3+k_4}{6}\times\Delta t}$ は、関数：calclate_nextとして与えられる。

**引数は、変数 $z$、時間 $t$ と時間の刻み $\Delta t$**

calclate_next(**COMPLEX z, double t, double step**)

**計算結果は複素数**

COMPLEX calclate_next(COMPLEX z, double t, double step)

**calclate_next 関数：** $z_{n+1}=z_n+\frac{k_1+2k_2+2k_3+k_4}{6}\times\Delta t$

$\overbrace{z_{n+1}}^{\text{next z}}=\boxed{z_n+\frac{k_1+2k_2+2k_3+k_4}{6}\times\Delta t}$ を計算し、計算結果は複素数(COMPLEX)になり return を使って計算結果を返す。

```
COMPLEX calclate_next(COMPLEX z, double t, double step)
{
    COMPLEX k1, k2, k3, k4;
    COMPLEX next_z;
    COMPLEX w, ww;

    k1 = f(z, t);⇐ k1 = f(zn, tn)
    //k2 = f(zn + k1×Δt/2, tn + Δt/2) = f(zn + Δt/2 k1, tn + Δt/2)

    w = multiply_number(step/2.0, k1);⇐ w= step/2.0 Δt/2  k1 : k2 = f(zn + step/2.0 Δt/2  k1, tn + Δt/2)

    w = add(z, w);⇐ z+w : k2 = f(zn + step/2.0 Δt/2  k1, tn + Δt/2)
```

$$k2 = f(w,\ t+step/2.0) \Leftarrow k_2 = f(\mathrm{w}, t_n + \frac{\Delta t}{2}) : k_2 = f(z_n + \overbrace{\frac{\Delta t}{2}}^{\text{step/2.0}}\ \overbrace{k_1}^{\text{k1}}, t_n + \frac{\Delta t}{2})$$

$$// k_3 = f(z_n + \frac{k_2 \times \Delta t}{2}, t_n + \frac{\Delta t}{2}) = f(z_n + \frac{\Delta t}{2} k_2, t_n + \frac{\Delta t}{2})$$

```
w = multiply_number(step/2.0, k2);
w = add(z, w);
k3 = f(w, t+step/2.0);
```

$$// k_4 = f(z_n + k_3 \times \Delta t, t_n + \Delta t) = f(z_n + \Delta t k_3, t_n + \Delta t)$$

```
w = multiply_number(step, k3);
w = add(z, w);
k4 = f(w, t+step);
```

$$w = multiply\_number(2.0,\ k2) \Leftarrow z_{n+1} - z_n = \frac{k_1 + \overbrace{2k_2}^{\text{w}} + 2k_3 + k_4}{6} \times \Delta t$$

$$ww = add(k1,\ w) \Leftarrow z_{n+1} - z_n = \frac{\overbrace{k_1 + \overbrace{2k_2}^{\text{w}}}^{\text{ww}} + 2k_3 + k_4}{6} \times \Delta t$$

$$w = multiply\_number(2.0,\ k3) \Leftarrow z_{n+1} - z_n = \frac{k_1 + 2k_2 + \overbrace{2k_3}^{\text{w}} + k_4}{6} \times \Delta t$$

$$ww = add(ww,\ w) \Leftarrow z_{n+1} - z_n = \frac{\overbrace{k_1 + 2k_2}^{\text{ww}} + \overbrace{2k_3}^{\text{w}} + k_4}{6} \times \Delta t$$

$$w = add(ww,\ k4) \Leftarrow z_{n+1} - z_n = \frac{\overbrace{k_1 + 2k_2 + 2k_3}^{\text{ww}} + \overbrace{k_4}^{\text{k4}}}{6} \times \Delta t$$

$$\Leftarrow z_{n+1} - z_n = \frac{k_1 + 2k_2 + 2k_3 + k_4}{6} \times \Delta t = \frac{\overbrace{\Delta t}^{\text{step}}}{6} \times \overbrace{k_1 + 2k_2 + 2k_3 + k_4}^{\text{w}}$$

```
ww = multiply_number(step/6.0, w);
```

$$next\_z = add(z,\ ww) \Leftarrow \overbrace{z_{n+1}}^{\text{next z}} = \overbrace{z_n}^{\text{z}} + \overbrace{\frac{\Delta t}{6} \times (k_1 + 2k_2 + 2k_3 + k_4)}^{\text{ww}}$$

```
    return next_z;
}
```

微分方程式 $\frac{dz}{dt} = f(z,t)$ は、関数：f(z, t)：$f(z,t) = (-0.1 + 0.5i)z$ から与えられる。

**引数は、変数 $z$、時間 $t$**

f(**COMPLEX z, double t**)

**計算結果は複素数**

COMPLEX f(COMPLEX z, double t)

**f 関数：** $f(z,t)=(-0.1+0.5i)z$

$f(z,t)=\boxed{(-0.1+0.5i)z}$ を計算し、計算結果は複素数(COMPLEX)になり return を使って返す。

```
COMPLEX f(COMPLEX z, double t)
{
    COMPLEX fz;
    COMPLEX a = {-0.1, 0.5};   ⇐ −0.1+0.5i
    fz = multiply(a, z);       ⇐ (−0.1+0.5i)z

    return fz;
}
```

fz を省略した次の関数でもよい:

```
COMPLEX f(COMPLEX z, double t)
{
    COMPLEX a = {-0.1, 0.5};   ⇐ −0.1+0.5i

    return multiply(a, z);     ⇐ (−0.1+0.5i)z
}
```

## 【５】微分方程式解法プログラム

$z$ の初期値( $t=0$ )として、

$$\begin{cases} t=0 \\ z=10 \end{cases} \Rightarrow \boxed{\begin{array}{l}\text{COMPLEX z0 = \{10, 0\};} \\ \text{z = z0;} \\ \text{t = 0;}\end{array}}$$

をとり、0.1秒 おき( $\Delta t=0.1$ )に $t=50$秒 まで計算する:

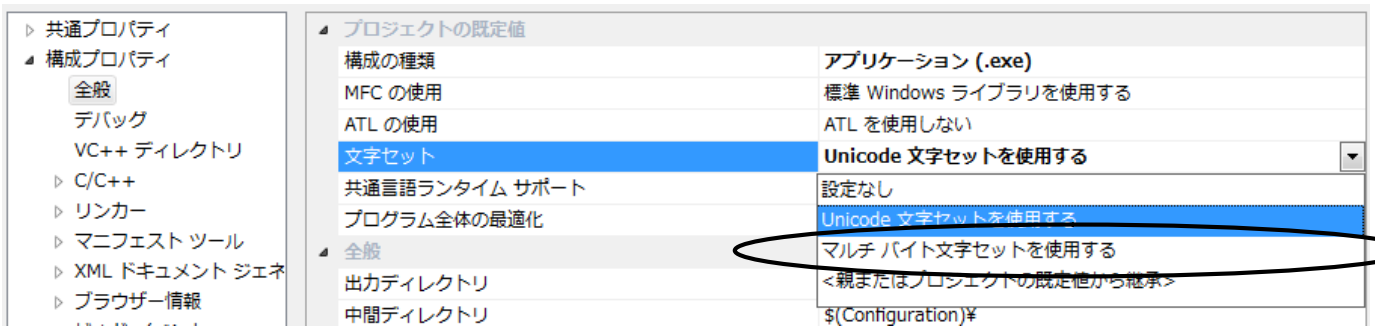
$\Delta t=0.1 \Rightarrow$ `#define STEP 0.1`

$t=50 \Rightarrow$ `#define LAST_TIME 50.0`

計算結果をファイル名 :**bibun.csv**に書き込み、エクセルを自動起動させる。

**メニュー[プロジェクト]-[プロパティー]**の

- **[文字セット] マルチバイト文字セットを使用する**を選択



以下のプログラムを、[**bibun.c**]として先ほどのプロジェクトbibun（**微分**）に追加する。

```
#define _CRT_SECURE_NO_WARNINGS

#include <windows.h>
#include <stdio.h>
#include <math.h>
#include "fukusosuu.h"

#define LAST_TIME 50.0
#define STEP  0.0025

COMPLEX f(COMPLEX z, double t)
{
   COMPLEX fz;
   COMPLEX a = {-0.1, 0.5};

   fz = multiply(a, z);

   return fz;
}

COMPLEX calclate_next(COMPLEX z, double t, double step)
{
   COMPLEX k1, k2, k3, k4;
   COMPLEX next_z;
   COMPLEX w, ww;

   k1 = f(z, t);

   w = multiply_number(step/2.0, k1);
   w = add(z, w);
   k2 = f(w, t);

   w = multiply_number(step/2.0, k2);
   w = add(z, w);
   k3 = f(w, t);

   w = multiply_number(step, k3);
   w = add(z, w);
   k4 = f(w, t);

   w = multiply_number(2.0, k2);
   ww = add(k1, w);

   w = multiply_number(2.0, k3);
   ww = add(ww, w);

   w = add(ww, k4);

   ww = multiply_number(step/6.0, w);

   next_z = add(z, ww);

   return next_z;
}
```

```
void writetitle(FILE *file_open)
{
   printf("t,実部,,t,虚部¥n");

   if(file_open != NULL){
      fprintf(file_open, "t,実部,,t,虚部¥n");
   }
}

void writedata(FILE *file_open, COMPLEX z, double t)
{
   printf("%5.2f,%10.3e,,%5.2f,%10.3e¥n", t, z.r, t, z.i);
   if(file_open != NULL){
      fprintf(file_open, "%5.2f,%10.3e,,%5.2f,%10.3e¥n", t, z.r, t, z.i);
   }
}

int main(void)
{
   // 作成ファイル名はbibun.csv
   char filename[] = "bibun.csv";
   // 作成ファイル追尾用変数
   FILE * file_open;
   double t, last_t;
   COMPLEX z, next_z;
   COMPLEX z0 = {10, 0};

   z = z0;
   t = 0.0;
   last_t = LAST_TIME*(1.0+STEP);

   // ファイルの作成に失敗すると数値(file_open=NULL)を返す
   file_open = fopen(filename, "wt");

   writetitle(file_open);

   while(t < last_t){
      writedata(file_open, z, t);
      next_z = calclate_next(z, t, STEP);
      z = next_z;
      t = t+STEP;
   }

   // NULLのときにファイル作成失敗
   if(file_open == NULL){
      printf("¥n====>ファイル(%s)の作成に失敗しています。¥n====>既にファイルが
開かれているかもしれません。", filename);

      getchar();
   }
   else{
      HINSTANCE hinst;

      fclose(file_open);

      hinst = ShellExecute((HWND)0, "open", filename, NULL, NULL, SW_SHOWNORMAL);

      if(hinst <= (HINSTANCE)32){
```

```
            printf("¥n====>表計算ソフトの自動起動に失敗しました。");

            getchar();
        }
    }

    return 0;
}
```

# 第８回目レポート

## レポートの回数(８回目)、学生証番号と氏名を明記すること

### 必ず表紙を付け、最後のページ(の添付用)を表紙の次に入れる

### （Ａ４レポート用紙使用のこと）

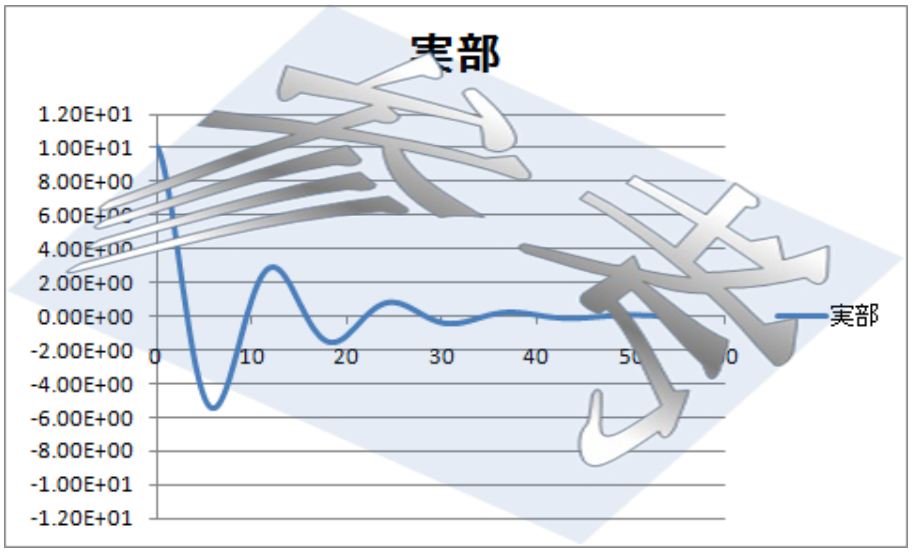
1） 3 つのプログラム(fukusosuu.h, fukusosuu.c, bibun.c)において、

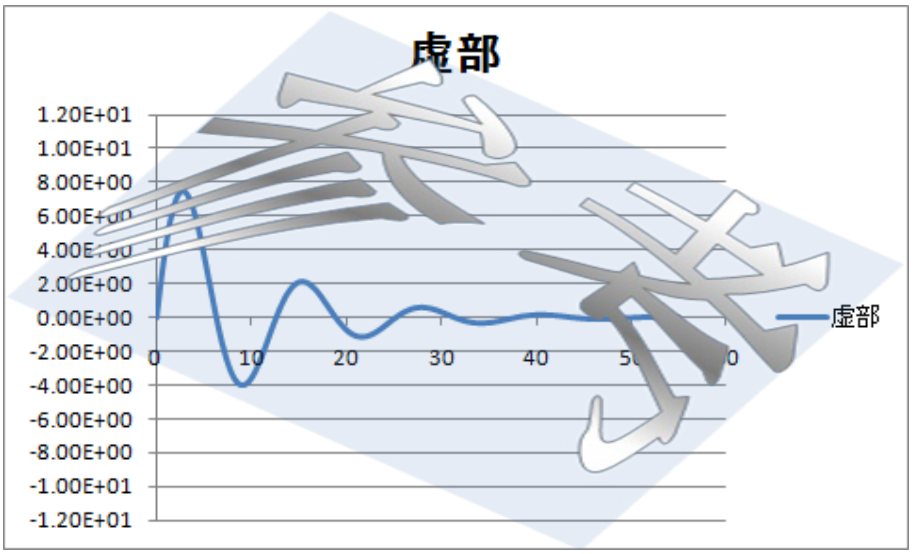
- 3 つプログラム
- プログラムを実行し、計算結果データを出力して、実部と虚部の2つグラフ
  - **実部**のグラフ
  - **虚部**のグラフ

の5点です。グラフ作成時は、

- 縦軸の**最小値を-12、最大値を 12、目盛間隔を 2** にして表示する。
- [**グラフの移動**]を”新しいシート”にする。

を実行する事。グラフは以下のようになります。




2） $\frac{dz}{dt}=(-0.1+0.5i)z$ を解き、1)のグラフの振る舞いを説明せよ。

3） $\frac{dz}{dt} = -0.05z + 0.01iz^2$ を解き、

- プログラム：自動作成のファイル名を **repo.csv** に**変更すること**。

```
// 作成ファイル名はrepo.csv
char filename[] = "repo.csv";
```

ヒントは・・・

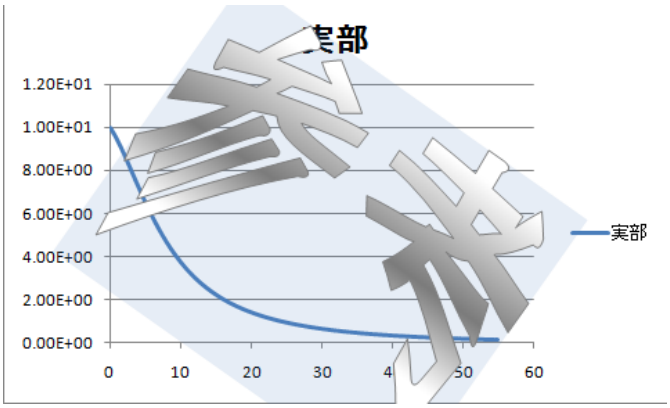
- 虚数 $i$ は、COMPLEX 型の {0, 1}として定義する。
- $0.01iz^2$ で $z^2$ を作るには multiply 関数、$iz^2$ を作るには $i$ と $z^2$ に multiply 関数、$0.01iz^2$ を作るには、$0.01$ と $iz^2$ に multiply_number 関数を順に使う。

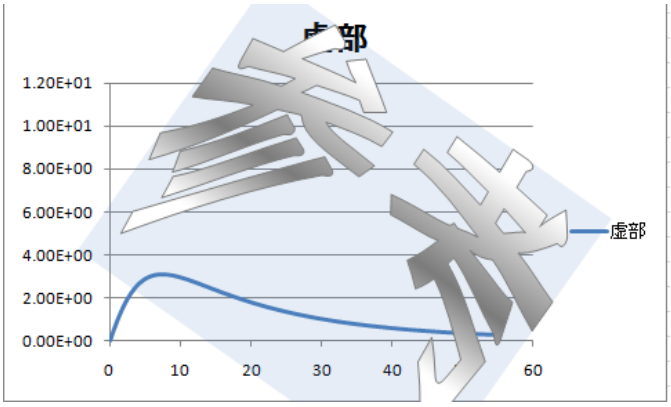
- 実部と虚部の2つグラフ
  - **実部**のグラフ
  - **虚部**のグラフ

の 3 点を提出です。グラフ作成時は、

- 縦軸の**最小値を 0、最大値を 12、目盛間隔を 2** にして表示する。
- [**グラフの移動**]を”新しいシート”にする。

を実行する事。グラフは以下のようになります。



提出例(使える表紙は次のページ)


１ページ目　　２ページ目　　３ページ目以降

# コンピュータ物理学演習 II

## 第 8 回目

月　日提出

学籍番号

氏名

# 第８回目レポート（添付用）

1）３つのプログラム(fukusosuu.h, fukusosuu.c, ryousi.c)において、

- ３つプログラムを作成
- プログラムを実行し、計算結果( $x-\left|\psi\left(x,t\right)\right|$ )データを出力して、次のｎの場合の5つグラフ(n_out = **N_MAX/4** の箇所を変更します)を作成
  - n=0
  - n=N_MAX/4
  - n=N_MAX/2
  - n=3*N_MAX/4
  - n=N_MAX

の合計７点です。グラフ作成時は、

- **縦軸**の最大値を 10 にして表示する。
- [**グラフの移動**]を”新しいシート”にする。

を実行する事。

2）グラフからは、最初１つの波束しかないが、ポテンシャルを通過すると２つの波束に分かれてゆく様子が読み取れる。1つの量子をポテンシャルにぶつけると2つの量子に分裂し、最後は ２ つの量子が残るという事だろうか？この結果を説明せよ。