College Analysis 総合マニュアル

―　数　　学　―

目次

１．数式

ここでは、プログラム全体を通して利用できる関数を紹介しておく。

実数数式内で利用可能な関数と定数

sin(), cos(), tan(), atan(), exp(), log(), log10(), sinh(), cosh(), tanh(), int(), abs()

theta() : x>=0 のとき theta(x)=1, x<0 のとき theta(x)=0

領域ごとに関数を分ける場合に利用できる。

pulse() : x=0 のとき pulse(x)=1, x≠0 のとき pulse(x)=0

システムダイナミクス等で利用する。if 文のような使い方もできる。

def() : x>=0 のとき def(x)=0, x<0 のとき未定義

グラフなどを描くとき、未定義の領域には描画しないことを利用して、描画区間を指定できる。

pi [=π], ep [=e], rnd [=Rnd()], nrnd 標準正規乱数

数学定数と一様乱数、標準正規乱数

ival [グリッドに対して利用可能で、先頭行から 1,2,3,...]

複素数数式内で利用可能な関数と定数（数式内で複素数が利用可能）

sin(), cos(), tan(), atan(), exp(), log(), log10(), int(), abs(), adj()[共役]

pi[=π], ep[=e], rnd[=Rnd()]

数学定数と一様乱数、標準正規乱数

ival [グリッドに対して利用可能で, 先頭行から 1,2,3,...]

２． １変数関数グラフ

１変数関数グラフは、 $y = f(x)$ の形の関数のグラフである。College Analysis のメニュー［分析－数学－グラフ－１変数関数グラフ］を選択すると図 1 のような実行画面が表示される。ここで、中央のテキストボックスの関数は後で入力した。

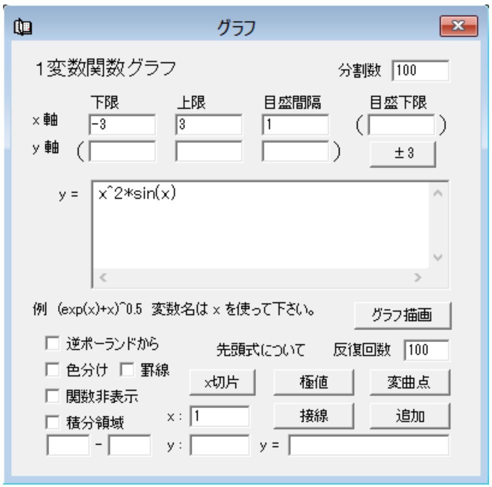


図１　１変数関数グラフ実行画面

「y＝ 」のテキストボックスに、描きたいグラフの関数形の右辺を入力し、x 軸の下限、上限、目盛間隔のテキストボックスに必要な数値を入力して、「グラフ描画」ボタンをクリックすると、y 軸の目盛が適当な値になり、グラフが描画される。例えば、数式として $y = \sin x$（テキストボックスには右辺 sin(x)を入力する）、下限を-3、上限を 3、目盛間隔を 1 とすると、表示結果は図 2a のようになる。また、同じ数式で、下限を-pi、上限を pi、目盛間隔を pi/2 とすると、表示結果は図 2b のようになる。

表示結果は範囲に数値を用いると図 2a のように軸目盛が数値となり、「pi」を使った値を用いると図 2b のようなπを使った表示になる。y 軸の目盛はメニューに値を入力することで変更が可能である。

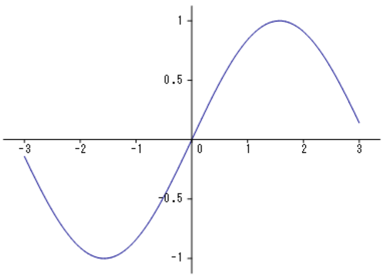


図 2a　$y = \sin x$ のグラフ１

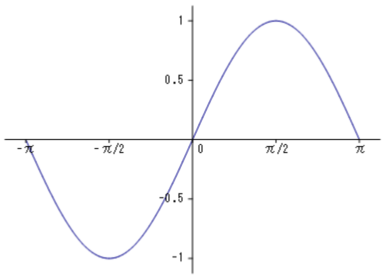
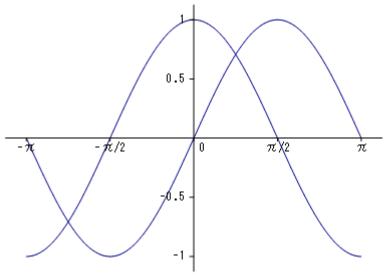


図 2b　$y = \sin x$ のグラフ２

「数式 y＝ 」テキストボックスに改行して複数の数式の右辺を入力することにより、複数のグラフを同時に表示できる。図 3a に $y = \sin x$, $y = \cos x$ のグラフ、図 3b に $y = x$, $y = 1/x$, $y = x + 1/x$ のグラフを示す。



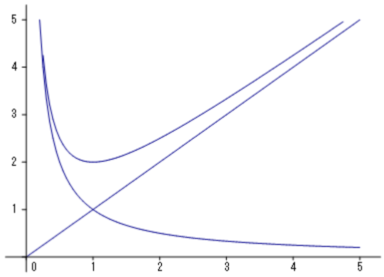


図 3a　$y = \sin x$, $y = \cos x$ グラフ　図 3b　$y = x$, $y = 1/x$, $y = x + 1/x$ のグラフ

このプログラムでは、1 行目に書いた関数の x 切片を求めることができる。例えば、$y = x^5 - x^3$ のグラフを描くと、図 4 のようになるが、描画メニューの「x 切片」ボタンをクリックすると、図 5 のように x 切片が表示される。

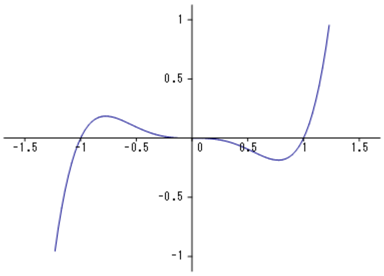


図 4　$y = x^5 - x^3$ のグラフ


計算結果（表示範囲内x切片）

	解 1	解 2	解 3
X	-1.0000	0.0000	1.0000
y値	0.0000	0.0000	0.0000
収束解の個数	19	54	27

図 5　$y = x^5 - x^3$ の x 切片

ここで、計算は初期値をランダムに x 軸表示範囲内に取り、ニュートン法を用いている。「収束解の個数」は、描画メニューで乱数発生の「反復回数」を 100 回として実行した場合の各 x 切片に収束した回数である。x 切片は表示境界上の値も含めるようにしている。

また、描画メニューの「極値」ボタンをクリックすると、表示領域内の極値を求めることができる。図 6 にその結果を示す。同様に、描画メニューの「変曲点」ボタンをクリックすると、変曲点の座標も求めることができる。図 7 にその結果を示す。


計算結果（表示範囲内極値）

	解 1	解 2	解 3
X	-0.7746	0.0000	0.7746
極値	0.1859	0.0000	-0.1859
判定	極大	他停留点	極小
収束解の個数	34	34	31

図 6　$y = x^5 - x^3$ の極値


計算結果（表示範囲内変曲点）

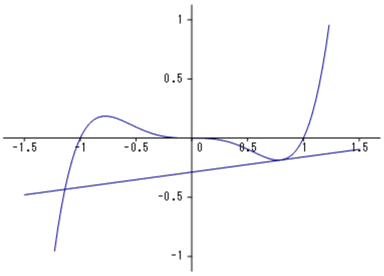
	解 1	解 2	解 3
X	-0.5477	0.0000	0.5477
y値	0.1150	0.0000	-0.1150
収束解の個数	41	18	32

図 7　$y = x^5 - x^3$　の変曲点

さらにこのプログラムでは先頭に記述した関数の、指定した x 座標における、接線の方程式を求めることができる。描画メニューの「x = 」テキストボックスに接点の x 座標を入力し、「先頭式接線」ボタンをクリックすると、その下の「y = 」テキストボックスに接線の方程式の右辺部分が表示される。例えば、図 4 の $x = 0.8$ での接線の方程式は、$y = 0.128x - 0.287$ となり、その式を「追加」ボタンで中央の「y =」テキストボックスに追加コピーして描画すると、図 7a のように表示される。このままでは接線が長いと思い、$0.2 \le x \le 1.4$ の範囲で描画しようと考えると、関数定義を以下のように変更して描画する。

0.128*x-0.287+def(x-0.2)+def(1.4-x)

def( )関数が $x \ge 0$ だけで $\mathrm{def}(x) = 0$ と定義されているため、結果は図 7b のようになる。



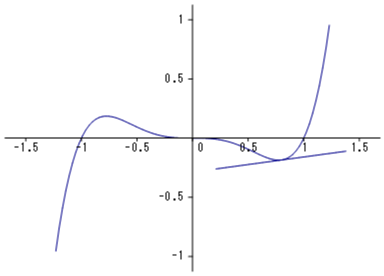
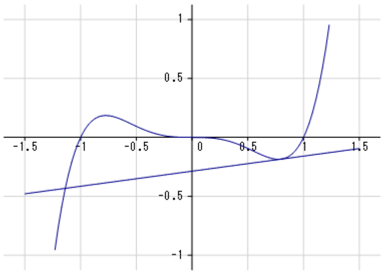


図 7a　接線を追加したグラフ　　　図 7b　接線の範囲を指定したグラフ

接線を追加した画面を「罫線」チェックボックスにチェックをして表示すると、図 8a のようになる。また、「非表示」チェックボックスをチェックして、グラフを表示せずに描いたものが図 8b である。



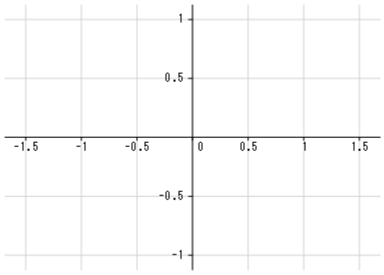
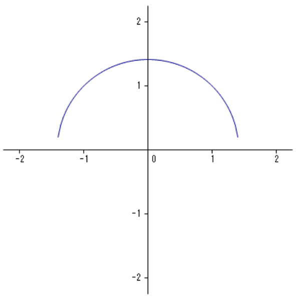


図 8a　罫線表示　　　図 8b　グラフ非表示

教員が問題を作る際にも、実際に図や数値を眺めながら作る方がかなり効率的である。図 8b の軸のみの表示は、グラフを確かめた上で軸のみ表示してくれるので、グラフを描かせるような演習のプリント作成などで役に立つ。

このグラフ表示のプログラムには問題がある。例えば $y=\sqrt{2-x^2}$ のグラフを描こうとした場合、デフォルトの「区間分割数」100 では図 9a のように表示される。これは、境界の近くで、描画要素の端が、定義されない領域に入るためで、「区間分割数」1000 にしても、図 9b のように多少は改善されるが、まだ隙間は残る。「区間分割数」を極端に多くしたり、数字をうまく調節したりして、繋がったように描くことは可能であるが、これでは問題を解決したとは言えない。



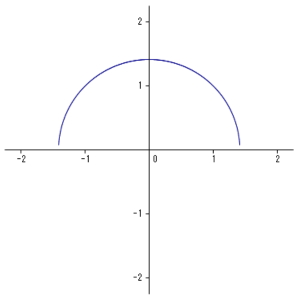


図 9a　$y=\sqrt{2-x^2}$（分割数 100）　　図 9b　$y=\sqrt{2-x^2}$（分割数 1000）

この問題を解決するのは、次章で述べる２次元パラメータ表示関数である。

問題１　$y=x^2-2x-3$ の極値とグラフを求めよ。

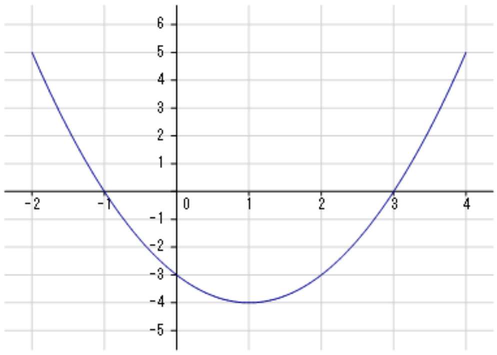
y 軸との交点　　$y=$

x 軸との交点　　$x=$

極値

(　　　,　　　)［極大・極小・停留点］

グラフの概形を描け。



問題２　$y=x^3-6x^2+9x$ の極値とグラフを求めよ。

y 軸との交点　　$y=$

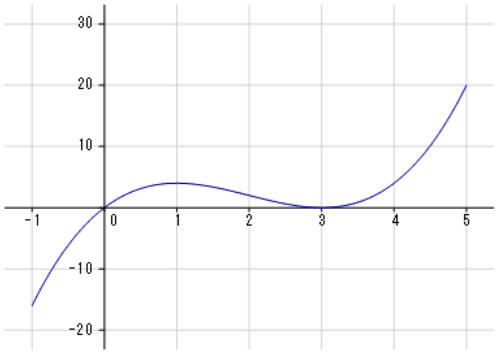
x 軸との交点　　$x=$

極値

(　　　,　　　)［極大・極小・停留点］

(　　　,　　　)［極大・極小・停留点］

グラフの概形を描け。



問題３　$y=x^3$ の極値とグラフを求めよ。

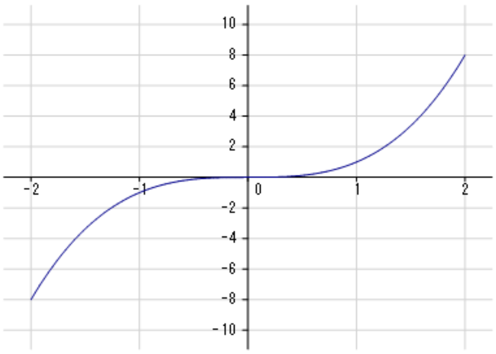
y 軸との交点　　$y=$

x 軸との交点　　$x=$

極値

(　　　,　　　)［極大・極小・停留点］

グラフの概形を描け。



問題４　$y=x+\dfrac{1}{x}$ のグラフを描き極値を求めよ。

y 軸との交点　　$y=$

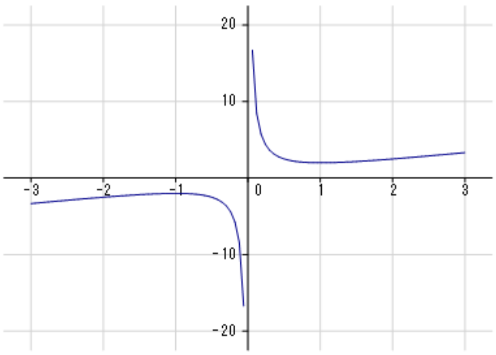
x 軸との交点　　$x=$

極値

(　　　,　　　)［極大・極小］

(　　　,　　　)［極大・極小］

グラフの概形を描け。



問題５　$y=e^{x}-x$　のグラフを描き、極値を求めよ。

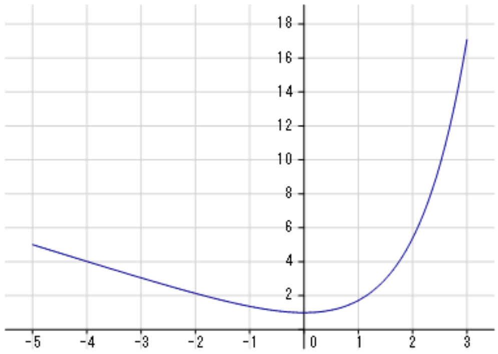
y軸との交点　　　$y=$

x軸との交点　　　$x=$

極値

(　　　　,　　　　)［極大・極小］

グラフの概形を描け。



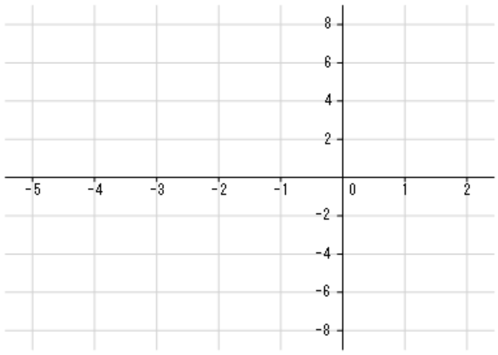
演習１　$y=-x^{2}-3x+4$　の極値とグラフを求めよ。

y軸との交点　　　$y=$

x軸との交点　　　$x=$

極値

(　　　,　　　)［極大・極小・停留点］



演習２　$y=e^{-x^{2}}$　のグラフを描き、極値と変曲点を求めよ。

y軸との交点　　　$y=$

x軸との交点　　　$x=$

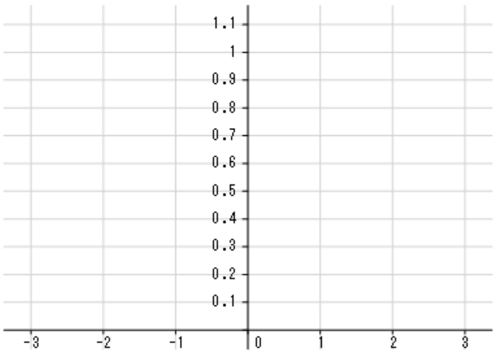
極値

(　　　　,　　　　)［極大・極小］

変曲点

(　　　　,　　　　)

(　　　　,　　　　)



問題１解答　$y=x^2-2x-3$ の極値とグラフを求めよ。

y 軸との交点　　　$y=$

x 軸との交点　　　$x=$

(　　　,　　　)［極大・極小・停留点］

問題２解答　$y=x^3-6x^2+9x$ の極値とグラフを求めよ。

y 軸との交点　　　$y=$

x 軸との交点　　　$x=$

(　　　,　　　)［極大・極小・停留点］

(　　　,　　　)［極大・極小・停留点］

問題３解答　$y=x^3$　の極値とグラフを求めよ。

y 軸との交点　　　$y=$

x 軸との交点　　　$x=$

(　　　,　　　)［極大・極小・停留点］

問題４解答　$y=x+\dfrac{1}{x}$　のグラフを描き極値を求めよ。

y 軸との交点　　　$y=$

x 軸との交点　　　$x=$

(　　　　,　　　　)［極大・極小］

(　　　　,　　　　)［極大・極小］

問題５　$y=e^x-x$　のグラフを描き、極値を求めよ。

y 軸との交点　　　$y=$

x 軸との交点　　　$x=$

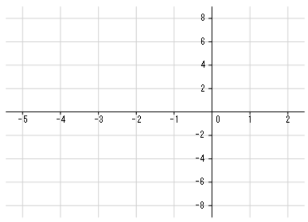
(　　　　,　　　　)［極大・極小］

演習１解答　$y=-x^2-3x+4$　の極値とグラフを求めよ。

y 軸との交点　　　$y=$

x 軸との交点　　　$x=$

(　　　,　　　)［極大・極小・停留点］



演習２解答　$y=e^{-x^2}$　のグラフを描き、極値と変曲点を求めよ。

y 軸との交点　　　$y=$

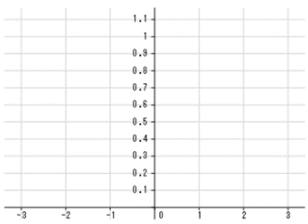
x 軸との交点　　　$x=$

(　　　　,　　　　)［極大・極小］

変曲点

(　　　　,　　　　)

(　　　　,　　　　)



３． ２次元パラメータ表示関数グラフ

前章の終りで、グラフを $y=f(x)$ の形で描く場合の定義域の境界の問題点を述べたが、これを解決する１つの方法がパラメータによる関数表示である。2 次元パラメータ表示関数は、ある範囲をとるパラメータ u を用いて、関数を $x=f(u),\ y=g(u)$ の形式で表現するものである。極座標表示の関数 $r=f(\theta)$ もパラメータ表示関数で、$r=f(u)$，$\theta=u$ のように表される。サイクロイドやリサージュ曲線はパラメータ表示関数の有名な例である。

この章では２次元パラメータ表示関数グラフの描画プログラムについて説明する。メニュー［分析－数学－グラフ－２次元パラメータ表示関数］を選択すると、図 1 に示す実行画面が表示される。

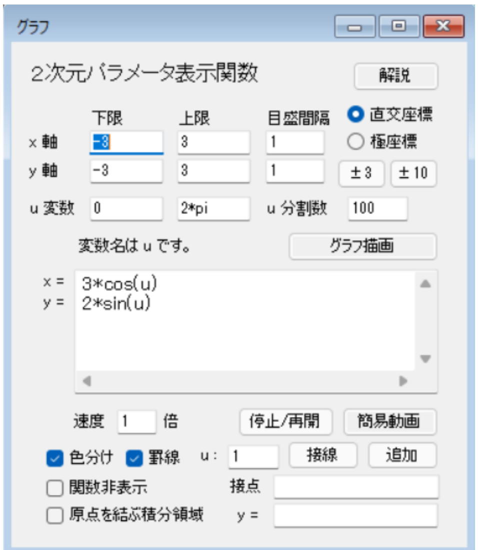


図１ ２次元パラメータ表示関数グラフ実行画面

利用法は、「u 変数」テキストボックスにパラメータ u の範囲を記述し、数式テキストボックスの、「x =」、「y =」の位置に u で表わされた関数形を記述して、「グラフ描画」ボタンをクリックする。上の x 軸と y 軸の下限、上限、目盛間隔は必要があれば記入する。空白の場合は、図が収まる範囲で適当な値となる。2 章の終りで述べた半円を描くには以下のようにする。また、円は描画範囲を 2 倍にする。

描画範囲：$0 \le u \le \pi$ （テキストボックス内で π は pi で表す。）

関数： $x=\sqrt{2}\cos u,\ y=\sqrt{2}\sin u$

ここで２つの関数は、間にカンマを入れず、図 1 の表示のように 2 行に並べて記入する。図 2a に半円、図 2b に円の描画結果を示す。ここでパラメータ u は、極座標の角度座標 θ の役割を果たす。

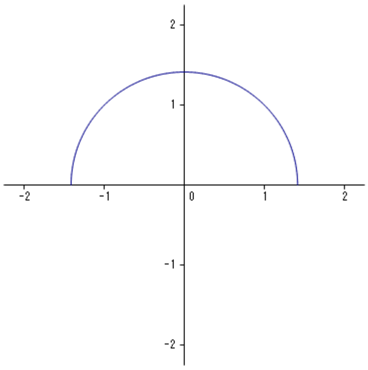


図 2a　半円

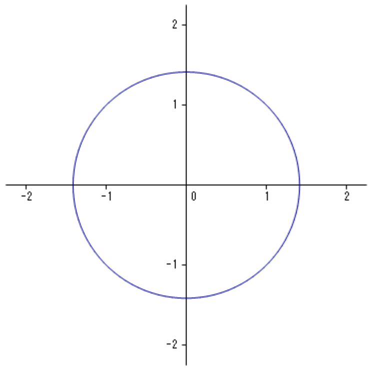


図 2b　円

パラメータ表示関数は、パラメータ値の変化による描画過程がアニメーションで表示されると効果的である。このプログラムでは「簡易動画」ボタンをクリックすると、図 3 のように描画過程とパラメータ値を表示する。

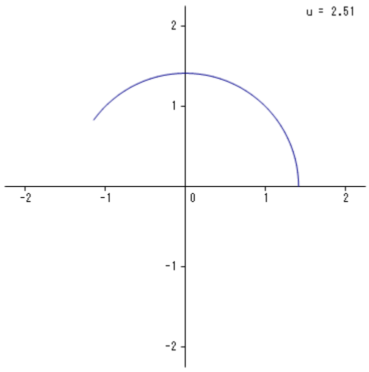


図 3　描画過程

描画は「停止／再開」ボタンや「速度」テキストボックスによって制御できる。

「接線」ボタンをクリックすることで、パラメータ u の値により、接点の位置と接線の方程式を求めることができる。さらに「追加」ボタンによって、求められた接線を図 4 のように描画に追加して行くことができる、これは直交座標でも極座標でも同様である。

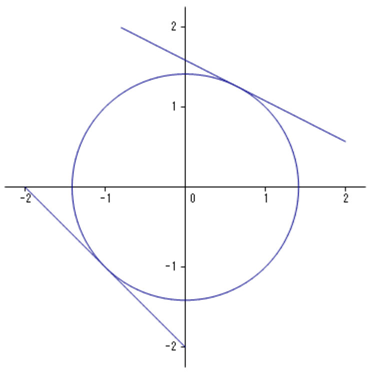


図 4　接線の追加

複数組の式を使った２次元パラメータ表示関数の例を図 5 に描いておく。これらの関数

は間にカンマを入れず、複数行に並べて記入する。

$y=1$ と $x=1$

描画範囲：$-2 \le u \le 2$

関数：　$x=u,\ y=1$，　$x=1,\ y=u$

放物線（軸の上限と下限も記入）

描画範囲：$-2 \le u \le 2$

関数：　$x=u,\ y=u^2$，　$x=u^2,\ y=u$

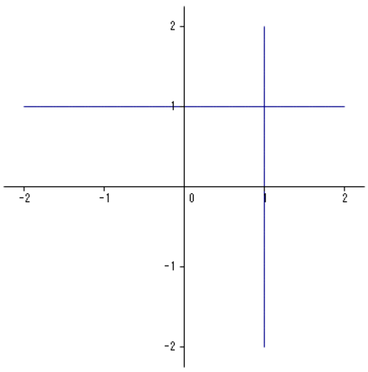


図 5a　$y=1$ と $x=1$

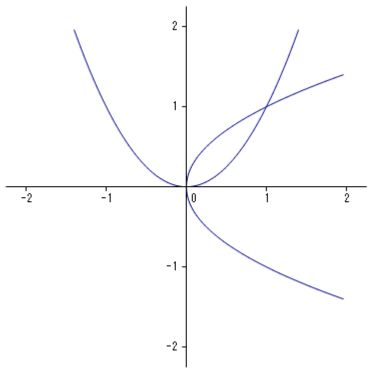


図 5b　放物線

双曲線

描画範囲：$-2 \le u \le 2$

関数：　$x=\cosh u,\ y=\sinh u$，　$x=-\cosh u,\ y=\sinh u$

らせん

描画範囲：$0 \le u \le 6\pi$

関数：　$x=u\cos u,\ y=u\sin u$，　$x=-u\cos u,\ y=-u\sin u$

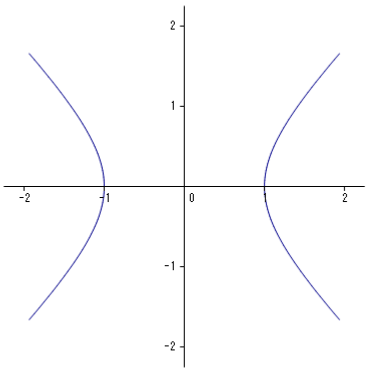


図 5c　双曲線

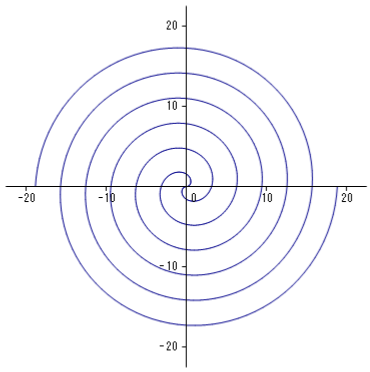


図 5d　らせん

描画メニューの「陰関数描画」は、等高線を描くアルゴリズムを用いて陰関数をグラフ化する機能である。テキストボックスに $f(x,y)=0$ の形式（$y=f(x)$ なども可能）で方程式を入力し、表示領域を両軸とも設定して（この場合は必須）、「描画」ボタンをクリックすると、方程式の解を表すグラフが表示される。描画の方法は領域を三角形（四角形を２つの

三角形で分ける）で区切り、３角形の頂点の値を用いて、符号の変化する線分上を比例分割して、等高線が通る点を２点求め、それを繋ぐ。これをすべての３角形で実行して小さな線分の集まりとして等高線を描く。精度は三角形（元々は四角形）に区切る「分割数」で与える。

例として図６に、この方法で描いた、$sin^2 x - \tan y + x + y = 0$ と $e^{x+y} - \sin y + xy - 1 = 0$ の２つのグラフを示す。いずれも、単純に１変数について解くことも、パラメータ表示にすることもできないグラフである。

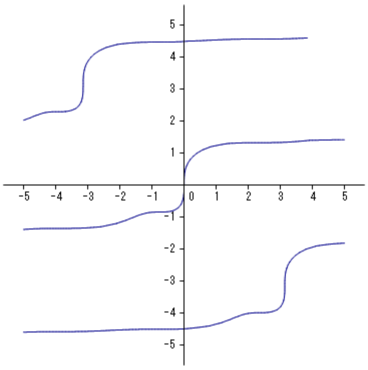


図 6a　$sin^2 x - \tan y + x + y = 0$ のグラフ

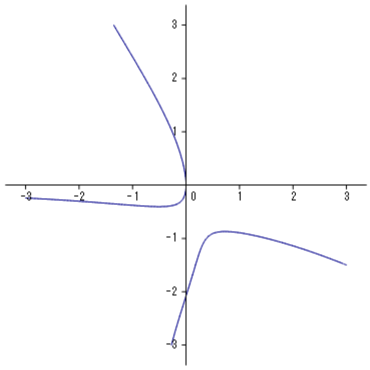


図 6b　$e^{x+y} - \sin y + xy - 1 = 0$ のグラフ

問題

１）$-3 \le x \le 3$，$-3 \le y \le 3$ の領域において、$0 \le u \le 2\pi$ の範囲で以下の関数を描け。

$$\begin{cases} x = 3\cos u \\ y = 2\sin u \end{cases}$$

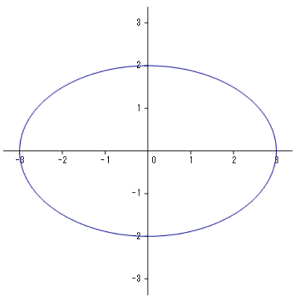
２）このグラフの描画過程をアニメーションせよ。

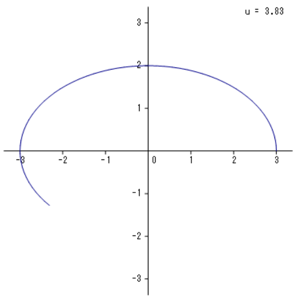
３）$u = 1$ における接線を求めよ。

４）上のグラフは、$\dfrac{x^2}{9} + \dfrac{y^2}{4} = 1$ とも表されるが、陰関数表示を用いて描画せよ。

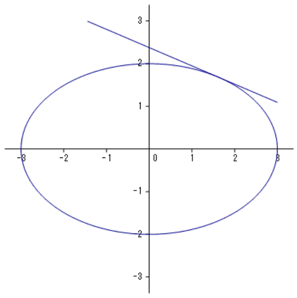
問題解答

１）２）



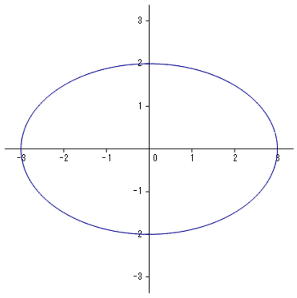


３）$u=1$における接線を求めよ。



y=0.4281*x+2.3768

４）上のグラフは、$\dfrac{x^2}{9}+\dfrac{y^2}{4}=1$とも表されるが、陰関数表示を用いて描画せよ。



４． ２変数関数グラフ

２変数関数グラフは、$z=f(x,y)$ の形の関数のグラフである。College Analysis のメニュー［分析－数学－グラフ－２変数関数グラフ］を選択すると図 1 のような描画メニューが表示される。

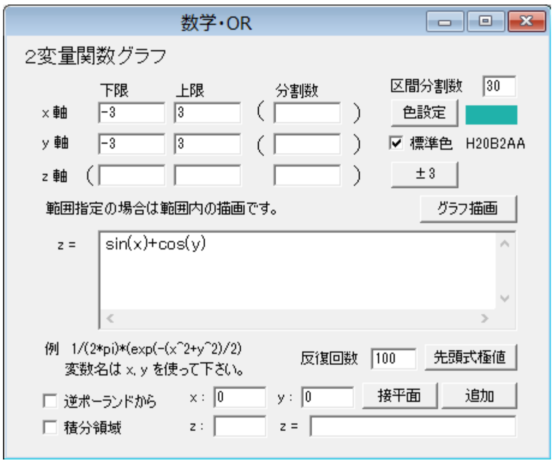


図１ ２変数関数グラフ実行画面

「z＝ 」のテキストボックスに描きたいグラフの関数形の右辺を入力し、x 軸の下限・上限、y 軸の下限・上限テキストボックスに数値を入力して、「グラフ描画」ボタンをクリックすると、z 軸目盛が適当な値になり、グラフが描画される。必要があれば、それぞれの目盛の分割数、z 軸の下限・上限を指定する。x 軸、y 軸の下限・上限でπ表示にすると、軸目盛は分数のπ表示となる。このグラフは１変数関数グラフと異なり、軸目盛の間隔は分割数で表わす。図 2a に $z=xe^{-x^2-y^2}$ のグラフを、図 2b に $z=\sin x+\cos y$ のπ表示のグラフを示す。

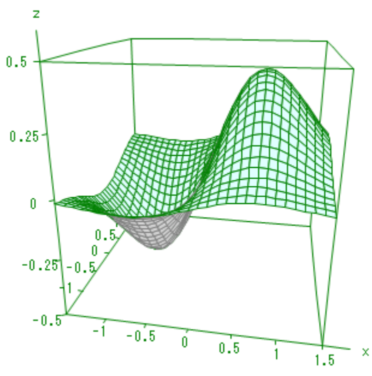


図 2a　$z=xe^{-x^2-y^2}$ のグラフ

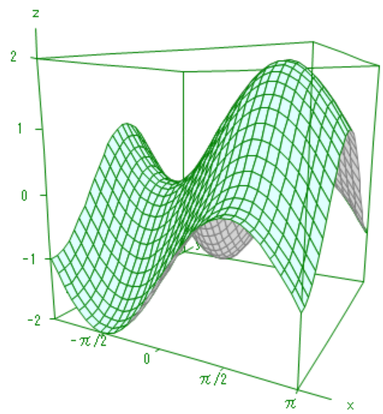
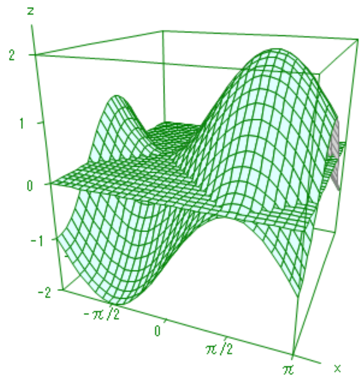


図 2b　$z=\sin x+\cos y$ のグラフ

１変数関数と同様、２変数関数の場合も複数のグラフの表示が可能である。図 3a は２つの関数を表示したもので、図 3b は色分けした３つの関数を表示したものである。



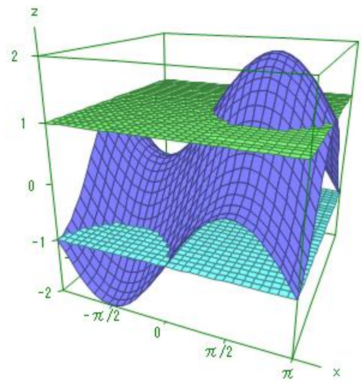


図 3a　２つの関数表示　　　　　　　図 3b　色分けした３つの関数表示

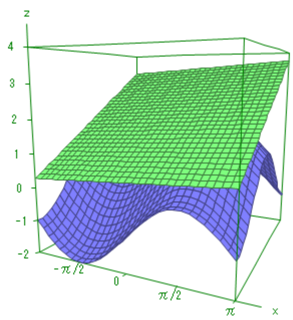
このプログラムでは、1 変数関数のプログラムと同様に、1 行目に書いた関数の表示領域内の極値を求めることができる。例えば $z = \sin x + \cos y$ のグラフで、描画メニューの「先頭式極値」ボタンをクリックすると、図 3 のように極値が表示される。ここに領域境界上の点は含めるようにしている。

計算結果（表示範囲内極値）

	解 1	解 2	解 3	解 4	解 5	解 6
X	-1.5708	-1.5708	-1.5708	1.5708	1.5708	1.5708
Y	-3.1416	0.0000	3.1416	-3.1416	0.0000	3.1416
極値	-2.0000	0.0000	-2.0000	0.0000	2.0000	0.0000
判定	極小	鞍点	極小	鞍点	極大	鞍点
収束解の個数	10	17	9	6	11	5

図 3　極値の表示

また、x, y 座標を指定し、「先頭式接平面」ボタンをクリックして、接平面の方程式を表示することができる。さらに、それを図に追加して表示することも可能である。例えば、$x = 1.5,\ y = -0.5$ の接平面は $z = 0.071x + 0.479y + 2.009$ で表わされ、それを表示すると図 4a のようになる。また、def( ) 関数を用いて、接平面の描画領域を半径 $\sqrt{2}$ の円形状に指定すると図 4b のようになる。但し、分割数を少し上げて、より円形に見えるようにしている。



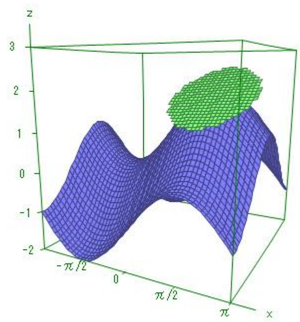
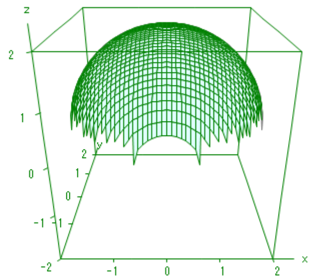


図 4a　接平面　　　　　　　図 4b　領域を円形状に指定した接平面

さて、このプログラムで図形を描くには限界がある。例えば、$z = \sqrt{4 - x^2 - y^2}$ の式で

半球を描こうとすれば、図 5a のように未定義の部分との境がきれいに表示されない。$z = \pm\sqrt{4 - x^2 - y^2}$ として、２つの関数を重ねて球体を表現しようとしても、図 5b のように描画されない部分も残るし、表面の裏表が変わる。



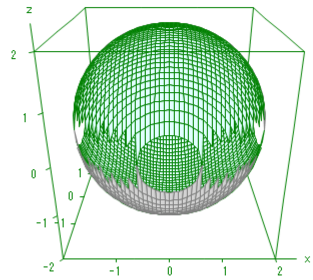


図 5a　上半分の球　　　　　図 5b　２つの曲面を利用した球

この問題は２章の１変数関数より深刻である。何故なら２変数関数の場合、分割数をあまり大きく取れないからで、しかも分割数の値を調節してきれいに描くこともできない。また、裏表の問題は面の描き方にも影響する大きな問題である。これらを解決してきれいな半球や球を描くプログラムが５章の３次元パラメータ表示関数である。

問題

$-3 \le x \le 3$，$-3 \le y \le 3$ の領域で以下のグラフを描き、極値を求めよ。また $x = 1, y = 1$ での接平面を求めよ。

１） $z = (\sin x + \cos y)^2$

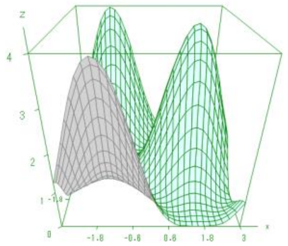
２） $z = \dfrac{-1}{\sqrt{x^2 + y^2 + 1}}$

３） $z = x\sin(x + y)$

問題解答

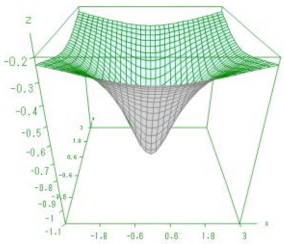
$-3 \le x \le 3$，$-3 \le y \le 3$ の領域で以下のグラフを描き、極値を求めよ。また $x = 1, y = 1$ での接平面を求めよ。

１） $z = (\sin x + \cos y)^2$



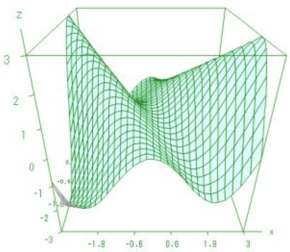
接平面　z=1.493*x−2.325*y+2.742

２） $z = \dfrac{-1}{\sqrt{x^2 + y^2 + 1}}$



接平面　z=-0.577*x-0.577*y-0.577

３） $z = x\sin(x + y)$



接平面　z= 0.493*x-0.416*y+0.832

５．３次元パラメータ表示関数グラフ

４章で述べたように、通常の 2 変数関数は１価関数であるので、球やトーラス等の形状を表現することはできない。我々はこのような関数を表示するために、図形を３次元パラメータ表示関数として表現するプログラムを作成した。３次元パラメータ表示関数は２変数パラメータ表示関数とも呼ばれ、ある範囲の値をとる２つのパラメータ u,v を用いて、$x = f(u,v),\ y = g(u,v),\ z = h(u,v)$ の形で表わされる関数である。

メニュー「分析－数学－グラフ－３次元パラメータ表示関数」を選択すると、図 1 のような描画メニューが表示される。

図１　３次元パラメータ表示関数グラフ実行画面

利用法は、「u 変数」、「v 変数」テキストボックスにパラメータ u,v の範囲を記述し、大きなテキストボックスの、「x =」、「y =」、「z =」の位置に u,v で表わされた関数形を記述して、「グラフ描画」ボタンをクリックする。例えば、４章の半球の場合は以下のようにする。

描画範囲：$0 \le u \le \pi/2,\ 0 \le v \le 2\pi$ （テキストボックス内で 2π は 2*pi で表す。）

関数：　$x = 2\sin u \cos v,\ y = 2\sin u \sin v,\ z = 2\cos u$

グラフは、図 2a が半球で、図 2b が球の場合である。球の場合、描画範囲を $0 \le u \le \pi$ に変え、u 分割数を 2 倍にしている。

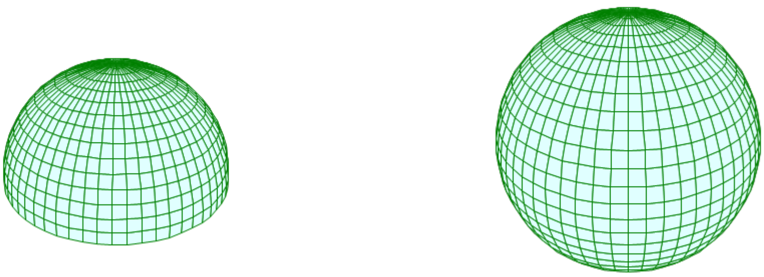
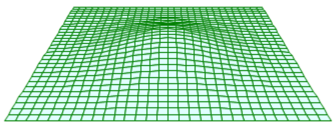


図 2a　半球の表示　　　　　　　　図 2b　球の表示

このプログラムでは、軸表示や描画範囲の表示も可能であるが、３Ｄビューアに標準で備わった機能を利用するので、詳細な設定はできない。また、図形の形が重要であるため、描画はすべての軸を同じスケールにしている。例えば 2 変数標準正規分布（相関 0）の密度関数を３次元パラメータ表示関数と 2 変数関数、2 つの表示法によって描くと、図 3a と図 3b のようになる。これらは、目的に応じて使い分ける必要がある。



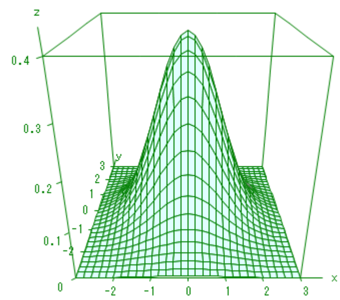


図 3a　３次元パラメータ表示関数表示　　　　図 3b　2 変数関数表示

「関数サンプル」コンボボックスには参考のための、以下のようなサンプルが入っている。これらのグラフは図 4 に示す。

円筒

描画範囲：$0 \le u \le 2\pi,\ 0 \le v \le 20$

関数：　$x = 5\cos u,\ y = 5\sin u,\ z = v$

メビュウス

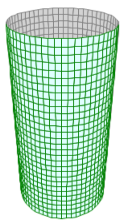
描画範囲：$0 \le u \le 2\pi,\ -3 \le v \le 3$

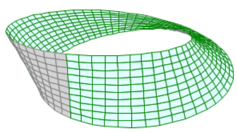
関数：　$x = \left(10 - v\sin\left(u/2\right)\right)\cos u,\ y = \left(10 - v\,sin(u/2)\right)\sin u,\ z = v\,cos\left(u/2\right)$

トーラス

描画範囲：$0 \le u \le 2\pi,\ 0 \le v \le 2\pi$

関数：　$x = \left(10 + 5\sin u\right)\cos v,\ y = \left(10 + 5\sin u\right)\sin v,\ z = 5\cos u$





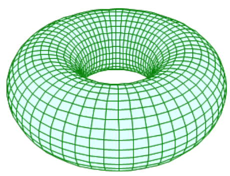


図 4a　円筒　　　　図 4b　メビュウス　　　　図 4c　トーラス

これらの他に、少し工夫をすると様々な図形が表示可能である。図 5 に簡単な例を示す

円錐

描画範囲：$-3 \le u \le 3,\ 0 \le v \le 2\pi$

関数：　　$x = u\cos v,\ y = -u\sin v,\ z = u$

巻貝のような図形

描画範囲：$0 \le u \le \pi,\ 0 \le v \le 4\pi$

関数：$x = 5(1+u)(1+v)\sin u\cos v,\ y = 5(1+u)(1+v)\sin u\sin v,$

$z = 5(1+u)(1+v)\cos u$

歪んだトーラス

描画範囲：$0 \le u \le 2\pi,\ 0 \le v \le 2\pi$

関数：　　$x = (10+5\sin u)\cos v,\ y = (10+5\sin u)\sin v,\ z = 5\cos u + 5\sin 2v$

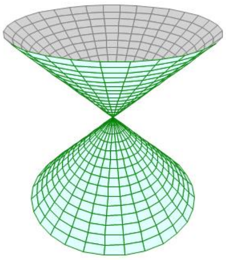


図 5a　円錐

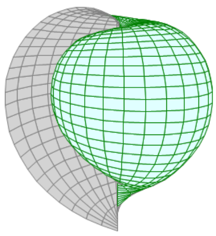


図 5b　巻貝のような図形

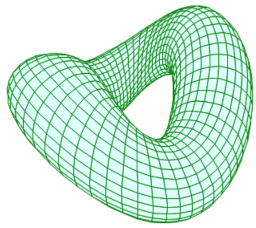


図 5c　歪んだトーラス

このプログラムでは、複数組の数式を並べて書くことによって、２つ以上のパラメータ表示関数を同時に表示することができる。式は省略するが、図 6 に複数の図形を組み合わせた例を示す。図 6a と図 6b は２つの図形、図 6c は４つの図形である。

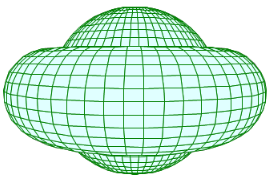


図 6a　２つの図形１

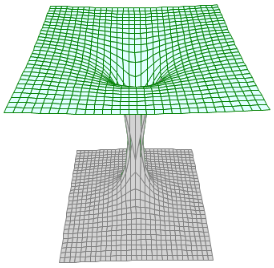


図 6b　２つの図形２

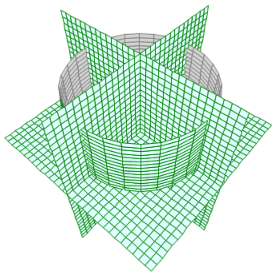


図 6c　４つの図形

複数の図形は２変数関数の場合と同様に、色で塗り分けることもできる。

３次元パラメータ表示関数は、複雑な表記のものが多いので、数式データをグリッドエディタに保存できるようにしている。図形を描いた後、描画メニューで、「グリッドへ」ボタンをクリックすると、描画データが、グリッドエディタの現在のページへ保存される。例えば半径 5 の球の場合、グリッドデータは図 7 のようになる。

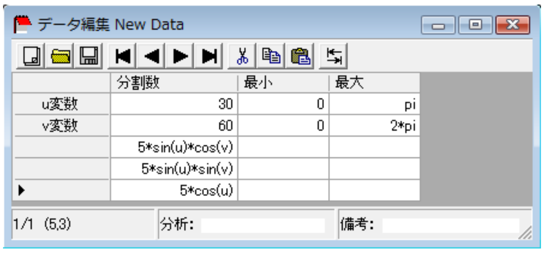


図７　グリッドへ保存された関数データ

これには、関数形だけでなく、u,v 変数の範囲や描画の際の分割数も保存されている。

この関数は場合によって非常に大きな描画範囲に渡ることがあるので、予め描画の最大値を中心からの距離で、「描画範囲」テキストボックスに入力しておく。描画要素の一部がこの範囲を超えると、その要素は描画されない。

表示する関数をワープロ等にコピーする場合、普通にコピーする場合は「コピー」ボタンを利用するが、変数の範囲や分割数などもコメントとして追加してコピーすることもできる。その際は「付加コピー」ボタンをクリックする。例えば球では以下のような形式でデータがクリップボードにコピーされる。

```
5*sin(u)*cos(v)
5*sin(u)*sin(v)
5*cos(u)
# u 下限：0
# u 上限：pi
# v 下限：0
# v 上限：2*pi
# u 分割数：30
# v 分割数：30
```

この表示を数式入力用のテキストボックスにそのまま貼り付けた場合、先頭に「#」の付いた行はコメント行となる。

６．方程式ソルバー

6.1　プログラムの利用法

方程式ソルバーは非線形の連立方程式を解くツールであり、今後のプログラム発展のための基礎技術を得る問題でもある。

メニュー［分析－数学－方程式ソルバー］を選択すると図 1 のような実行画面が表示される。

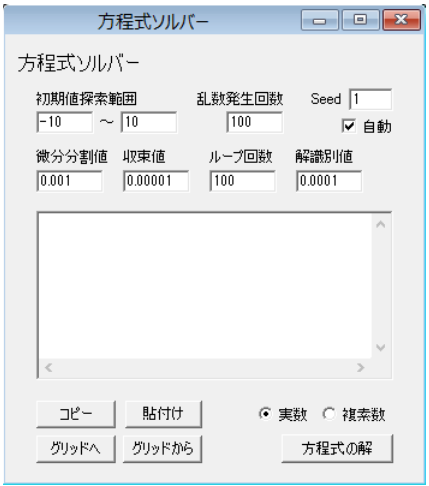
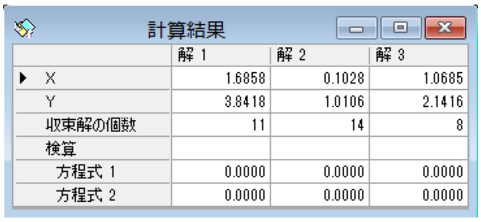


図 1　方程式ソルバー実行画面

メニューの中のテキストエディタに方程式を入力し、「方程式の解」ボタンをクリックして解を求める。例えば、以下の方程式を入力すると図 2 のような結果が出力される。

sin(x+y)+x-1=0

y-x^2-1=0



	解 1	解 2	解 3
X	1.6858	0.1028	1.0685
Y	3.8418	1.0106	2.1416
収束解の個数	11	14	8
検算			
方程式 1	0.0000	0.0000	0.0000
方程式 2	0.0000	0.0000	0.0000

図 2　実行結果

図 2 の検算の部分は誤差などによる不当な結果を排除するためのものである。正しい解であれば、「0」の値になる。

解の探索条件はメニューの「初期値探索範囲」と「乱数発生回数」や「乱数 seed」などで与えられる。「初期値探索範囲」は変数ごとではないので細かな設定方法ではないが、解が全く分からない最初の設定としては最も簡単である。またニュートン・ラフソン法に使われる微分を計算するための「微分分割値」や解の収束を判断する「収束値」及び、逐次計算を行う最大回数である「ループ回数」や異なった解が計算された場合それを判定する「解識別値」も計算に利用される。それらのデフォルト値はメニューにある通りであるが、必要に応じて変更する。

複素数解を求める場合は、メニューで「複素数」ラジオボタンを選択する。例えば、

「x^2+x+1=0」の方程式では、以下の結果となる。

計算結果

	解1	解2
X_Re	-0.5000	-0.5000
X_Im	-0.8660	0.8660
収束解の個数	40	60
検算		
方程式1_Re	0.0000	0.0000
方程式1_Im	0.0000	0.0000

図３　複素数解の計算結果

ここでは、変数が１つだけであるが、それ以上の方程式にも対応している。

問題１　実方程式

以下の方程式の解を実数の範囲で求めよ。

１）$x^2 - 2x - 5 = 0$

２）$\begin{cases} x + y = 3 \\ 3x - 2y = 7 \end{cases}$

３）$\begin{cases} \sin(x + y) + x - 1 = 0 \\ y = x^2 + 1 \end{cases}$

問題２　複素方程式

以下の方程式の解を複素数の範囲で求めよ。

１）$x^2 + x + 1 = 0$

２）$\begin{cases} x + iy^2 - 1 = 0 \\ y - x^2 - i = 0 \end{cases}$

問題１解答　実方程式

１）$x^2 - 2x - 5 = 0$

計算結果

	解 1	解 2
X	3.4495	-1.4495
収束解の個数	43	57
検算		
方程式 1	0.0000	0.0000

２）$\begin{cases} x + y = 3 \\ 3x - 2y = 7 \end{cases}$

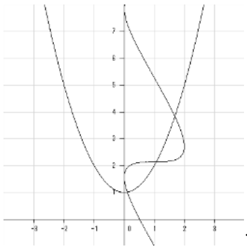
計算結果

	解 1
X	2.6000
Y	0.4000
収束解の個数	100
検算	
方程式 1	0.0000
方程式 2	0.0000

３） $\begin{cases} \sin(x+y)+x-1=0 \\ y=x^2+1 \end{cases}$

計算結果

	解 1	解 2	解 3
X	1.6858	1.0685	0.1028
Y	3.8418	2.1416	1.0106
収束解の個数	11	25	20
検算			
方程式 1	0.0000	0.0000	0.0000
方程式 2	0.0000	0.0000	0.0000


←不等式グラフで描いた。

問題２解答　複素方程式

１） $x^2+x+1=0$

計算結果

	解1	解2
X_Re	−0.5000	−0.5000
X_Im	0.8660	−0.8660
収束解の個数	44	56
検算		
方程式1_Re	0.0000	0.0000
方程式1_Im	0.0000	0.0000

２） $\begin{cases} x+iy^2-1=0 \\ y-x^2-i=0 \end{cases}$

計算結果

	解1	解2	解3	解4
X_Re	0.3143	0.9876	−0.0087	−1.2933
X_Im	−1.0906	−0.5106	0.7146	0.8866
Y_Re	−1.0906	0.7146	−0.5106	0.8866
Y_Im	0.3143	−0.0087	0.9876	−1.2933
収束解の個数	25	27	28	20
検算				
方程式1_Re	0.0000	0.0000	0.0000	0.0000
方程式1_Im	0.0000	0.0000	0.0000	0.0000
方程式2_Re	0.0000	0.0000	0.0000	0.0000
方程式2_Im	0.0000	0.0000	0.0000	0.0000

6.2　方程式ソルバーの理論

　ここでは方程式ソルバーの計算法について簡単に触れておく。非線形連立方程式を以下の形に書く。

$$f_i(x_1,x_2,\cdots,x_n)=0 \qquad (i=1,2,\cdots,n)$$

ここでは上の方程式に対してニュートン・ラフソン法を適用して解を求める方法を示す。

　今以下の関数を考える。

$$y=f_i(x_1,x_2,\cdots,x_n) \qquad (i=1,2,\cdots,n)$$

これらの関数に対して変数 x_i （$i=1,2,\cdots,n$）に $x_i=x_i^0$ の初期値を与える。この初期値における各関数の接平面（$n-1$ 次元平面）を考えるとその方程式は以下になる。

$$y=\sum_{j=1}^{n} f_{i,j}^0(x_j-x_j^0)+f_i^0 \qquad (i=1,2,\cdots,n)$$

ここに、 $f_i^0=f_i(x_1^0,x_2^0,\cdots,x_n^0)$， $f_{i,j}^0=\left.\dfrac{\partial f_i(x_1,x_2,\cdots,x_n)}{\partial x_j}\right|_{x_1=x_1^0,x_2=x_2^0,\cdots,x_n=x_n^0}$

これらの接平面が、 $y=0$ 平面と交わる点を求めると以下の方程式となる。

$$\sum_{j=1}^{n} f_{i,j}^{0}(x_j - x_j^{0}) = -f_i^{0} \qquad (i=1,2,\cdots,n)$$

これをクラメールの方法で解くと以下の解を得る。

$$x_j = x_j^{0} + \frac{M_j}{D} \qquad (j=1,2,\cdots,n)$$

ここに、 $$M_j = \begin{array}{c} \begin{array}{ccccc} \phantom{f_{1,1}^0} & & j\text{列} & & \phantom{f_{1,n}^0} \end{array} \\ \begin{vmatrix} f_{1,1}^{0} & \cdots & -f_1^{0} & \cdots & f_{1,n}^{0} \\ f_{2,1}^{0} & \cdots & -f_2^{0} & \cdots & f_{2,n}^{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ f_{n,1}^{0} & \cdots & -f_n^{0} & \cdots & f_{n,n}^{0} \end{vmatrix} \end{array}, \quad D = \begin{array}{c} \begin{array}{ccccc} \phantom{f_{1,1}^0} & & j\text{列} & & \phantom{f_{1,n}^0} \end{array} \\ \begin{vmatrix} f_{1,1}^{0} & \cdots & f_{1,j}^{0} & \cdots & f_{1,n}^{0} \\ f_{2,1}^{0} & \cdots & f_{2,j}^{0} & \cdots & f_{2,n}^{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ f_{n,1}^{0} & \cdots & f_{n,j}^{0} & \cdots & f_{n,n}^{0} \end{vmatrix} \end{array}$$

この x_j （$j=1,2,\cdots,n$）を新しい初期値 x_j^1 として上の処理を繰り返す。我々のプログラムでは収束条件を、ある微小指定値 ε に対して以下とした。

$$\max_j \left| x_j^{r} - x_j^{r-1} \right| < \varepsilon$$

ニュートン・ラフソン法で問題になるのは、初期値の与え方である。我々はこれに対して少し時間がかかるが乱数で初期値を与えて、収束する解を集める方法を選択した。もちろんすべての解が求まる保証はないので、解の個数には十分注意する必要がある。

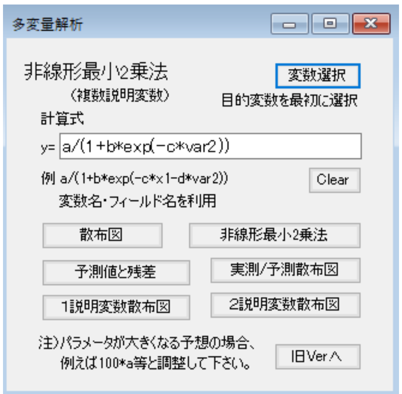
方程式ソルバーを複素数に対応させるのはそれほど困難ではない。基本的に方程式の数が実数部と今日数部で 2 倍になり、同様にパラメータの数も実数部と虚数部で 2 倍になるだけであり、計算の基本は、複素計算を行うサブルーチンを作れば変わらない。

７．非線形最小２乗法

ある変数を他のいくつかの変数で予測する場合、線形の式がよく利用されるが、線形性が認められない場合に利用される手法の１つが非線形最小２乗法である。この手法では関数形を仮定するため、ある程度データの性質が分かっていなければならない。これに対して関数形を仮定しない局所重回帰分析等の手法もあり、状況によって使い分けられる。

7.1　プログラムの利用法

メニュー［分析－数学－非線形最小２乗法］または［分析－多変量解析－予測手法－非線形最小２乗法］を選択すると、図 1 のような分析実行画面が表示される。このプログラムは以前に作っていた旧バージョンのものを改訂したものである。ここでは図 2 に与えられる非線形最小２乗法 2.txt の４頁目のデータを用いてプログラムの機能を説明する。



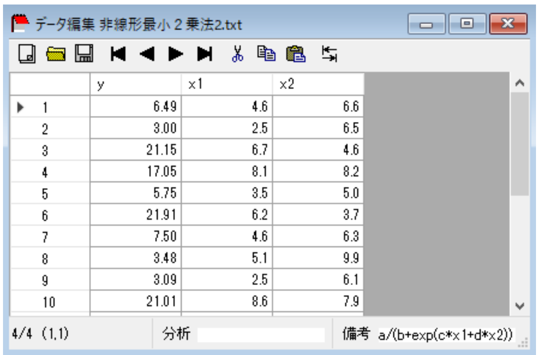


図１　分析実行画面　　　　　　　　　　　図２　データ

データの性質を見るために、説明変数が２つまでは「散布図」ボタンで状況を見ることができる。図３に説明変数が２つの場合の３次元散布図を描く。

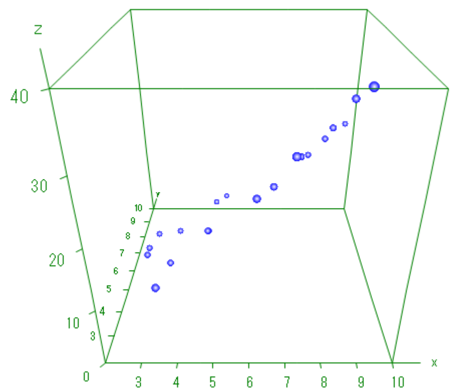


図３　３次元の散布図

計算式にパラメータの値を除き予め予想される計算式を記入し、重回帰分析と同様に変数

を選択して、「非線形最小 2 乗法」ボタンをクリックする。表示される結果を図 4 に示す。

計算結果

	推定値	標準誤差	z統計量	確率値	95%下限	95%上限
A	5.7044	0.1755	32.5070	0.0000	5.3605	6.0484
B	0.1095	0.0036	30.2095	0.0000	0.1024	0.1166
C	-0.4733	0.0067	-70.4771	0.0000	-0.4864	-0.4601
D	0.2859	0.0051	56.5447	0.0000	0.2760	0.2958
実測・予測 R	1.000	R^2	1.000			

図 4　非線形最小 2 乗法結果

この解によるデータの「予測値と残差」ボタンをクリックした際に表示される結果を図 5 に、「実測/予測散布図」ボタンをクリックした際に表示される結果を図 6 に示す。

予測値と残差

	実測値	予測値	残差
1	6.49	6.651	-0.161
2	3	2.751	0.249
3	21.15	21.457	-0.307
4	17.05	17.024	0.026
5	5.75	6.293	-0.543
6	21.91	21.719	0.191
7	7.5	7.165	0.335
8	3.48	3.507	-0.027
9	3.09	3.064	0.026
10	21.01	20.901	0.109
11	33.82	33.400	0.420
12	25.81	25.672	0.138
13	19.95	20.386	-0.436
14	39.59	39.770	-0.180
15	2.46	2.293	0.167

図 5　予測値と残差

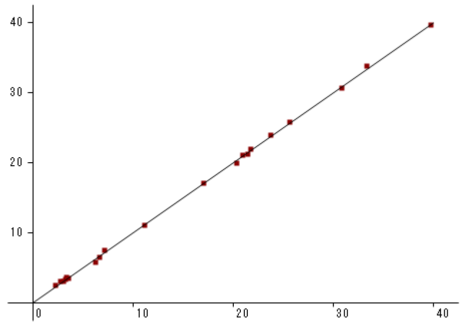


図 6　実測/予測散布図

説明変数が 2 つの場合、「２説明変数散布図」ボタンをクリックすると、図 3 の散布図に図 7 のような予測曲面が追加されたグラフが表示される。

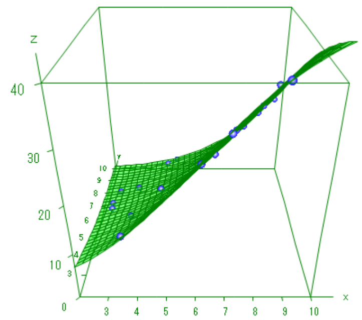


図 7　２説明変数散布図

これで曲線の適合性を視覚的に確認することができる。

プログラムのアルゴリズムとしては、これまで乱数で初期設定をし、その中から最適値を選んでいたものを、MCMC で初期値を設定するように変更している。そのため、解を求める操作は 1 回だけで、計算時間は短縮される。初期値の手動による指定については、非常に残念であったが、新しい画面では省略した。しかし大きなパラメータ値に対しては対応しきれていないので、旧バージョンの利用もできるようにしている。その他の設定については、必要なくなったものや故意に省いたものもある。必要があれば「旧 Ver.へ」ボタンで旧バ

ージョンのものを使うこともできる。

問題１

非線形最小 2 乗法 2.txt の 1 ページ目のデータを用いて、非線形最小 2 乗法の計算を行い、以下の問いに答えよ。但し、計算式は　y=a*x+b*sin(x)+c　とすること。

１）パラメータの値を求めよ。

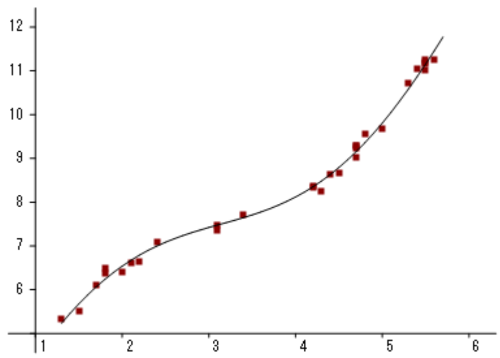
a= [　　　　　　　　], b= [　　　　　　　　], c= [　　　　　　　　]

２）R^2 の値を求めよ。[　　　　　　　　]

３）先頭の人の実測値、予測値、残差を求めよ。

実測値 [　　　　　　　　], 予測値 [　　　　　　　　], 残差 [　　　　　　　　]

４）関数形を描け。



注）これは重回帰を使っても計算できるのでやってみてもらいたい。

問題２

非線形最小 2 乗法 2.txt の 2 ページ目のデータを用いて、非線形最小 2 乗法の計算を行い、以下の問いに答えよ。但し、計算式は　y=a*x1+b*x2^2+c　とすること。

１）パラメータの値を求めよ。

a= [　　　　　　　　], b= [　　　　　　　　], c= [　　　　　　　　]

２）R^2 の値を求めよ。[　　　　　　　　]

３）先頭の人の実測値、予測値、残差を求めよ。

実測値 [　　　　　　　　], 予測値 [　　　　　　　　], 残差 [　　　　　　　　]

注）これは重回帰を使っても計算できる。

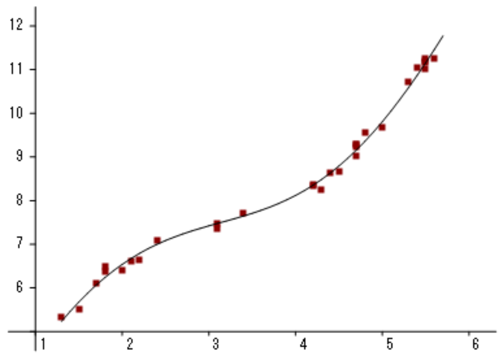
問題３

非線形最小 2 乗法 1.txt の 1 ページ目のデータを用いて、非線形最小 2 乗法の計算を行い、以下の問いに答えよ。但し、計算式は　売上高=100*a/(1+100*b*exp(-c*週))　とすること（a, b は大きな数になる）。

１）パラメータの値を求めよ。

a= [　　　　　　　　], 　b= [　　　　　　　　], 　c= [　　　　　　　　]
２）R^2 の値を求めよ。[　　　　　　　　]
３）先頭の人の実測値、予測値、残差を求めよ。
実測値 [　　　　　　　　], 予測値 [　　　　　　　　], 残差 [　　　　　　　　]
注）これは重回帰では計算できない。

問題１解答
計算式は　y=a*x+b*sin(x)+c　とすること。
１）パラメータの値を求めよ。
a= [1.9774], 　b= [1.4258], 　c= [1.2896]
２）R^2 の値を求めよ。[0.995]
３）先頭の人の実測値、予測値、残差を求めよ。
実測値 [8.34], 予測値 [8.352], 残差 [-0.012]
４）関数形を描け。



問題２解答
計算式は　y=a*x1+b*x2^2+c　とすること。
１）パラメータの値を求めよ。
a= [0.500], 　b= [0.186], 　c= [15.305]
２）R^2 の値を求めよ。[0.951]
３）先頭の人の実測値、予測値、残差を求めよ。
実測値 [42.89], 予測値 [46.398], 残差 [-3.508]

問題３解答
計算式は　売上高=100*a/(1+100*b*exp(-c*週))　とすること（a,b は大きな数になる）。
１）パラメータの値を求めよ。
a= [7.8572], 　b= [1.1024], 　c= [0.5144]
２）R^2 の値を求めよ。[0.999]
３）10 週目の実測値、予測値、残差を求めよ。
実測値 [485], 予測値 [478.180], 残差 [6.820]

7.2　プログラムの利用法（旧版）

ここでは簡単に旧版のプログラムについて説明しておく。図 1 に分析実行画面を示す。

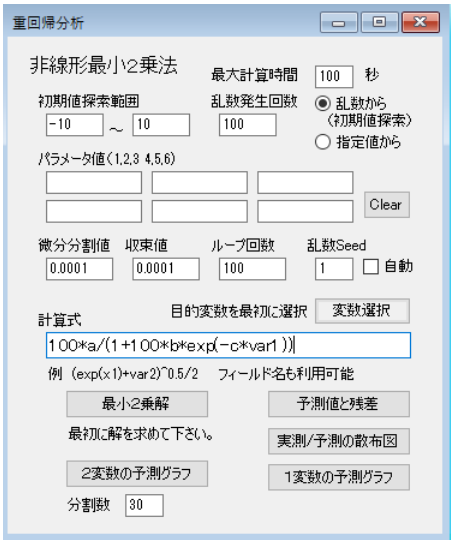
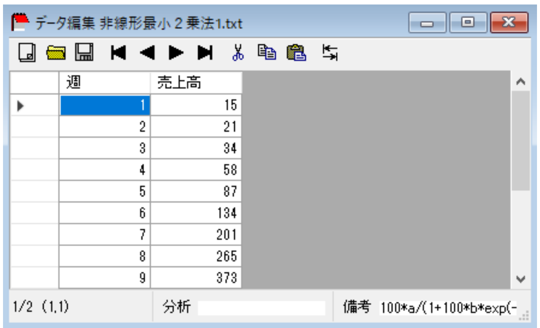


図 1　分析実行画面（旧版）

解を求める際の初期値の設定は、乱数による設定と指定による設定がある。ここでは最小 2 乗法 1.txt の 1 頁目のデータを用いて説明をする。



]

図 2　データ画面

まず図 1 の画面の目的変数コンボボックスを設定し、計算式のテキストボックス内にパラメータを含む予測式を書く。これはロジスティック曲線と呼ばれる例である。我々のプログラムでは方程式ソルバーと同じく、初期値の設定は乱数で行う。図 1 の設定は初期値の設定を「乱数から」にした場合のデフォルト値である。初期設定が終わったら、変数を目的変数、説明変数の順に選択し、「最小２乗解」ボタンをクリックして計算を実行する。実行結果を図 3 に示す。

図３　解表示画面１

解を求めた後、その解による実測値、予測値、残差は、「予測値と残差」ボタンをクリックすることで求まる。実行例を図４に示す。この実測値と予測値の関係を見る散布図は「実測／予測の散布図」ボタンで求まる。その例を図５に示す。

予測値と残差

	実測値	予測値	残差
▸	15	11.743	3.257
	21	19.447	1.553
	34	31.995	2.005
	58	52.089	5.911
	87	83.408	3.592
	134	130.213	3.787
	201	195.956	5.044
	265	280.680	-15.680
	373	378.523	-5.523
	485	478.180	6.820
	568	567.507	0.493
	648	638.856	9.144
	681	690.779	-9.779
	738	726.059	11.941
	748	748.926	-0.926
	755	763.299	-8.299

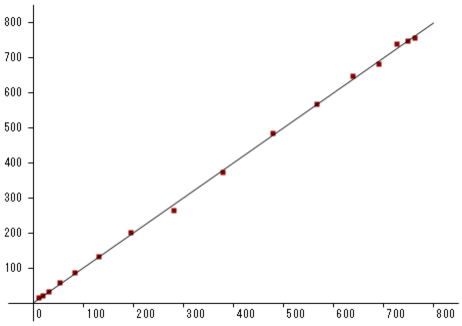


図４　予測値と残差　　　　　　　図５　実測／予測散布図

また、説明変数が１変数の場合に限り、目的変数がその説明変数によってどのように予測されるかを「１変数の予測グラフ」をクリックすることで見ることができる。実行例を図６に示す。

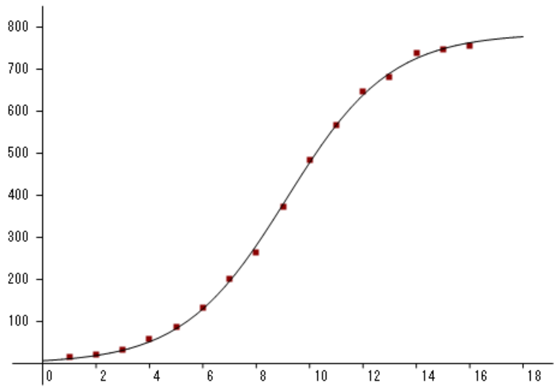


図６　１変数の予測グラフ画面

初期値の設定方法で、「乱数から」を選択した場合、状況によっては解が求められない場合がある。その場合は、「収束値」の値を少し大きくして粗い解を求め、その後その解を初期値として、精度の高い解を求めることを考える。その際は図７のように一度求めた解が、パラメータ値のところに入っているので、そのまま初期値を「指定値から」にして、最小２乗解を求めればよい。

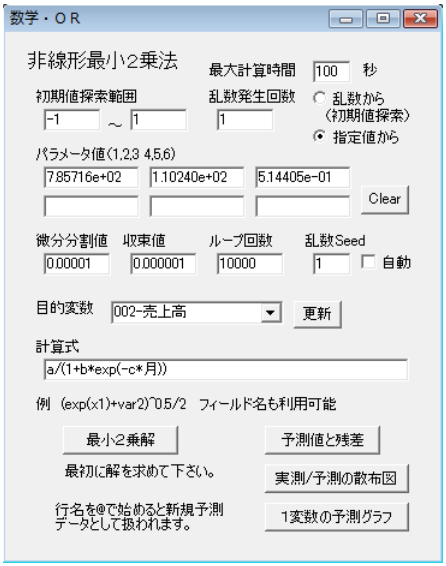


図７　パラメータ値を初期値にする設定

7.3　非線形最小２乗法の理論

非線形最小２乗法は、ある目的変数 y を m 個の説明変数 x_i（$i=1,2,\cdots,m$）と p 個のパラメータ a_k（$k=1,2,\cdots,p$）を含む非線形関数 $f(x_1,\cdots,x_m;a_1,\cdots,a_p)$ で予測することを目的とする。

$$y = f(x_1,\cdots,x_m;a_1,\cdots,a_p)+u$$

ここに u は誤差項である。

まず観測された N 個のデータについてパラメータ a_k を最適化する式を考える。今レコードを λ（$\lambda=1,2,\cdots,N$）として、以下の残差の２乗を表す統計量 Z を考える。

$$Z = \sum_{\lambda=1}^{N}\left[y_\lambda - f(x_{1\lambda,},\cdots,x_{n\lambda};a_1,\cdots,a_p)\right]^2 \qquad \text{最小化}$$

我々は Z の値を最小化するために、パラメータ a_k（$k=1,\cdots,p$）で微分する。

$$\frac{\partial Z}{\partial a_k} = -2\sum_{\lambda=1}^{N}\frac{\partial f(x,a)}{\partial a_k}\left[y_\lambda - f(x,a)\right]=0 \qquad (k=1,\cdots,p) \qquad (7.1)$$

ここに書式の簡単化のため、$f(x,a)\equiv f(x_{1\lambda,},\cdots,x_{n\lambda};a_1,\cdots,a_p)$ としている。

式（7.1）は y_λ と $x_{i\lambda}$ に実測値を使うことから、パラメータ a_k についての非線形連立方程式となる。よって前に述べた連立方程式（6.1）の場合の解法が利用できる。即ち、$a_k = a_k^0$ を初期値として以下の値を考える。

$$a_k = a_k^0 + M_k/D \qquad (k=1,\cdots,p)$$

$$M_k = \begin{array}{c} \begin{array}{ccccc} \phantom{Z_{11}^0} & & k\text{列} & & \phantom{Z_{1p}^0} \end{array} \\ \begin{vmatrix} Z_{11}^0 & \cdots & -Z_1^0 & \cdots & Z_{1p}^0 \\ Z_{21}^0 & \cdots & -Z_2^0 & \cdots & Z_{2p}^0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ Z_{p1}^0 & \cdots & -Z_p^0 & \cdots & Z_{pp}^0 \end{vmatrix} \end{array}, \quad D = \begin{array}{c} \begin{array}{ccccc} \phantom{Z_{11}^0} & & k\text{列} & & \phantom{Z_{1p}^0} \end{array} \\ \begin{vmatrix} Z_{11}^0 & \cdots & Z_{1k}^0 & \cdots & Z_{1p}^0 \\ Z_{21}^0 & \cdots & Z_{2k}^0 & \cdots & Z_{2p}^0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ Z_{p1}^0 & \cdots & Z_{pk}^0 & \cdots & Z_{pp}^0 \end{vmatrix} \end{array}$$

ここに、 $Z_{ij}^0 = \partial^2 Z \big/ \partial a_i \partial a_j \big|_{a_1=a_1^0, a_2=a_2^0, \cdots, a_p=a_p^0}$ である。

これから先はここで求めた a_k を新たな初期値として同じ処理を繰り返す。これは前節で述べた逐次近似と同じ計算である。しかし、前節の場合は１回微分までであったが、今回は２回微分を使っている。数値計算の２回微分は一般に精度が悪くなり、さらに解を見つけにくくなる。そのため、我々はこの問題に対して、まずマルコフ連鎖モンテカルロ法を用いて、最小２乗法の解になるべく近い初期値を求め、その後比較的収束が良いとされる、レーベンバーグ・マーカート法を用いて解を求めている。しかしこの方法でも、パラメータが 1 に比べて極端に大きい場合や小さい場合には解を求めにくくなるので、予め解がどの程度の大きさか予想がつく場合には、それに合わせてパラメータに定数をかけ、パラメータの大きさが概ね１になるように式を調節しておく方がよい。

次に、パラメータの標準誤差を求める方法を与えておく。回帰式を以下のように仮定する。

$$y_\lambda = f(\mathbf{x}_\lambda, \boldsymbol{\beta}) + \varepsilon_\lambda, \quad Y_\lambda = f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})$$

ここに、$\boldsymbol{\beta}$ は回帰係数の真の値、$\hat{\boldsymbol{\beta}}$ はその推定値とする。

最小化する関数 L を以下のようにする。

$$L = \sum_{\lambda=1}^{N} [y_\lambda - f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})]^2$$

これを用いて、

$$\frac{\partial L}{\partial \beta_i} = -2 \sum_{\lambda=1}^{N} \frac{\partial f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})}{\partial \hat{\beta}_i} [y_\lambda - f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})] = 0$$

この式を解いて係数 $\hat{\boldsymbol{\beta}}$ を求めるが、 y_λ に上の式を代入すると、

$$\sum_{\lambda=1}^{N} \frac{\partial f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})}{\partial \hat{\beta}_i} [f(\mathbf{x}_\lambda, \boldsymbol{\beta}) + \varepsilon_\lambda - f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})] = 0$$

これより、

$$\sum_{\lambda=1}^{N} \frac{\partial f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})}{\partial \hat{\beta}_i} [f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}}) - f(\mathbf{x}_\lambda, \boldsymbol{\beta})] = \sum_{\lambda=1}^{N} \frac{\partial f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})}{\partial \hat{\beta}_i} \varepsilon_\lambda$$

ここで、１次の近似として、以下を仮定する。

$$f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}}) - f(\mathbf{x}_\lambda, \boldsymbol{\beta}) \simeq \sum_{j=1}^{p} \frac{\partial f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})}{\partial \hat{\beta}_j} (\hat{\beta}_j - \beta_j)$$

これを用いると

$$\sum_{j=1}^{p} \sum_{\lambda=1}^{N} \frac{\partial f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})}{\partial \hat{\beta}_i} \frac{\partial f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})}{\partial \hat{\beta}_j} (\hat{\beta}_j - \beta_j) = \sum_{\lambda=1}^{N} \frac{\partial f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})}{\partial \hat{\beta}_i} \varepsilon_\lambda$$

ここで、$(\mathbf{A})_{ij} = \sum_{\lambda=1}^{N} \frac{\partial f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})}{\partial \hat{\beta}_i} \frac{\partial f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})}{\partial \hat{\beta}_j}$ 、$(\mathbf{c})_i = \sum_{\lambda=1}^{N} \frac{\partial f(\mathbf{x}_\lambda, \hat{\boldsymbol{\beta}})}{\partial \hat{\beta}_i} \varepsilon_\lambda$

とすると、上式は以下となる。

$$\mathbf{A}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta})=\mathbf{c}$$

これを解いて、

$$\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}=\mathbf{A}^{-1}\mathbf{c}$$

即ち、

$$\hat{\beta}_i-\beta_i=\sum_{j=1}^{N}(\mathbf{A}^{-1})_{ij}\sum_{\lambda=1}^{N}\frac{\partial f(\mathbf{x}_\lambda,\hat{\boldsymbol{\beta}})}{\partial\hat{\beta}_j}\varepsilon_\lambda=\sum_{\lambda=1}^{N}\left(\sum_{j=1}^{N}(\mathbf{A}^{-1})_{ij}\frac{\partial f(\mathbf{x}_\lambda,\hat{\boldsymbol{\beta}})}{\partial\hat{\beta}_j}\right)\varepsilon_\lambda$$

ここで、$\varepsilon_\lambda\sim N(0,\sigma_\lambda^2)$ であるから、標準誤差は以下で与えられる。

$$Std\left(\hat{\beta}_i-\beta_i\right)\to\sqrt{\sum_{\lambda=1}^{N}\left(\sum_{j=1}^{N}(\mathbf{A}^{-1})_{ij}\frac{\partial f(\mathbf{x}_\lambda,\hat{\boldsymbol{\beta}})}{\partial\hat{\beta}_j}\right)^2\varepsilon_\lambda^2}$$

ここで、この関係を具体的に見てみよう。最も簡単な例として、

$$f(\mathbf{x}_\lambda,\hat{\boldsymbol{\beta}})=ax_\lambda+b$$

$$(\mathbf{A})_{11}=\sum_{\lambda=1}^{N}\frac{\partial(ax_\lambda+b)}{\partial a}\frac{\partial(ax_\lambda+b)}{\partial a}=\sum_{\lambda=1}^{N}x_\lambda^2$$

$$(\mathbf{A})_{12}=(\mathbf{A})_{21}=\sum_{\lambda=1}^{N}\frac{\partial(ax_\lambda+b)}{\partial a}\frac{\partial(ax_\lambda+b)}{\partial b}=\sum_{\lambda=1}^{N}x_\lambda=N\overline{x}$$

$$(\mathbf{A})_{22}=\sum_{\lambda=1}^{N}\frac{\partial(ax_\lambda+b)}{\partial b}\frac{\partial(ax_\lambda+b)}{\partial b}=\sum_{\lambda=1}^{N}1=N$$

$$\mathbf{A}=\begin{pmatrix}\sum_{\lambda=1}^{N}x_\lambda^2 & N\overline{x}\\ N\overline{x} & N\end{pmatrix},$$

$$(\mathbf{c})_1=\sum_{\lambda=1}^{N}\frac{\partial(ax_\lambda+b)}{\partial a}\varepsilon_\lambda=\sum_{\lambda=1}^{N}x_\lambda\varepsilon_\lambda\,,\qquad(\mathbf{c})_2=\sum_{\lambda=1}^{N}\frac{\partial(ax_\lambda+b)}{\partial b}\varepsilon_\lambda=\sum_{\lambda=1}^{N}\varepsilon_\lambda=N\overline{\varepsilon}$$

解は以下で与えられる。

$$\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}=\mathbf{A}^{-1}\mathbf{c}=\frac{1}{N\sum_{\lambda=1}^{N}x_\lambda^2-N^2\overline{x}^2}\begin{pmatrix}N & -N\overline{x}\\ -N\overline{x} & \sum_{\lambda=1}^{N}x_\lambda^2\end{pmatrix}\begin{pmatrix}\sum_{\lambda=1}^{N}x_\lambda\varepsilon_\lambda\\ N\overline{\varepsilon}\end{pmatrix}$$

$$=\frac{1}{\sum_{\lambda=1}^{N}(x_\lambda-\overline{x})^2}\begin{pmatrix}\sum_{\lambda=1}^{N}x_\lambda\varepsilon_\lambda-N\overline{x}\overline{\varepsilon}\\ -\overline{x}\sum_{\lambda=1}^{N}x_\lambda\varepsilon_\lambda+\overline{\varepsilon}\sum_{\lambda=1}^{N}x_\lambda^2\end{pmatrix}$$

$$=\frac{1}{\sum_{\lambda=1}^{N}(x_\lambda-\overline{x})^2}\begin{pmatrix}\sum_{\lambda=1}^{N}(x_\lambda-\overline{x})\varepsilon_\lambda\\ -\overline{x}\sum_{\lambda=1}^{N}(x_\lambda-\overline{x})\varepsilon_\lambda+\overline{\varepsilon}\sum_{\lambda=1}^{N}(x_\lambda-\overline{x})^2\end{pmatrix}$$

$$=\begin{pmatrix}\sum_{\lambda=1}^{N}(x_\lambda-\overline{x})\varepsilon_\lambda\Big/\sum_{\lambda=1}^{N}(x_\lambda-\overline{x})^2\\ -\sum_{\lambda=1}^{N}\overline{x}(x_\lambda-\overline{x})\varepsilon_\lambda\Big/\sum_{\lambda=1}^{N}(x_\lambda-\overline{x})^2+\overline{\varepsilon}\end{pmatrix}$$

これは直接計算した結果と一致する。

８．定積分

ここで扱う定積分は以下のような積分である。

- 積分変数は 1～3 個とする。
- 座標系については、2 次元及び 3 次元の直交座標と極座標で、3 次元についてのみ円柱座標を扱う。
- 関数については、通常の x, y, z 座標の関数とパラメータ表示関数を扱う。
- 積分の目的は、2 次元及び 3 次元の曲線の長さ、面積、体積、3 次元での x 軸周りの回転体の体積及び、一般的な 3 変数関数の積分を求めるものである。

積分は、1 変数及び 2 変数の場合、1 次元多い空間に座標軸を設けて、面積や体積を求める積分に帰着させることができる。我々は、グラフ描画の際に、この解釈を用いている。3 変数の場合も 4 次元上で同様に考えられるが、図に描けないため一般的な関数の積分として区別しておく。ここでは積分の特性によって 3 つの節に分け、議論を進める。

8.1　直交座標での積分

メニュー［分析－数学－定積分－直交座標での積分］を選択すると図 1 のような実行画面が表示される。

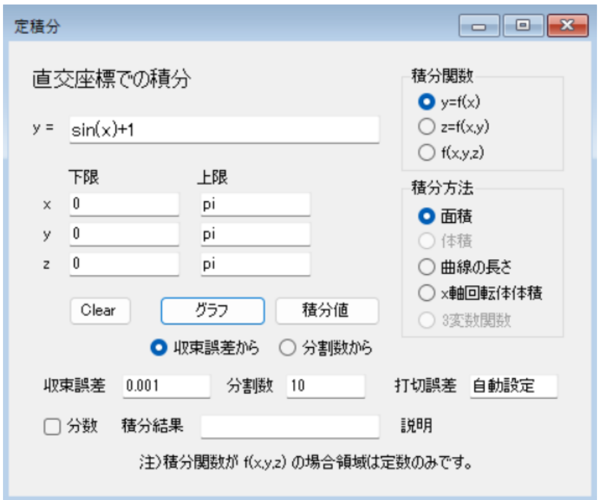


図 1　直交座標での積分実行画面

ここでは「積分関数」を選ぶことによって、「積分方法」の選択肢が示される。積分関数として「y = f(x)」を選択すると、積分方法は、面積、曲線の長さ、x 軸回転体体積となる。積分関数として「z = f(x,y)」を選択すると、面積、体積となる。また、積分関数として「f(x,y,z)」を選択すると、3 変数関数だけとなる。

図 1 の例の場合、「グラフ」ボタンをクリックすると、1 変数関数グラフのメニューが表示され、描画結果は図 2 のようになる。

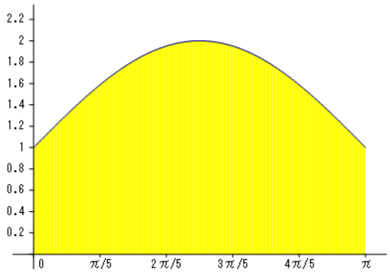


図 2　(y =) sin(x)+1 の積分範囲

積分範囲の色は、積分値が正の場合は黄色、負の場合は赤色になる。

「積分値」ボタンをクリックすると「積分結果」テキストボックスに計算結果が表示される。積分値は領域の分割数によって誤差が生じるが、多次元の場合、細かくし過ぎると計算機の能力により計算時間が 10 秒以上（ハングアップと間違えない程度を基本にしている）かかることがある。デフォルトでは変数数に応じて適当な数値を与え、利用者が必要に応じて変更できるようにしている。

同じの設定で、「x 軸回転体体積」では「グラフ」ボタンで、3 次元パラメータ表示関数グラフのメニューが表示され、描画結果は図 3 のようになる。

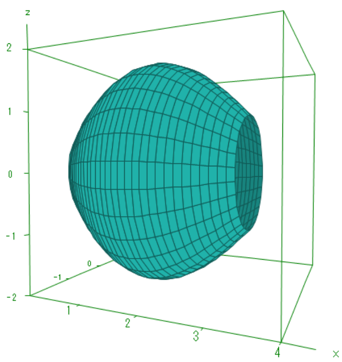


図 3　x 軸回転体体積での積分範囲

次に積分関数が「z = f(x,y)」の設定で、関数を(z =) sin(x)+cos(y)とし、積分範囲を $0 \leqq x \leqq 1$, $0 \leqq y \leqq x$ としてグラフを描画すると、図 4 のような、領域が切断されたグラフになる。

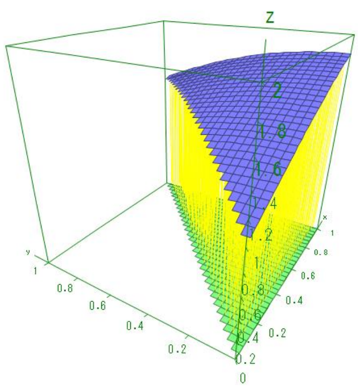


図 4　関数で表される積分範囲

ここでは積分範囲内に、正の場合は黄色、負の場合は赤色で縦線を入れることができる。

直交座標での積分はよく利用されるので、積分の分割数の設定を「収束誤差」を指定して自動的に求める方法と直接入力する方法の２つから選べるようにした。通常計算誤差は重要なので前者の方法を利用する。分割数は、２つの分割数で積分値の差が収束誤差の値を下回るまで、各軸の分割数を同時に 2 倍ずつにすることにしている。この方法では、次元が高くなると収束までに計算時間が必要となる。例えば 3 次元の積分の場合、分割数を 2 倍するごとに計算量は 8 倍になる。そのため、その間を埋めるような計算も必要になると考え、分割数を指定して計算する方法も加えることにした。但し、以前のバージョンでは軸ごとに分割数を設定していたが、ここでは 3 つの軸で同じ分割数を用いるようにしている。また、以前のバージョンでは２つの関数間の積分を計算することも考えていたが、新しいバージョンでは１つの関数についてだけ扱えるように機能を縮小させた。これは２つの積分の引き算で解答が得られるため、特に必要ではないと考えたからである。

問題　直交座標での積分

以下の定積分を実行せよ。また、領域のグラフを描け。

１）$\int_0^\pi \sin^2 x dx$

２）上の非積分関数の曲線の長さ

３）上の非積分関数を x 軸周りに回転させた図形の体積

４）$\int_0^\pi \int_0^\pi (2\sin x + 2\cos y - 1) dxdy$

５）上の非積分関数の面積

６）$\int_0^\pi \int_0^y (2\sin x + 2\cos y - 1) dxdy$

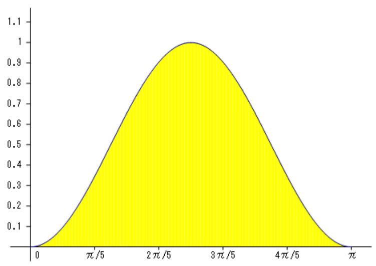
問題解答　直交座標での積分

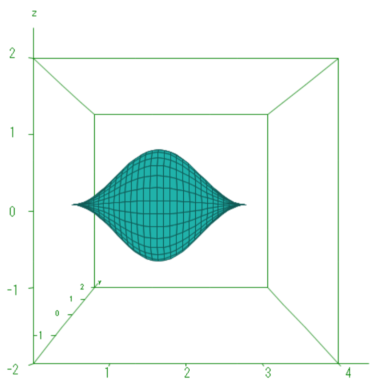
以下の定積分を実行せよ。また、領域のグラフを描け。

１）$\int_0^\pi \sin^2 x dx$　　1.5708　図は左下

２）上の非積分関数の曲線の長さ　　3.8202

３）上の非積分関数を x 軸周りに回転させた図形の体積　　3.7011　図は右下

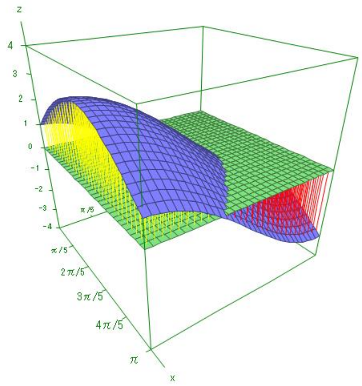


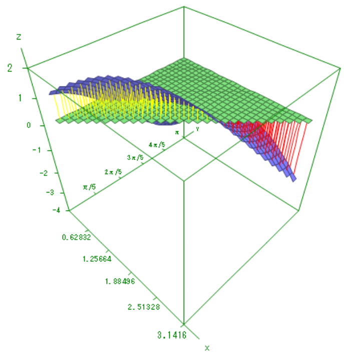


4） $\int_0^{\pi}\int_0^{\pi}(2\sin x+2\cos y-1)dxdy$　　2.6965　図は左下

5） 上の非積分関数の面積　　22.6270

6） $\int_0^{\pi}\int_0^{y}(2\sin x+2\cos y-1)dxdy$　　-2.6516　図は右下





8.2　２次元パラメータ表示積分

メニュー［分析－数学－定積分－2 次元パラメータ表示積分］を選択すると図 1 のようなメニュー画面が表示される。

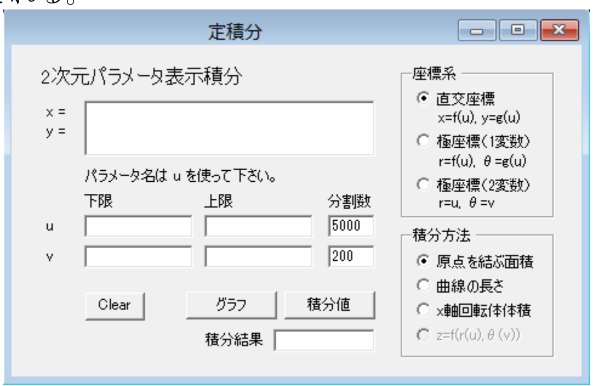


図 1　2 次元パラメータ表示積分実行画面

座標系により計算できる積分方法は、座標系が「直交座標」または「極座標（1 変数）」のとき、「原点を結ぶ面積」と「曲線の長さ」であり、座標系が「極座標（2 変数）」のとき、「2 変数関数 f(u,v)」である。

例えば、カージオイドと呼ばれる曲線

$$r=a(1+\cos\theta)\quad(0\le\theta\le 2\pi)$$

を我々の「極座標（1 変数）」の表示にすると、例えば $a=2$ として、以下のようになる。

(r =) 2*(1+cos(u))

(θ =) u　　　　　　　　　0≦u≦2*pi

この関数の原点を結ぶ面積のグラフは、2 次元パラメータ表示関数の描画として図 2 のようになる。

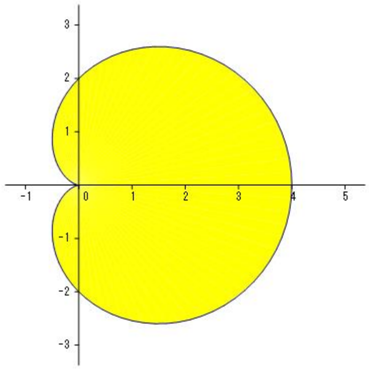


図２　原点を結ぶ面積のグラフの描画

座標系が「極座標（2 変数）」の場合は、$f(r,\theta)$ の $r(=u)$ と $\theta(=v)$ による積分であるが、$z=f(r,\theta)$ と考えると 3 次元空間における円柱座標による体積の積分と考えられる。例えば、以下のように結果を求めることができる。

(f =) 3*exp(-u^2)，　0≦u≦3,　0≦v≦2*pi

グラフは図 3 のようになる。

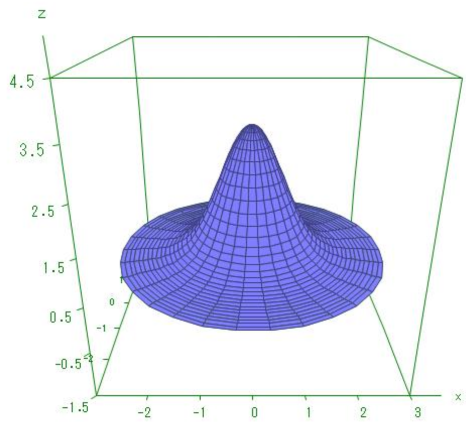


図３　円柱座標での体積積分

2 次元パラメータ積分の場合、直交座標の場合と同様に、x 軸回転体のグラフやその体積も求めることができる。

問題　２次元パラメータ表示積分

１）$\begin{cases}\cos u \\ \sin u\end{cases}$　$0 \le u \le \pi$　の原点と曲線を結ぶ面積

２）上の関数の曲線の長さ

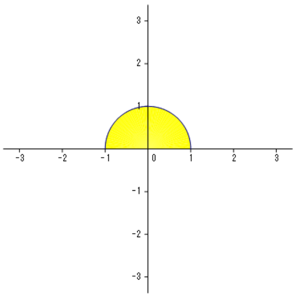
３）x 軸周りに回転させた図形の体積

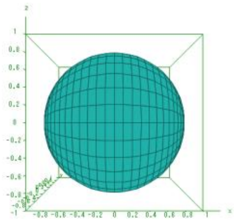
問題解答　２次元パラメータ表示積分

1） $\begin{cases} \cos u \\ \sin u \end{cases}$ 　$0 \le u \le \pi$ 　の原点と曲線を結ぶ面積　　1.5708（半径 1 半円面積）

2）上の関数の曲線の長さ　　3.1416（半径 1 半円円周）

3）x 軸周りに回転させた図形の体積　　4.1888（半径 1 球体積）





8.3　3 次元パラメータ表示積分

メニュー［分析－数学－定積分－3 次元パラメータ表示積分］を選択すると図 1 のようなメニュー画面が表示される。

図 1　メニュー画面

座標系が「直交座標」、「円柱座標（2 変数）」、「極座標（2 変数）」のいずれかの場合、積分方法は「原点を結ぶ体積」、「面積」、「曲線の長さ（u のみ）」が選択できる。曲線の長さは、パラメータ表示関数が u だけで表示されているもので、3 次元上の曲線となる。例えば、以下の式ではグラフは図 2 のようになる。

(x =) 3*cos(u),　(y =) 3*sin(u),　(z =) u/5,　 0≦u≦20*pi

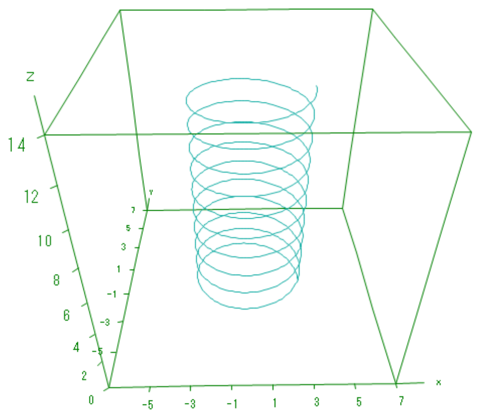


図２　３次元パラメータ表示曲線

ここでは描画の分割数を 1000 にして、滑らかになるようにしている。

座標系が「円柱座標（3 変数)」、「極座標（3 変数)」の場合、積分方法は「3 変数関数」だけが選択できる。この場合、グラフは表示できない。

参考文献

1) 高橋大輔, 理工系の基礎数学８ 数値計算, 岩波書店, 1996.

９．常微分方程式

常微分方程式は、時間変化など、1 つの変数による関数の変化の様子を式で表わしたもので、自然科学や工学などで広く利用される。特に重要な 2 階常微分方程式には、初期値問題と呼ばれる、ある変数の点での関数と関数の微分の値を与えて解を求めるものと、境界値問題と呼ばれる、2 つの変数の点での関数の値を与えて解を求めるものとがある。我々は、2 階（または 1 階）の一般的な常微分方程式の初期値問題と 2 階の線形常微分方程式の境界値問題を扱うプログラムを参考文献 1）に習って作成した。これらの解法のアルゴリズムは多くの応用分野で利用可能であり、今後 College Analysis に物理シミュレーションを加えて行く際には役に立つ。

一般に、2 階の常微分方程式は以下の形に書かれるが、

$$\frac{d^2y}{dx^2} = f\left(\frac{dy}{dx}, y, x\right)$$

初期値問題の場合は、初心者の入力も考えて、より一般に以下の形で入力するようにした。

$$f\left(\frac{d^2y}{dx^2}, \frac{dy}{dx}, y, x\right) = 0$$

1 階常微分方程式についても同様である。また、連立常微分方程式も同様の形式で、式を改行して並べて入力する。解は Runge-Kutta 法により求める。

境界値問題の場合は、1 関数で、以下の線形の形に限定している。

$$p(x)\frac{d^2y}{dx^2} + q(x)\frac{dy}{dx} + r(x)y + s(x) = 0$$

解は 3 重対角行列の LU 分解による方法で求める。

メニュー［分析－数学－常微分方程式］を選択すると、図 1 の実行メニューが表示される。

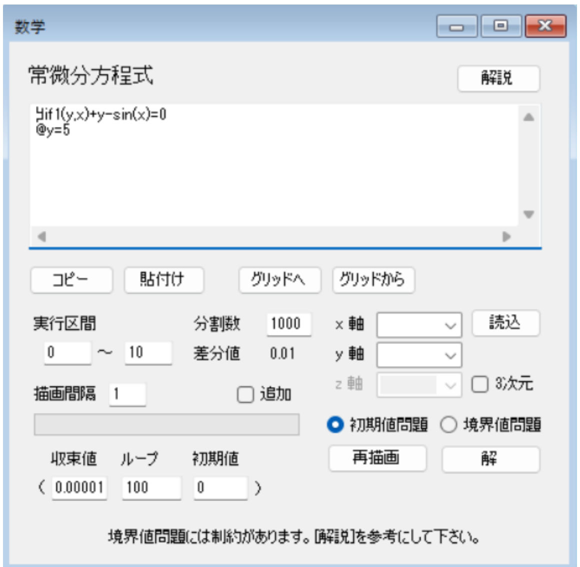


図 1　実行メニュー

微分の表記法や入力例などは、「解説」ボタンをクリックすると、図 2 のように示される。

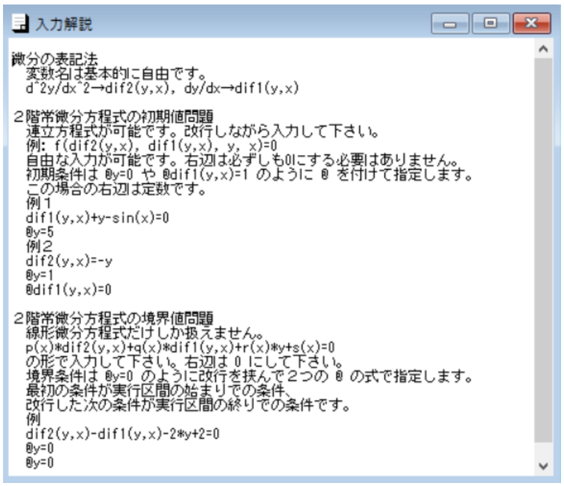


図２　解説画面

数式の中で、微分 dy/dx は dif1(y, x)、d^2y/dx^2 は dif2(y, x) で表わす。初期値問題の初期値は、@y=5 や @dif1(y, x)=0 のように表すが、メニューの「実行区間」の開始点の値として与えられる。また、境界値問題の境界値も図２の下のように、@y=0 改行 @y=0 のように表すが、これはそれぞれ実行区間の開始点と終了点に相当する。初期値問題と境界値問題の切り替えは、実行メニューの「初期値問題」、「境界値問題」ラジオボタンで行う。以後、サンプルとして表示されている以下の微分方程式（初期値問題）を用いて実行メニューの各機能を紹介する。

dif1(y, x)+y−sin(x)=0

@y=5

最初にデフォルトの設定で、「読込」ボタンで横軸と縦軸の候補を読み込み、そのまま横軸 X，縦軸 Y として、「解」ボタンをクリックし、微分方程式の解を求める。図３にその結果を示す。

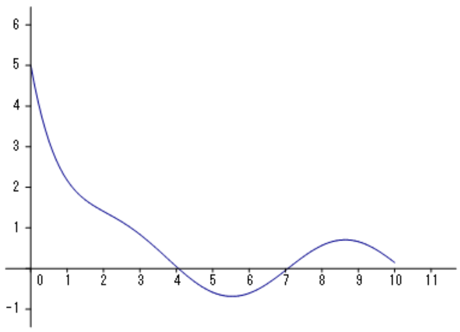


図３　X-Y グラフ

数値的な結果が知りたいときは、グラフメニュー［変数－データ表示］を利用する。

解を求める範囲を変更する場合は、「実行区間」に範囲を入力し、計算の精度を高めたい場合は、「分割数」を多くする。デフォルトでは、初期値問題で「分割数」1000，境界値問題で 100 になっている。「差分値」は実行区間と分割数から、(終了点－開始点) ÷分割数で計算され、数値微分の差分値となる。

「描画間隔」は計算した値のうち、何個飛ばしで、グラフに表示させるかを決めるもので、10 のときは分割点が 0, 10, 20, … のときに表示し、1 のときは計算した点すべてを表示する。初期値問題ではデフォルトで 10、境界値問題では 1 になっている。図 4 に「描画間隔」を 100 にして解を求めた結果を示す。

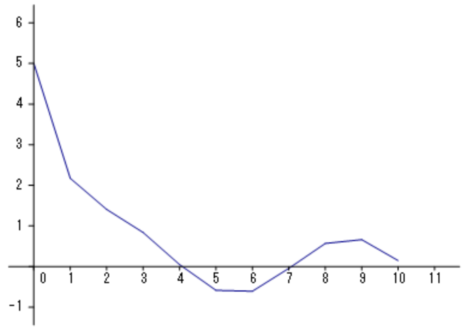


図 4　描画間隔を変えた表示

描画間隔を大きくするとグラフは粗くなるが、計算精度には影響がない。

次に、以下のような連立 1 階微分方程式の例を見る。

```
dif1(y,x)-z=0
dif1(z,x)+y=0
@y=1
@z=0
```

これは、連立 1 階微分方程式の形をしているが、以下の単振動の方程式と同じである。

$$\frac{d^2y}{dx^2}+y=0,\ y\Big|_{x=0}=1,\ \frac{dy}{dx}\Big|_{x=0}=0$$

この方程式の場合、関数が 2 つあるので、軸設定でどちらの関数を表示させるか選ぶことができる。横軸に X 、縦軸に Y を選択し、「解」ボタンをクリックすると、図 5a の結果を得る。

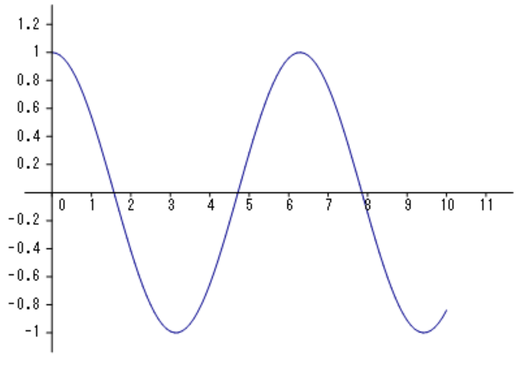


図 5a　X-Y グラフ

軸設定で、横軸に X 、縦軸に Z を選択した場合の結果を図 5b に、横軸に Z、縦軸に Y を選択した結果を図 5c に示す。

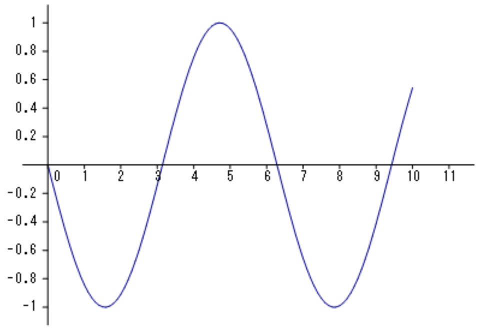


図 5b　X-Z グラフ

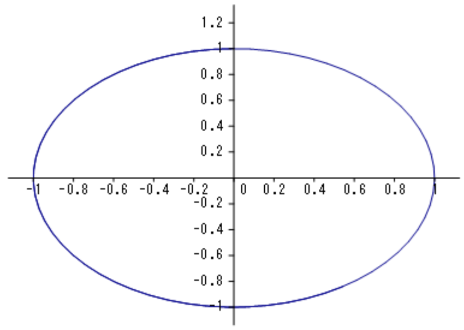


図 5c　Y-Z グラフ

この連立１階微分方程式と同等な２階微分方程式は、以下のように入力される。

```
dif2(y,x)+y=0
@y=1
@dif1(y,x)=0
```

これは、解を求める初期段階で連立１階微分方程式に書き直して計算される。この場合も上と同様、軸設定に X, Y, dY/dX（軸選択コンボボックスでは DY_DX で表示される）が選択できる。軸を変えて結果を出力する場合、長く時間のかかる計算など、「再描画」ボタンをクリックすることで、計算時間を省略することができる。

最後に原点からの太陽の重力による惑星の運動モデルを示しておく。但し、万有引力定数×太陽質量は１に設定している。入力例は以下である。

```
dif2(x,t)+x/(x^2+y^2)^1.5=0
dif2(y,t)+y/(x^2+y^2)^1.5=0
@x=1
@y=0
@dif1(x,t)=0.5
@dif1(y,t)=0.5
```

この問題は精度が必要なので、「分割数」を 10000 に設定しておく。

横軸に T 、縦軸に X を選択し、解を求めると図 6a、横軸に T 、縦軸に Y を選択すると図 6b のようになる。

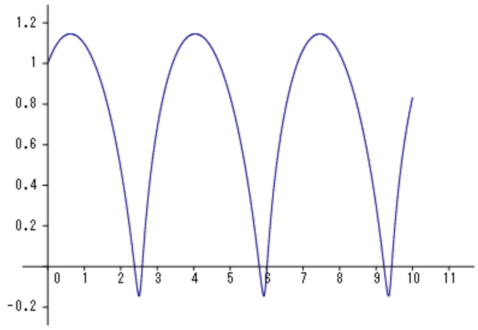


図 6a　T-X グラフ

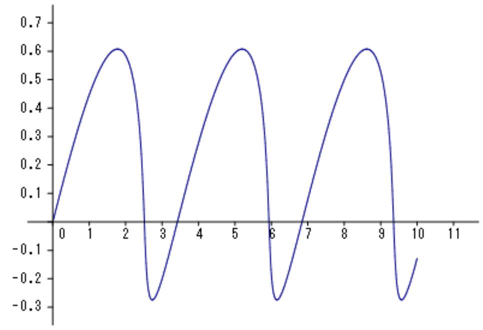


図 6b T-Y グラフ

また、軌道を見るために、横軸に X 、縦軸に Y を選択すると、図 6c のような楕円軌道になる。

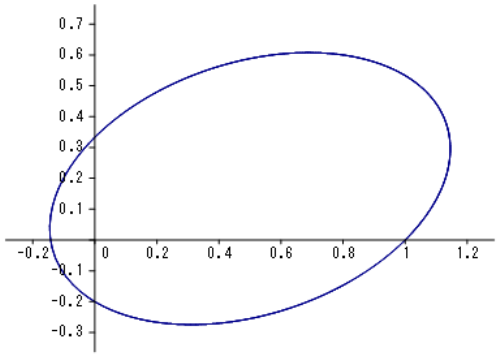
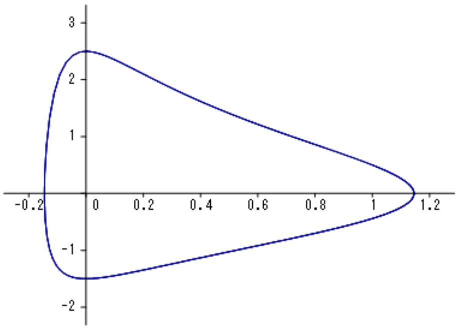


図 6c X-Y グラフ

次に、常微分方程式とは直接関係ないが、ケプラーの第３法則（面積速度一定）を見ておくことにする。まず、横軸に X、縦軸に DX_DT を選択して図 7a を描き、次に横軸に Y 、縦軸に DY_DT を選択して図 7b を描く。



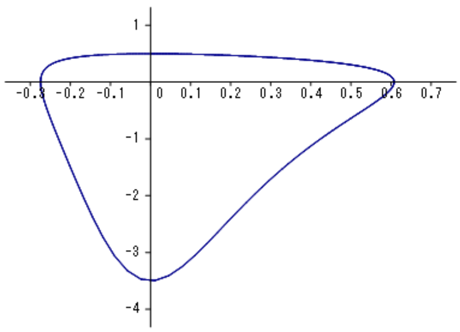


図 7a　X-DX_DT グラフ　　　　図 7b Y-DY_DT グラフ

これらのグラフメニュー［編集−データ出力］で、各時点のデータを出力し、Excel にコピーして（ここでは、時間の部分もコピーしておいた）、面積速度 $\left(xv_y - yv_x\right)/2$ を計算する。その結果を図 8 に示す。結果は右端の列のように 0.25000±0.00001 になり、面積速度は一定であることが示された。

F2 =(B2*E2-C2*D2)/2

	A	B	C	D	E	F
1	T	X	DX/DT	Y	DY/DT	x*vy-y*vx
2	0	1	0.5	0	0.5	0.25000
3	0.01	1.00495	0.49005	0.005	0.499975	0.25000
4	0.02	1.009801	0.480196	0.009999	0.499902	0.25000
5	0.03	1.014554	0.470438	0.014998	0.499782	0.25000
6	0.04	1.01921	0.460772	0.019995	0.499615	0.25000
7	0.05	1.02377	0.451195	0.02499	0.499404	0.25000
8	0.06	1.028235	0.441706	0.029983	0.49915	0.25000
9	0.07	1.032605	0.432302	0.034973	0.498854	0.25000
10	0.08	1.036881	0.42298	0.03996	0.498516	0.25000
11	0.09	1.041064	0.41374	0.044943	0.498139	0.25000
12	0.1	1.045156	0.404578	0.049922	0.497722	0.25000
13	0.11	1.049156	0.395493	0.054897	0.497268	0.25000
14	0.12	1.053066	0.386482	0.059868	0.496776	0.25000
15	0.13	1.056886	0.377544	0.064833	0.496248	0.25000

図 8　面積速度一定の検証

最後に、ローレンツの微分方程式によるローレンツ・アトラクタについて見てみる。C.Analysis ではカオスのところで現れているが、ここでは「汎用的な」常微分方程式の中での計算であることを強調しておく。

ローレンツの微分方程式はプログラム上で以下のように与えられる。

```
dif1(x,t)=-10*x+10*y
dif1(y,t)=-x*z+28*x-y
dif1(z,t)=x*y-8/3*z
@x=1
@y=0
@z=0
```

実行区間を 0～50、分割数を 5000 として、3 次元表示で変数を（X,Y,Z）に選んで解を与えると、図 9 のようになる（時間がかかる）。

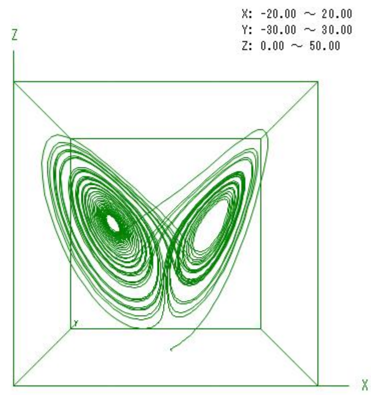


図 9　ローレンツ・アトラクタ

これを同じ 3 次元表示で変数を（T,X,Y）に選んで「再描画」ボタンをクリックすると図 10 のような結果になる。

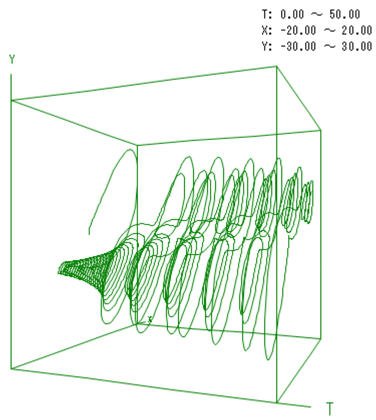


図 10　ローレンツ・アトラクタの時間変化

また、2次元表示で変数を（X,Y）に選んで「再描画」すると図11のような結果になる。

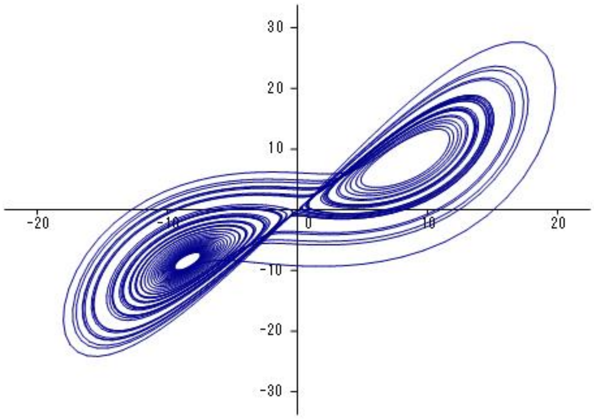


図11　ローレンツ・アトラクタ2次元表示

このように、2次元と3次元が自由に変更でき、様々な視点が与えられる。

参考文献

1）高橋大輔, 理工系の基礎数学８　数値計算, 岩波書店, 1996.

１０． ２次元幾何アニメーション

メニュー［分析ー数学ー2 次元幾何アニメーション］を選択すると、図 1 のような実行メニューが表示される。

図 1　2 次元幾何アニメーション実行画面

デフォルトで描画範囲は、$-4 \leqq x \leqq 4$, $-4 \leqq y \leqq 4$ 、デモ時間は 1 つのプログラム当たり 12 秒（描画スピードの遅れによりずれることもある）、乱数発生は Seed=1、関数等の描画の分割数は 50 である。

プログラムは中央のテキストボックス内に記述するが､｢解説｣ボタンをクリックすると、図 2 のような、記述法が表示される。

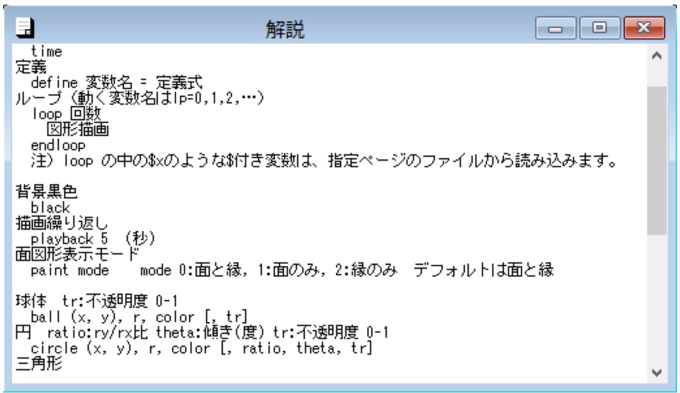


図 2 プログラム記述法画面

利用者はこの画面を参照しながらプログラムを作成する。コマンド内での数式の使い方については、メニュー［ヘルプー数式内利用可能関数］を選択して表示される関数群を参照する。

ここでは各コマンドについて、少し詳しく説明する。コマンドは大文字と小文字を区別しないので、どちらを使ってもよい。また、パラメータ間の空白は、あってもなくてもよい。

描画範囲変更

rangex xmin, xmax　(rangey ymin, ymax)

range min, max

描画範囲を一時的に変更する。メニューの描画範囲を表すテキストボックスは変化しない。2 番目の例のように、range だけだとすべての軸の描画範囲を同一に変更する。

時間変数（time=0, 0.1, 0.2, … 計算機の実行が追いつく限り実時間）

time

時間を表す予約変数で、0, 0.1, 0.2, … （秒）と増加して行くアニメーションの基本となる変数である。描画は 0.1 秒ごとなので、簡単な動画で描画が追いつけば、実時間となる。

定義

define 変数名=定義式

変数名を定義式で置き換えることができる。定義式に乱数が含まれる場合は、描画の度に乱数が発生するので、同じ値は保持しない。プログラムの実行の最初に置換処理を実行するので、描画スピードのロスは少ない。

初期定義

set 変数名=定義式

最初に定義式を計算して、変数名を置き換える。乱数を用いて初期位置を決めることができるので、define が利用できないところで有効である。定義式は数値として評価可能なものに限られる。変数の置換処理はその都度行われるので、これが含まれると実行速度が遅くなる。次で説明するループと連動して使用する。

ループ（ループ変数名は lp=0,1,2,…）

loop 回数

図形描画

endloop

複数回の処理を連続で行うときに利用する。ループの回数を表す予約変数は「lp」で、これは 0, 1, 2, … ,回数-1 まで変化する。現在のバージョンではループは 1 重までである。1 重ループを多重ループとして機能させるために、整数の割り算の余りを求める演算 % が用意されている。

$付き変数（データページ参照）

$x, $y 等

ループの中の$付き変数は、実行メニュー中のデータページにある（$を取った）同名変数の値を読み込む表記法である。ループの lp 予約変数+1 に相当する行番号のものが利用される。但し、$付き変数がなく、実行メニューの「データページ」が 0 以下ならば、データペ

ージは必要ない。

実行の繰り返し

playback 時間（秒）

実行メニューの「動画」ボタンをクリックした際の描画の実行時間を決める。その描画時間が終了したら、再描画が繰り返される。

ペイントモード

paint mode

mode 1：面と縁 2：面のみ 3：縁のみ

以下に述べる、box, circle, tri, poly, param2 コマンドにおいて、面と縁、面のみ、縁のみを表示させるモードを指定する。描画の前に指定し、次に変更して指定されるまでは、そのままの状態になっている。デフォルトは面と縁である。

描画開始・終了時間

starttime 時間（秒）

stoptime 時間（秒）

描画コマンドの開始時間と終了時間を設定する。複雑な描画では時間のずれが生じる。一度設定すると、その情報はその後の描画で保持される。

描画条件

disp 式

式≧0 のときそれ以降を描画

グリッド

grid [widths, widthl]

小さな間隔を widths、大きな間隔を widthl として、グリッドを引く。図を描くガイド用として利用し、後で消すことを想定している。

背景黒色

black

これは描画の背景を黒色にするコマンドである。

球体

ball (x, y), r, color [, tr]

座標(x,y)を中心にした半径 r の立体的な球を描画する。色を指定する color は&Hffaa80 の

ように 16 進数で表示するか、10 進数で表示する。値がよく分からない場合は、実行メニューの「色参照」ボタンで調べることができる。tr は不透明度（0～1）を与える。tr は省略可能でデフォルトは 1（不透明）である。color と tr については、他でも同じように使用する。一般に、[ ] 内のパラメータは省略可能である。

直方体

box (x, y), wx, wy, color [, theta, tr]

座標(x,y)を中心にした、横幅 wx、高さ wy、色 color の直方体を描く。theta は図形中央を中心とした回転角（単位は度）で、デフォルトは 0 度、tr は不透明度である。

正 n 角形

poly (x, y), r, n, color [, theta, tr]

座標(x,y)を中心にした、半径 r、色 color の n 角形を描く。theta は回転角（度）、tr は不透明度である。

円・楕円

circle (x, y), r, color [, ratio, theta, tr]

座標(x,y)を中心にした、横半径 r、色 color、縦横比 ratio の楕円を描く。縦横比は、縦／横で与え、デフォルトは 1 である。theta は回転角（度）、tr は不透明度である。

三角形

tri (x1, y1)-(x2, y2)-(x3, y3), color [, tr]

3 点の座標(x1, y1), (x2, y2), (x3,y3) を繋ぐ、色 color の 3 角形を描く。tr は不透明度である。

連結

connect (x1, y1)-(x2, y2), color [, mode, r, dr]

mode 1-2：線，3-4：矢印，5-6：バネ，7-8：抵抗，9-10：波線，11-12：点線，13-14 点矢印,

15-16：バネ（持ち手なし），17-18：コイル，19-20：コイル半分

21-22：左半円，22-23：右半円，31-32:左半波円，33-34：右半波円

41-42：左半 gluon，43-44：右半 gluon　　　注）偶数は太線

2 点の座標(x1,y1), (x2,y2)を繋ぐ、色 color の、線分、矢印、バネ等を描く。描画の mode は上に書かれた通りである。r は、バネ/コイルの場合は半径、波線/抵抗の場合は厚みである。dr は、バネ/コイルの場合は巻き数、波線/抵抗の場合は山と谷数、矢印の場合は矢印の位置（始点 0 終点 1）である。

関数

func y=f(x), color [, mode, min, max, div]

func f(x), color [, mode, min, max, div]

mode 1：細線 2：太線

色 color の y=f(x)の関数形を描く。x=f(y)の関数形を描くこともできる。mode は上に示した通りである。min と max は関数を描く領域の最小と最大で、デフォルトは実行メニューで定めた描画範囲である。div は描画間隔を決めるための分割数で、デフォルトは実行メニューで定めた分割数である。

関数値による色付け

colorz f(x, y) [, mode, xmin, ymin, xmax, ymax, div, tr]

関数 f(x,y) の大きさにより、領域に虹色の色付けを行う。最小が紫、最大が赤である。mode=1 は 1 回の描画の値による描画、mode=2 はそれまでの最大値を保持させるものとする。後者は、例えば波の干渉などで、振幅の大きな領域を見つけるのに役に立つ。デフォルトは mode=1 である。xmin, ymin, xmax, ymax は描画領域（固定）、div はその領域の x,y 方向の分割数である。

パラメータ曲線［0<=u<=1］

param1 (x(u), y(u)), mode, color [, div]

mode 1：細線 2：太線 3：矢印 4：太矢印

予約パラメータ「u」で指定するパラメータ表示関数を色 color で描く。パラメータが動く範囲は 0≦u≦1 である。mode は上に示した通りである。矢印は u=1 の側に付く。r は矢印の長さである。

パラメータ面［0<=u,v<=1］

param2 (x(u,v), y(u,v)), color [,div1, div2, tr]

2 つの予約パラメータ「u, v」で指定するパラメータ表示関数を色 color で描く。パラメータ u, v は 0 から 1 までの間を分割数 div1 と div2 で動くものとする。tr は不透明度である。通常は 3 次元空間内に描くものだが、ここでは 2 次元平面内で面を描くのに利用する。

陰関数

impfunc f(x,y), color [, mode, div]

mode 1：細線 2：太線

f(x,y)=0 で与えられる陰関数のグラフを色 color で描く。mode は上に示した通りである。

div は表示領域の x,y 方向をいくつに分割して描画するかを決める分割数である。

クリップボード中の画像貼り付け

clip (x, y), wx [, theta]

クリップボードにある画像を、座標(x,y)を中心に、wx を横幅として貼り付ける。その際、画像の縦横比は保持する。theta は画像の回転角度（度）である。

文字列

string (x, y), ”文字列”, color [, size]

座標(x, y)の位置に指定された文字列（” ” でくくる）を色 color で描く。文字のフォントサイズは size で指定するが、デフォルトは 9 point である。

コマンド開設の中でも述べたが、描く図形の色は RGB の 16 進表示（10 進表示でもよい）で指定する。なじみの薄い利用者もいるので、実行メニューに「色参照」ボタンを加え、色を選びながら、数字を指定できるようにしている。描く図形によって図形表示の最大数が決まっており、指定した数が多い場合は、例えば「ball 数<=最大数」のようにエラーメッセージが表示される。これを変更する場合は、例えば「maxball 20000」のように、max の後に描画コマンド名を続けて、必要な数値を書く。図形描画コマンドのパラメータは、時間変数 time や繰り返し変数 lp などを使って帰ることができるが、数値で表し、固定値として扱うものもある。例えば、描画モードを表す mode や分割数を表す div、colorz コマンドの描画領域などである。また、図形描画以外のコマンドのパラメータは殆ど固定である。

以後は例を用いてコマンドの利用法を説明する。行の最初の数字は説明用の行番号であり、実際には入力しない。図は通常正方形の描画領域内に表示されるが、画面の都合により、必要部分を切り取って示す。

例 1　バネ

1. box (0,3.5),6,1,&H804000
2. define y=sin(2*time)
3. connect (0,3)-(0,y),&Hff0000,5
4. ball (0,y-0.2),0.3,&Hff

1 行目はバネをつるす天井を描いている。(0,3.5)を中心として、幅 6、高さ 1 の直方体を色番号&H804000 で描く。2 行目は運動するおもりの位置を y として与える定義文である。運動は位相 θ=2t（秒）で与えられる。3 行目は、コマンド connect の mode が 5 で、赤色のバネを表している。4 行目は中心を y-0.2 とした、青色のボールを表している。図 3 に描

画結果を示す。

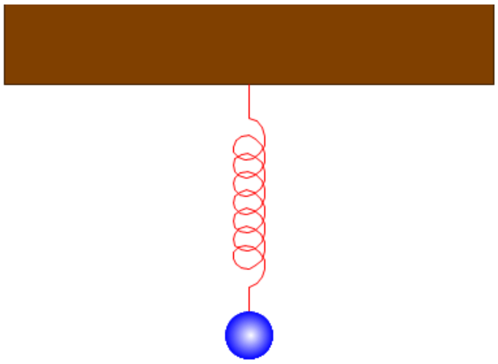


図 3 バネ

例２　ボックスとボールの回転

1. circle (0,0),2,&Hff,0.5,90*time
2. box (0,0),1,0.5,&Hff00,90*time

1 行目は座標(0, 0)を中心とした長半径 2、縦横比率 0.5 の楕円を表す。楕円は 1 秒間に 90 度の角度で回転をさせている。但し、パソコンの描画速度によって遅れる場合もある。2 行目は(0, 0)を中心とした、幅 1、高さ 0.5 の緑色の長方形を描く。長方形は楕円と同じ速さで回転させている。描画はプログラムの順番に描かれるので、circle の上に box が描かれている。図 4 に描画結果を示す。

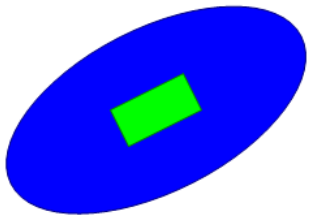


図 4 ボックスとボールの回転

例３　多角形の回転

1. poly (-1,0),2,6,&Hff,-90*time,0.2
2. poly (1,0),1.5,5,&Hff00,90*time,0.2

1 行目は半径 2 の青色で半透明な（不透明度 0.2）六角形を描くコマンドで、2 行目は半径 1.5 の緑色で半透明な（不透明度 0.2）五角形を描くコマンドである。2 つの多角形の回転速度は 1 秒間に 90 度で、逆方向に回っている。図 5 に描画結果を示す。

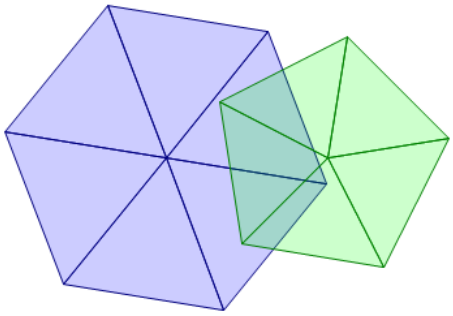


図 5 多角形の回転

例４　円運動

1. define x=3*cos(time)
2. define y=3*sin(time)
3. ball (x,y),0.2,&Hff
4. connect (x,y)-(0,0),&H606060,1
5. connect (x,y)-(2*x/3,2*y/3),&H8080,4
6. connect(x,y)-(x-y/3,y+x/3),&Hff0000,4

1,2 行目は円運動の位置の定義文である。3 行目はその位置に青色のボールを描くコマンドである。4 行目では、connect コマンドの mode=1 を使って、中心とボールを繋ぐ線、5 行目は加速度の方向、6 行目は速度の方向を mode=4 の太矢印で描画している。図 6 に描画結果を示す。

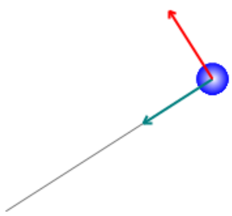


図 6 円運動

例５　坂と滑車

1. playback 3
2. define l=(time/1.5)^2
3. define x1=-2.2+l*cosd(30)
4. define y1=-2.11+l*sind(30)
5. define x2=3.2
6. define y2=-0.5-l

```
7.  connect (x1,y1)-(3,1/3^0.5+0.4),&H404040,1
8.  connect (x2,y2)-(3.2,1/3^0.5+0.2),&H404040,1
9.  tri (-3.2,-3)-(2.8,-3)-(2.8,-3+6*tand(30)),&H804000
10. box(x1,y1),0.7,0.5,&Hff0000,30
11. box(x2,y2),0.7,0.5,&H800000
12. circle(3,1/3^0.5+0.2),0.2,&Hff8040
13. connect (3,1/3^0.5+0.2)-(2.8,-3+6*tand(30)),&Haa0000,2
```

1 行目では、描画の繰り返し時間を 3 秒に設定している。3,4,9,13 行目の cosd(), sind(), tand()関数は、引数を度単位で表した三角関数である。9 行目の tri は台の三角形を描くコマンドである。図 7 に描画結果を示す。

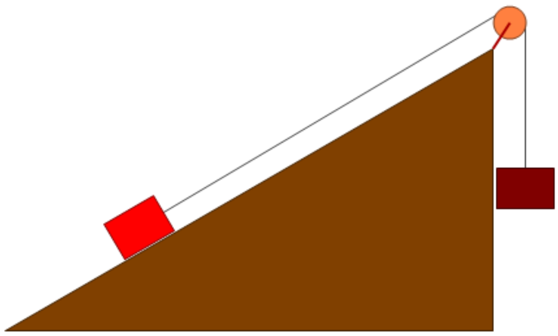


図 7 坂と滑車

例６　クリップボードからの描画

```
1. loop 10
2. define x=2*cos(vth(time-(10-lp)/2.5))
3. define y=2*sin(vth(time-(10-lp)/2.5))
4. clip (x,y),2-(10-lp)*0.1
5. endloop
```

1行目から5行目の loop 10 ～ endloop はその間の描画を 10 回繰り返すコマンドである。その間、予約変数 lp は 0 から 9 まで値が動く。define はループの間でも前後でも同じ働きをする。4 行目の clip はクリップボードにコピーした画像を表示するコマンドである。大きさは描画幅で指定するが、縦横比は保持される。2,3 行目で使われた vth()は引数が負の時は 0、正の時はその値となる関数である。必要に応じて図を回転させることもできる。図 8 に描画結果を示す。

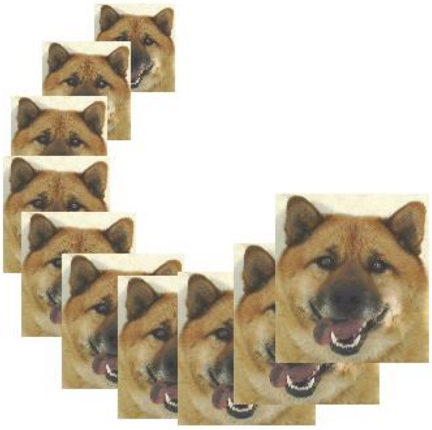


図 8 クリップボードからの描画

例７　定常波

1. define y1=0.5*sin(2*pi*(x-time/5))
2. define y2=0.5*sin(2*pi*(x+time/5))
3. func y1,&Hff,,-3,3,100
4. func y2,&Hff00,,-3,3,100
5. func y1+y2,&Hff0000,2,-3,3,100
6. connect (-0.5,1.5)-(0.5,1.5),&Hff,4
7. connect (0.5,-1.5)-(-0.5,-1.5),&Hff00,4
8. box (-3.2,0),0.4,2,&H804040
9. box (3.2,0),0.4,2,&H804040

この例の 3,4,5 行の func は、y=f(x)または x=f(y)の形の関数を描画するコマンドである。この場合は y1,y2 の中に変数 x が含まれているため、y=f(x)の描画となる。描画の mode を指定しない場合は mode=1 となり細線、2 の場合は太線となる。ここでは 5 行目で記述された赤色の定常波が太線である。図 9 に描画結果を示す。

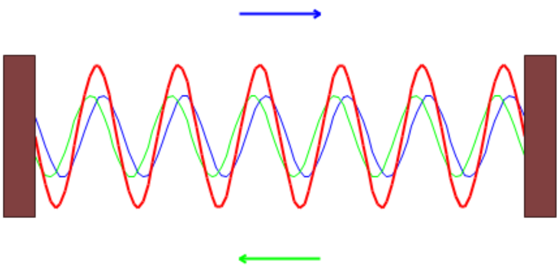


図 9 定常波

例８　超音速

1. playback 15

```
2.  define x1=(lp*0.04)^2-2.5
3.  define x2=(0.2*time)^2-2.5
4.  paint 3
5.  loop 100
6.  circle (x1,0),vth(time/2-lp/10), &Hff
7.  endloop
8.  paint 1
9.  tri (x2,0)-(x2-0.2,0.1)-(x2-0.2,-0.1),&Hff0000
```

これは音速を超える際の音波の伝搬を描画したものである。1 行目は 15 秒ごとに再描画するコマンドである。4 行目と 8 行目の paint は、それぞれ描画を縁のみと縁と面にするコマンドである。図 10 に描画結果を示す。

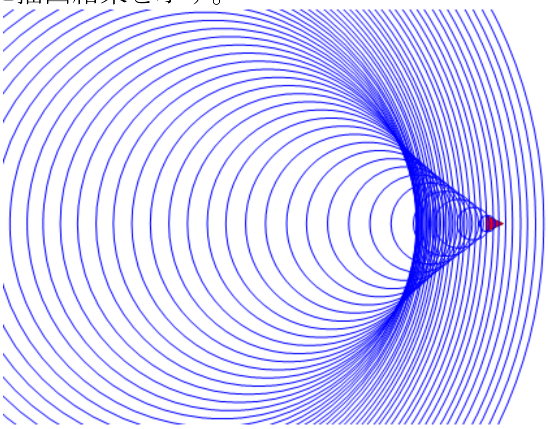


図 10 超音速

例９　リサジュー図形

```
1.  define x1=2*cos(2*pi*u*0.9+time)
2.  define y1=sin(4*pi*u*0.9-pi/4+time)
3.  param1 (x1,y1),rainbow(u),4,100
```

これは時間とともに動くリサジュー図形の例である。3 行目の param1 は 2 次元のパラメータ表示関数を表すコマンドである。パラメータは、予約変数として u が使われ、0≦u≦1 の間の値を取る。パラメータ u は通常、実行メニューの分割数で指定された数で分割され描画されるが、描画を滑らかにする場合など、3 行目の最後の 100 のように、分割数を増やすこともできる。3 行目の rainbow()関数は引数の値が 0 から 1 の範囲で、紫色から赤色の虹色の数値を与える関数である。また、mode の値は太矢印の mode=4 が与えられている。図 11 に描画結果を示す。

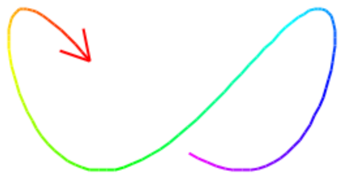


図 11 リサジュー図形

例 10　等電位面

```
1.  define x1=2*sin(time)
2.  define x2=-2*sin(time)
3.  define y1=2*cos(time)
4.  define y2=-2*cos(time)
5.  define r1=((x+x1)^2+y^2)^0.5
6.  define r2=((x+x2)^2+y^2)^0.5
7.  define r3=(x^2+(y+y1)^2)^0.5
8.  define r4=(x^2+(y+y2)^2)^0.5
9.  define p=1/(r1+0.001)+1/(r2+0.001)+1/(r3+0.001)+1/(r4+0.001)
10. loop 3
11. impfunc p-1.1*lp-2,&Hff,2,100
12. end loop
13. ball(x1,0),0.2,&Hff00
14. ball(x2,0),0.2,&Hff0000
15. ball(0,y1),0.2,&Hffff
16. ball(0,y2),0.2,&Hffff00
```

ここでは 4 つの等しい電荷が作る等電位面を陰関数を用いて描いている。11 行目の impfunc は f(x,y)=0 で表される図形を描画する。ここで f(x,y)に相当する p-1.1*lp-2 の p は 9 行目に定義されている。また impfunc コマンドで描画を滑らかにするために、領域を x,y 方向で 100 分割にしている。図 12 に描画結果を示す。

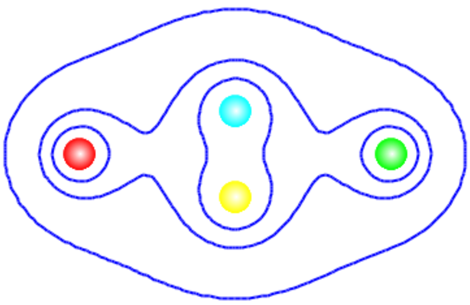


図 12 等電位面

例 11　重ね合わせの原理

```
1.  playback 10
2.  maxcircle=200
3.  define d=1
4.  define f=2
5.  define r1=((x+d)^2+y^2)^0.5
6.  define r2=((x-d)^2+y^2)^0.5
7.  define z1=sin(2*pi*f*vth(time-r1))+sin(2*pi*f*vth(time-r2))
8.  colorz z1+z2,2,,,,,101
9.  loop 40
10. paint 3
11. circle(d,0),vth(time-lp/f-0.25/f),&H0
12. circle(-d,0),vth(time-lp/f-0.25/f),&H0
13. endloop
```

2 点から発せられる円形波の干渉縞を振幅の大きい部分と小さい部分で塗り分けた図である。波の重なった部分は振幅が大きく、赤くなっている。8 行目の colorz が関数の値によって領域を塗り分けるコマンドである。この場合は過去の最大振幅を記憶する mode=2 となっている。また、分割は結果をきれいに表示するために、101 とデフォルトより多くしている。7 行目によると、この波は速さ 1、周期 0.5、速さ×周期である波長が 0.5、円形波の中心間の距離が 2 の設定になっており、波の線は 11,12 行目から 1 波長ごとに描かれている。図 13 に描画結果を示す。

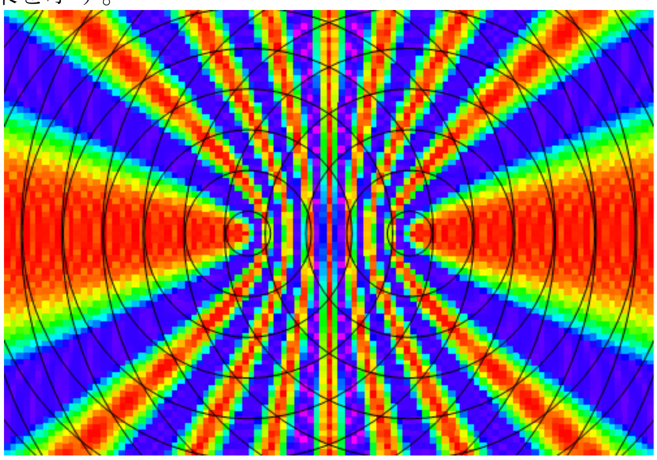


図 13 干渉縞

１１．３次元幾何アニメーション

これまで 3 次元物理シミュレーションで、質点系や静電磁場の問題を扱ってきたが、その際に環境を含めて描くシミュレーションを目指すことにした。ここではその準備のために種々の描画に必要な要素を開発している。シミュレーションとの連携は今後の課題であるが、この図形の描画法だけでも数学関数や空間座標概念の把握に役に立つ。

メニュー［分析－数学－3 次元幾何アニメーション］を選択すると図 1 のような実行メニューが表示される。

図 1 実行メニュー

利用法は 2 次元幾何アニメーションとほぼ同じである。コマンドについて、基本的な事項は 2 次元のコマンドと同様であるので、重複する部分は簡単な説明に留める。結果はこれまでに開発した 3D ビューアに表示する [1]。以下のコマンドについて、利用法は 2 次元の場合と同じである。

描画範囲変更

rangex xmin, xmax （rangey ymin, ymax / rangez zmin,zmax）

range min, max

2020/12 の改訂で、異なる範囲でも直方体領域で描画されるようにした。

時間変数（time=0, 0.1, 0.2, … 計算機が追いつく限り実時間）

time

定義

define 変数名=定義式

初期定義

set 変数名=定義式

ループ（ループ変数名は lp=0,1,2,…）

loop 回数

図形描画

endloop

$付き変数（データページ参照）

$x, $y 等

実行の繰り返し

playback 時間（秒）

ペイントモード

paint mode

mode 1：面と縁 2：面のみ 3：縁のみ

描画開始・終了時間

starttime 時間（秒）

stoptime 時間（秒）

描画条件

disp 式

式≧0 のときそれ以降を描画

背景黒色

black

これ以後のコマンドは 2 次元の場合とは異なるので、少し詳しく説明する。また 2 次元の場合と同じく、コマンド等は大文字と小文字を区別しない。

切り取りモード

cutmode

これは図形がある距離まで近づくと、近づいた部分の表示が消えるモードで、内部構造などを見るときに利用する。

座標軸表示

axis

これは実行メニューで与えられる描画領域を囲む座標軸を描くコマンドである。

初期角度

angle 角度（度）

これは図形が最初に表示されるときの見る角度を指定するコマンドである。0（度）の場合は正面から、90 度の場合は真上から見た図になる。

小さな球体

ball (x, y, z), r, color [, tr]

座標(x,y,z)を中心にした、半径 r、色 color の、立体的に見える球体を描く。不透明度 tr を指定すると、立体表示は無くなり通常の半透明な球になる。デフォルトは不透明な立体的球体である。

大きな球体（分割は指定分割）

spher (x, y, z), r, color [, theta, phi, rot, tdiv, pdiv, tr]

座標(x, y, z)を中心とした、半径 r、色 color の球体を描く。その際、中心軸の z 軸からの傾きを theta（度）、軸の旋回角度を phi（度）とし、軸の周りの回転角度を rot（度）とする。また tdiv と pdiv はそれぞれ theta 方向と phi 方向の分割数、tr は不透明度である。デフォルトは theta=0, phi=0, rot=0, tr=1 であり、tdiv と pdiv は実行メニューに与えられた分割数である。これらの分割数を少なくすることによって多面体を表現することもできる。回転を分かり易くするために、予約変数「phi」を利用した経線方向の色付けが可能である。

三角形

tri (x1, y1, z1)-(x2, y2, z2)-(x3, y3, z3), color [, tr]

3 点の座標(x1, y1, z1), (x2, y2, z2), (x3, y3, z3)を繋ぐ、色 color の三角形を描く。tr は不透明度である。

正 n 多角形

poly (x, y, z), r, n, color [, theta, phi, rot, rdiv, tr]

座標(x,y,z)を中心にした、半径 r、色 color の正 n 角形を描く。正 n 角形と垂直な軸の z 軸からの傾きを theta（度）、軸の旋回角度を phi（度）とし、軸の周りの回転角度を rot（度）とする。回転を分かり易くするために、予約変数「phi」を利用した経線方向の色付けが可能である。rdiv はデフォルトが 1 の半径方向の分割数、tr は不透明度である。

小さな直方体（描画に多少の乱れが生じる）

box (x, y, z), wx, wy, wz, color [, theta, phi, rot, tr]

座標(x,y,z)を中心にした、x 方向 wx、y 方向 wy、z 方向 wz の大きさ、色 color の直方体を描く。z 軸との傾きを theta（度）、軸の旋回角度を phi（度）とし、軸の周りの回転角度を rot（度）とする。

連結

connect (x1, y1, z1)-(x2, y2, z2), color [, mode, r]

種類 1：細線，2：太線，3：矢印，4：太矢印，5：バネ

2 点の座標(x1, y1, z1), (x2, y2, z2)を繋ぐ、色 color の、線分、矢印、バネ等を描く。描画の mode は上に書かれた通りである。r はバネを描いた場合の半径である。

床・天井・平面波

func z=f(x, y), color [, min1, min2, max1, max2, div, tr]

func f(x, y), color [, min1, min2, max1, max2, div, tr]

色 color の z=f(x, y)の関数形を描く。x=f(y, z), y=f(z, x) の関数形を描くこともできる。描画領域は、座標(min1,min2)と(max1,max2) を対角線とする四角形である。分割は 1 方向当たり div で指定する。デフォルトは実行メニューの分割数である。

パラメータ曲線［0<=u<=1］

param1 (x(u), y(u), z(u)), mode, color [, div, r]

mode 1：細線 2：太線 3：矢印 4：太矢印

予約パラメータ「u」で指定するパラメータ表示関数を色 color で描く。パラメータ u は 0 から 1 までの間を分割数 div で動くものとする。描画 mode は、細線、太線、矢印、太矢印がある。r は矢印の長さである。

パラメータ曲面［0<=u,v<=1］

param2 (x(u,v), y(u,v), z(u,v)), color [, div1, div2, tr]

2 つの予約パラメータ「u,v」で指定するパラメータ表示関数を色 color で描く。パラメータ u,v は 0 から 1 までの間を分割数 div1 と div2 で動くものとする。tr は不透明度である。

文字列

string (x, y, z), ”文字列”, color [, size]

座標(x, y, z)の位置に指定された文字列（” ” でくくる）を色 color で描く。文字のフォントサイズは size で指定するが、デフォルトは 10 point である。

以後は 2 次元の場合と同様、例を用いてコマンドの利用法を説明する。行の最初の数字は説明用の行番号であり、実際には入力しない。図は通常正方形の描画領域内に表示されるが、印字の都合により、必要部分を切り取って示す。

例 1　銀河のような形状

1. black
2. ball(0,0,0),0.3,&Hffffff,0.5
3. loop 800
4. define x=$r*cos(time/($r+1)+$a)
5. define y=$r*sin(time/($r+1)+$a)
6. ball(x,y,$z),0.05,&H8888ff
7. endloop

1 行目では背景を黒色にし、2 行目は原点に白い球形を描いている。3 行目から 7 行目は繰り返しで、800 個の小さな球形を描く。4,5 行目は、loop の前にあっても中にあっても、機能は同じである。$r, $a には実行メニューのデータページにある r, a 変数のデータの値をループ予約変数 lp の値に応じて、lp+1 行目から読み出して行く。動きは time 変数で与える。データさえ与えておけば星の数は 10000 個程度でも問題なく動く。図 2a に描画結果を示す。

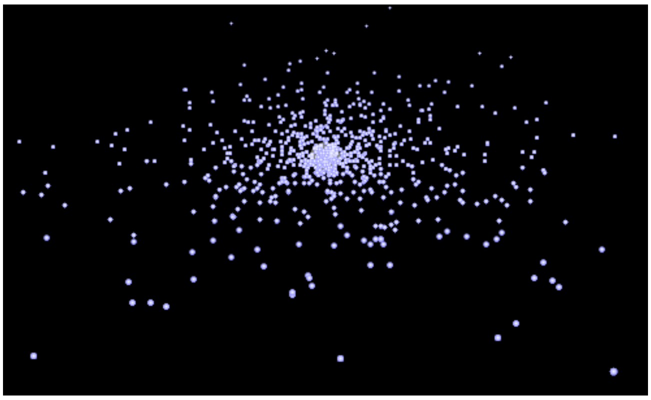


図 2a　銀河のような形状

これは力学的法則を用いてシミュレーションを実行しているわけではない。あくまで単なる点の回転である。またデータページにデータを与えない限り、サンプルとして実行することはできない。

同様な処理を ball の数を増やして、データページを使わずに示しておく。これならデータなしにすぐに実行できる。

1. black
2. angle 10
3. ball(0,0,0),0.3,&Hffffff,0.5

```
4. set r=3*abs(nrnd)
5. set a=2*pi*rnd
6. set z1=rnd-0.5
7. loop 9000
8. define x=r*cos(time/(r+1)+a)
9. define y=r*sin(time/(r+1)+a)
10. define z=exp(-r^2)*z1
11. ball(x,y,z),0.03,&Haaaaff
12. endloop
```

ここでは ball の数を 9000 個にしている。4, 5, 6 行目の set コマンドは、要素 1 個 1 個の定義データ（計算式）を作る際に乱数に予め数値を割り当て、描画の段階ではその数値を利用する方法で描画する。描画スピードは落ちるが、データページを使わず似た効果を出せる利点がある。2 行目の angle コマンドでは、最初の見る角度を水平から 10 度に設定している。図 2b に描画結果を示す。

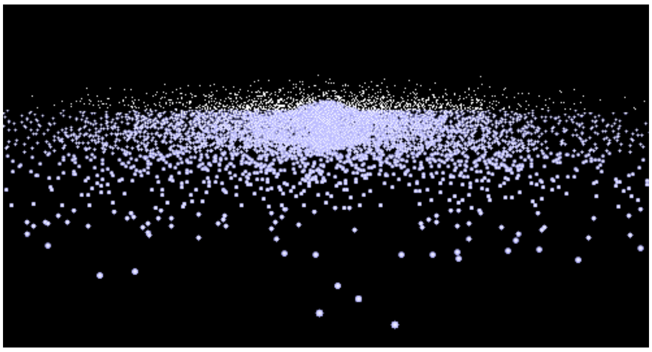


図 2b データページを使わない銀河のような形状

例２　2 つの球体

```
1. define col1=&Hffff*theta(phi-pi*0.01)+&Hff*theta(pi*0.01-phi)
2. define col2=&Hffff00*theta(phi-pi*0.01)+&Hff00*theta(pi*0.01-phi)
3. spher(2.5*cos(time),2.5*sin(time),0),2,col1,,,90*time,20,,0.25
4. spher(2.5*cos(time),2.5*sin(time),0),0.5,&Hff0000,,,90*time,20,,0.5
5. spher(2.5*cos(time+pi),2.5*sin(time+pi),0),1,col2,-30,,-180*time,20
```

ここでは、spher コマンドを使って 2 つの球体を描いているが、大きい方の中にはもう 1 つ球体がある。5 行目の小さい方の球体は軸を 30 度傾けている。軸方向の回転速度は大きい球体で 90 度／秒、小さい球体で、-180 度／秒である。分割数は、theta 方向を 20、phi 方向はデフォルトの 30 に設定している。また回転が分かるように予約変数 phi が 0 の近傍

で色を変えるようになっている。1,2 行目の theta()は引数が 0 以上の時に 1、負のときに 0 となる関数である。図 3 に描画結果を示す。

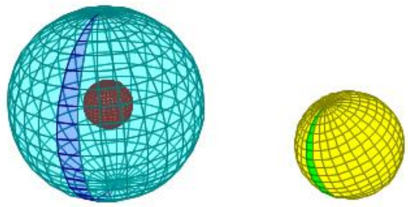


図 3　2 つの球形

例３ 波とプール

1. playback 10
2. define z1=0.2*sin((x^2+y^2)^0.5*3-4*time)
3. define def=theta(4*time-(x^2+y^2)^0.5*3)
4. func z=z1*def,&Hffcc,-5,-5,5,5,,0.5
5. define dz=0.2*sin(-4*time)
6. ball(0,0,dz),0.3,&Hff
7. connect (0,0,3)-(0,0,dz+0.2),&Hff0000,5
8. box (0,0,0),10,10,5,&Hffff,,,,0.1

これは円形波の例で、3 行目で定義された変数 def で最初の波の到達を与える。波の形は 4 行目の func コマンドで､z1 の中に x と y が time 変数と共に含まれることで与えられる。波の描画範囲は x,y の範囲として、(-5,-5)と(5,5)を対角とする正方形で与えられる。6 行目でボールを描き、7 行目で connect コマンドを用いてバネを描いている。8 行目はプール自体を半透明に描いている。図 4 に描画結果を示す。

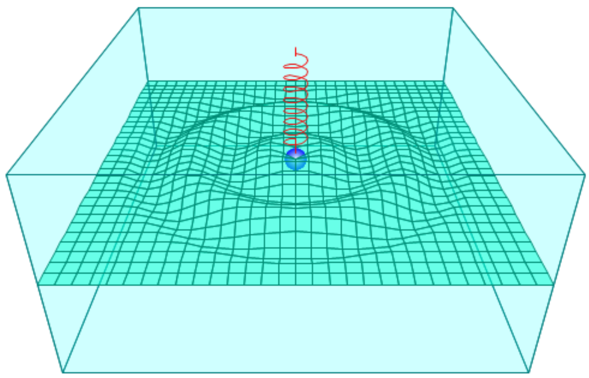


図 4 波とプール

例４　力学（バネ）

```
1.  axis
2.  define z=2*sin(2*time)
3.  ball(0,0,z),0.5,&Hff00
4.  connect (0,0,4)-(0,0,z+0.51),&H8888,5
5.  connect (0,-0.2.,z)-(0,-0.2,z-2),&Hff,4
6.  connect (0,-0.2.,z)-(0,-0.2,2),&Hff0000,4
7.  connect (1,-0.2,z)-(1,-0.2,0),&Hff00ff,4
8.  string (1.7,-0.2,z/2), "合力",&Hff00ff,20
```

これは今後応用して行く物理シミュレーション用に作られたものである。バネとおもりの力の向きをベクトルで表し、その中に文字を入れている。また、動きが分かり易いように、1 行目の axis コマンドで座標軸も加えている。8 行目の string コマンドは文字列を描くコマンドで、フォントサイズは
20 であるが、表示の大きさは文字の位置により変わってくる。図 5 に描画結果を示す。

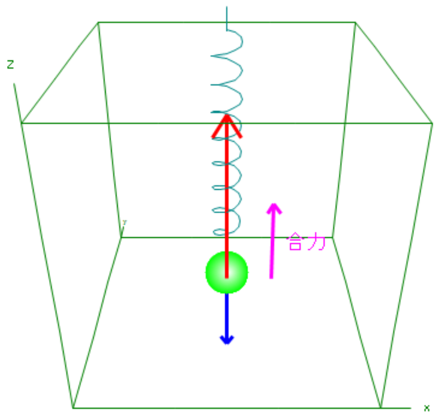


図 5 力学（バネ）

例５　斜面

```
1.  playback 3.5
2.  func z=tan(pi/6)*x,&Hccffff,-10,-10,10,10
3.  define x1=-time^2+4
4.  define z1=x1*tan(pi/6)+0.5
5.  box (x1,-3,z1),1,1,0.5,&Hff0000,-30,0,-180*time
6.  ball (x1,0,z1),0.4,&H8800
7.  box (x1,3,z1+0.1),1,2,0.5,&Hff,-30,0,90*time
```

これは 2 つのボックスとボールが斜面を回転しながら滑り落ちるプログラムで、2 行目で斜面を与え、3, 4 行目でボックスとボールの位置を定義している。それぞれの動きは、5, 6,

7 行目で与えるが、ボックスにはそれぞれ-180, 90（度／秒）の回転を与えている。図 6 に描画結果を示す。

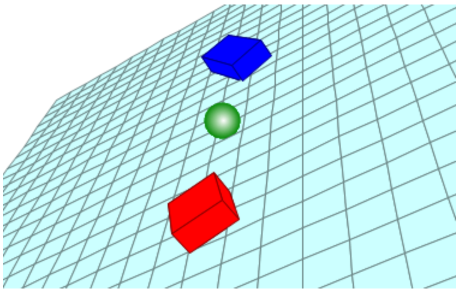


図 6 斜面

例６　矢印

1. define x1=3*cos(2*pi*u+2*time)
2. define x2=3*sin(2*pi*u+2*time)
3. define x3=4*u-2+sin(2*time)
4. param1 (x1,x2,x3),rainbow(u),4
5. param1 (x2,x3,x1),rainbow(u),4
6. param1 (x3,x1,x2),rainbow(u),4

これは param1 コマンドを使った矢印の動きを表す例で、1,2,3 行目で与えた定義により、4,5,6 行目で座標の役割をサイクリックに変えて表示したものである。色は rainbow 関数で予約パラメータ u の値に応じて付けられている。また、param1 コマンドの mode は矢印・太線を与える 4 である。図 7 に描画結果を示す。

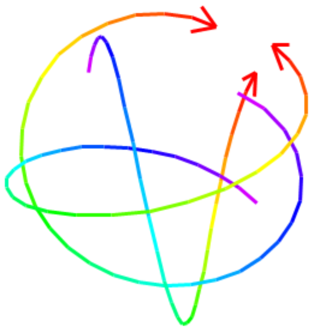


図 7 矢印

例７　トーラス

1. define x=(3+1.5*sin(2*pi*u))*cos(2*pi*v)
2. define y=(3+1.5*sin(2*pi*u))*sin(2*pi*v)
3. define z=1.5*cos(2*pi*u)+2*sin(sin(time)*u)
4. param2(x,y,z),rainbow(u)

これは param2 コマンドの使用例で、3 次元パラメータ表示のトーラスに少し時間的な動きを加えたものである。3 行目の第 2 項によって動きが得られる。図 8 に描画結果を示す。

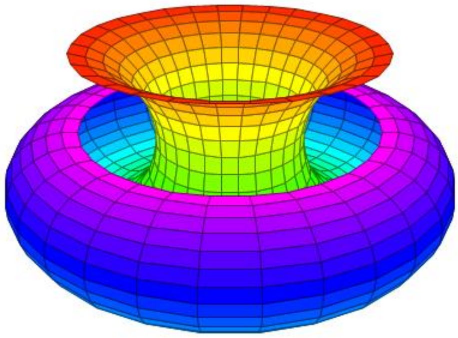


図 8 トーラス

例８　コマ

```
1.  define col=&Hff*theta(phi-175)+&Hffff*theta(175-phi)
2.  define x1=2*sind(30)*cosd(90*time)
3.  define y1=2*sind(30)*sind(90*time)
4.  define z1=2*cosd(30)-2
5.  define x2=4*sind(30)*cosd(90*time)
6.  define y2=4*sind(30)*sind(90*time)
7.  define z2=4*cosd(30)-2
8.  define th=30
9.  define alpha=-(90*time) mod 360
10. poly (x1,y1,z1),3,6,col, th, alpha, 270*time, 3
11. connect (x1,y1,z1)-(x2,y2,z2), &Hff0000,2
12. connect (x1,y1,z1)-(0,0,-2), &Hff0000,2
13. connect (0,0,-2)-(0,0,-4),&H6600,2
14. poly (0,0,-4),2,30,&Hff00, , , , 6
```

これは支柱の上で回転する 6 角形のコマを表現している。10 行目の poly は 6 角形のコマを描画するコマンドで、軸の傾きが th、軸の旋回角度が alpha である。これらはそれぞれ 8 行目と 9 行目で定義されている。コマ自身の回転は軸の周りに 270 度／秒であり、コマの半径方向の分割数は 3 となっている。13, 14 行目が台座の描画である。図 9 に描画結果を示す。

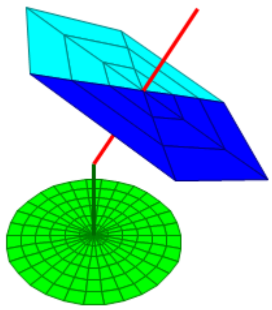


図 9 コマ

例９　スイカ

```
1.  paint 2
2.  cutmode
3.  angle 0
4.  define y1=2-5.5*abs(sin(time))
5.  define col=&Hff00*(phi % 36)/18)
6.  spher (0,y1,0), 3, col, , , , , 20
7.  spher (0,y1,0), 2.9, &Hffffff, , , , , 20
8.  spher (0,y1,0), 2.5, &Hff0000, , , , , 20
9.  loop 100
10. define x=$r2*sin($theta2)*cos($phi2)
11. define y=2+$r2*sin($theta2)*sin($phi2)
12. define z=$r2*cos($theta2)
13. define y2=y-5.5*abs(sin(time))
14. ball (x,y2,z),0.1,&H0,1
15. endloop
```

これはスイカが近づいたり遠ざかったりする描画である。2 行目の cutmode コマンドで、近づいた際、中が見えるようになっている。スイカらしくするために、1 行目の paint 2 コマンドでフレームを表示しないようにしている。また、cutmode を効果的にするために 3 行目の angle 0 で、最初の見る角度を水平面からにしている。スイカの位置は、4 行目で y 方向に振動させている。また色の区分けは 5 行目で予約変数 phi の値が飛び飛びになることを利用して付けられている。図 10 に描画結果を示す。

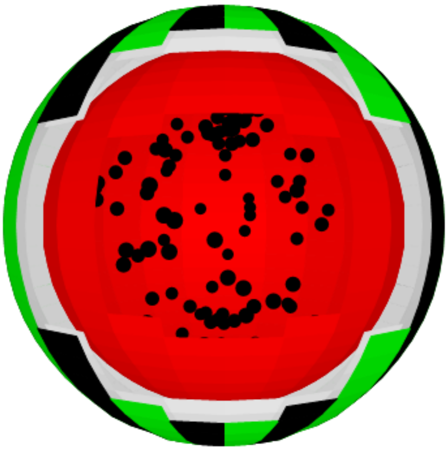


図 10　スイカ

１２．行列計算

数学の行列計算は College Analysis の中で統計学をはじめとする様々な分野のプログラムで用いられてきたが、我々はこの計算プログラムを分析の表題として表に出すことはなかった。その理由は授業等で利用する機会がなかったからだけでなく、どのような形式にすれば簡単に利用できるプログラムを作ることができるのか分からなかったからである。今回この問題について考え、１つの回答を見出したので紹介する。

行列はデータとして、グリッドエディタのセルに「A=」のように名前を付けて自由に記述し、実行メニューの中で名前を利用した数式表現を用いて、計算を実行する。結果はグリッド出力を用いて行列の形式で出力される。このプログラムの面白いところは、単純に数値計算を行うだけでなく、行列を文字列として計算できるところで、例えば固有方程式などを（形は見易くないが）数式として表現して、結果をそのまま方程式ソルバーにコピー・ペーストして固有値などを求めることもできる。同様のことは連立 1 次方程式についても実行できる。また、行列表示した楕円の式などの結果を陰関数表示のプログラムにコピーして結果をグラフ表示することも可能である。これは行列の教科書で習う計算をそのままシミュレートすることに相当しており、教育効果も期待できそうである。College Analysis に含まれる分析相互のやり取りを通して活用の幅が広がる例である。

College Analysis のメニュー［分析－数学－行列計算］を選択すると図 1 のような実行画面が表示される。

図 1　実行画面

行列データは、グリッドエディタに例えば「a3=」のように、行列名を付けて入力する。行列名は必ず後ろに半角の「=」を付け、行列の成分は行列名の右から下にかけて連続的に入力する。空欄が行列の区切りである。図 2 にデータ入力の例を示す。

a3=	4	1	2		i3=	1	0	0	
	1	2	1			0	1	0	
	2	1	3			0	0	1	
a2=	4	2			i2=	1	0		
	2	3				0	1		
b3=	1				x3=	x		x2=	x
	2					y			y
	3					z			
b2=	3								
	2			d=	1.0000	0.7394	0.9065	0.4803	0.9141
					0.7394	1.0000	0.7025	0.6007	0.6995
c=	1	3	2		0.9065	0.7025	1.0000	0.5345	0.9044
	2	2	-1		0.4803	0.6007	0.5345	1.0000	0.5323
	-1	2	1		0.9141	0.6995	0.9044	0.5323	1.0000

図 2　データ入力画面

ここには、a3, a2, b3, b2, c, d, i3, i2, x3, x2 という名前の 10 個の行列が含まれている。行列の成分には単に数字だけでなく、数式や文字列も利用可能である。行列名は小文字と大文字を区別せず、すべて大文字に変換して処理される。但し、このプログラムは複素行列に対応していない。複素行列については将来このプログラム内で対応させるか、独立に扱うか検討中である。

利用者は「行列数式」のテキストボックスに、データで入力した行列名を含む数式を記述する。例えば「a3*(c-i3)」として実行ボタンをクリックすると、図 3 のような結果が表示される。

	1 列	2 列	3 列
1 行	0	17	7
2 行	3	7	0
3 行	-1	13	3

図 3　「a3*(c-i3)」計算結果

計算式の演算では、実数同士の加減乗除、行列同士の加減乗算、行列とスカラーの乗除（スカラーで割る）、が可能である。また関数として扱われるのは、実数の College Analysis で標準的に利用できる関数、行列の転置（mt()）、トレース（mtrace()）、行列式（mdet()）、逆行列（minv()）、固有値対角行列（mival()）、固有ベクトル（mivec()）である。但し、後者 2 つについては対称行列のみ対応するようになっている。これについては今後改良を加えて行く必要がある。また、固有値と固有ベクトルの関数名について、本来は meival() と meivec() とすべきであろうが、作者のミスで上のような形になってしまった。訂正も可能であるが、文系学生の読みやすさを優先してそのままにしている。行列の関数で行列式はスカラーとなるが、これも図 3 と同様に、1 行 1 列の行列として結果が表示される。行列計算は基本的に行列もスカラーもすべて行列として処理している。これらの計算は図 1 の中の「数値処理」ラジオボタンで示されるように、数値として計算される。

このプログラムを利用すると、通常の逆行列や固有値、固有ベクトルを求める計算だけでなく、教科書などでよく例が表示される、行列と逆行列の積が単位行列であることや固有ベクトルによる実対称行列の対角化などを手軽に見ることができる。例えば前者は、テキスト

入力で「c*minv(c)」として「実行」ボタンをクリックすると、結果は図 4 のように単位行列となる。

	1 列	2 列	3 列
1 行	1	0	0
2 行	0	1	0
3 行	0	0	1

図 4　行列と逆行列の積

後者では、「mt(mivec(a3))*a3*mivec(a3)」とすると、対角成分に固有値が並んだ、図 5 のような対角行列を得る。

	1 列	2 列	3 列
1 行	6.0489	0	0
2 行	0	1.6431	0
3 行	0	0	1.308

図 5　固有ベクトルによる実対称行列の対角化

ここで、もう少し複雑な計算をするときの、新しく追加された仮の定義名について述べておく。「行列数式」の中に以下のような記述をすると図 5 の結果を得る。

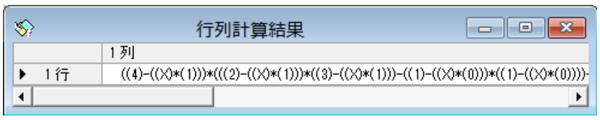
define <d>=mivec(a3)

mt(<d>)*a3*<d>

<d>のように、「<>」で挟んで定義名を指定すると、計算の際に置換えが行われる。また、先頭文字を「#」にするとその行はコメント行になる。

このような数値計算だけでなく、行数と列数が小さな行列では図 1 の「文字列処理」ラジオボタンによって、行列の計算を文字列で実行することもできる。結果は数式処理ソフトのようにきれいではないが、これを利用すると、実際に固有方程式や 1 次連立方程式を作り、方程式ソルバーでそれを解いて固有値を求めたり、2 次曲線の式を行列表示して陰関数グラフの表示プログラムでグラフ化したりすることができる。

図 1 で「文字列処理」ラジオボタンを選択し、テキスト入力で「mdet(a3-x*i3)」とすると、図 6 のような結果が得られる。



	1 列
1 行	((4)-((X)*(1)))*(((2)-((X)*(1)))*((3)-((X)*(1)))-((1)-((X)*(0)))*((1)-((X)*(0))))-

図 6　行列式の文字列計算

数式処理が可能ならばもっと簡単な表式になるのであるが、ここでは区切りのための括弧が多く付いた表式が与えられている。この行列式を 0 とおいて固有値が求められるが、これにはメニュー［分析－数学－方程式ソルバー］を選択して得られる方程式ソルバーメニューを利用する。これにこの表式をコピーして、「方程式の解」ボタンをクリックすると、表式=0 で求められる解が図 7 のように求められる。

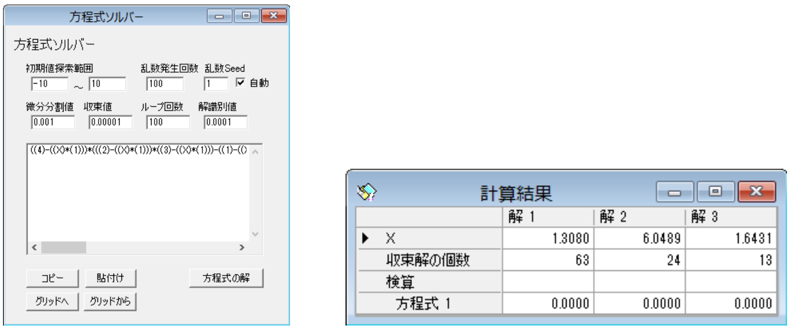


図 7　固有方程式の解

次に、テキスト入力で「c*x3-b3」として「実行」ボタンをクリックすると、図 8 のような結果が得られる。これは連立方程式「c*x3-b3=0」の左辺である。

図 8　連立方程式の文字列計算

この表式を方程式ソルバーにコピーし、結果を求めたものが図 9 である。

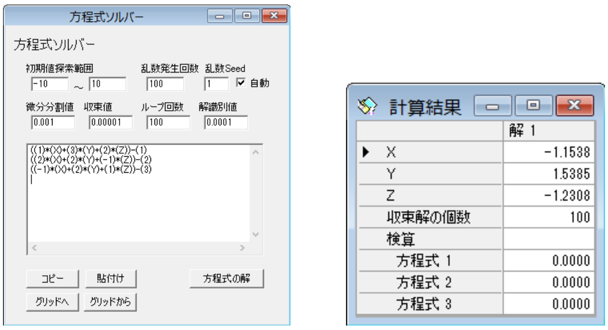


図 9　連立方程式の解

最後に 2 次曲線の行列表示について見てみよう。テキスト入力で「mt(x2)*a2*x2-9」として「実行」ボタンをクリックすると、図 10 のような結果が得られる。表式=0 とすると、これは 2 次曲線の方程式である。

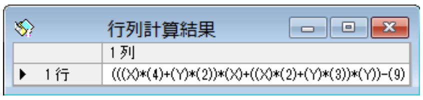


図 10　2 次曲線方程式の文字列計算

これを、[分析－数学－2 次元パラメータ表示関数] のメニューの「陰関数描画」のテキストボックスに代入し、「描画」ボタンをクリックすると、図 11 のような結果を得る。

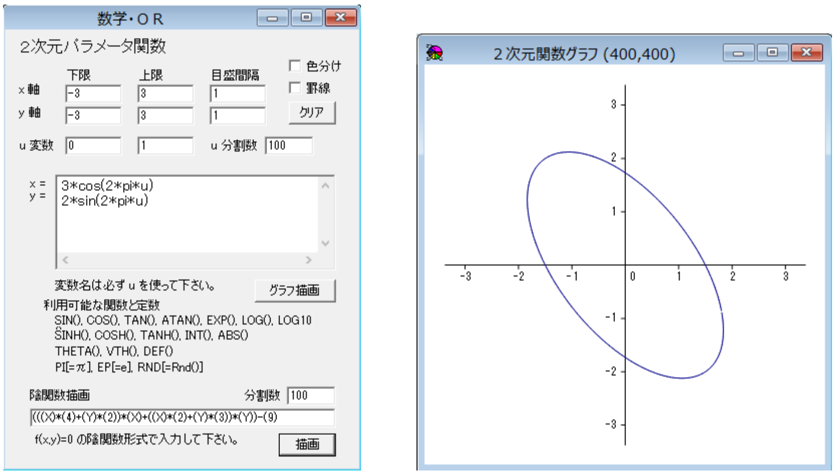


図 11　２次曲線の描画

ここでは２次元のグラフを描いたが、これには３次元のグラフを描くものもある。

これまで分析プログラムの内部で利用されていた行列計算のプログラムを表に出し、行列計算が数式で行えるようにした。また、行列計算を文字列で行い、その結果を用いて方程式を解いたり、グラフを描いたりできるようにした。このプログラムは、数学や統計の授業で役に立つと考える。数学の教科書の演習問題の答え合わせに、公式のチェックに利用することが考えられる。また統計の多変数解析では行列計算が多用される。これらの計算を追っていく際、数式によって計算を進めて行くこのプログラムは非常に有効である。また、文字列計算も利用法によって用途が拡がるものと考えられる。

行列計算の問題点としては、複素数の行列の計算に制限があることである。例えば非対称行列の固有値は一般に複素数となり、今後これらの解を求める方法も追加して行かなければならない。

問題１

行列 $\mathbf{A}, \mathbf{B}, \mathbf{C}$ が以下のように与えられているとき、次の結果を求めよ。

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 3 & 5 \end{pmatrix},\quad \mathbf{B} = \begin{pmatrix} 3 & 0 \\ -2 & 1 \end{pmatrix},\quad \mathbf{C} = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 1 & -2 \\ 2 & -2 & 3 \end{pmatrix}$$

１） ${}^{t}\mathbf{A}$

２） $\mathbf{A} + 2\mathbf{B}$

３） $\mathbf{BA}$

４） $\mathrm{tr}\,\mathbf{C}$

５） $|\mathbf{C}|$

６） $\mathbf{C}^{-1}$

７） $\mathbf{C}$ の固有値

８） $\mathbf{C}$ の固有ベクトル（固有値大きい順）

問題２

以下の連立方程式を行列表示し、その解を求めよ。

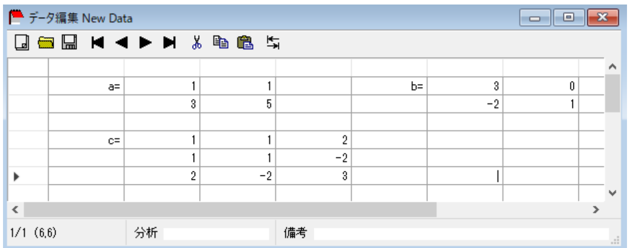
１） $$\begin{aligned} x_1 + 2x_2 - x_3 &= 1 \\ x_1 + x_2 + 2x_3 &= 2 \\ x_1 - x_2 &= 1 \end{aligned} \qquad \begin{pmatrix} \quad & \quad & \quad \\ & & \\ & & \end{pmatrix}\begin{pmatrix} \quad \\ \\ \end{pmatrix} = \begin{pmatrix} \quad \\ \\ \end{pmatrix}$$

$$\mathbf{Ax} = \mathbf{b}$$

２）解 $$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \quad \\ \\ \end{pmatrix} \qquad \mathbf{x} = \mathbf{A}^{-1}\mathbf{b} =$$

問題１解答

以下のように入力する。


データ編集 New Data

a=	1	1		b=	3	0
	3	5			-2	1
c=	1	1	2			
	1	1	-2			
	2	-2	3			

1/1 (6,6) 分析 備考

手計算でも簡単だが、プログラムを使った結果を示す。

１） ${}^t\mathbf{A}$　mt(a)


行列計算結果

	1 列	2 列
1 行	1.0000	3.0000
2 行	1.0000	5.0000

２） $\mathbf{A}+2\mathbf{B}$　a+2*b


行列計算結果

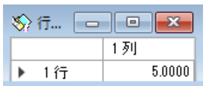
	1 列	2 列
1 行	7.0000	1.0000
2 行	-1.0000	7.0000

３） $\mathbf{BA}$　b*a


行列計算結果

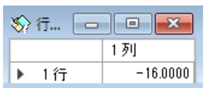
	1 列	2 列
1 行	3.0000	3.0000
2 行	1.0000	3.0000

４） $\mathrm{tr}\,\mathbf{C}$　mtrace(c)


行...

	1 列
1 行	5.0000

５） $|\mathbf{C}|$　mdet(c)


行...

	1 列
1 行	-16.0000

６） $\mathbf{C}^{-1}$　minv(c)　分数表示も可


行列計算結果

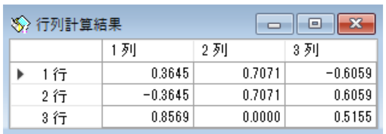
	1 列	2 列	3 列
1 行	0.0625	0.4375	0.2500
2 行	0.4375	0.0625	-0.2500
3 行	0.2500	-0.2500	0.0000

７） $\mathbf{C}$ の固有値　mival(c)


行列計算結果

	1 列	2 列	3 列
1 行	4.7016	0.0000	0.0000
2 行	0.0000	2.0000	0.0000
3 行	0.0000	0.0000	-1.7016

８） $\mathbf{C}$ の固有ベクトル　mivec(c)


行列計算結果

	1 列	2 列	3 列
1 行	0.3645	0.7071	-0.6059
2 行	-0.3645	0.7071	0.6059
3 行	0.8569	0.0000	0.5155

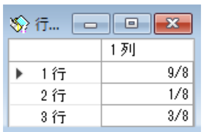
問題２解答

以下の連立方程式を行列表示し、その解を求めよ。

1） $\begin{aligned} x_1+2x_2-x_3&=1\\ x_1+x_2+2x_3&=2\\ x_1-x_2&=1\end{aligned}$ $\qquad \begin{pmatrix}1&2&-1\\1&1&2\\1&-1&0\end{pmatrix}\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}=\begin{pmatrix}1\\2\\1\end{pmatrix}$

以下のように入力する。

2）解 $\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}=\begin{pmatrix}1.1250\\0.1250\\0.3750\end{pmatrix}$ minv(a)*b

or


補足　特異値分解について

実行列 $\mathbf{X}(m\times n)$ （$m\le n$、複素行列でも可）について、

$$\mathbf{XX'u}=\lambda\mathbf{u}(m\times 1)$$

ここに、$\mathbf{u'XX'u}=(\mathbf{X'u})'(\mathbf{X'u})=\lambda\mathbf{u'u}(m\times 1)$ であるから、$\lambda\ge 0$

また、$\mathbf{v}=c\mathbf{X'u}$ とおくと

$$\mathbf{X'Xv}=c\mathbf{X'XX'u}=c\lambda\mathbf{X'u}=\lambda c\mathbf{X'u}=\lambda\mathbf{v}(n\times 1)$$

より、$\mathbf{XX'}$ と $\mathbf{X'X}$ の固有値は 0 を除いて一致する。

ここで、$c=1/\sqrt{\lambda}$ とすると、

$$\mathbf{X'u}=\sqrt{\lambda}\mathbf{v},\quad \mathbf{Xv}=\sqrt{\lambda}\mathbf{u}$$

の関係も得る。

これらより、

$$\mathbf{XX'}=\mathbf{UWU'}(m\times m),\quad \mathbf{X'X}=\mathbf{V\tilde{W}V'}(n\times n)$$

但し、

$$\mathbf{W}=\begin{pmatrix}\lambda_1&\cdots&0\\ \vdots&\ddots&\vdots\\ 0&\cdots&\lambda_m\end{pmatrix},\quad \tilde{\mathbf{W}}(n\times n)=\begin{pmatrix}\mathbf{W}(m\times m)&\cdots&0\\ \vdots&\ddots&\vdots\\ 0&\cdots&0\end{pmatrix}$$

また、

$$\mathbf{X'U}=\mathbf{X'}\begin{pmatrix}\mathbf{u}_1&\cdots&\mathbf{u}_m\end{pmatrix}=\begin{pmatrix}\sqrt{\lambda_1}\mathbf{v}_1&\cdots&\sqrt{\lambda_m}\mathbf{v}_m\end{pmatrix}=\mathbf{V}_m\sqrt{\mathbf{W}},\quad \mathbf{U'U}=\mathbf{UU'}=\mathbf{I}_m$$

の関係から、

$$\mathbf{X'}=\mathbf{V}_m\sqrt{\mathbf{W}}\mathbf{U'}$$

これより、

$$\mathbf{X} = \mathbf{U}\sqrt{\mathbf{W}}\mathbf{V}'_m = \left(\mathbf{U}\sqrt[4]{\mathbf{W}}\right)\left(\sqrt[4]{\mathbf{W}}\mathbf{V}'_m\right) \qquad \text{for} \quad m \le n$$

同様にして、

$$\mathbf{X} = \mathbf{U}_n\sqrt{\mathbf{W}}\mathbf{V}' = \left(\mathbf{U}_n\sqrt[4]{\mathbf{W}}\right)\left(\sqrt[4]{\mathbf{W}}\mathbf{V}'\right) \qquad \text{for} \quad m \ge n$$

この関係を用いて、データ圧縮を行う。主要な固有値 p 個と固有ベクトル p 個ずつを利用する。

$$\mathbf{X} \simeq \mathbf{U}_p\sqrt{\mathbf{W}_p}\mathbf{V}'_p$$

１３．不等式グラフ

これまで我々は等式に関するグラフを描くプログラムを多く作成してきたが、ここでは不等式について考えてみたい。3 次元における不等式の領域は表現しにくいことから、今回のプログラムでは 2 次元についてのみ考える。不等式は基本的に陰関数の領域として処理し、不等式の論理式も考えられるようにする。また、不等式の領域として円等を考えることにより、ベン図なども描けるようにする。できるだけ汎用的に使えるツールとして考えて行きたい。

メニュー［分析－数学－グラフ－不等式グラフ］を選択すると図 1 のような不等式グラフの実行メニューが表示される。

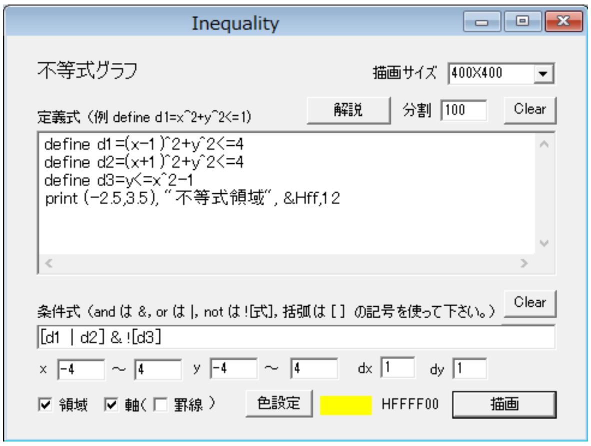


図 1　不等式グラフ実行メニュー

メニューは、大きく、定義式の部分と条件式の部分に分かれ、定義式の部分では 1 つずつの不等式が定義され、表示領域上に表示される文字列が設定される。条件式の部分では、複数の不等式の論理演算式が指定される。ここで指定された「真」の領域は「色設定」ボタンで選択される色で塗られる。以後、具体的に例を見ながら説明する。

図 1 に示される設定で、「描画」ボタンをクリックすると、図 2 に与えられるグラフが表示される。

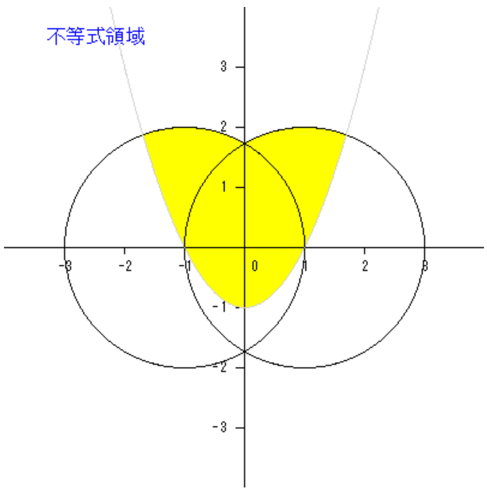


図 2　不等式領域グラフ

定義式の部分の以下の形は、その変数名で右辺の不等式（通常の式も可）を利用することを意味する。

define 変数名=不等式（含等式）

これは幾何アニメーションの定義式と同じである。

条件式の部分では、定義式のところで定義した変数名と通常の数式を使って論理式を記述する。上の例では、d1 と d2 の和集合と d3 の補集合の積集合が「真」となる領域である。この領域が指定された色で、集合の境界と共に表示される。集合の境界は、その集合に含まれる場合は黒、含まれない場合はグレーで表示される。領域は 1 画素毎に真偽を判定し、境界は陰関数の描画法である等高線を表示するアルゴリズムを使って描画しているので、多少の粗さは残る。また、描画にある程度の時間を必要とするので、通常のグラフ描画のように、Window の枠の大きさの変化には対応していない。

定義式の部分の書式をまとめておく。

不等式の定義

define d1=(x-1)^2+y^2<=4

define ex1=x^2+y^2

define 変数名=式（等式または不等式、定義式）で指定する。

文字列の描画

print (x,y), ″文字列″[, 色番号, サイズ]

(x,y) の位置に文字列を描画する。色番号は&Hff00 のような RGB 形式，サイズはポイント数である。

条件式の部分の書式は以下である。

条件式

[d1 | d2] & ![x^2+y^2<=5]

定義式で定義された変数名か、式を直接使う。

論理積は &、論理和は |、式の括弧は[ ] を用い、否定は ![論理式] である。

境界を含む場合は黒、含まない場合はグレーで境界を表示する。

描画される画面サイズは 400×400 から 900×900 まで、100 刻みで選択できるが、ドットごとに真偽を判定するので、画面サイズが大きいほどサイズの 2 乗に比例して計算時間がかかる。

画面の下の領域は描画範囲を表し、dx と dy は軸目盛の幅を表す。「軸」チェックボックスは座標軸の表示・非表示を決め、罫線チェックボックスは、座標軸が表示された場合に罫線を描くか否かを決める。図 3a に座標軸と罫線を表示した場合のグラフ、図 3b に領域を表示しない場合のグラフを示す。後者はベン図などの演習問題の作成に適している。

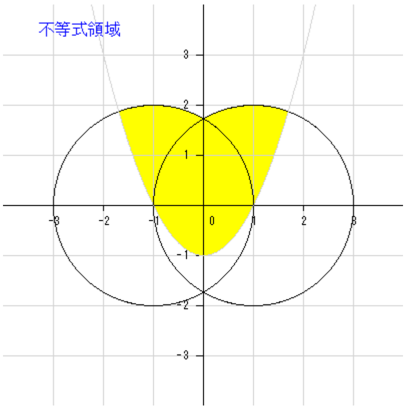


図 3a　座標軸と罫線表示

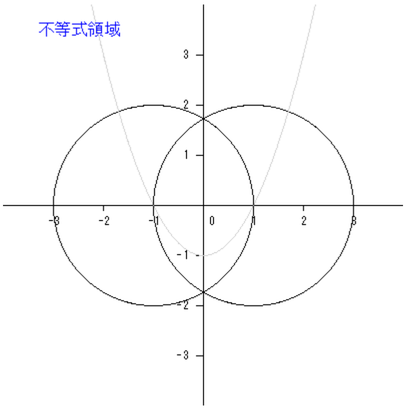


図 3b　領域非表示

１４． ２次元陰関数グラフ

2 次元陰関数グラフは、2 次元パラメータ表示関数のメニューの下に入れていたが、単独でメニューを作って機能追加を行った。メニュー［分析－数学－グラフ－2 次元陰関数グラフ］を選択すると、図 1 のような実行画面が表示される。

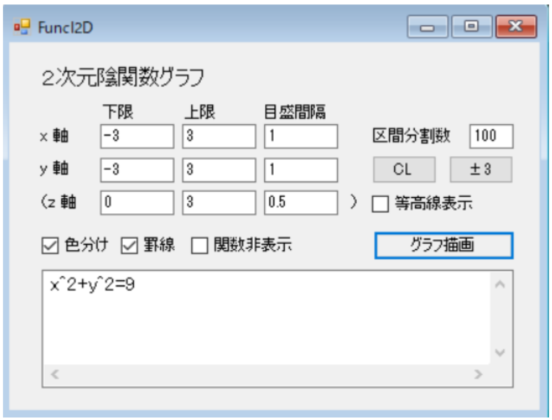


図 1　2 次元陰関数グラフ実行画面

今回のプログラムの改良点は以下の通りである。

1. 複数の関数に対応している。
2. 通常の 1 次元関数と同様に、「色分け」、「罫線」、「関数非表示」機能がある。
3. z 軸を含む 3 次元陰関数を記入した場合、「等高線表示」で、指定区間、指定間隔の等高線を描く。

複数の関数を描く問題で、図 2 のようなことが可能となった。それぞれの数式に対応するグラフを下に示している。

x^2+y^2=4　　　　　　　　x^2+y^2=4+def(x^2-1-y)

y=x^2-1　　　　　　　　　y=x^2-1

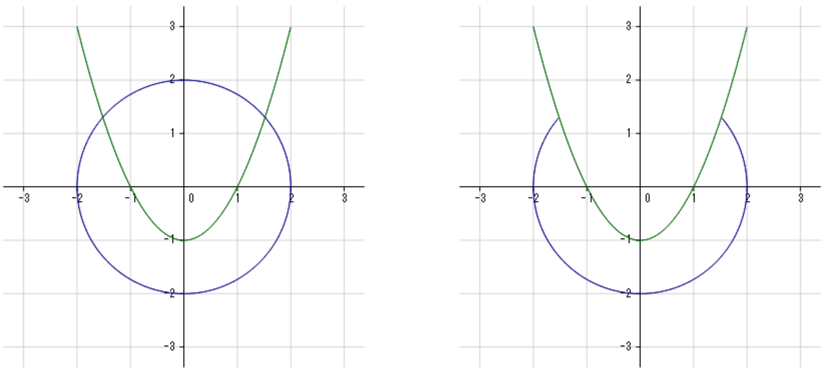


図 2　範囲を指定した表示

左は通常の表示であるが、右は円の範囲を 2 次曲線の下に設定している。これは関数 def(z) が z≧0 で 0、z<0 でエラーとなるような関数だからである。表示の式の中にエラーが含まれる場合、描画は実行されない。

次に等高線表示を見てみよう。z = 0～3 まで 0.5 間隔で 2 つの関数を表示してみる。

z=x*y　　　　　　　　　　　z=sin(x)+cos(y)

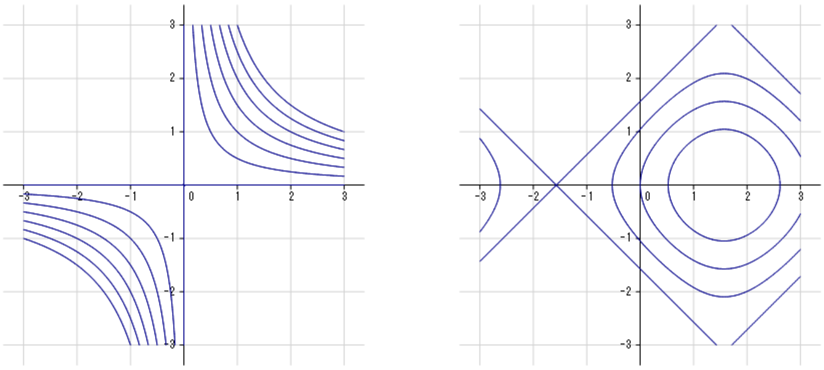


図３　等高線表示

この２つの等高線は同時に示すこともできる。

次に面白い例を図 4 に示しておこう。以下の関数は以前 3 次元陰関数の例として描いたものである。この関数の z = -3～3 まで 0.5 間隔で描いた等高線が左の図である。右はその立体図形である。

x*y*z-sin(x^2+y^2+z^2)=0

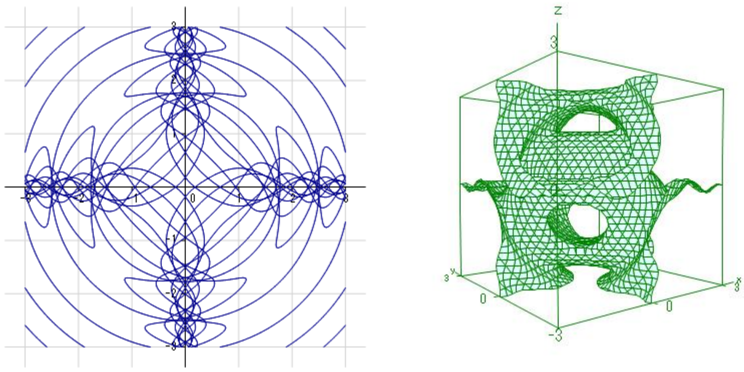


図４　命名カエル曲面

ある人が右の図を Google Lens で検索したところワイヤーフレームのカエルが出て来たそうである。なぜこれがカエルに見えるのか。右の図の z=1 の等高線を調べたところ、図 5 のようなカエル足が出現した。なかなか奥深いものである。

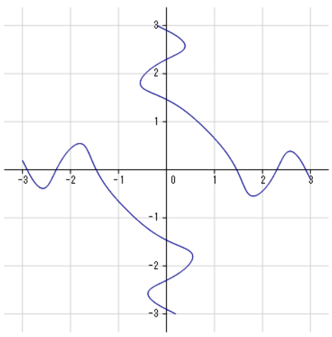


図５　カエル足

１５．３次元陰関数グラフ

3 次元陰関数グラフは、2 次元と同様、3 次元パラメータ表示関数のメニューの下に入れていた。しかし複数のグラフの同時表示や細かな設定ができなかったため、単独でメニューを作り、機能追加を行った。メニュー［分析－数学－グラフ－3 次元陰関数グラフ］を選択すると、図 1 のような実行画面が表示される。

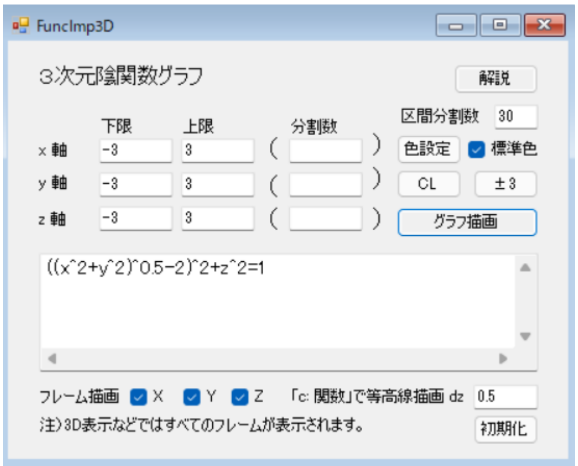


図 1　3 次元陰関数グラフ実行画面

前章のカエル曲面はこのメニューでそのまま「グラフ描画」ボタンをクリックして描いたものである。

3 次元曲面については、これまでフレーム線の描画に不満を感じていた。x, y, z 軸すべての方向に描画するとフレーム線が多すぎるように感じてしまう。理想的には画像の精度はそのままで、フレーム線の本数を減らして表示したい。また、フレーム線の方向を指定できるようにもしたい。但し、C.Analysis のコンセプトの「初心者にも簡単」を守って設定を複雑にしないことが制約である。これらの条件の下で機能追加した今回のプログラムの改良点は以下の通りである。

1. 複数の関数に対応している。
2. 描画領域を関数として指定できる。
3. 見た目階段のような関数も表示できる（多少の勾配はあるが）。
4. 図形に表示されるフレームを軸単位で設定できる。
5. 図形に表示されるフレームの間隔を設定できる。
6. 関数の先頭に「c:」(contour)を付けることにより、指定区間、指定間隔の等高線を描画領域最下部に描く。
7. グラフの色指定を通常のグラフと同様に色パネルによって指定できる。

まず、改良点 1.と 2.であるが、複数のグラフの表示は図 2 の左側、右側は中央の球に上半分（z≧0）の制限を加えた。

x^2+y^2+z^2=1　　　　　　　x^2+y^2+z^2=1+def(z)

x^2-y^2-z^2=1　　　　　　　x^2-y^2-z^2=1

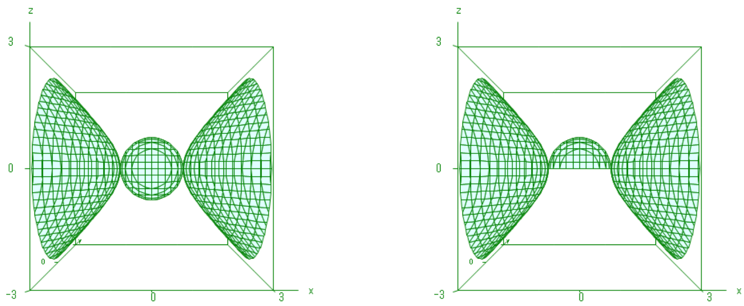


図 2　陰関数の複数表示と描画領域の指定

右のグラフで球の下側が表示されていないが、これには描画用の数式の中の +def(z) の部分が重要である。計算式の中で、0 で割る、負の値の平方根など、定義されていなかったり、実数の値を取らなかったりする場合は、特別な数値 G_DATAERROR（実際は負の非常に大きな値）が返される。多くのグラフではその値が出るとその部分は表示をしないように設定しているが、今回これを徹底して、陰関数グラフにも適用するようにした。しかし、グラフの「区間分割数」の値がデフォルトの 30 程度だと、def(z-x-y) などとした場合、図 3 左のように切り口が荒くなってしまう。このような場合、せめて図 3 右の区間分割数 50 程度にしたいが、それでもまだきれいとはいえない。切り口をきれいに見せる方法については考察中である。

x^2+y^2+z^2=9+def(z-x-y)

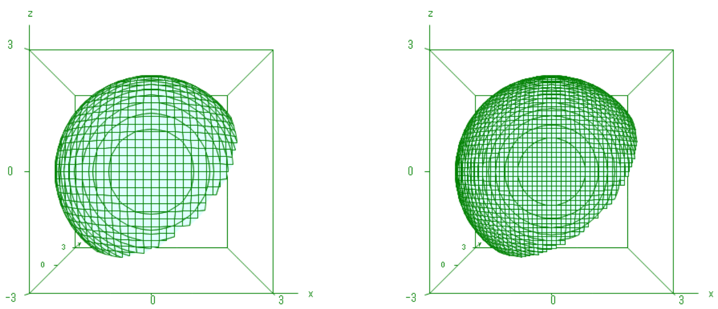


図 3　関数による領域指定（区間分割数 30 と 50）

3 番目の段差のある関数について、改めてプログラムを書き加えたわけではないが、例えば、z=x*y の関数の後ろに theta(1-z)を掛けると、図 4 のように階段状に表示されることが分かった。ここで theta(x)は、x≧0 で 1、x<0 で 0 となる関数である。これは描画のアルゴリズムを見ると納得できるものであるが、詳細は省略する。ただ、この場合、段差の上が

完全に水平ではない、区間分割の幅だけの傾きを持っていることを注意しておく。

z=x*y*theta(1-z)

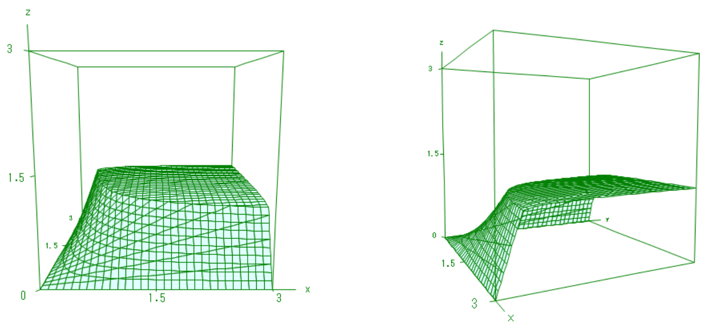


図 4　段差の簡易的な表示法

この水平面を厳密に描く方法もある。それは 2 つの領域を描く関数を合わせることである。

z=x*y+def(1-z)

z=1+def(x*y-1)

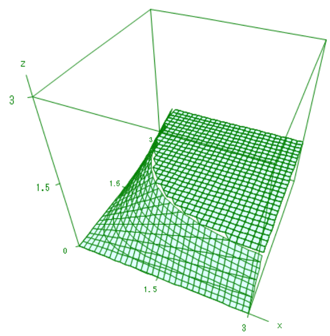


図 5　2 つの関数の合成

しかし、この表示法では図 5 のように、接合部分に隙間が生じる。

この簡易法を用いると図 6 のように複数の段差を積み上げることも可能である。

z=x*y*theta(1-z)+(x*y-2)*theta(z-1.01)

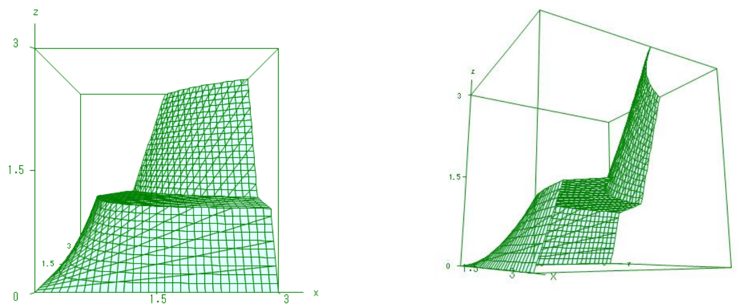


図 6　2 段目の積み上げ

但し、この場合微妙な誤差が生じ、theta()関数の領域が数学のように厳密に規定できない。例えば、2 番目の theta()関数を theta(z-1)にすると図に少しずれが生じる。

4 番目のフレームの軸単位描写の機能は、実行画面のフレーム描画の x, y, z チェックボックスにチェックを入れて描画することで可能となる。図 7 の左側はすべてのフレームにチェックを入れたグラフ、右側は z 軸に垂直なフレームだけを表示したグラフである。

z=x*y

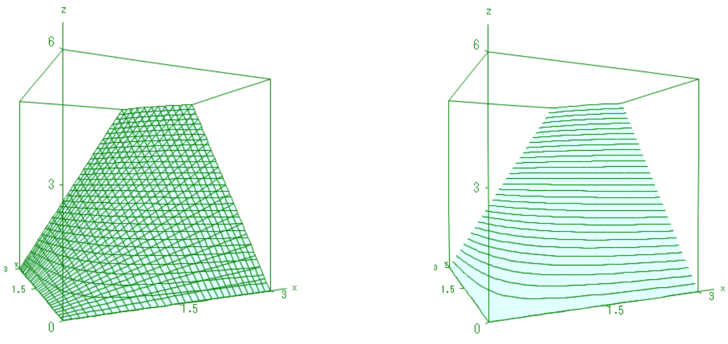


図 7　フレーム描画

5 番目のフレームの間隔は、図 1 のメニューではなく、図 8 のように、3D 表示用の 3D ビューアのメニューに加えた。

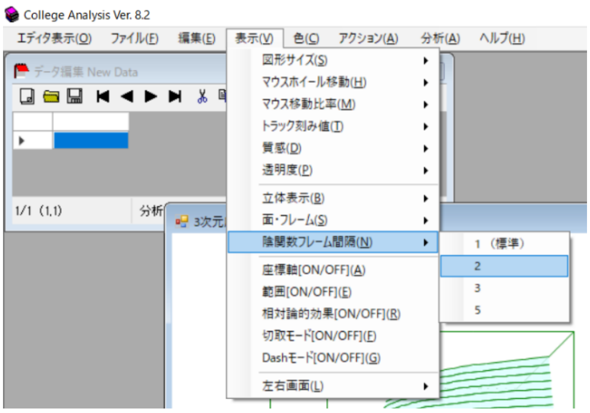


図 8　フレーム間隔設定用メニュー（3D ビューア）

このメニューを使って図 7 のグラフのフレームの間隔を 2 にしたグラフが図 9 である。

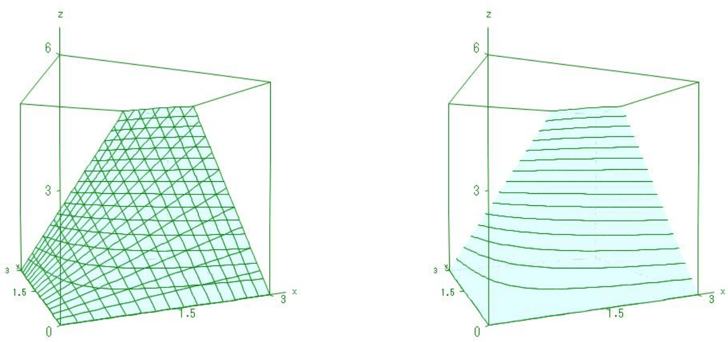


図 9　フレーム間隔を 2 にしたグラフ

試してみて感じることであるが、あまりフレーム間隔を空けすぎると、グラフ自体が雑なもののように見えてくるので注意が必要である。

テキストなどで曲面の下に等高線を表示したものがたまに見られる。これに対応させたものが 6 番目の等高線描画機能である。これはグラフの先頭に「c:」を付けるだけである。上のグラフで例を見てみよう。

c:z=x*y

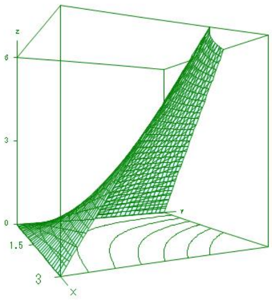


図 10　等高線描画

等高線は z 軸の下限から、等高線描画の「dz」の値の間隔で引かれたものである。

最後に、改良点の 7 番、グラフの色付けについて見てみよう。これまでの図は、実行画面右上の「標準色」チェックボックスにチェックを入れた描画結果である。標準色チェックボックスからチェックを外すと、グラフは指定された色で描画される。デフォルトは棒グラフなどで使われる青色である。これは色が強すぎるので、他の色を選択することになるが、実行画面の「色設定」ボタンで色のパターンと使う色を選択する。図 11 が色設定ボタンをクリックした場合に表示される画面である。

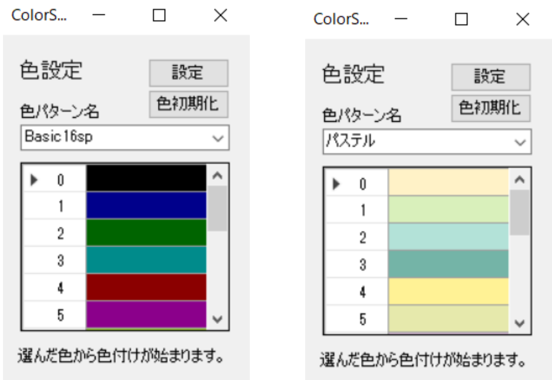


図 11　色設定画面

色パターンを選択し、色をクリックすると、その色から順番に使用される。等高線を付けて、図 11 の右側一番上の色を選択した場合の 2 つのグラフ描画の結果を図 12 に示す。

c:z=x*y

(x-0.5)^2+(y-0.5)^2+(z-2)^2=0.25

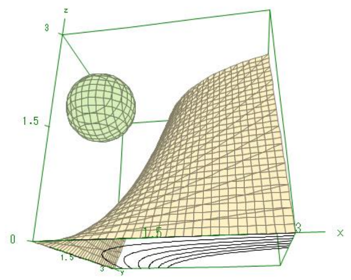


図 12　各種設定をした場合の描画

これまで陰関数に対しての設定で､表示されるフレームの向きと表示間隔の設定が可能になったという話をしたが、これは全体に対する設定で、個々の関数に対する設定ではない。また色についても､色パターンの設定は可能になったが､完全に好きな色を決められるわけではない。これらはこのプログラムの「初心者にやさしく」という方向性にそって改良が可能になったとき、また変更を加えることにする。

１６．和・積計算

　数列の計算では和と積の記号、ΣとΠがよく表れる。これらは巧妙な計算によって厳密な値が計算できることも多いが、利用者にとって予め概数を知っておくことも必要である。また収束の問題では、収束の速さを具体的に見ることも重要である。そこで我々は、これらの数値計算を簡単に行うプログラムを作成することにした。対象は和と積であるが、よく見かける素数を使った計算にも使えるようにした。図 1 に分析実行画面を示す。

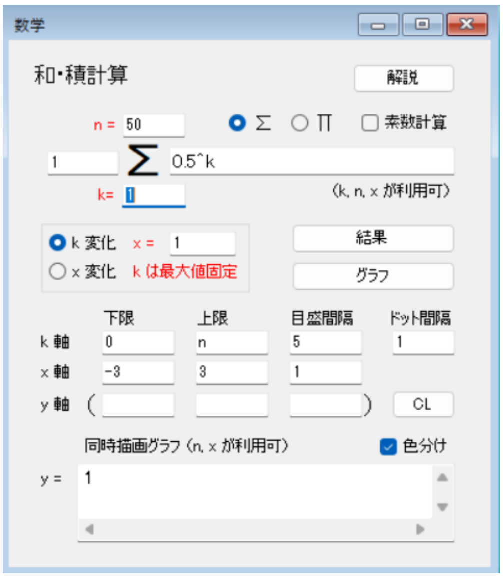


図 1　分析実行画面

デフォルトの設定は、以下の等比数列の和である。

$$\sum_{k=1}^{50} 0.5^k$$

数式の入力は見た目どおりである。和か積かは上のラジオボタンで選択する。上の計算結果は、「計算」ボタンをクリックすると収束の様子が図 2 のように表される。

k	関数1	和
1	1.000000	0.500000
2	1.000000	0.750000
3	1.000000	0.875000
4	1.000000	0.937500
5	1.000000	0.968750
6	1.000000	0.984375
7	1.000000	0.992188
8	1.000000	0.996094
9	1.000000	0.998047
10	1.000000	0.999023

図 2　計算結果

ある値に収束する場合は､同時描画グラフにその値を書いておけば関数値として表示される。計算結果の小数点以下桁数は後で変更可能である。

　この和の収束をグラフで見るには「グラフ」ボタンをクリックする。結果を図 3 に示す。

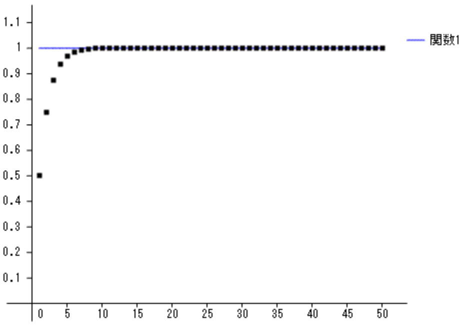


図 3　収束グラフ

ここで、描画の表示範囲は k 軸の「下限」、「上限」、「目盛間隔」で設定する。グラフはこの下限と上限の範囲内の変数 k までの和（積）で表示される。「同時描画グラフ」は複数表示可能であるが、基本的には収束値を表示することを想定している。

次に、テイラー展開などの変数に依存した以下の展開について見てみよう。

$$\sum_{k=0}^{\infty}\frac{(-1)^k x^{2k+1}}{(2k+1)!}=\sin x$$

左の「k 変化」のラジオボタンのところで、$x=1$ と固定して調べると、図 4 のように $k\le 5$ まででもかなり収束が速い。

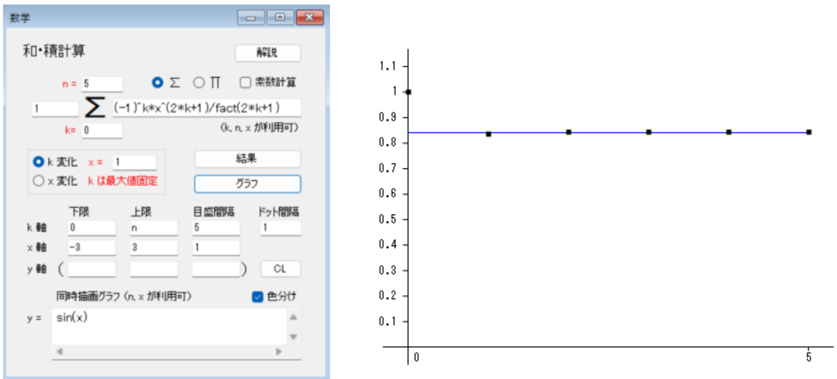


図 4　$k\le 5$ までの収束状況

しかし、「x 変化」のラジオボタンに設定し直し、x 軸の範囲を $-6\le x\le 6$ に設定すると、図 5 に見るように n=5 では、両端が大きく外れている。

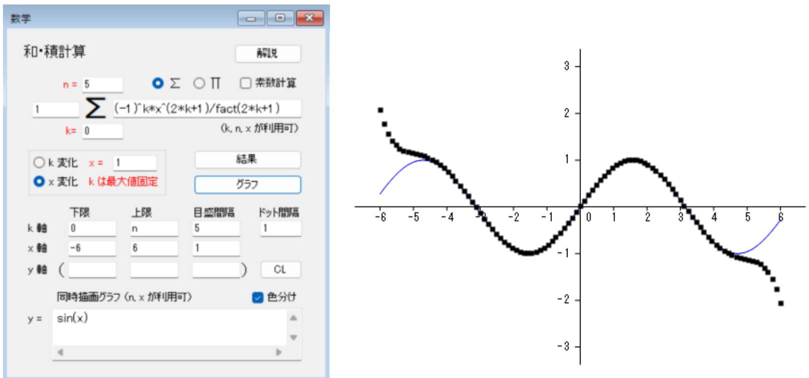


図 5　n=5 の場合の x 依存性

x に依存する級数の場合、x を変化させて収束を見る機能は重要である。

　次に積の公式としてウォリスの公式を示す。

$$\prod_{k=1}^{\infty}\frac{(2k)^2}{(2k-1)(2k+1)}=\frac{\pi}{2}$$

この公式で 50 までの結果を示す実行画面は図 6 である。集計方法は積に変えている。

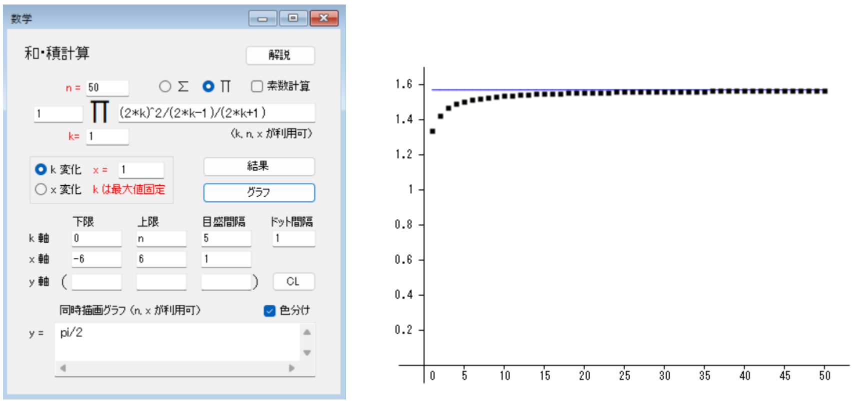


図 6　ウォリスの公式の収束

　以上のように使い方は非常に簡単であるが、今度はこのプログラムを作るきっかけになったオイラーの素数の公式について見てみよう。この公式は以下のように与えられる。

$$\frac{1}{1-1/2^2}\times\frac{1}{1-1/3^2}\times\frac{1}{1-1/5^2}\times\frac{1}{1-1/7^2}\times\frac{1}{1-1/11^2}\times\cdots=\prod_{k(prime)}^{\infty}\frac{1}{1-1/k^2}=\frac{\pi^2}{6}$$

左辺はすべての素数についての掛け算である。左辺の設定では、分析実行画面の例は図 7 のようにする。「素数計算」のチェックボックスに注意してもらいたい。実行結果は図 8 に示す。

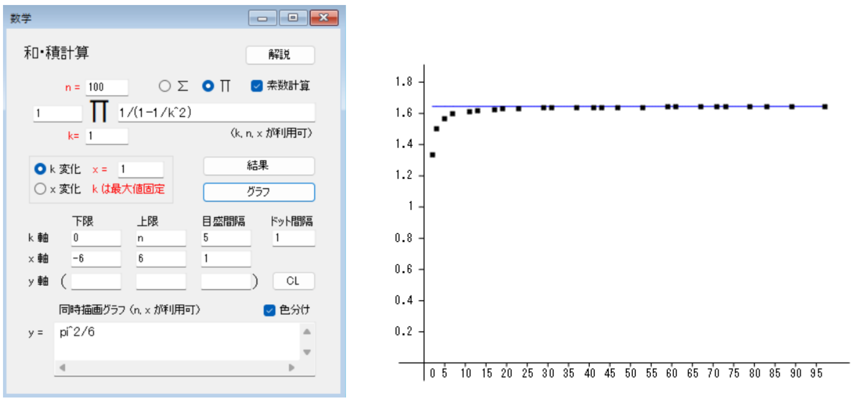


図 7　オイラーの素数公式

97 までの素数で計算すると、収束値 1.644934 に対して、1.641945 の結果を得る。

　素数計算には多少時間がかかると思われるが、この例は 100 までであるから、ほとんど時間がかからない。しかし 100,000 までの素数を使っても時間は 1 秒とかからない。結果は 1.644934 に対して 1.644933 まで近づけることができる。1,000,000 になると 5 秒程度は必要であった。もちろんこれはパソコンの性能にもよるが、一応実用の範囲内と考えたい。

　最後にラマヌジャン型の円周率公式をやってみよう。式は以下の通りである。

$$\frac{\sqrt{8}}{99^2}\sum_{k=0}^{\infty}\frac{(4k)!}{(4^k k!)^4}\frac{1103+26390k}{99^{4k}}=\frac{1}{\pi}$$

非常に大きな数が含まれるので、5 回までの収束結果を図 8 に示す。

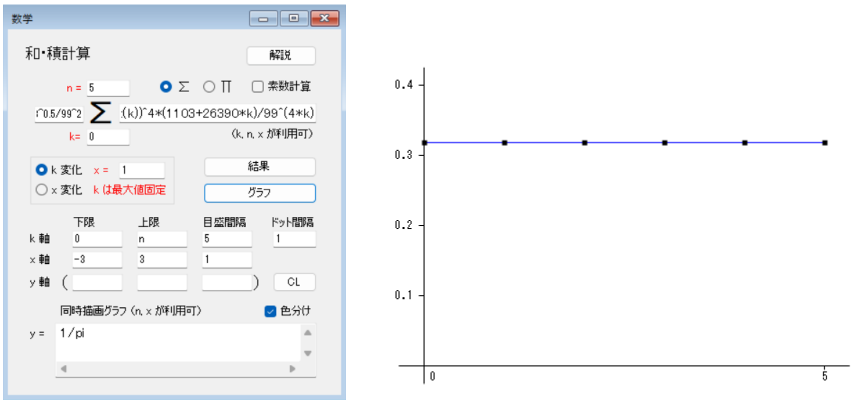


図 8　ラマヌジャン型の円周率公式の収束

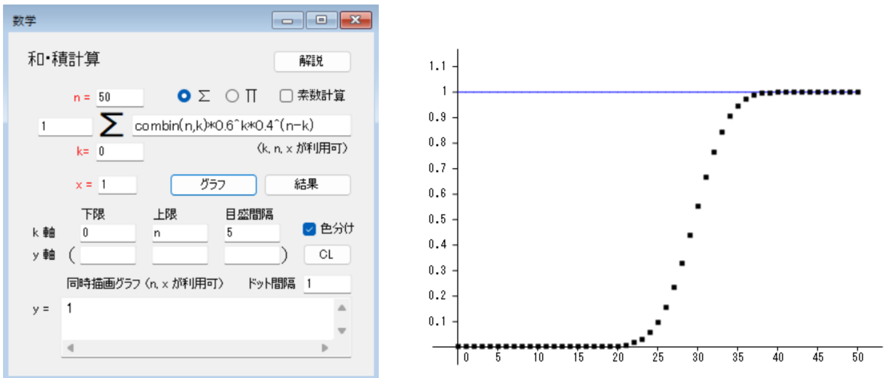
実はこの結果を見て著者はプログラムのエラーを疑った。しかしこの公式は第 1 項で 7 桁まで合うことが分かり、グラフにすること自体に問題があることを理解すると同時に、自分が応用分野の人間であることを実感した。

次ページから数列の和や積を用いた例をいくつか取り扱う。このような数列の例は参考文献 [1]に詳しい。

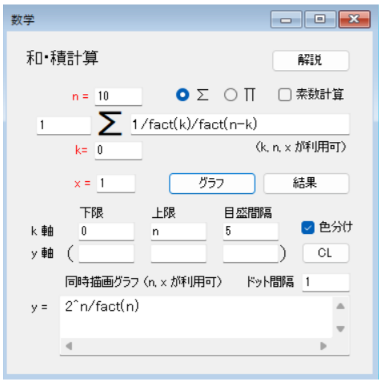
プログラムを使った例

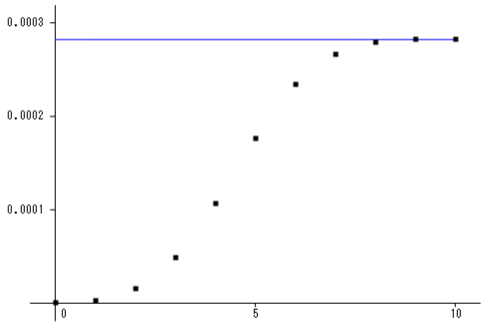
2 項定理（2 項分布）

$$\sum_{k=0}^{n} {}_nC_k a^k b^{n-k}=(a+b)^n \qquad \text{特に } a+b=1 \text{ のとき、} \sum_{k=0}^{n} {}_nC_k p^k (1-p)^{n-k}=1$$

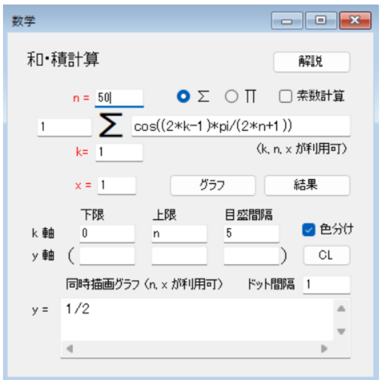


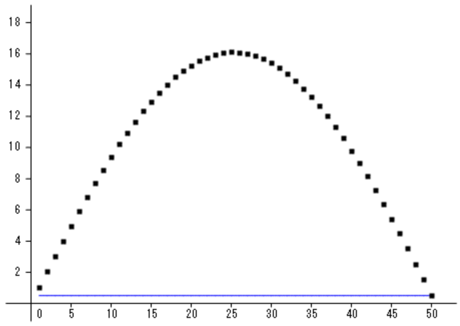
$$\sum_{k=0}^{n}\frac{1}{k!(n-k)!}=\frac{2^n}{n!}$$



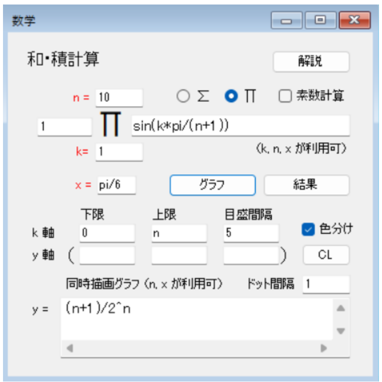


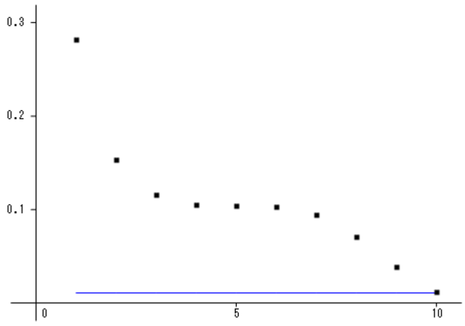
$$\sum_{k=1}^{n}\cos\left(\frac{(2k-1)\pi}{2n+1}\right)=\frac{1}{2}$$





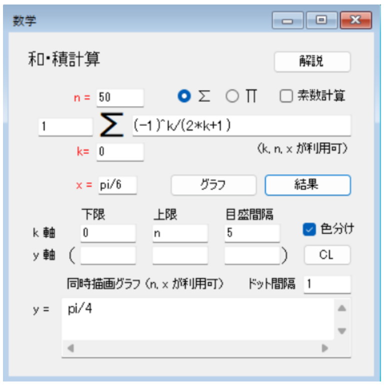
$$\prod_{k=1}^{n}\sin\frac{k\pi}{n+1}=\frac{n+1}{2^n}$$

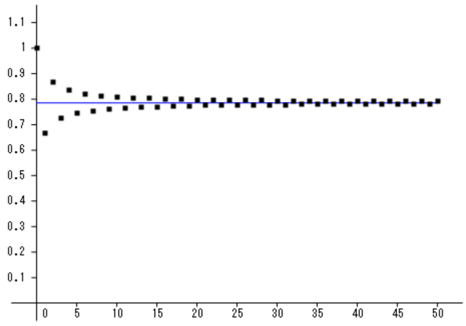




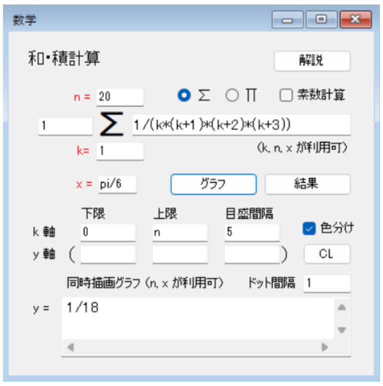
ライプニッツの公式

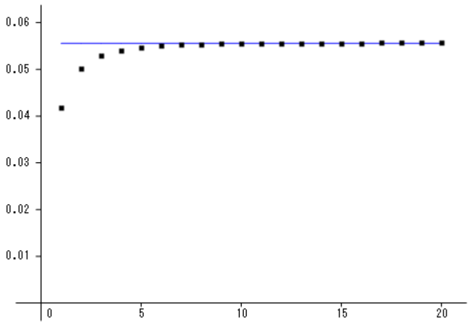
$$\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1}=\frac{\pi}{4}$$



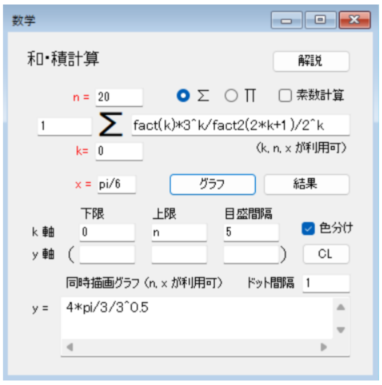


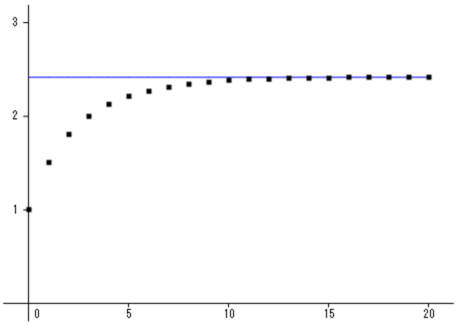
$$\sum_{k=1}^{\infty}\frac{1}{k(k+1)(k+2)(k+3)}=\frac{1}{18}$$





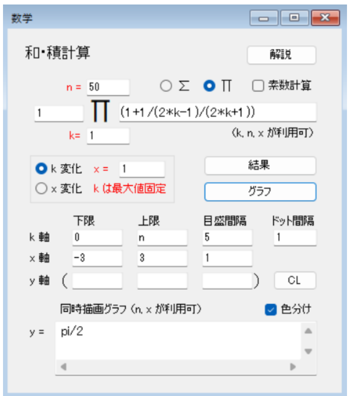
$$\sum_{k=0}^{\infty}\frac{k!3^k}{(2k+1)!!2^k}=\frac{4\pi}{3\sqrt{3}}$$

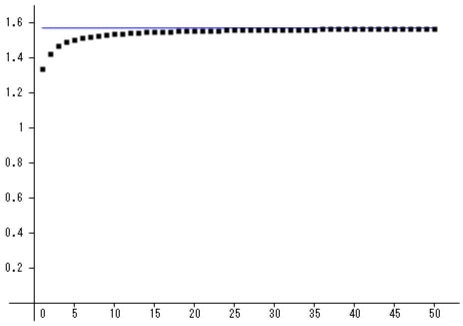




ウォリスの公式

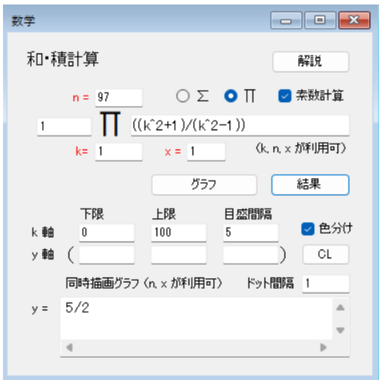
$$\prod_{k=1}^{\infty}\left(1+\frac{1}{(2k-1)(2k+1)}\right)=\frac{\pi}{2}$$

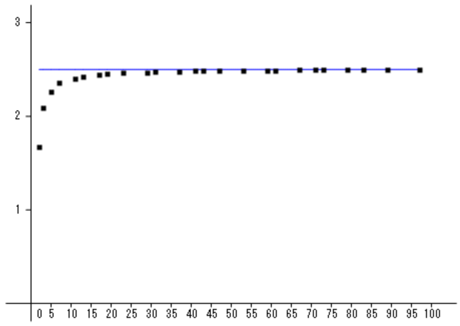




ラマヌジャンの素数公式

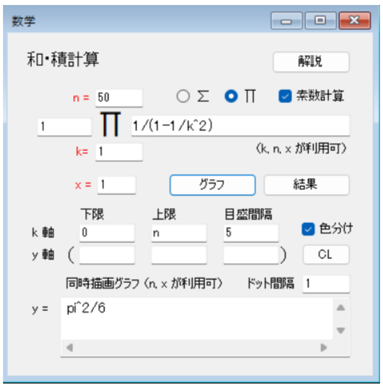
$$\prod_{k(prime)}^{\infty}\left(\frac{k^2+1}{k^2-1}\right)=\frac{5}{2}$$

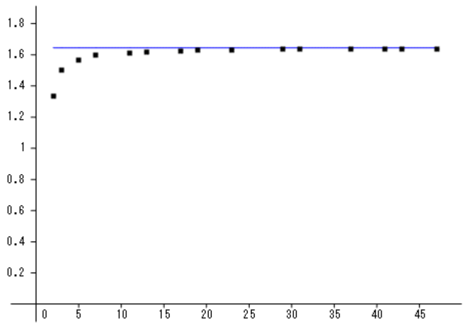




オイラーの素数公式

$$\prod_{k(prime)}^{\infty}\frac{1}{1-1/k^2}=\frac{\pi^2}{6}$$

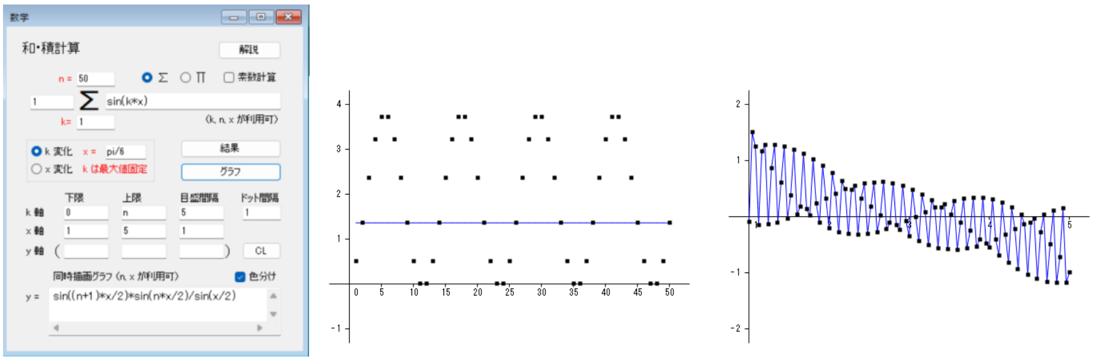




数式中に変数を含む場合

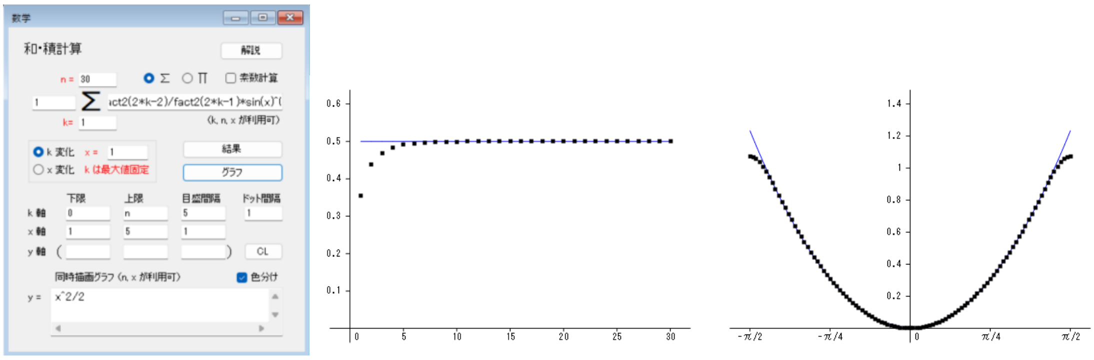
$$\sum_{k=1}^{n}\sin(kx)=\sin\frac{(n+1)x}{2}\sin\frac{nx}{2}\Big/\sin\frac{x}{2}$$

$x=\pi/6$ のとき



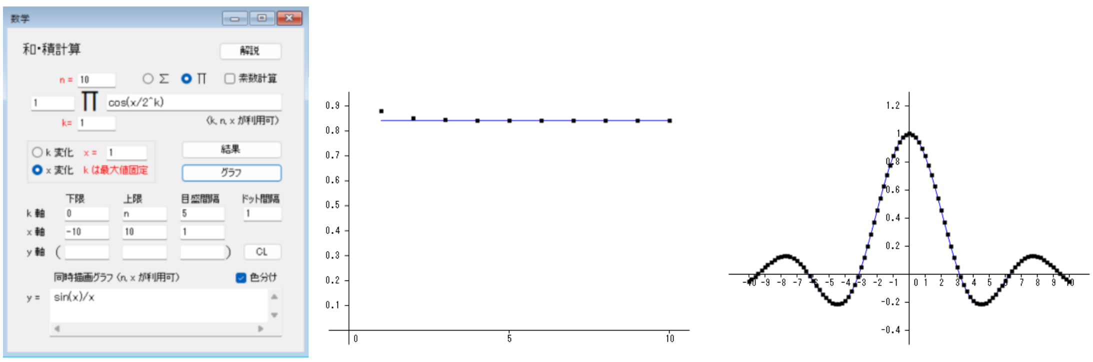
$$\sum_{k=1}^{\infty}\frac{(2k-2)!!}{(2k-1)!!}\frac{\sin^{2k}x}{2k}=\frac{x^2}{2}\qquad |x|\le\frac{\pi}{2}$$

$x=1$ のとき



オイラーの公式

$$\prod_{k=1}^{\infty}\cos\frac{x}{2^k}=\frac{\sin x}{x}$$



参考文献

[1] 森口繁一，宇田川銈久，一松信，数学公式Ⅱ－級数・フーリエ解析－，岩波全書, 1957

１７．フーリエ級数

フーリエ級数（Fourier series）は、ある有限区間の関数（有限個の不連続点も可）を三角関数の級数で表示する応用範囲の広い数式である。その公式は、区間を$[a,b]$にした場合、$f(x+(b-a))=f(x)$，$\int_a^b |f(x)|dx$：有限　のとき、$L=(b-a)/2$として、以下で与えられる。

$$f(x)\sim\frac{a_0}{2}+\sum_{n=1}^{\infty}\left(a_n\cos\frac{n\pi}{L}x+b_n\sin\frac{n\pi}{L}x\right)$$

$$a_n=\frac{1}{L}\int_a^b f(x)\cos\frac{n\pi}{L}xdx,\quad b_n=\frac{1}{L}\int_a^b f(x)\sin\frac{n\pi}{L}xdx$$

我々は教育用にこの公式を実際に計算し、グラフにして、どのように収束して行くか調べるプログラムを作成した。

メニュー［分析－数学－フーリエ級数］を選択すると以下の分析実行画面が表示される。

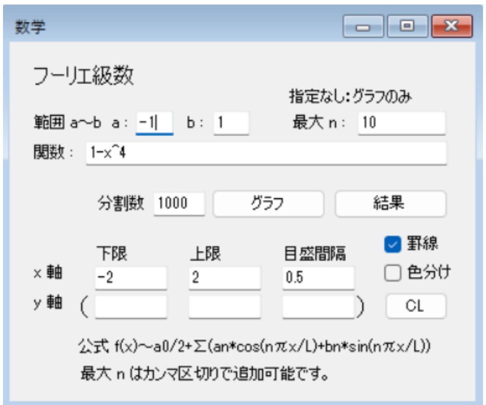


図１　分析実行画面

級数展開の範囲を「範囲 a～b」のところで設定し（”pi”の表記も利用可能）、その範囲で級数展開する関数を「関数」テキストボックスに書き込む。範囲外ではその範囲の関数が繰り返されるものとする。展開の最大の n の値は「最大 n」のテキストボックスで指定する。また、カンマ区切りで入力することで、グラフを追加することもできる。

図１のデフォルトの設定で「結果」ボタンをクリックすると以下の計算結果を得る。

計算結果

n	nπ/L	an[cos()]	bn[sin()]
0	0.0000	1.6000	
1	3.1416	0.3178	0.0000
2	6.2832	-0.1718	0.0000
3	9.4248	0.0840	0.0000
4	12.5664	-0.0487	0.0000
5	15.7080	0.0316	0.0000
6	18.8496	-0.0221	0.0000
7	21.9911	0.0163	0.0000
8	25.1327	-0.0125	0.0000
9	28.2743	0.0099	0.0000
10	31.4159	-0.0081	0.0000

図２　実行結果

ここでは、n の値に対する $n\pi/L$ の値、係数 a_n, b_n の値が表示される。ただ、この結果からは、関数がどのように近似されているかは分からない。そこで、関数の値と級数で表された

近似結果をグラフで表すことにする。「グラフ」ボタンをクリックすると、図 3 の結果が表示される。

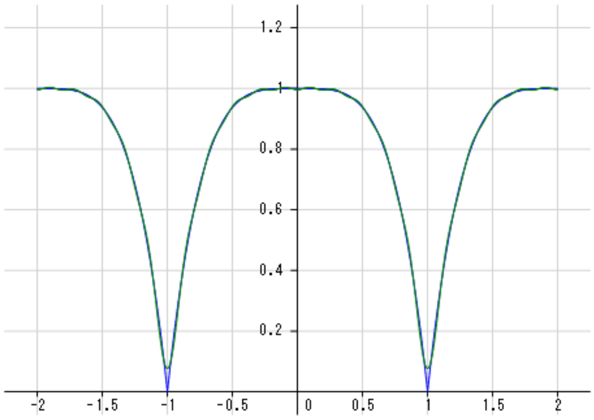


図 3　結果のグラフ表示

ここでは重なって分かりにくいが、青色のグラフが関数、緑色のグラフが近似である。見やすくするために、「最大 n」を 1,2,3,4 と増やして実行結果を見てみよう。図 4 に結果を示す。

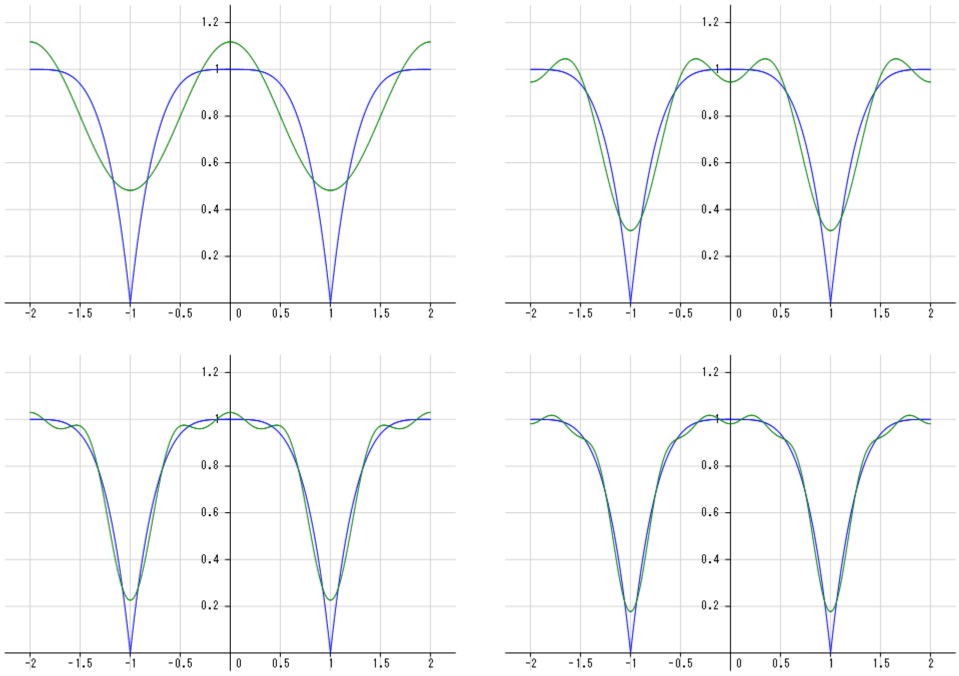


図 4　n=1, n=2, n=3, n=4 と順番に入力した場合の結果

次第に関数に近づいて行っているのが分かる。使い方は通常のグラフと同じであるので、試してみてもらいたい。

「最大 n:」テキストボックスに「1,2,3,4」とカンマ区切りで入力すると、図 5 のような結果になる。

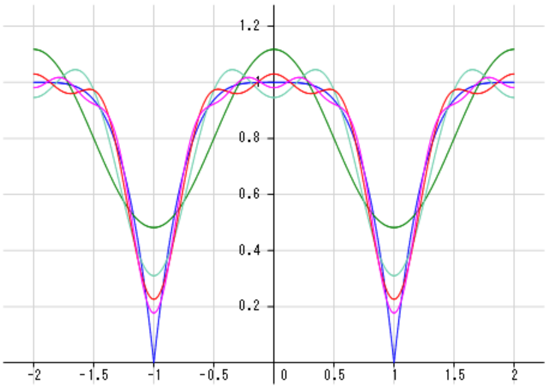


図 5　n=1,2,3,4 とまとめて入力した結果

１８．漸化式

ベルヌーイ数の計算に触発されて、漸化式の各項の値を求めるプログラムを作成した。メニュー［分析－数学－漸化式］を選択すると図１のような分析実行メニューが表示される。

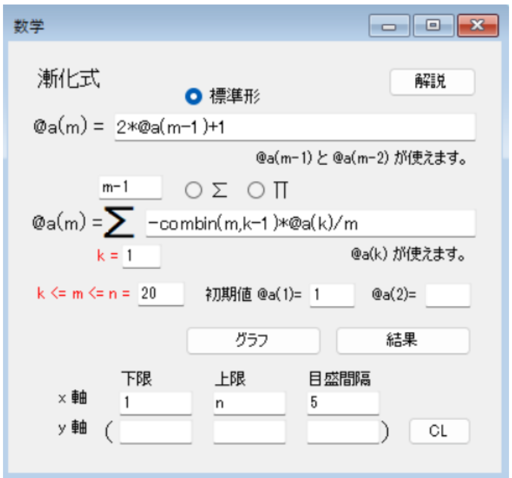


図 1　漸化式実行画面

ここでは 2 つの形式で計算が実行できるが、「標準形」ラジオボタンか「Σ」または「Π」ラジオボタンで、どちらかの形式を選択する。上の形式（「標準形」ラジオボタン）では、

$$a_n = f(a_{n-1}) \quad \text{または} \quad a_n = f(a_{n-1}, a_{n-2})$$

の形式が計算できる。左の場合は a_1 の値を、右の場合は a_1, a_2 の値を下の「初期値」のところで指定して、a_1 から「n =」のテキストボックスで指定した a_n までの値を求める。

書式は a_n を他の関数と区別して「@a(n)」で表す。即ち、右辺には「@a(n-1)」または、「@a(n-1)」と「@a(n-2)」を使った関数が入力される。例えば、上の例で、「実行」ボタンをクリックすると、図 2 の実行結果が表示される。

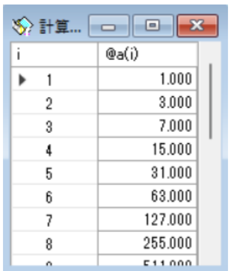


図 2　上の形式の実行結果

下の形式は、漸化式の中にΣ記号の和またはΠ記号の積が含まれる場合である。和と積は「Σ」か「Π」のラジオボタンで指定する。和及び積で利用する記号は「k」に指定しておく。そのため右辺では「@a(k)」が使われる。初期値については、k の動く範囲に応じて、a_1 または、a_1, a_2 を下の「初期値」のところで指定する。

図 1 の例は、関孝和も発見したことで有名なベルヌーイ数を計算するものである。ベルヌーイ数には様々な表式があるが、ここで利用するのは以下の表式である。

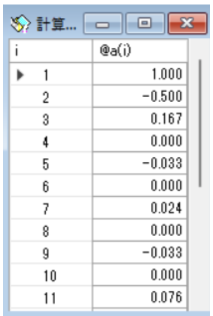
$n=0$ からの表式

$$B_0=1,\quad B_n=-\frac{1}{n+1}\sum_{k=0}^{n-1}{}_{n+1}C_k B_k$$

$n=1$ からの表式

$$B_1=1,\quad B_n=-\frac{1}{n}\sum_{k=0}^{n-2}{}_{n}C_k B_{k+1}=-\frac{1}{n}\sum_{k=1}^{n-1}{}_{n}C_{k-1} B_k$$

通常上の $n=0$ からの表式を使うが、我々の数列の定義は a_1 からなので、下の定義を使っている。計算結果は図 3 のように与えられる。



i	@a(i)
1	1.000
2	-0.500
3	0.167
4	0.000
5	-0.033
6	0.000
7	0.024
8	0.000
9	-0.033
10	0.000
11	0.076

図 3　下の形式の実行結果

利用法については「解説」ボタンで図 4 のような簡単な説明が表示される。

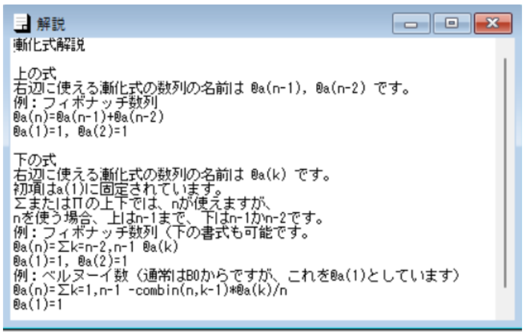


図 4　解説

有名なフィボナッチ数列については、上の形式と下の形式共に与えられ、図 5 のような設定となる。

図 5　フィボナッチ数列の設定（上下）

「グラフ」ボタンは数列の増減の様子を視覚化する。見たい項の番号を「下限」と「上限」で指定し、グラフ表示時の「目盛間隔」を入力する。a_1 から見たい場合は「下限」を 1 にする。「上限」が「n =」の値（この場合 20）よりも小さければその値まで、大きければ「n =」の値までが表示される。この場合は n = 20 までが表示される。結果を図 6 に示す。

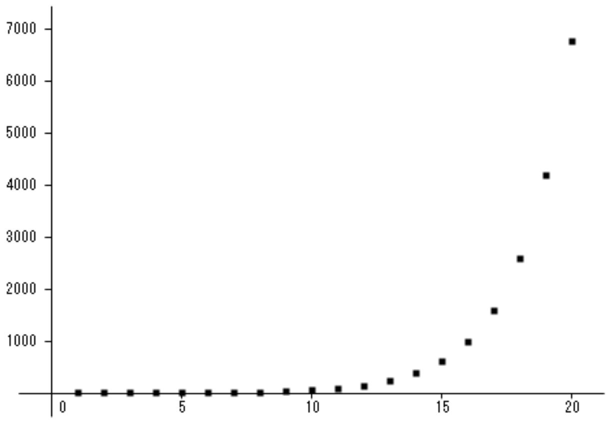


図 6　グラフ表示結果