複素関数を描く

*瓜　　生　　　　等

How to draw graphs of functions of a complex variable

URYU Hitoshi

Abstract

I consider the graphs of the functions of a complex variable in this paper.

I cannot draw it directly, the functions from complex number to complex number. Because a thought method to look around the perspective from the part is necessary, high content will be found mathematically. If it can be visualized in a part easily, I become able to understand the total picture by putting them on top of one another in imagination. And this is regarded as typical mathematical activity. Therefore I decided to use an information appliance to be able to perform visualization instantly. I calculated the theoretical numerical value by C language and used drawing software Gnuplot with practice numerical value. We are dealing with fluid dynamics as a specific example.

I really report the contents which let a university student test it by a class here.

Key words：Complex number　複素数
Function of a complex variable　複素関数
Programming language　プログラミング言語
Drawing software　描画ソフトウエア
Fluid mechanics　流体力学

目次

＊　数学教育講座

1　序文

平成 8 年度に宮城教育大学には生涯教育総合課程が設置され，新しく情報数理専攻が立てられ，11年の間学生を育てた歴史をもつ．この専攻は数学的な素養を基礎として「情報数理」を理解することが目的であった．その専門科目のなかに「現象の数理」があり，複素関数論を基礎として，複素解析の具体的な応用例を紹介する内容であった．著者はこの講義を担当して，教室で理論を講義をした後，C言語によるプログラミングと描画ソフトである Gnuplot により， 複素関数を可視化する実験を演習とした．理論としての複素関数論の理解を深めるため，コンピュータアルゴリズムを理解をさせ，複素関数による可視化することにより，理論の実感を構築することが目的であった．本稿では，筆者の作った講義ノートの内容を踏まえて，複素関数論の理解を深めるための実践例について説明をする．

2　演習のしかた

演習は本学情報演習室の端末 Windows と Macintosh を利用した．必要なプログラムやスクリプトは情報処理センターの特定のフォルダーにおいておき，学生に利用させた．基本となるプログラムやスクリプトは筆者が作成し，その内容を講義で説明をし，学生は必要箇所を適宜改変することにより簡易に実習が実行できるようにした．

この演習は Unix 上で全ての作業を行うため，以下のスキルが必要であった．

- ターミナルソフトを用いて Windows か Macintosh から Unix につなぐことができること．
- Unix の基本的なシェル操作ができること．
- Unix 上のエディターを用いてテキストファイルの編集ができること．
- Unix 上のC言語プログラムを実行できること．
- Gnuplot スクリプトを実行できること．

3　複素数演算

複素数をC言語で扱うために数学的な準備が必要となる．

3.1　四則演算

まず複素数の基本的な演算を復習する．

複素数は $z = x + iy$ と表現される．

z の共役複素数は $\bar{z} = x - iy$ とかかれ，

絶対値は $|z| = \sqrt{x^2 + y^2}$,

偏角は $\arg z = \arctan \dfrac{y}{x}$ となる．

$z_1 = x_1 + iy_1$, $z_2 = x_2 + iy_2$ に対して

加減は

$$z_1 \pm z_2 = (x_1 \pm x_2) + i\,(y_1 \pm y_2),$$

乗算は

$$z_1 z_2 = (x_1 x_2 - y_1 y_2) + i\,(x_1 y_2 + x_2 y_1),$$

除算は

$$\frac{z_1}{z_2} = \frac{(x_1 x_2 + y_1 y_2)}{x_2^2 + y_2^2} + i\,\frac{(y_1 x_2 - x_1 y_2)}{x_2^2 + y_2^2}$$

となる．

3.2　複素関数

次に複素関数の基本について復習する．

指数関数は $z = x + iy$ に対して

$$\exp z = \mathrm{e}^x(\cos y + i \sin y).$$

対数関数は $z = re^{i\theta}$ に対して

$$\log z = \log r + i\,\theta\,.$$

ここで

$$r = \sqrt{x^2 + y^2},\ \theta = \arctan \frac{y}{x}.$$

三角関数は

$$\sin z = \frac{\mathrm{e}^{iz} - \mathrm{e}^{-iz}}{2i},\ \cos z = \frac{\mathrm{e}^{iz} + \mathrm{e}^{-iz}}{2},$$

$$\tan z = \frac{\sin z}{\cos z}\,.$$

双曲線関数は

$$\sin z = \frac{\mathrm{e}^{iz} - \mathrm{e}^{-iz}}{2i},\ \cos z = \frac{\mathrm{e}^{iz} + \mathrm{e}^{-iz}}{2},$$

$$\tan z = \frac{\sin z}{\cos z}\,.$$

以上より

$$\sin(x + iy) = \sin x \cosh y + i \cos x \sinh y$$
$$\cos(x + iy) = \cos x \cosh y - i \sin x \sinh y$$
$$\sinh(x + iy) = \sinh x \cos y + i \cosh x \sin y$$
$$\cosh(x + iy) = \cosh x \cos y + i \sinh x \sin y$$

が得られるので，複素関数である左辺を右辺の実部と虚部を実関数で表して計算することができる．

平方根は

$\sqrt{x+iy} = \sqrt{r}\,(cos(\theta/2) + i\sin(\theta/2))$.

$=\sqrt{\frac{r+x}{2}} + i\sqrt{\frac{r-x}{2}}$.

累乗はをつぎを利用する.

$$z^{\alpha} = \exp(\alpha \log z).$$

以上を考慮して奥村氏[1]が作成したプログラムを参考に若干手を加えたものが以下の complex.c である．本質的なルーチンには変更を加えていない．ここに，奥村氏に感謝する．この基本的なルーチンを理解することは，後に主プログラムからこれらの関数を引用する際に重要な内容であることになる．これは学生にとって重要な学習過程となるので，あえて全文を掲げた.

```
/*********************************************************
complex.c -- 複素数の四則と初等複素関数
*********************************************************/
typedef struct { double re, im; } complex; }
/ * 複素数型 * /
complex c_conv(double x, double y)
/ * $x$, $y$ を複素数 $z = x + iy$ に変換 * /
{
        complex z;

        z.re = x; z.im = y;
        return z;
}
char *c_string(complex z)
/ * 複素数 $z = x + iy$ を文字列に変換 * /
{
        static char s[40];

        sprintf(s, "%g%+gi", z.re, z.im);
        return s;
}
complex c_conj(complex z)
/ * 共役複素数 $\overline{z}$ * /
{
        z.im = - z.im;
        return z;
}
double c_abs(complex z)
/ * 絶対値 $|z|$ * /
{
        double t;

        if (z.re == 0) return fabs(z.im);
        if (z.im == 0) return fabs(z.re);
        if (fabs(z.im) > fabs(z.re)) {
                t = z.re / z.im;
                return fabs(z.im) * sqrt(1 + t * t);
        } else {
                t = z.im / z.re;
                return fabs(z.re) * sqrt(1 + t * t);
        }

double c_arg(complex z)
/ * 偏角 ($-\pi \le \varphi \le \pi$) * /
{
        return atan2(z.im, z.re);
}

complex c_add(complex x, complex y)
/ * 和 $x + y$ * /
{
        x.re += y.re;
        x.im += y.im;
        return x;
}

complex c_sub(complex x, complex y)
/ * 差 $x - y$ * /
{
        x.re -= y.re;;
        x.im -= y.im;
        return x;
}

complex c_mul(complex x, complex y)
/ * 積 $xy$ * /
{
        complex z;

        z.re = x.re * y.re - x.im * y.im;
        z.im = x.re * y.im + x.im * y.re;
```

```
	return z;
}

#if 0
complex c_div(complex x, complex y)
/ * 商 $x / y$（単純版）* /
{
	double r2;
	complex z;

	r2 = y.re * y.re + y.im * y.im;
	z.re = (x.re * y.re + x.im * y.im) / r2;
	z.im = (x.im * y.re - x.re * y.im) / r2;
	return z;
}
#endif

complex c_div(complex x, complex y)
/ * 商 $x / y$（上位桁あふれ対策版）* /
{
	double w, d;
	complex z;

	if (fabs(y.re) >= fabs(y.im)) {
		w = y.im / y.re; d = y.re + y.im * w;
		z.re = (x.re + x.im * w) / d;
		z.im = (x.im - x.re * w) / d;
	} else {
		w = y.re / y.im; d = y.re * w + y.im;
		z.re = (x.re * w + x.im) / d;
		z.im = (x.im * w - x.re) / d;
	}
	return z;
}

complex c_exp(complex x)
/ * 指数関数 $e^x$ * /
{
	double a;

	a = exp(x.re);
	x.re = a * cos(x.im);
	x.im = a * sin(x.im);
	return x;
}

complex c_log(complex x)
/ * 自然対数 * /
{
    complex z;

    z.re = 0.5 * log(x.re * x.re + x.im * x.im);
    z.im = atan2(x.im, x.re);
    return z;
}

complex c_pow(complex x, complex y)
/ * 累乗 * /
{
	return c_exp(c_mul(y, c_log(x)));
}

complex c_sin(complex x)
/ * 正弦 $ * /
{
	double e, f;

	e = exp(x.im); f = 1 / e;
	x.im = 0.5 * cos(x.re) * (e - f);
	x.re = 0.5 * sin(x.re) * (e + f);
	return x;
}

complex c_cos(complex x)
/ * 余弦 * /
{
	double e, f;

	e = exp(x.im); f = 1 / e;
	x.im = 0.5 * sin(x.re) * (f - e);
	x.re = 0.5 * cos(x.re) * (f + e);
	return x;
}
```

```
complex c_tan(complex x)
/ * 正接 * /
{
        double e, f, d;

        e = exp(2 * x.im); f = 1 / e;
        d = cos(2 * x.re) + 0.5 * (e + f);
        x.re = sin(2 * x.re) / d;
        x.im = 0.5 * (e - f) / d;
        return x;
}

complex c_sinh(complex x) / * 双曲線正弦 * /
{
        double e, f;

        e = exp(x.re); f = 1 / e;
        x.re = 0.5 * (e - f) * cos(x.im);
        x.im = 0.5 * (e + f) * sin(x.im);
        return x;
}

complex c_cosh(complex x)
/ * 双曲線余弦 * /
{
        double e, f;

        e = exp(x.re); f = 1 / e;
        x.re = 0.5 * (e + f) * cos(x.im);
        x.im = 0.5 * (e - f) * sin(x.im);
        return x;
}

complex c_tanh(complex x)
/ * 双曲線正接 * /
{
        double e, f, d;

        e = exp(2 * x.re); f = 1 / e;
        d = 0.5 * (e + f) + cos(2 * x.im);
        x.re = 0.5 * (e - f) / d;
        x.im = sin(2 * x.im) / d;
        return x;
}

#define SQRT05 0.707106781186547524
/ * $\sqrt{0.5}$ * /
complex c_sqrt(complex x)
/ * 平方根 $\sqrt{x}$ * /
{
    double r, w;

    r = c_abs(x);
    w = sqrt(r + fabs(x.re));
    if (x.re >= 0) {
      x.re = SQRT05 * w;
      x.im = SQRT05 * x.im / w;
    } else {
      x.re = SQRT05 * fabs(x.im) / w;
      x.im = (x.im >= 0) ?
      SQRT05 * w : -SQRT05 * w;
      }
    return x;
}
```

3.3　計算例

実際の複素数の計算例

```
/*********************************************************
csisoku.c -- 複素数の四則
*********************************************************/

#include <stdio.h>
#include <math.h>
#include "complex.c"

int main()
{
        double x1, y1,x2,y2;
        complex z1,z2,z3;
        x1=1;y1=2;x2=2;y2=3;
        z1 = c_conv(x1, y1);
        z2 = c_conv(x2, y2);
        printf("z1 = %s\n", c_string(z1));
        printf("z2 = %s\n", c_string(z2));
        z3=c_conj(z1);
```

```
        printf("conjugate of z1= %s\n", c_string(z3));
        / * 共役 * /
        printf("rorm of z1= %f\n", c_abs(z1));
        / * 絶対値 * /
        printf("argument of z1= %f\n", c_arg(z1));
        z3=c_add(z1,z2);
        printf("z1+z2 = %s\n", c_string(z3));
        / * 和 * /
        z3=c_sub(z1,z2);
        printf("z1-z2 = %s\n", c_string(z3));
        / * 差 * /
        z3=c_mul(z1,z2);
        printf("z1*z2= %s\n", c_string(z3));
        / * 積 * /
        z3=c_div(z1,z2);
        printf("z1/z2= %s\n", c_string(z3));
        / * 除 * /
}
```

実際の複素関数の計算例

```
/*******************************************************
function.c 複素数の初等関数
*******************************************************/
#include <stdio.h>
#include <math.h>
#include "complex.c"

int main()
{
    double x, y,u,v;
    complex z,w,a;
    x=1;y=2;/*z=x+iy*/
    u=0;v=1;/*a=u+iv*/
    z = c_conv(x, y);
    a = c_conv(u, v);
    printf("z = %s\n", c_string(z));
    printf("a = %s\n", c_string(a));
    /*****************************/
    w = c_exp(z);
    printf("exp(z) = %s\n", c_string(w));
    w = c_log(z);
    printf("log(z) = %s\n", c_string(w));
    w = c_sin(z);
    printf("sin(z) = %s\n", c_string(w));
    w = c_cos(z);
    printf("cos(z) = %s\n", c_string(w));
    w = c_tan(z);
    printf("tan(z) = %s\n", c_string(w));
    w = c_sinh(z);
    printf("sinh(z) = %s\n", c_string(w));
    w = c_cosh(z);
    printf("cosh(z) = %s\n", c_string(w));
    w = c_sqrt(z);
    printf("sqrt(z) = %s\n", c_string(w));
    w = c_pow(z,a);
    printf("z^a= %s\n", c_string(w));
}
```

4　複素数関数の可視化

以上のプログラムにより複素数関数の値が計算できるようになったので，次に複素関数を可視化することにより，関数の行動を理解することが可能となる．

理論を理解し，Ｃプログラムや Gnuplot スクリプトを作成したうえで，Ｃ言語でデータを計算し，Gnuplot でグラフを描くが手順である．

4.1　初等関数の視覚化

一般に複素関数 $w = f(z)$ が与えられたとする．

$z = x + iy, w = u + iv$ と書くと，

$f(x + iy) = u(x, y) + iv(x, y)$ と実部虚部に分けて表示すると $u = u(x, y)$, $v = v(x, y)$ なる関係式がでてくる．

$z = re^{i\theta}, w = u + iv$ と書くと，

$f(re^{i\theta}) = u(r, \theta) + iv(r, \theta)$ と実部虚部に分けて表示すると，$u = u(r, \theta)$, $v = v(r, \theta)$ なる関係式が得られる．

ガウス平面上で複素数を考えると複素関数は 2 次元から 2 次元への写像となり，4 次元を扱うことになるが，これを 3 次元の空間に住む人間が理解することは極めて難しい．従って，何らかの工夫をすることにより，可視化が実現すること考える．

そこで上の表示式を利用して，z 平面および w 平面において各変数の一方を固定し，像曲線および逆像曲

線を描くことにする．これらの曲線群の変化を眺めることにより，複素関数の行動を観察していくことにするわけである．

以下この節で考察する問題は共通して以下である．

$f(z)$ に対して$u(x, y)$, $v(x, y)$, $u(r, \theta)$, $v(r, \theta)$ をそれぞれ求め，次の各問に答えよ．

- (1) x が一定のとき，$(u(x, y),\ v(x, y))$ が描く曲線を求めよ．
- (2) y が一定のとき，$(u(x, y),\ v(x, y))$ が描く曲線を求めよ．
- (3) r が一定のとき，$(u(r, \theta),\ v(r, \theta))$ が描く曲線を求めよ．
- (4) θ が一定のとき，$(u(r, \theta),\ v(r, \theta))$ が描く曲線を求めよ．
- (5) $u(x, y)$ が一定のとき，(x, y) が描く曲線を求めよ．
- (6) $v(x, y)$ が一定のとき，(x, y) が描く曲線を求めよ．

注意

(5)，(6) は$f(z)$ の逆関数に対して(1)，(2) を考えることと同じことである．

4.2　多項式関数

$w = f(z) = z^2$ を用いて，方法の大筋を説明する．

Ｃプログラムはグラフとなる基礎データを計算させることに用い，実際のグラフの描画は Gnuplot なる描画ソフトに任せる．

以下のプログラムは設問の(1)，(2) に答えたものである．

```
#sqr1.c
#include <stdio.h>
#include <math.h>
#define pai 3.141592653589793
/ * 必要なとき使用する * /
/ * 関数 f(z)=z*z * /
double funcf(double x, double y)
/ * f(z) の実部 * /
{
        return x*x -y*y;
}
double funcg(double x, double y)
/ * f(z) の虚部 * /
{
        return 2*x*y;
}
/ *x を一定としたときの像の描く曲線群の関数 * /
double conf_x(double xl, double xr,double yl,
double yr, double xstep, double ystep)
{
        double x,y;
        for(x=xl;x<=xr;x=x+xstep)
        {
          for(y=yl;y<=yr;y=y+ystep)
          printf("%f\t%f\n",
          funcf(x,y),funcg(x,y));
          printf("\n");
        }
}
/ *y を一定としたとき, 像の描く曲線群の関数 * /
double conf_y(double xl, double xr,double yl,
double yr, double xstep, double ystep)
{
        double x,y;
        for(y=yl;y<=yr;y=y+ystep)
        {
          for(x=xl;x<=xr;x=x+xstep)
          printf("%f\t%f\n",
          funcf(x,y),funcg(x,y));
          printf("\n");
        }
}
/*****************************************/
int main()
{
        double xl=-1,xr=1;
        double yl=-1,yr=1;
```

```
        double xstep=0.1,ystep=0.1;
        conf_x(xl, xr,yl,yr, xstep,ystep);
    conf_y(xl, xr,yl,yr, xstep,ystep);
}
/*****************************************/
```

以下のプログラム sqr2.c は設問の(3), (4) に答えたものである.

```
#sqr2.c
#include <stdio.h>
#include <math.h>
#define pai 3.141592653589793/
* 必要なとき使用する * /

/ * 関数 f(z)=z*z * /
/ * 極座標 * /
double funcf(double x, double y)
/*f(z) の実部 * /
{
        return x*x*cos(2*y);
}
double funcg(double x, double y)
/* f(z) の虚部 * /
{
        return x*x*sin(2*y);
}
/* x を一定としたときの像の描く曲線群の関数 * /
double conf_x(double xl, double xr,double yl,
double yr, double xstep, double ystep)
{
        double x,y;
        for(x=xl;x<=xr;x=x+xstep)
        {
          for(y=yl;y<=yr;y=y+ystep)
            printf("%f\t%f\n",
            funcf(x,y),funcg(x,y));
            printf("\n");
        }
}
/ * y を一定としたとき, 像の描く曲線群の関数 * /
double conf_y(double xl, double xr,double yl,
double yr, double xstep, double ystep)
{
        double x,y;
        for(y=yl;y<=yr;y=y+ystep)
        {
            for(x=xl;x<=xr;x=x+xstep)
            printf("%f\t%f\n",
            funcf(x,y),funcg(x,y));
            printf("\n");
        }
}
/*****************************************/
int main()
{
        double xl=0,xr=1;
        double yl=0,yr=pai;
        double xstep=0.1,ystep=yr/20;
        conf_x(xl, xr,yl,yr, xstep,ystep);
    conf_y(xl, xr,yl,yr, xstep,ystep);
}
/*****************************************/
```

次は設問(5), (6) に答えるための Gnuplot のスクリプトである.

```
#sqr.gp
set nologscale
set noparametric
set time
set title 'root of z'
set xlabel ''
set ylabel ''
set zlabel ''
set view 0,0,1,1.5
set isosamples 30
set nohidden3d
set contour base
set no surface
set cntrparam levels 17
set cntrparam levels incremental -2.0, 0.25
i={0.0,1.0}
f(x,y)=(x+i*y)*(x+i*y)
splot [-2.5:2.5][-2.5:2.5]
real(f(x,y)), imag(f(x,y))
set isosamples 10
```

4.3　多項式以外の関数

同様にして，他の複素関数も描くことが可能である．ここでは典型的な問いの形を示す．

分数関数

問

$f(z)=\frac{1}{z}$ は $z=re^{i\theta}$ とおくと $w=f(z)=r^{-1}e^{-i\theta}$

となるので，z 平面の単位円内は w 平面の単位円外に写り，z 平面の単位円外は w 平面の単位円内に写っている．また z 平面での同心円の動きと w 平面での同心円の動きは逆になっている．以上のことを視覚的に確認せよ．

指数関数

問

$f(z)=\exp z$ は周期 $2\pi i$ を持っている．このことを視覚的に確認せよ．

対数関数

問

$f(z)=\log z$ は定義は $z=re^{i\theta}$ に対して，$\log z=\log r+i\theta$ であった．これにより z 平面を全て動いたとしても w 平面では細長い帯状の領域しか動けない．このことを視覚的に確認せよ．

三角関数

問

$\sin z=\frac{e^{iz}-e^{-iz}}{2i}$ であった．

このことから$\sin z$ は有界関数ではないことがわかるが，このことを視覚的に確認せよ．

逆三角関数

問

$z=\tan w$ が逆関数 $w=\arctan z$ の定義と言える．この関数について最初の問題(1)，(2) に答えるためには $\tan w=z$ より，$z=a+ib$ と置くことにより，$\Re\tan w=a$ および $\Im\tan w=b$ の w の曲線を描けば良いことになる．したがって，$\Re\tan w$ と $\Im\tan w$ の等高線を描かせることができればよい．このことを利用して，$w=\arctan z$ のグラフを視覚的に確認せよ．

4.4　等角写像

複素関数の重要な性質についても可視化による確認ができる．

領域 D で連続な複素関数 $f(z)$ が点 $z_0\in D$ で次の性質を持っているとき，$f(z)$ はz_0 において等角であるという．

(1) z_0 で接線をもつ全ての曲線は $w=f(z)$ によって w 平面の点 $w_0=f(z_0)$ をとおり，そこで接線を持つ曲線に写像される．

(2) z_0 で接線をもつ 2 つの曲線が z_0 でなす角と，その 2 つの像曲線が w_0 でなす角とが，大きさ，向き，ともに等しい．

このとき一般的に次の定理が成立している．

定理4.1

複素関数 $f(z)$ が点 z_0 で微分可能で，$f'(z_0)\neq 0$であるとき，$f(z)$ は点 z_0 で等角である．

問

例えば $f(z)=z^2$ を考えてみると，$f'(z)=2z$ であるから，この定理より原点以外では等角であることが解る．$x=c$, $y=d$ は直交する直線であるがこれはそれぞれ放物線に写されている．実際に等角であるかどうを理論的にまた視覚的に確認せよ．

5　2次元流体力学

これまでの内容を具体的な応用（流体力学）に結びつけるために今井功[2]，[3] を参考にした．今井氏には物理と数学を結びつけるための著作が多くある．ここに今井氏に感謝します．

2 次元非圧縮性流体は複素関数と密接な関連があることが知られている．流体の流れは複素関数を用いて表示でき，その像の虚部が一定である複素平面の曲線が流線を表し，実部が一定である複素平面の曲線が等ポテンシャル線を表している．複素関数を可視化する意味の一つの理由をここに求めても良い．また，数学的に基本となる複素関数が物理的にどのように関与しているかを知っていることは複素関数への親しみとしての意味がある．

5.1　非圧縮性流体の方程式

2次元の非圧縮性渦無しの流れに話を限定する．このとき，次のことが知られている．

平面内の点 (x, y) における流体の速度を $(u(x, y), v(x, y))$ とする．平面上の曲線 $(x(t), y(t))$ が流線であるとはこの曲線の接線が流体の速度ベクトルに平行なときを言う．数学的に言えば，ベクトル場を扱っていることになる．

このとき非圧縮性流体の連続の方程式は

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0$$

と書け，渦無しの条件は

$$\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}=0$$

とかけている．

渦無しの条件より，Φ が存在して，

$$u=\frac{\partial \Phi}{\partial x}, v=\frac{\partial \Phi}{\partial y}$$

と書くことができる．Φ を速度ポテンシャルとよんでいる．また連続の方程式より，Ψ が存在して，

$$u=\frac{\partial \Psi}{\partial y}, v=-\frac{\partial \Psi}{\partial x}$$

と表される．$\Psi(x, y)$ を流れの関数とよんでいる．

いま $(x(t), y(t))$ が流線であるとする．このとき，$\Psi(x(t), y(t))$ を t で微分すると，

$$\frac{d\Psi(x(t), y(t))}{dt}=\frac{\partial \Psi}{\partial x}x'(t)+\frac{\partial \Psi}{\partial y}y'(t)=0$$

となり，Ψ は流線上で一定である．$\Psi(x, y)$ が流れの関数とよばれる理由である．

これまでの得られた式を並べ変えてみると

$$u=\frac{\partial \Phi}{\partial x}=\frac{\partial \Psi}{\partial y}$$
$$v=\frac{\partial \Phi}{\partial y}=\frac{\partial \Psi}{\partial x}$$

が得られ，これは複素関数論の基礎となるコーシーリーマンの方程式に他ならず，$\Phi+i\Psi$ が $z=x+iy$ の正則関数であることを示している．そこで

$$f(z)=\Phi(x, y)+i\Psi(x, y), z=x+iy$$

とおき，$f(z)$ を複素速度ポテンシャルと呼んでいる．この様にして，物理と数学が綺麗に結びつく．

ここで，$f(z)$ の両辺を x で偏微分すると

$$f'(z)=u-iv$$

が得られる．この $w=f'(z)$ を複素速度と呼んでいる．w の共役である $\bar{w}$ が流体の速度ベクトル (u, v) を表しているといえる．

$$f(z)=\Phi(x, y)+i\Psi(x, y)$$

であるので $\Re f(z), \Im f(z)$ の等高線がそれぞれ等ポテンシャル線と流線となる．

以上が非圧縮性渦なし流の理論であるが，このように複素関数と流体力学は密接に関係していることがわかる．そこで，これを前提として具体的な流れを扱う．

5.2　単一の流れ

最も基本的な流れを考察する．

5.2.1　一様な流れ

$U>0$ として，次の関数で表される流れを考える．

$$f(z)=Uz$$

複素速度は $w=U$ であるので，速度ベクトルは一定である．また $z=x+iy$ であるので，

$$\Psi=Ux, \Phi=Uy$$

であり，等ポテンシャル線と流線はそれぞれ，y 軸および x 軸に平行な直線である．

問

（1）$\alpha>0$ のとき，$f(z)=Ue^{-i\alpha}z$ の等ポテンシャル線と流線はどのような曲線を描くか．

（2）Gnuplot を利用して，実際に等ポテンシャル線と流線を描いてみよ．

（2）についての解説

Gnuplot は等高線が簡単に描けるので流体力学の解析には非常に便利である．以下のスクリプトでは $\Re f(x+iy)$ の等高線を描かせている．

```
set nologscale
set noparametric
set time
set title ’Re(f(z))=const : f(z)=z’
set xlabel ”
set ylabel ”
```

```
set zlabel ”
set view 0,0,1,1.5
set isosamples 30
set nohidden3d
set contour base
set nosurface
set cntrparam levels 17
set cntrparam levels incremental -2.0, 0.25
i={0.0,1.0}
U=1.0
f(x,y)=U*(x+i*y)
splot [-2.5:2.5][-2.5:2.5] real(f(x,y))”
set isosamples 10
```

5.2.2　わき出しと吸い込み

m を実数として，$f(z) = m \log z$ で表される流れを考える．

このとき複素速度は $f'(z) = \dfrac{m}{z}$ である．$z = re^{i\theta}$ を用いると，$f(z) = m(\log r + i\theta)$ であるので，

$$\Phi = m\, log\, r, \Psi = m\theta.$$

等ポテンシャル線と流線はそれぞれ，原点を中心とした円群および原点からの放射状の直線群である．

$$v_r = \frac{\partial \Phi}{\partial r} = \frac{m}{r}$$

であるので，$m > 0$ であれば $v_r > 0$ となり，$m < 0$のとき $v_r < 0$. そこで，それぞれのに対応する流れをわき出し，吸い込みと呼んでいる．

問

わき出しと吸い込みの現象を視覚的に確認せよ．

5.2.3　渦糸

k を実数として，$f(z) = ik \log z$ で表される流れを考える．

このとき複素速度は $f'(z) = i\,\dfrac{k}{z}$ である．$z = re^{i\theta}$ を用いると，$f(z) = ik(\log r + i\theta)$. これより，

$$\Phi = -k\theta, \Psi = k \log r.$$

等ポテンシャル線と流線はそれぞれ，原点からでる放射状の線群および原点を中心とした円群である．また，

$$v_\theta = \frac{\partial \Phi}{\partial \theta} = -k$$

である．$k > 0$ のとき $v_\theta < 0$, $k < 0$ のとき$v_\theta > 0$であるので，原点に対して，時計回り，反時計回りに流れている．が，このような流れを渦糸と呼んでいる．

問

渦糸の流れを視覚的に確認せよ．

5.3　流れの重ね合わせ

基本の流れを重ね合わせると新たな流れが得られる．数学的には複素関数の加法を考える．

5.3.1　一様流とわき出し

一様流とわき出しを重ねあわせてみる．

$$f(z) = Uz + m\log z$$

$z = re^{i\theta}$ を用いると，

$$\begin{aligned} f(z) &= Ure^{i\theta} + m(\log r + i\theta) \\ &= (Ur\cos\theta + m\log r) + i(Ur\sin\theta + m\theta) \end{aligned}$$

x 軸は明らかに流線であるが，流線は他にもあり，右側に開いた半無限の曲線がある．

また複素数速度は

$$f'(z) = U + \frac{m}{z}$$

であり，$z \to 0$ で w $\to U$ となり一様流であり，原点の近くではわき出しの振る舞いをしていることがわかる．

問

あたかも特定の物体がおかれている様に流体が流れる様子が観察されることを視覚的に確認せよ．

5.3.2　わき出しとすいこみ

$z = a$ と $z = -a$ に同一係数 m をもつわき出しとすいこみを重ねると，関数は次となる．

$$f(z) = m\log(z - a) - m\log(z + a)$$

$$z - a = r_1 e^{i\theta_1}, z + a = r_2 e^{i\theta_2}$$

とおくと，

$$\Phi + i\Psi = m\{\log \frac{r_1}{r_2} + i\,(\theta_1 - \theta_2)\}$$

従って

$$\Phi = m\log\frac{r_1}{r_2}, \Psi = m(\theta_1 - \theta_2)$$

問

等ポテンシャル線と流線は 2 点からの距離の比と，見込む角がそれぞれ一定である点の軌跡である．両者ともに円となることを理論的および視覚的に確認せよ．

5.3.3　渦対

次に $z = a$ と $z = -a$ に同一係数 k をもつ両方時計回りの渦があるとする．このとき，関数は次となる．

$$f(z) = ik\log(z-a) + ik\log(z+a)$$

$$z - a = r_1 e^{i\theta_1},\ z + a = r_2 e^{i\theta_2}$$

とおくと，

$$\Phi + i\Psi = ik\{\log r_1 r_2 + i(\theta_1 + \theta_2)\}$$

従って

$$\Phi = -k(\theta_1 + \theta_2),\ \Psi = k\log r_1 r_2$$

問

流線は 2 点からの距離の積が一定である点の軌跡である．これは Cassini の卵形と呼ばれているが，この形を視覚的に確認せよ．

5.4　速度ベクトル場

すでに述べた様に，$(u(x, y), v(x, y))$ を速度ベクトルとする流体は流線上の点の座標を (x, y) に対してベクトル場が与えられていると考えることができる．

このベクトル場を描く視覚化プログラムを考察する．

各点 (x, y) を始点として，ベクトル $(u(x, y),\ v(x, y))$ の方向を矢線で描いていく．このベクトルの大きを単位ベクトルにそろえることも考えられるが，分母が 0 になることもあるので，以下の工夫を行う．

$$N = \sqrt{u(x, y)^2 + v(x, y)^2}$$

とおき，ベクトル $\frac{1}{1+N}(u(x, y), v(x, y))$ を描くことにする．このベクトルの大きさは常に $\frac{N}{1+N}$ となることに注意する．そして，N が 0 になればベクトルも 0 になるので速度ベクトルの大きさもある程度反映できている．

さて，具体的にわき出しであるときを例として考える．

$f(z) = \log z$

であるので，これを微分すると

$f'(z) = \frac{1}{z} = u - iv$

が得られる．

これを計算すると

$$u(x, y) = \frac{x}{x^2+y^2},\ v(x, y) = \frac{y}{x^2+y^2}$$

となり具体的な速度ベクトル場が得られる．

Gnuplot による必要なデータをはき出すプログラムが以下の wakidashi.c である．

```
/*wakidashi.c*/
/*方向場ベクトル場(u(x,y),v(x,y)) を描く*/
/*gnuplot で描くデータの掃き出し*/
/*plot "data" with vector*/

/*流体のベクトル場に応用*/
/*f(z)=log z, f'(z)=1/z=u-iv 原点がわき出し*/
/*(u(x,y),v(x,y))*/

#include <stdio.h>
#include <math.h>
#define beginx    -1. /*描く x の左端*/
#define endx    1. /*描くx の右端*/
#define beginy    -1./*描くy の左端*/
#define endy    1. /*描くy の右端*/
#define step    0.1 /*ステップ幅*/

float u(float x,float y )
{
   return x/(x*x+y*y);
}
float v(float x,float y )
{
   return y/(x*x+y*y);
}
int main()
{
   float x, y,deltax,deltay,norm;
   for( x = beginx; x <= endx; x = x + step )
                  {
   for( y = beginy; y <= endy; y = y+ step )
                  {
   norm=sqrt(u(x,y)*u(x,y)+v(x,y)*v(x,y));
   deltax=u(x,y)/(1+norm)*step;
   deltay=v(x,y)/(1+norm)*step;
```

```
    printf( "%f %f %f %f\n", x, y ,
                              deltax,deltay);
                                }
                            }
     return 0;
}
```

5.5 合成関数による流れ

新たな関数を発生させる機構として関数の合成がある．すなわち $f(z)$ で表される流れがあったとする．このとき，$z = g(\zeta)$ である複素正則関数に対して合成関数 $F(\zeta) = f(g(\zeta))$ を作ると ζ 変数に対する新たな流れを表す関数が得られたと考えることができる．

5.5.1　関数の合成により保存される性質

複素関数の一般論として，以下が知られている．

定理5.1

正則関数 $z = g(\zeta)$ による等角写像に際して，等ポテンシャル線と流線はそれぞれ等ポテンシャル線と流線に写される．

定理5.2

正則関数 $z = g(\zeta)$ による等角写像に際して，わき出しと渦糸はそれぞれわき出しと渦糸に写される．

以上より，流れの本質的な性質は等角写像により変化していないことがわかる．このことをは理論的に証明することが求められるが，これから得られる流体力学的内容は非常に興味深い．

5.5.2　リーマンの写像定理

複素関数論において極めて重要かつ意味深い次のリーマンの写像定理が成立している．

定理5.3

z 平面と ζ 平面に単連結領域 D，D' がそれぞれあるとき，D を D' に等角に写像する正則関数 $z = g(\zeta)$ が存在する．ただし，D，D' のどちらか一方だけが全平面の場合を除く．

リーマンの写像定理より D を D' に等角写像する正則関数が存在するのであるから，これはある領域で性質が良く分かっている流れがあるとき，他の領域での流れが解ることを意味している．

しかし，リーマンの写像定理は存在定理であって，関数の形 $z = g(\zeta)$ を具体的に示してくれるわけではない．具体的な対象には個別にあたるしか無いが，存在は保証されているので，安心して探せばよいことになる．

これを考察するとき，複素関数の鏡像の原理が有効であることが知られているが，ここでは 2 つの実例を紹介する．

5.5.3　円の内部を上半平面へ変換

$$z = -\frac{\zeta - i}{\zeta + i}$$

は原点を中心とした半径 1 の円の内部は上半平面へ写され，円の外部は下半平面に写されている．

問

以上のこと理論的に，また視覚的に確認せよ．

応用例

平面に沿った一様流を利用して，上の変換で円に沿った一様流を観察することができるようになる．

x 軸にに沿った一様流を考えと，$f(\zeta) = U\zeta$ であるので，z 変数の関数として，次が得られる．

$$f(z) = iU\frac{1 - z}{1 + z}.$$

問

無限に長い壁があるときの下半平面を流れる一様流が円の外部でどの様に流れるかを可視化して確認してみよ．

5.5.4　平板の外部から円の外部へ変換

$$z = \frac{1}{2}\left(\zeta + \frac{1}{\zeta}\right)$$

は z 平面の実軸上（−1, 1）に沿って切れ目を入れた領域を ζ 平面の原点を中心とした半径 1 の円の外部に写す変換となる．これをジューコフスキー変換と呼んでいる．

問

以上のこと理論的に，また視覚的に確認せよ．

応用例

まず，x 軸にに沿った一様流を考えると，$f(z) = Uz$ であるので，円に沿った一様流はζ変数の関数として

$$f(\zeta) = \frac{1}{2} U(\zeta + \frac{1}{\zeta})$$

となる．

つぎに，ζ 平面に渦糸$F(\zeta) = ik \log \zeta$ があったとする．

ジューコフスキー変換によって z 平面のどのような流れになるかを考察する．

$\zeta = \rho e^{i\theta}$ とおくと

$$\Phi = -k\theta, \Psi = k \log \rho$$

一方，ジューコフスキー変換より

$$z = \frac{1}{2}(\rho e^{i\theta} + \frac{1}{\rho} e^{-i\theta})$$

となり

$$x = \frac{1}{2}(\rho + \frac{1}{\rho}) \cos\theta, y = \frac{1}{2}(\rho - \frac{1}{\rho}) \sin\theta$$

問

等ポテンシャル線が双曲線となり，流線が楕円となることを理論的に示し，視覚的に確認せよ．

5.5.5　飛行機の翼

ジューコフスキー変換

$$z = \frac{1}{2}(\zeta + \frac{1}{\zeta})$$

で原点以外を中心としたζ平面の円が z 平面のどのような形になるかを考察する．

$\zeta = u + iv$ とおくと，中心を$\alpha = a + ib$ とする半径 r の円は

$$(u - a)^2 + (v - b)^2 = r^2$$

まず，この円が $\zeta = 1$ と $\zeta = -1$ を通る様にする．すなわち z 平面の切れ目 (−1, 1) の右端と左端は動かない様にしておく，$a = 0$ となり$1 + b^2 = r^2$ である．このときは弓の形が観察される．

次に，この円が $\zeta = 1$ を通る様にしておく．すなわち z 平面の切れ目 (−1, 1) の右端は動かない様にしておく．この条件より $(1 - a)^2 + b^2 = r^2$ を課すことになるが，このとき飛行機の翼の形が観察される．

以上より円の外部と飛行機の翼の形に対応がつき，円の外部の流体の流れの情報から飛行機の翼の外の空気の流れを理解する事が可能となる．

以下はジューコフスキー変換により Gnuplot によるデータをはき出すＣプログラムである．

```
#include <stdio.h>
#include <math.h>
#define pai 3.141592653589793
/*Joukowski 変換z=0.5*(w+1/w)*/
/*w 平面の円を z 平面の飛行機の翼に変換*/
/*gcc joukowski.c でコンパイルし，
./a.out > test.dat でデータをはき出し*/
/*gnuplot でload "joukowski.gp" */
double freal(double x,double y,
double a,double b)
{
  return x*cos(y) +a;
}
double fimag(double x,double y,
double a,double b)
{
  return x*sin(y)+b;
}

double fbunbo(double x,double y,
double a,double b)
{
  return freal(x,y,a,b)* freal(x,y,a,b)+
          fimag(x,y,a,b)* fimag(x,y,a,b);
}
double funcf(double x, double y,
double a,double b)
{
  return 0.5*(1+
  1/fbunbo(x,y,a,b))*freal(x,y,a,b);
        }
double funcg(double x, double y,
double a,double b)
{
  return 0.5*(1-
```

```
    1/fbunbo(x,y,a,b))*fimag(x,y,a,b);
}
/*******************************************/
int main()
{
        double x,y,yl,yr,ystep,a,b;
        yl=0;yr=2*pai;ystep=0.1;
        a=-0.1;b=0.5;
        /*a=-0.1;b=0.5; わりと良い形？*/
        /*a=0 の時は対称型*/
        x=sqrt((1-a)*(1-a)+b*b);
        /*(1,0) を通る設定*/
        for(y=yl;y<=yr;y=y+ystep)
        printf("%f\t%f\n",
        funcf(x,y,a,b),funcg(x,y,a,b));
        printf("\n");
}
/*******************************************/
```

問

プログラム joukowski.c において円の中心を$\alpha = a + ib$ を適当に変化させてみて翼の形の変化を観察せよ．

講義では複素数関数を利用した数値計算の方法も紹介し，具体的な計算もＣ言語で行った．本稿では，視覚化して理解する方法に限定したため省略した．

実際にこのような講義を実践してみたところ，学生は十分に複素関数に興味を示したこと，そして理論の重要性も理解できたと受け取ることができた．一つの授業実践の例として，記録にとどめておくことも重要であると考えた．

参考文献

[1]奥村晴彦，Ｃ言語によるアルゴリズム事典，技術評論社，1991
[2]今井功，等角写像とその応用，岩波書店，1979
[3]今井功，流体力学と複素解析，日本評論社，1981

（平成27年９月30日受理）