Mathematica 入門

桂田 祐史

1999 年 7 月 1 日～

この文書 (の訂正版) は

https://m-katsurada.sakura.ne.jp/syori2/mathematica/ (HTML) あるいは
https://m-katsurada.sakura.ne.jp/syori2/mathematica.pdf (PDF)

でアクセスできます。HTML 版がメインのつもりです。

情報処理 2 のホームページは http://nalab.mind.meiji.ac.jp/~mk/syori2/

代表的な**数式処理系**である **Mathematica**（マセマティカ）を体験しましょう。数式処理でどういうことが出来るのか大体の雰囲気をつかんで、今後の学習・研究のヒントにしてもらうのがねらい。

1 Mathematica ってこんなもの

Mathematica をどのように操作するかは後で説明することにして、まずはどういうことが出来るか見てみましょう。

(ドキュメントさえ開ければ、`In[数] :=` というプロンプトの右に書いてあるコマンドをタイプした後に shift+return すれば真似が出来ます。)

いわゆる数式処理

分数計算, $\sqrt{3}$ や i 等の取り扱い、多倍長演算、$(x+y)^6$ の展開のような文字式の計算、方程式の解、微積分の計算など。

```
In[1]:= 1/2 + 1/3

Out[1]= 5/6

In[2]:= ((-1 + Sqrt[3] I) / 2) ^ 3
          |平方根    |虚数単位

Out[2]= 1/8 (-1 + i Sqrt[3])^3

In[3]:= Simplify[%]
        |簡単な形式に

Out[3]= 1

In[4]:= N[Pi, 50]
        |... |円周率

Out[4]= 3.1415926535897932384626433832795028841971693993751

In[5]:= Expand[(x + y) ^ 6]
        |展開

Out[5]= x^6 + 6 x^5 y + 15 x^4 y^2 + 20 x^3 y^3 + 15 x^2 y^4 + 6 x y^5 + y^6

In[6]:= Solve[x ^ 3 + 2 x == 1, x]
        |解く

Out[6]= {{x -> -2 (2/(3 (9 + Sqrt[177]))^(1/3) + ((1/2) (9 + Sqrt[177]))^(1/3)/3^(2/3)},
  {x -> (1 + i Sqrt[3]) (2/(3 (9 + Sqrt[177]))^(1/3) - ((1 - i Sqrt[3]) ((1/2) (9 + Sqrt[177]))^(1/3))/(2 × 3^(2/3))},
  {x -> (1 - i Sqrt[3]) (2/(3 (9 + Sqrt[177]))^(1/3) - ((1 + i Sqrt[3]) ((1/2) (9 + Sqrt[177]))^(1/3))/(2 × 3^(2/3))}}

In[7]:= Integrate[Log[x], x]
        |積分        |対数

Out[7]= -x + x Log[x]
```

(2021/8/7 追記) 少し前から、(数値を係数とする) 3 次方程式を解くのに、根号を使わなくなっていた。上のように根号を用いた結果を得るには、, Cubics->True というオプションを指定する。

```
Solve[x^3 + 2 x == 1, x, Cubics -> True]
```

(また , x という指定は省略できるようになっている。“すべての変数について解く” 場合は不要、とのこと。)

線形代数の計算

線形代数に出て来るような行列の計算も出来る。

```
In[8]:= a = {{0, 1}, {6, 1}}

Out[8]= {{0, 1}, {6, 1}}

In[9]:= MatrixForm[a]
        行列形式

Out[9]//MatrixForm=
        0  1
        6  1

In[10]:= Det[a]
         行列式

Out[10]= -6

In[11]:= Inverse[a]
         逆行列

Out[11]= {{-1/6, 1/6}, {1, 0}}

In[12]:= Eigenvalues[a]
         固有値

Out[12]= {3, -2}

In[13]:= Eigenvectors[a]
         固有ベクトル

Out[13]= {{1, 3}, {-1, 2}}
```

(2023/4/21 追記) Eigenvalues[] と Eigenvectors[] を分けて実行するのは無駄かもしれない。代わりに Eigensystem[a] とするとよい。{{3, -2}, {{1, 3}, {-1, 2}}} が得られる。

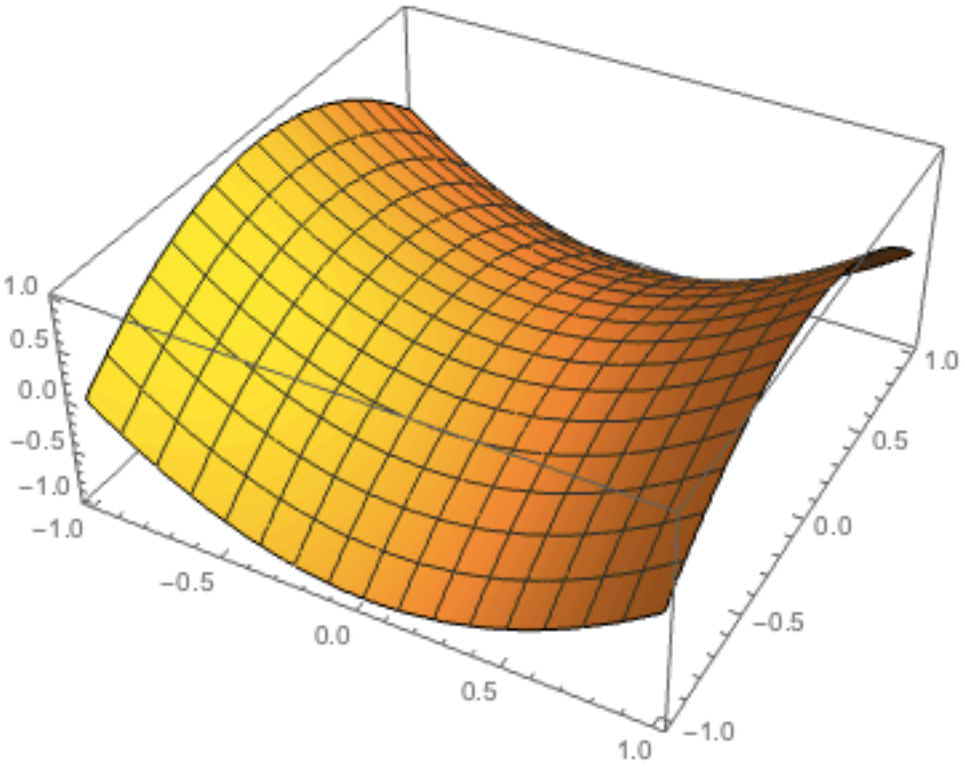
```
{{lam1,lam2},{v1,v2}}=Eigensystem[a]
a.v1-lam1 v1
a.v2-lam2 v2
```

グラフィックス

2 変数関数のグラフ、パラメーター曲面もお手の物。

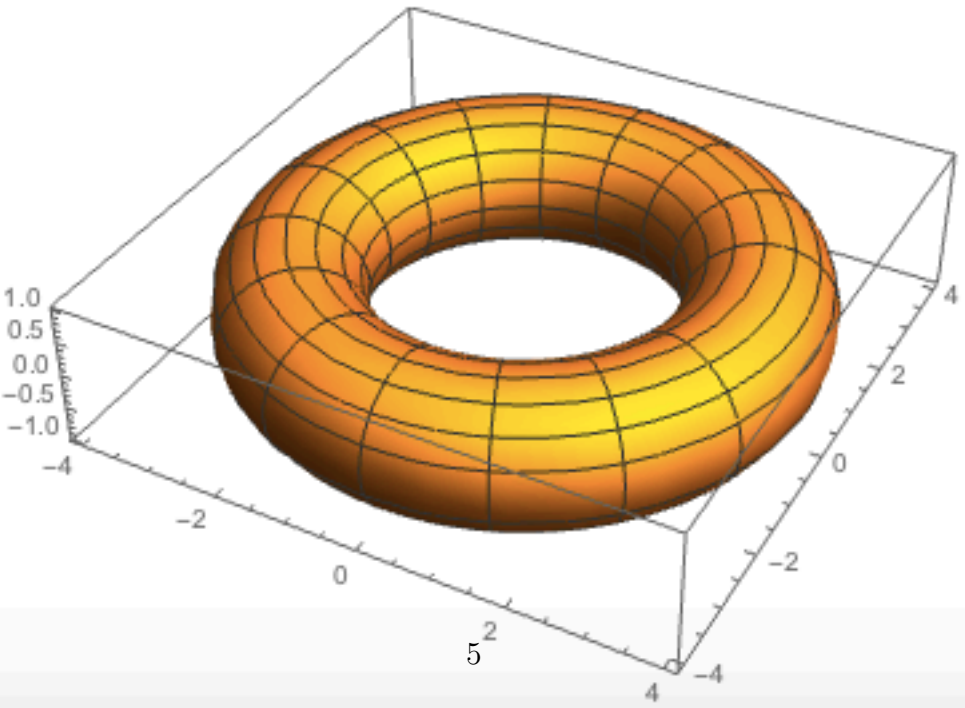
```
In[14]:= Plot3D[x^2 - y^2, {x, -1, 1}, {y, -1, 1}]
         |3Dプロット
```

Out[14]=




```
In[15]:= ParametricPlot3D[{Cos[t] (3 + Cos[u]), Sin[t] (3 + Cos[u]), Sin[u
         |3Dパラメトリックプロット |余弦          |余弦          |正弦          |余弦          |正弦
         {u, 0, 2 Pi}]
                |円周率
```

Out[15]=



コピペ用

```
Plot3D[x^2-y^2,{x,-1,1},{y,-1,1}]

ParametricPlot3D[{Cos[t](3+Cos[u]),Sin[t](3+Cos[u]),Sin[u]},{t,0,2Pi},{u,0,2Pi}]
```

2 数式処理とは

プログラミング言語 (計算機言語) の中には、数値や文字だけでなく、

数式をデータとして扱うことの出来る「**数式処理言語**」

と呼ばれるものがあります。数式処理言語を使えるソフトウェアを**数式処理系**と呼びます。現在、一般向けの数式処理系としては **Mathematica**, **Maple**(メイプル) が双璧と言われています。

(その他に **MuPAD**(ミューパッド)[1], **REDUCE**(リデュース)[2], **Risa/Asir**[3], **Macsyma**[4], **MAXIMA**[5] などが有名。)

C 言語や BASIC のようなプログラミング言語は、プログラムの中では「数式」を書けますが、関数 `scanf()` や `printf()` 等で入出力可能なデータは、数や文字列だけで、例えば `-2/5` のような分数式の入力は出来ません。またグラフを描くプログラムを作る場合に、範囲や、分割数の指定等は実行時に入力出来ても、グラフを描こうとしている関数自体は (普通の方法では) 入力できず、プログラムの中に自分で埋め込むしかなかったわけです。そういう意味では C は不自由な言語であると言えます[6]。

Mathematica は、グラフィックスやサウンドなども便利に扱えるようになっていて、ひょっとすると「数式処理系」とだけ説明するのはもう間違いかもしれません。)

3 この講義で用いる Mathematica

C 言語や BASIC のようなプログラミング言語には、国際規格があり、無償で利用できる処理系もありますが、Mathematica は **Wolfram Research** という一企業の所有物で、処理系は同社が作成・販売しているものしかありません。

---

[1] ずっと以前は、個人・非商用利用には無償で利用できるバージョンがありましたが、今では MATLAB に取り込まれました。

[2] 筆者が学生の頃 (三十ん年前)、大型計算機で REDUCE を使って、計算するのがおしゃれだった。今ではフリーソフトになっている。

[3] Made in Japan. グレブナー基底の計算など得意です。

[4] かつて MIT でしか使えなかった憧れの (歴史的) 処理系。古い本を読むと良く出て来ます。

[5] Macsyma の子孫。GPL (GNU GENERAL PUBLIC LICENSE) で配布されている (ゆえに、いわゆるフリーソフト)。メジャーになれるか ??

[6] もちろん不自由さを補って余りある大きな利点があるから、現在でも盛んに使われているわけです。例えば、実際の処理系の (反復の多い) 数値計算の速さで比べると C が圧勝します。原理的には一つのプログラミング言語があれば、どんな計算でも出来るはずなのですが、実際的な意味で万能のプログラミング言語と呼べるものは存在せず、適材適所を心がけることが重要です。みなさんも、あまり一つの言語、一つのシステムにこだわらずに、機会があったら色々なものを勉強してみましょう。

かなり高価なソフトウェアですが、教育機関に所属している人向けには、比較的安価で利用可能になっています。現象数理学科 Mac にはインストールされていて、利用できるようになっています。

Wolfram Research は、Wolfram Alpha[7] という WWW サイトで、計算サービスを提供しています。使い方は Mathematica に近いので、Mathematica に慣れていて、インターネットへの接続があれば、「ちょっと計算してもらう」ことが出来ます。

4 基本的な操作 (現象数理学科 Mac 向け説明)

言語としての Mathematica の説明の前に、操作法に類することをざっと紹介しておきます。キーワードくらい覚えておいて、後で必要になってから戻って来ると良いでしょう。

4.1 起動

色々なやり方で起動できます。

4.1.1 アプリケーション・フォルダを表示してアイコンをダブル・クリック

1. Finder でアプリケーションを表示すると、`Mathematica.app` というアイコンが見つかるはず。それをダブルクリックして起動する。

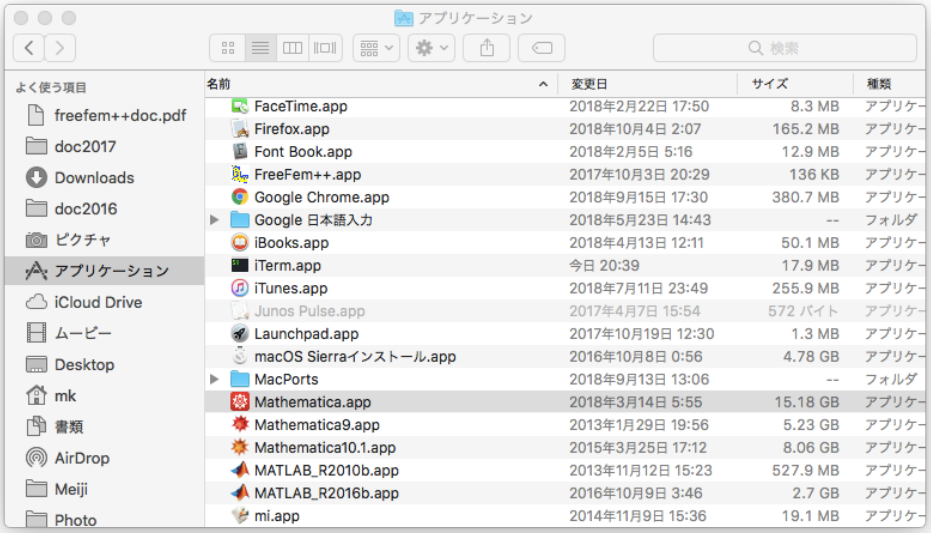


図 1: [アプリケーション・フォルダに Mathematica.app がある

[7] `https://www.wolframalpha.com/`

図 2: [Wolfram Mathematica へようこそ] 新規ドキュメントをクリック

2. 起動すると「名称未定義-1」という名前のウィンドウが現れる。例えばここにキーボードからコマンドをタイプして、最後に Shift+return を打って (やや珍しい、要注意！)、コマンドを入力するのが基本である。

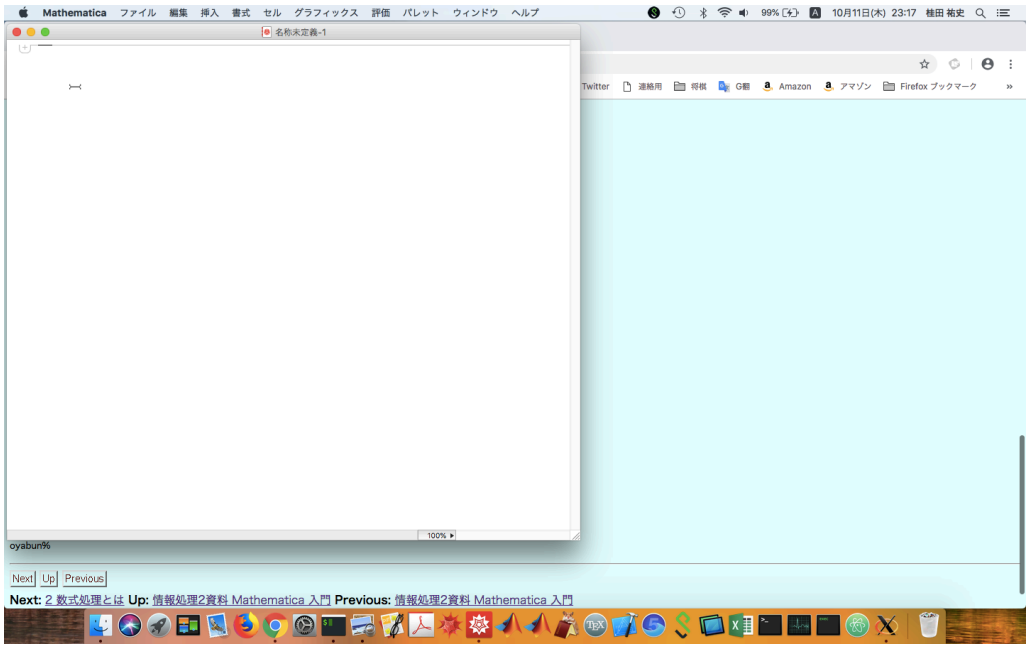


図 3: Mathematica 起動直後の様子

3. 実際にコマンド入力＆実行させてみた。

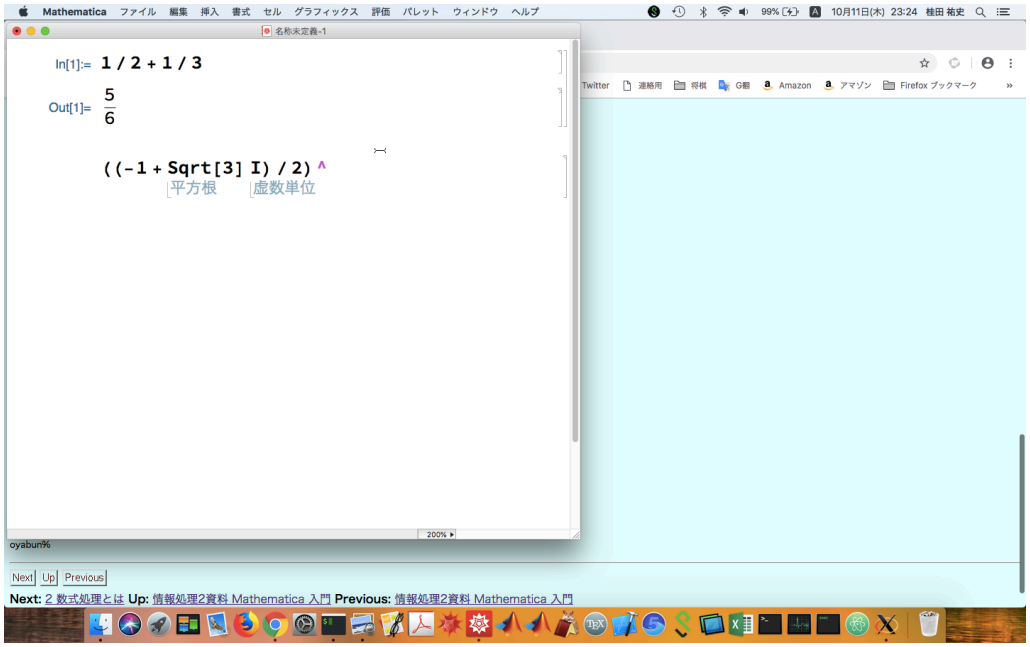


図 4: コマンドを一つ入力実行、二つ目のコマンドを入力途中

多分、次のこと以外は覚える苦労はないはずと思う。

Mathematica のコマンド入力は、最後に shift+return

4.2 起動後の基本操作

- コマンドをキーボードから入力してから、最後に shift+return をタイプする。
  コマンド (command) の部分に “In[番号]:= ” というラベルが加えられ、“Out[番号]= ” に続いて計算結果 (result) が出る。
  `In[番号]:= コマンド` を**入力セル**、`Out[番号]= 結果` を**出力セル**と呼ぶ。

- 一度評価 (計算) した入力セルを書き換えて再び shift+return を入力すると、再評価して、その結果で出力セルを上書きする。

- コマンド入力の最後にセミコロン (;) を書くと、計算結果は表示されない[8]。

- 入力に関して便利な記号として

%	直前の結果
%%%$\cdots$% (パーセントを k 個重ねる)	k 回前の結果
%n (n は自然数)	Out[n] と同じ (n 番目の結果)。

[8]意味がないように思うかも知れないが、`a=2^1000` のような変数への代入など、副作用のある処理をする場合に使うことがある。

4.3 計算を強制的に止める

時々計算が暴走することがある (決して終わらない計算を実行するなど)。そういう場合など、現在実行中の計算を止めさせたくなったときは、[評価] メニューから [評価を放棄] を選択する。コンピューターが計算に夢中だとすぐには止まってくれないが…

Mac では、Command-. で評価を放棄。

Windows では、Alt-. で評価を放棄、Alt-, で評価を中断。

4.4 作業内容の保存 (ノートブックの利用)

Mathematica では、入力されたコマンドを、**ノートブック**という形式で保存することが出来る。そのためには [ファイル] メニューから [別名で保存] を選んで、ファイルを置くフォルダとファイル名を指定する。

ファイル名は “名称未定義-1.nb” から変更するのが無難。ファイル名から中身が推測できるようなものが望ましい。拡張子は .nb のままにしておくこと。

ファイルの名前は衝突しないように気をつけよう。くれぐれも一つのファイルを二つのプロセスで扱ったりしないこと。

レポート提出用のノートブックを作るには、例えば次のようにして整理すると良い。

1. 課題をこなすための計算を一通り実行した後で、[ファイル] メニューから [新規作成] を選んで新しいノートブックを開き、そちらに必要なコマンドをコピー, 貼付けして、再実行 (shift+return で OK) することで、必要なところだけを抜き出したノートブックを作る。

   あるいは、ノートブックのウィンドウで、セルの右側をクリックして選択して、Delete で掃除が出来る。

2. [ファイル] メニューから [別名で保存] を選び、マイドキュメントに適切な名前 (“kadai10” とか “syori2-0630” とか) で保存する (拡張子は .nb となる)。

保存してあるノートブックを開く

Mathematica のノートブック (拡張子 ‘.nb’) を、ダブルクリックすれば開くことが出来る。

Mathematica の [ファイル] メニューから [開く] コマンドを選んでファイルを選択することでも開くことが出来る。

実行したいコマンドのところで、shift+return として再計算しても良いが、[評価] メニューで「ノートブックを評価」を選ぶと、ノート全体の内容の再計算が出来る。

4.5 計算結果を含めた記録文書を作る

ノートブックには、計算結果は含められない。計算結果を含めた文書が欲しい場合は、TEX で作る方法もあるが、お手軽には、次のやり方が勧められる。

(1) まずノートブックを読み込み、[評価] → [ノートブックを評価] を実行する。

(2) ファイルの種類として、PDF を選んで出力する。

4.6 パレット

[パレット] → [基本数学アシスタント] で、入力を補助するためのウィンドウ (**パレット**) が現れる。

Mathematica 使い始めのうちは、パレットが便利に感じられる人も多いであろうが、これで入力できるものは限られているし、キーボードからコマンドを打ち込むのに慣れるにつれ、必要なくなると思われる。

4.7 カレント・ディレクトリィ

(以前は必須の知識であったが、もう不要？)

現在注目しているフォルダ (カレント・ディレクトリィ) を知るには、`Directory[]` を実行する。

カレント・ディレクトリィを変えるには、`SetDirectory[]` を用いる。

通常は起動直後は、カレント・ディレクトリィは、ホームディレクトリィ(`~` 普通は `/Users/`ユーザー名) になっている。

Mac で書類 (`~/Documents`) に移動

```
SetDirectory["~/Documents"]
```

Windows の場合は、パスでディレクトリィを区切るのに通常はバックスラッシュ \ を用いるが、例えば `C:` ドライブのルート・ディレクトリィに移動するには、`SerDirectory["C:\\"]` と重ねる必要がある。バックスラッシュの代わりにスラッシュを使えば `SerDirectory["C:/"]` と重ねずに済む。

長いパス名のフォルダを作業フォルダにしてそこに移動するのが面倒な場合、そのディレクトリィへ移動するコマンドのみ書いたノートブック (“ここに行く.nb”) を作って、それをダブルクリックして Mathematica を起動するという手もある。

カレントディスクトリィにあるファイルの名前を列挙

```
FileNames[]
```

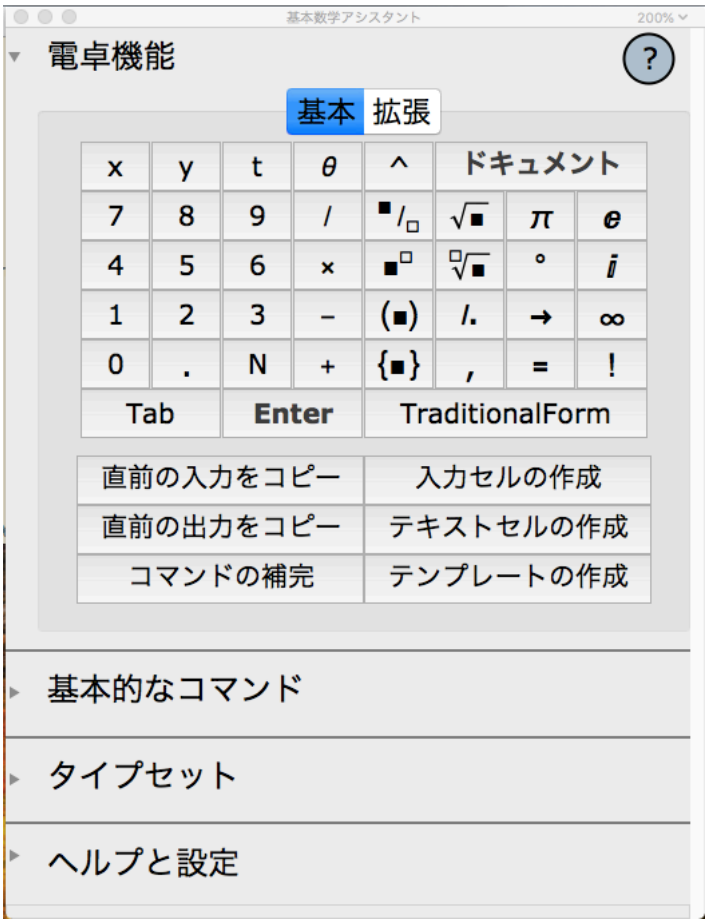


図 5: 基本数学アシスタントの基本タブ

図 6: 基本数学アシスタントの拡張タブ

5　電卓的な使用

最初の「Mathematica ってこんなもの」を試してみた後は、簡単な文法と、関数を知るだけでかなり便利に使えるはずです。

5.1　いくつかの定義済みの定数

`Pi`(円周率 π), `E`(自然対数の底 e), `Degree`($= 180/\pi$), `I`(虚数単位 $i = \sqrt{-1}$), `Infinity` (無限大 $+\infty$), `ComplexInfinity` (複素平面の無限大), `GoldenRation` (黄金比 $= (1+\sqrt{5})/2$) などの定数があらかじめ定義されています。

Mathematica に説明してもらおう

```
??Pi
??Degree
??I
??Infinity
??GoldenRatio
```

図 7: `??Pi` とすると簡単な説明が得られる。`Local >>` をクリックすると…

5.2　かっこ

C 言語のようなプログラミング言語に慣れていると、かっこによって意味の違いがあるのはすんなり受け入れやすいでしょうか。

- `( )` は演算の結合順位をグループ化する普通のかっこ。
- `[ ]` は関数 (コマンド) に渡す引数を示す。
- `{ }` はリストを作るかっこ (リストについては後述)。

図 8: 詳しい説明が出て来る。

- [[ ]] はリストの要素のアクセス (これも後述)。

色々なかっこ

```
(1+2)*3
Sin[Pi/3]
a={{1,2},{3,4}}
a[[2]][[1]]

Remove[a]
```

5.3 変数 (名前) の宣言、変数への代入

- 名前では大文字小文字を区別します。組み込みの名前は先頭が大文字になっています。ユーザーはなるべく小文字を使うことを奨励されています (衝突を回避するため) 。
- a=式, a=b=式, 等で代入。
- ?Global`* とすると、(自分が) 現在何を定義しているか表示します。
- ?*Sin* とすると、"Sin" を含む名前の一覧が表示されます。
- a=. またはClear[a] で変数 a の内容を除きます (内容を除く — 名前は残ります)。Clear[a,b,···] のように複数の変数の内容を同時に除くこともできます。
- Remove[a,b,···] 名前まで込めて完璧に除く。

どういう名前が定義されているかは、??Global`* でチェックできます (一度やってみることを勧めます)。

不要になった変数 (または後で出て来るユーザー定義の関数) は、Clear[名前] または Remove[名前] 等でこまめに削除しておくことを勧めます。一度使った変数に残っているゴミのせいで首をひねることが良くあります。名前を指定するためにアスタリスクが利用できます。Remove["Global`*"] とすると、Mathematica を起動してから、ユーザーが定義したものは全部削除できます (こういうことをして良いのか、若干自信がありませんが…)。

変数の定義は不要になったら消さないと副作用が出る

```
In[1]:= f=x^2+2x+3

                     2
Out[1]= 3 + 2 x + x

In[2]:= x=1

Out[2]= 1

In[3]:= f

Out[3]= 6

In[4]:= D[f,x]

General::ivar: 1 is not a valid variable.

Out[4]= D[6, 1]

In[5]:= Clear[x]

In[6]:= D[f,x]

Out[6]= 2 + 2 x

In[7]:= Remove[f,x]
```

5.4 演算子

- 四則演算は普通の記号で OK。a+b, a-b, a*b または a b (空白抜きで ab は一つの名前。a5 も一つの名前だが、5a は 5 かける a の意味), a/b.
- べき乗 a^b (これは右結合[9]する演算子です), 階乗 a! もあります。
- (C 言語にどっぷり使っている人向けの注意) パーセント % は整数の余りではなくて、直前の結果を表す記号です。

[9] a^b^c は $a^{b^c} = a^{(b^c)}$.

+,-,*,/ は他の言語と同じだけれど

```
2^10
2^2^3+1
1 2 3 4 5 6 7 8 9 10
10!
E^(I Pi)
```

5.5 組み込み関数

Sqrt[x], Exp[x], Log[x], Log[b,x], Sin[x], Cos[x], Tan[x], ArcSin[x], ArcCos[x], ArcTan[x], Abs[x], Round[x], Random[], Max[x,y,...,z], Min[x,y,...,z] のように普通のプログラミング言語にそなわっているような関数はもちろん、色々なものがあります (いざとなったら Help を見て下さい)。

とりあえず大学 2 年生がよく知っていそうな初等関数

```
Sin[Pi/6]
Cos[45Degree]
Tan[Pi/2]
ArcTan[1]
Log[10,2.0]
Sin[ArcSin[x]]
ArcSin[Sin[x]]
Exp[Log[x]]
Log[0]
```

以下、重要なことを (重複をいとわず) いくつか紹介します。

- 関数の引数を囲む括弧は [ ]
- システム組み込みのものは名前の先頭の文字が大文字です。
- ?文字列* で一覧表、さらに ?関数名 や??関数名 でその関数の説明が出ますが、Help から探す方が良いかもしれません。
- 関数がどういうオプションを持っているか見るに Options[] を用います。

Options[Plot]	Plot のオプション全部を表示
Options[Plot, PlotRange]	Plot の PlotRange オプションを表示

- オプションを一時的に変更するには “SetOptions[]” を用います。

5.6 無限多倍長整数 (小数も任意精度まで)

整数や有理数等は (可能な限り) 誤差のない計算をします。小数に対しては有限精度の計算となります。その場合 Sin[] などの関数の精度はデフォールトでは C 言語の double と同程度 (つまりパソコンや WS では 10 進 16 桁弱) ですが、精度はかなり自由に指定できます。

整数や有理数については、特に意識せずに使うことが出来ます。100! や2^100 などで試せます。

```
a=100!
b=99!
a/b

a=3^100
b=3^98
a/b

a=1.732050800000000000000000000000000000000000000  (0 を多めにつける)
a=(3/a+a)/2
a=(3/a+a)/2
 ...                何回か繰り返す
a^2
Remove[a,b]
```

5.7 分数

1/2+1/3 のようにして分数が入力できて、分数のまま計算できます。
文字式の分数については、5.11「結果の簡単化」などを見て下さい。

5.8 ルートなど

ルートなどのシンボリックな数も使えます。例えば Expand[(-1+Sqrt[3]I)^3] など。

1 の原始三乗根 $\omega = \dfrac{-1+\sqrt{3}i}{2}$ で遊ぶ

```
w=(-1+Sqrt[3]I)/2
w^2+w+1
Simplify[%]
w^3
Simplify[%]
Remove[w]
```

5.9 数値への変換 (近似値の計算)

分数や Sqrt[3] のようなシンボリックな数を、数値 (有限桁の小数) に変換するのに、N[式] あるいは 式 // N が使えます: N[(1+Sqrt[5])/2]

特に N[Sqrt[2], 100] のように精度を指定できます。

数値への変換

```
(1+Sqrt[3])^2; N[%,100]  直前の結果を 100 桁
N[Pi,1000]
N[E,1000]
2^100 // N
```

桁数が多いときは、NumberForm[] を使って、適当な桁数でブロックにまとめると良い。

```
NumberForm[N[Pi, 1000000 + 1], DigitBlock -> 10]
```

5.10 文字式 (多項式、分数式、その他)

これが「数式処理」という言葉にもっともしっくり来る計算でしょうか (英語を覚えていないと大変かも知れませんが…頑張って下さい)。

文字式の計算 (1)

```
Expand[(1+x)^10]                          多項式の展開
Factor[x^3+y^3+z^3 - 3 x y z]             多項式の因数分解
Apart[1/(x^3-1)]                          部分分数への分解
2 f[x] + 3 f[x]

p1 = Expand[(1+x)^10]                     多項式の展開結果を変数に代入
p2 = (1+x)^3
PolynomialQuotient[p1,p2,x]               多項式の商
PolynomialRemainder[p1,p2,x]              多項式の剰余
PolynomialQuotientRemainder[p1,p2,x]      多項式の商と剰余 (同時に求める)
Remove[p1,p2]                             おそうじ
```

有理数・有理式に対して、Numerator[], Denominator[] で分子、分母を取り出すことが出来る。

多項式 $f(x)$ と $g(x)$ の最大公約多項式を $d(x)$ とするとき、

$$d(x) = p(x)f(x) + q(x)g(x)$$

を満たす多項式 $p(x)$, $q(x)$ が存在する (Euclid の互除法によって、$d(x)$ と同時に計算できる) が、PolynomialExtendedGCD[] は $d(x)$, $f(x)$, $g(x)$ を求めることが出来る。

文字式の計算 (2)

```
f=x^6+1
g=x^3-2x^2+x-2
PolynomialGCD[f,g]              最大公約多項式
PolynomialExtendedGCD[f,g,x]    最大公約多項式

Discriminant[a x^2+b x+c,x]     おなじみ ax^2+bx+c の判別式
Discriminant[x^3+p x+q,x]       3 次式の判別式は見たことがないかな？
Remove[f,g]                     おそうじ
```

(余談: 最近 (2011 年のこと)、16 次多項式の判別式の計算に成功したというニュースがありました。38 億近い項数だそうです。次数が高くなると、急激に大変になるんですね。)

終結式を計算する Resultant[], グレブナー基底を計算する GroebnerBasis[] なども用意されています (高校数学には出て来ない、ちょっと高級な概念)。

5.11 結果の簡単化

計算結果が自分の望んでいる形に表されないことがしばしばあります。まず Simplify[] という関数を覚えておきましょう。結果が複雑だったら、取りあえず Simplify[%] として、「直前の結果の簡単化」を試みるのが良いでしょう (変わらないことも多いですが)。

ちなみに、通分は Together[], 因数分解は Factor[], 分数式の約分[10]は Cancel[], 展開は Expand[] です。例えば

色々な式変形

```
y=1/(x^2-1)
Apart[y]        部分分数への分解
Together[%]     通分
Simplify[%]     とりあえず簡単化
Factor[%]       因数分解
1/%             逆数を計算してから
Expand[%]       展開
Remove[y]
```

その他に FullSimplify[] などがありますが、少し時間がかかります。

仮定を与えて簡単化させることも出来ます (ここは結果を見てじっくり考えて下さい)。

[10] 共通因数を約する。

仮定を与えて簡単化

`Sqrt[x]^2`	これはすぐ簡単になる。
`Sqrt[x^2]`	これは簡単にならない。
	x について何か仮定がないと。
`Simplify[Sqrt[x^2],x<0]`	$x < 0$ ならば $\sqrt{x^2} = -x$ と簡単化
`Simplify[Sqrt[x^2],Element[x,Reals]]`	$x \in \mathbf{R}$ ならば $\sqrt{x^2} = \|x\|$ と簡単化

`Element[x,D]` の領域 `D` としては、`Reals` ($\mathbf{R}$), `Integers` ($\mathbf{Z}$), `Complexes` ($\mathbf{C}$), `Primes` (素数全体の集合) などがあります。

Mathematica は (最近では) かなり注意深い computer (計算者) です。どうしてこういう計算をしてくれないのか？と思ったら、理由を考えてみることを勧めます。

わかりますか？

`(-1)^(2n+1)` $(-1)^{2n+1} = -1$ のはず？

5.12 整数 (素因数分解、素数判定)

整数に関する演算

`n=2^2^5+1`	$n = 2^{2^5} + 1 = 4294967297$
	2^{2^5} は $2^{(2^5)} = 2^{32}$ という意味です ($(2^2)^5 = 2^{10}$ ではない)。
`PrimeQ[n]`	n は素数かどうか判定する
	素数は英語で prime という。'`Q`' は Question の頭文字。
`EvenQ[n]`	n は偶数かどうか判定する
	偶数は英語で even number という。
`OddQ[n]`	n は奇数かどうか判定する
	偶数は英語で odd number という。
`FactorInteger[n]`	n の素因数分解
`641 6700417`	掛け算してみる
`Remove[n]`	おそうじ
`Mod[123456,123]`	`123456` を `123` で割った余り
`GCD[96,18]`	最大公約数
`Table[Prime[n],{n,100}]`	最初の 100 個の素数
`Divisors[48]`	48 の約数全体

5.13 複素数

虚数単位 $i = \sqrt{-1}$ が、`I` という名前で定義済みであることを繰り返しておきます。

複素数あれこれ

`E^(I Pi)+1`	Euler の公式 $e^{i\pi}+1=0$
`z=(3 + 4I) (1 + 2I)`	$(3+4i)(1+2i)$
`Re[z]`	実部 (real part)
`Im[z]`	虚部 (imaginary part)
`Conjugate[z]`	共役複素数 (conjugate)
`Abs[z]`	絶対値 (absolute value)
`Arg[z]`	偏角 (argument)
`ComplexExpand[E^(Pi I/6)]`	$a+ib$ の形にする
`Remove[z]`	おそうじ

例えば $x^3 = 1$ を解かせると、$1,\ -(-1)^{1/3},\ (-1)^{2/3}$ と答えて来ます。そういうときは、`ComplexExpand[]` をしないと分かりづらいでしょう。

```
Solve[x^3==1,x]
ComplexExpand[%]
```

5.14 一時的な代入 /.

式に含まれる変数に一時的に値を代入して式の値を求めたい場合は、置換演算子 /. (スラッシュ・ドットと読む) を用いて、

式 /. 名前 -> 値

のようにします。ここで「名前 -> 値」は置き換えの規則を表わします。

具体的には、例えば

一時的な代入 (1)

```
y = x^2
y /.  x->3
x
y                x, y 自体は変っていない
Remove[x,y]
```

一度に複数の代入をするには、括弧 { } でくくって、

式 /. {名前 -> 値, 名前 -> 値,...}

とします。

一時的な代入 (2)

```
(x + y + z + w)^2
% /.  {y->1,z->2}   y に 1 を、z に 2 を代入
Remove[x,y,z,w]
```

方程式の解を、Solve[] (詳しいことは後述) を用いて求めたときの結果は、この代入をするのに便利です。例えば $x^2+x+1=0$ の根の 3 乗を計算するには、次のようにすれば OK です。

方程式の解を代入してチェックする

```
sol=Solve[x^2+x+1==0,x]
                          (-1)^(1/3) みたいのが出て来て良く分からないので…
ComplexExpand[sol]        a+ib の形にする
x^3 /.  sol               解の 3 乗を計算してみる
Simplify[%]               簡単化
Remove[x,sol]
```

(なお、Mathematica は細かい部分でバージョンにより動作が異なり、以前は sol のような置き換え規則に ComplexExpand[] を施しても何もしてくれませんでした。)

一方、$x^2-3x+2=0$ の解を x1, x2 とおくには、

```
sol=Solve[x^2-3x+2==0,x]
{x1,x2}=x /. sol
```

とすれば OK です。

もう少し複雑な 3 次方程式の根 (結果はかなり複雑) のチェックをしてみましょう。

3 次方程式を解いて、解であるか確かめる

```
f[x_]:=x^3+2x^2+3x+4
sol=Solve[f[x]==0,x]
f[x] /. sol
  → すごいことになるが
Simplify[%]
  → 納得するはず
ComplexExpand[sol]  または  ComplexExpand[x /. sol]
  → うーん…(訳が分からない !? )
N[sol]
  → 近似値を見ると複素平面上で大体どの辺にあるか分かります

Remove[x,f,sol]
```

解けたらしいけれど、簡単な答を教えてくれない…どうしてでしょう ??

(2019/11/2 追記: Mathematica 11.x では、3 次方程式を `Solve[]` で解くと、Cardano の公式を適用した結果を表示してくれたのですが、Mathematica 12 ではそれはやらなくなりました。ちょっと残念。`Solve[f[x]==0,x,Cubics->True]` とするのかな。)

5.15 微分

微分をする関数 `D[]` があります。

`D[x^n,x]`	$(x^n)'$
`f=x^2+2x y+3y^2+4x-5y+6`	
`D[f,x]`	f_x
`D[f,x,y]`	f_{xy}
`D[f,{x,2}]`	f_{xx}
`D[Exp[2x+3y],{x,2},{y,3}]`	$f(x,y) := \exp(2x+3y)$ に対して f_{xxyyy}
`Remove[f,x,y,n]`	
`f[x_]:= x^2+2x+3`	
`f''[x]`	$f''(x)$
`D[f[x]*g[x],{x,5}]`	積の高階微分 (Leibniz の公式 5 階の場合)
`Remove[f,g,x]`	

テーラー展開をするには、微分をたくさんする必要がありますが、そのための専用の命令 `Series[]` があって便利です。

`Series[Exp[x],{x,0,10}]`	$x=0$ のまわりの 10 次までの Taylor 展開！

微分方程式の独立変数の変数変換は面倒ですが、かなりの部分、Mathematica で出来ます。次は 1 次元波動方程式を解くための有名な変数変換の例の確認です[11]。

```
u[x_,t_]:=v[x-c t,x+c t]
D[u[x,t],{t,2}]/c^2-D[u[x,t],{x,2}]
Simplify[%]
```

逆に

```
V[xi_,eta_]:=U[(xi+eta)/2,(xi-eta)/(2c)]
D[V[xi,eta],xi,eta]
Simplify[%]
```

[11] $u(x,t)=v(\xi,\eta)$, $\xi=x-ct$, $\eta=x+ct$ とするとき、$\dfrac{1}{c^2}\dfrac{\partial^2 u}{\partial t^2}-\dfrac{\partial^2 u}{\partial x^2}=-4\dfrac{\partial^2 v}{\partial\xi\partial\eta}$.

5.16 不定積分、定積分

不定積分は Integrate[式, 変数], 定積分は Integrate[式, 変数の範囲] とします。

```
Integrate[1/(1+x^2),x]
```
$$\int \frac{1}{1+x^2}dx$$

```
Integrate[1/(1+x^2),{x,0,1}]
```
$$\int_0^1 \frac{1}{1+x^2}dx$$

```
Integrate[E^(-x^2),{x,0,Infinity}]
```
$$\int_0^\infty e^{-x^2}dx$$

$\int_0^1 \sqrt{x^2+1}\,dx$ を計算させると $\frac{1}{2}\left(\sqrt{2}+\mathrm{ArcSinh1}\right)$ と言う結果を返します。ArcSinh $z = \sinh^{-1} z = \log\left(z+\sqrt{z^2+1}\right)$ であるので、log を使って表示できます。そうさせたい場合は TrigToExp[] を作用させます。

```
In[]:=TrigToExp[Integrate[Sqrt[x^2+1],{x,0,1}]]
```
$$\text{Out[]=}\frac{1}{\sqrt{2}}+\frac{1}{2}\log(1+\sqrt{2})$$

定積分の場合、数値積分版 NIntegrate[] もあります。近似値しか計算できませんが、Integrate[] では計算できないような定積分も扱えます。

```
NIntegrate[Sqrt[Sin[x]], {x,1,2}]
NIntegrate[Sqrt[Sin[x]], {x,1,2}, WorkingPrecision->50,AccuracyGoal->40]
```

文字定数を含んだ関数の積分をするとき、その定数が正数であるとか、実数であるとか、Mathematica に教えてやらないと計算出来ないこともあります。

```
Assuming[a>0, Integrate[Exp[-x^2]Cos[2a x],{x,-Infinity,Infinity}]

Integrate[Exp[-x^2]Cos[2a x],{x,-Infinity,Infinity}, Assumptions->{a>0}]

Integrate[Exp[-x^2]Cos[2a x],{x,-Infinity,Infinity}, Assumptions->Im[a]==0]
```

計算してから、Simplify[%],Assumptions->仮定 とする方法もあります。

これと少し似ている問題に、z[u_]:=NIntegrate[Sqrt[1-Exp[-2*t]],{t,0,u}] のように、NIntegrate[] を使って定義した関数を含む式を評価するときに、積分範囲が数値として定まらないうちに評価してしまうと問題が生じる、というものがあります。こういう場合は、z[u_?NumericQ]:=NIntegrate[Sqrt[1-Exp[-2*t]],{t,0,u}] のように定義すると、u が数値であるかどうか判定出来るまで、z[] の評価が後回しにされて問題を回避出来る場合があるようです。

```
x[u_] = Exp[-u]
z[u_?NumericQ] := NIntegrate[Sqrt[1 - Exp[-2*t]], {t, 0, u}]
p[u_, v_] := {x[u]*Cos[v], x[u]*Sin[v], z[u]}
ParametricPlot3D[p[u, v], {u, 0, 3}, {v, 0, 2*Pi}]
```

(Mathematica のバージョンによっては、?NumericQ がなくても問題なく動きますが、警告が表示されるバージョンもあります。)

限定的ですが、$\iint_{x^2+y^2\leq 1} e^{-x^2-y^2}dx\,dy$ のような重積分も出来ます (重積分の計算については、誰かレポートを書いてくれないかな…)。

```
Integrate[Exp[-x^2-y^2] Boole[x^2+y^2<=1], {x,-1,1}, {y,-1,1}]
```

5.17 級数

いわゆる数列の和 (シグマ) $\sum$ を計算する Sum[] があります。無限級数の和も計算できることに注意。

Sum[n,{n,1,5}]	$\sum_{n=1}^{5} n$
Sum[k^2,{k,n}]	$\sum_{k=1}^{n} k^2$
Sum[r^n,{n,0,Infinity}]	等比級数
Sum[1/n^2,{n,Infinity}]	$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$

余談: 確率がらみの計算などで、2 項係数 ${}_n\mathrm{C}_r = \binom{n}{r}$ が必要になることがありますが、Binomial[n,k] で計算できます。Mathematica はかなり複雑な計算もしてくれるので驚きです。

5.18 方程式を解く

方程式を解くために、Solve[], NSolve[] という二つの手続きがあります。

Solve[左辺==右辺, 未知数] は式変形で解きます (等式を表すために = を二つ続けた == を使うのが大事なところ)。

```
Solve[x^2+3x+2==0, x]
```

この結果は {x->1} のような、置き換え規則のリストの形で得られますが、x /. % とすれば (置き換えを実行するわけです)、解のリストが得られます (5.14 参照)。

次のような連立方程式も解けます。

```
Solve[{x+y+z==6, 2x-y+z==5,-3x+y+2z==0},{x,y,z}]
```

Mathematica は 2 次方程式だけでなく、3 次方程式、4 次方程式の根の公式も覚えていて解くことが出来ますが、やってみれば分かるように、結果は分かりにくくなることが多いです。複素数について、5.13 を見ておいて下さい (特に ComplexExpand[] を使うなど)。

どの程度の値なのか知りたい場合、つまり近似値でよければ、直後に % // N あるいは N[%, 50] のように入力すれば求められます。

```
Solve[x^3+2x^2+3x+4==0, x]
% // N                              直前の結果を小数で
N[%%, 50]                          二つ前の結果を 50 桁の小数で
```

文字を係数に含む方程式も解くことが出来ます。

```
Solve[a x + b ==0, x]
Reduce[a x + b ==0, x]   ちゃんと場合わけをする
```

もっとも、特別簡単なものを除けば、方程式は式変形のみで解くことは難しく、できないことの方が多いです。そういう場合は、近似値を求めることで我慢することにすれば、NSolve[], FindRoot[] などの関数が利用できます。

```
NSolve[x^3+2x^2+3x+4==0, x, 40]              精度 40 桁で解く
FindRoot[x^3+2x^2+3x+4==0, {x, 0}]           Newton 法で 0 の近くの解を探す。
FindRoot[x^3+2x^2+3x+4==0, {x, 0},
    WorkingPrecision->100,AccuracyGoal->50]
```

実数解だけが欲しい場合は、Reals を指定する。

```
f[x_, y_] := x^4 - x y + y^4
sol = Solve[{D[f[x, y], x], D[f[x, y], y]} == {0, 0}, {x, y}, Reals]

(, Reals を指定しないと虚数解も表示される。)
```

なお、未知数の消去は Eliminate[]

5.19 極限

その名もずばり Limit[] という関数があります。その際 ∞ を意味する “Infinity” が使えることに注目。

```
Limit[Sin[x] / x, x-> 0]
Limit[(x^2 + 2 x + 3)/(3 x^2 + 2 x + 1), x->Infinity]
```

いわゆる片側極限 $\lim_{x\uparrow a} f(x)$, $\lim_{x\downarrow a} f(x)$ も計算できます。

```
Limit[Tan[x], x-> Pi/2, Direction -> 1]    下から (左から) の極限
Limit[Tan[x], x-> Pi/2, Direction -> -1]   上から (右から) の極限
Limit[Tan[x], x-> Pi/2]                    (結果はどうなるでしょう？)
```

Mathematica の古いバージョンは、Limit[ ] に対して少々信用できない結果を返すことがありました。

以前は (私は Mathematica と 30 年近いつきあいです)、Mathematica は結構おかしな結果を返すことも多かったのですが、最近はかなりちゃんとしてきていて、かえって危なくなったような気がします (結果を無批判に受け入れる人がいるのではないか？ということです)。計算結果をうのみにしないで検討する態度が大事だと思います。

5.20 リストとその応用 (行列、ベクトル)

急いでいるときは、使用頻度の高いベクトル、行列演算、Table[] だけでもチェックすると良い (？)。時間があるときは、じっくり試しながら読むことを勧めます。

- いわゆる中括弧 { と } の中にカンマで区切って複数のものを並べて作った「リスト」というデータ構造があります。

  ```
  list = {1,2,3}   1, 2, 3 という 3 つの要素からなるリスト list を定義
  ```

- 関数の多くはリストに対して作用します。

  ```
  Log[{a,b,c}]
  N[{1/2,1/3,1/4}]
  Sin[Pi/{1,2,3,4,5,6,7,8}]
  ```

  自分が作成した関数も (知らぬ間に[12]) リストに対して作用するようになっていたりするので、意外と便利なことがあります。

- 結果をリストとして返す関数は多いので (方程式の解が複数個ある場合とか、固有値、固有ベクトルを求める関数など)、リストの中から特定の要素を取り出す Part[リスト, 番号] や リスト [[番号]] は重要です。

[12] 自分が作成した関数で、Mathematica の組み込み関数を使っていて、それがリスト対応だったりすると、自作の関数も自然とそうなったりすることがあります。

```
list = {1,2,3}
Part[list,2] または list[[2]]                          list の第 2 要素
a={{1,2},{3,4}}                                        行列はこんなふうにリストで表現
Part[a,1] または a[[1]]                                a の第 1 行
Part[a,1,2] または a[[1,2]] または a[[1]][[2]]         a の (1, 2) 成分
e=Eigenvalues[a]                                       行列の固有値を計算
lam1 = e[[1]]; lam2 = e[[2]]                           それぞれ変数に代入
{lam1,lam2}=e                                          同時に代入 (注目!)
Remove[list,a,e,lam1,lam2]
```

- リストは集合、ベクトル、行列、テンソル、積分やプロットなどの範囲指定、関数の引数のグループなどを表すのに使われます。
Total[]	リストの要素の合計
Mean[]	リストの要素の平均
Max[]	リストの要素の最大値
Min[]	リストの要素の最小値

- 集合的な演算をするための関数として、

Union[]	合併 $A \cup B$
Intersection[]	共通部分 $A \cap B$
Complement[]	差集合 $A \setminus B$
Subsets[]	部分集合すべて (いわゆる冪集合)
DeleteDuplicates[]	リストから重複を省く

  などがあります。Complement[] が補集合だけでなく、差集合の計算に使えることに注意。

- ベクトル的な利用法として

```
{1,2,3}+{a,b,c}
2 {1 3 5}
{1,3,5} / 3
{1,2,3} .  {3,4,5}   内積
{1,2,3} {2,3,4}      成分ごとの積
{1,2,3} / {2,3,4}    成分ごとの商
```

- 行列演算もできます。

```
A={{a11,a12,a13},{a21,a22,a23},{a31,a32,a33}}
y={y1,y2,y3}
A . y                                                  行列とベクトルの積
                                                       (ドットが必要なことに注意)
```

行列用の関数としては、行列式 Det[], 転置行列 Transpose[], 逆行列 Inverse[], 固有値 Eigenvalues[], 固有ベクトル Eigenvectors[] などがあります。

{p,j}=JordanDecomposition[a] とすると、a の Jordan 標準形 j が求まります (p^{-1}a p が j です)。

- リストを生成する Table[] という関数も慣れると便利です (他のプログラミング言語では繰り返し処理を行って解決するような問題を、Table[ ] でリストを生成してお仕舞い、ということが多い)。

```
Table[i^2, {i,6}]
Table[Sin[n Pi/5], {n,0,4}]
Table[x^i+2i, {i,5}]
Table[Sqrt[x], {x, 0, 1, 0.25}]
Table[x^i+y^j, {i,3}, {j,2}]
Table[PrimeQ[2^2^p+1],{p,8}]
Table[{2^2^n+1,PrimeQ[2^2^n+1]},{n,6}]
```

単純な場合は Range[] が便利かも。

```
Range[10]       Table[n,{n,1,10}] と同じ
Range[3,10]     Table[n,{n,3,10}] と同じ
Range[1,101,2]  1 から 101 までの奇数
```

- Lisp 言語風の関数として First[], Rest[] などがあります (car, cdr)。

```
a={1,2,3,4,5}
First[a]
Rest[a]

Last[a]
Remove[a]
```

- その他 Length[{a,b,c,d,e}], MemberQ[{a,b,c,d,e,f},a], Count[{a,b,a,b,a,b},a], Reverse[], Sort[], RotateLeft[], RotateRight[] 等々。

- ReadList[] という関数を使って、外部ファイルからリストに読み込めます (この項工事中)。

```
ReadList["ファイル名", Number]
ReadList["ファイル名", Number, RecordLists->True]
ReadList["!外部コマンド名", Number]
ReadList["!外部コマンド名", Number, RecordLists->True]
```

“square.data” を

```
1 1
2 4
3 9
4 16
```

という内容のファイルとする時、以下のコマンドで何が起こるか？

```
ReadList["square.data", Number]
ReadList["square.data", Number, RecordLists -> True]
```

付録の A (「その他使う可能性の高いリスト処理用の関数」) も参考にして下さい。

5.21 微分方程式を解く DSolve[], NDSolve[]

(ただ今工事中です。現在の Mathematica は、一部の偏微分方程式や、境界値問題も解けるようになっていますが、ここでは常微分方程式やその初期値問題の解き方のみ紹介します。2019/11/3 LaTeX2HTML が ’’ を二重引用符 ’’ にすることがあって、コードが壊れていたのを直す。)

常微分方程式を解くコマンド DSolve[] があります。
一つの使い方は、次のように y[x] について解くというものです。

```
DSolve[y'[x]==a y[x],y[x],x]
```

結果は {{y[x]->Exp[a x]C[1]}} となります。つまり

$$\frac{dy}{dx} = ay$$

の一般解は $y = C_1 e^{ax}$ である、ということですね。ここから何かするには、演算子 /. を使うことになるでしょう。
次の例では、単振動の方程式の初期値問題

$$x''(t) = -x(t), \quad x(0) = 1, \quad x'(0) = 2$$

を解いています。任意定数は残っていないので、結果をグラフで表示できます。(少々強引に) 関数 x を上書き定義してから、もう一度グラフを描いています。

```
Remove[x,t]
sol=DSolve[{x''[t]==-x[t], x[0]==1, x'[0]==2},x[t],t]
Plot[x[t] /.  sol, {t,-10,10}]                          ← グラフを描く (分かり辛い？)

x[t_]:=Evaluate[x[t] /.  sol[[1]]]                      ← 関数 x を定義
                                                        Evaluate[] を使うのが一つのポイ
Plot[x[t], {t,-10,10}]                                  ← グラフを描く (分かりやすい？)
```

連立の微分方程式を解くことも出来ます。

$$x'(t) = -y(t), \quad y'(t) = 2x(t), \quad x(0) = 1, \quad y(0) = 2$$

という問題は次のようにして解けます。

```
Remove[t,x,y]
sol=DSolve[{x'[t] == -y[t], y'[t]==2x[t], x[0]==1, y[0]==2},{x[t],y[t]},t]
ParametricPlot[{x[t],y[t]}/.sol,{t,0,2Pi}]
```

もう一つのやり方は、y[x] でなく、y について解くと言うものです。結果は (説明は省略しますが) 「純関数」 {{y->Function[{x},2Exp[a x]]}} になります。つまり y に /. したときに、y[] という関数が出来るだけでなく、y'[x] や y[1] などの式が使えるのが嬉しい点です。

```
Remove[a,x,y]
sol=DSolve[{y'[x]==a y[x],y[0]==2},y,x]
y[x] /.  sol                              ← この段階では差がない
y'[x]-a y[x] /.  sol                      ← こういうことが出来る (解であることの確かめ)
```

こちらでも初期値問題が解けます。

```
Remove[t,x,x0,v0]
s=DSolve[{x''[t] == - omega^2 x[t], x[0] == x0, x'[0] == v0},x,t]
```

微分方程式は、いわゆる式変形では解けない場合が多いことは良く知られています。そのため数値的に近似解を求める方法が色々と開発されています。NDSolve[] は数値的に微分方程式の初期値問題を解くコマンドで、結果は離散的な点での関数値を近似的に求めたものになります。計算していない点での値は補間により簡略計算することになります (こういうものを、Mathematica では InterpolatingFunction (直訳すると補間関数？) と呼んでいるようです)。

次の例では、いわゆる単振り子の方程式

$$\theta''(t) = -\sin\theta(t)$$

を解いています。良く知られているように (?)、この解の表示には楕円関数などを使う必要があり (http://nalab.mind.meiji.ac.jp/~mk/labo/text/furiko/ とか)、ちょっと難しめのせいか、Mathematica も DSolve[] では解いてくれません。そこで NDSolve[] の出番となります。

振り子の方程式を解く (1)

```
Remove[t,x]
sol=NDSolve[{x''[t]==-Sin[x[t]],x[0]==1,x'[0]==0}, x, {t,0,10}]
Plot[x[t] /.  sol, {t,0,10}]
```

以前は Plot[Evaluate[x[t] /. sol], {t,0,10}] とかする必要があったのですが、今では Evaluate[] は不要になったようです。ただし、オンライン・ヘルプには Evaluate[] するように書いてあるし、今でも NDSolve[] とプロットを同時にやる場合には、Evaluate[] した方が無駄なく効率的に計算できます (逆に言うと、そうでない場合の処理は昔と変わったのかな？)。そこら辺が気になるならば、Timing[] で時間の計測をして比較すると良いでしょう。

連立 1 階の方程式

$$x' = y(t), \quad y'(t) = -\sin x(t)$$

に直して解いてみる。

振り子の方程式を解く (2)

```
Remove[t,x,y]
sol=NDSolve[{x'[t]==y[t], y'[t]==-Sin[x[t]],x[0]==1,y[0]==0}, {x,y}, {t,0,10}]
ParametricPlot[{x[t],y[t]}/.sol, {t,0,10}]
```

おまけ (未整理)

```
Remove[t,x,y,t2]
Manipulate[
 ParametricPlot[Evaluate[{x[t2], y[t2]} /.
   NDSolve[{x'[t] == y[t], y'[t] == -Sin[x[t]], x[0] == x0,
     y[0] == 0}, {x, y}, {t, 0, 20}]], {t2, 0, 20},
  PlotRange -> {{-1.2 Pi, 1.2 Pi}, {-Pi, Pi}}], {x0, 0, Pi}]
```

とか

```
Remove[t,x,y,x0,t2,a]
my[x0_] :=
 ParametricPlot[Evaluate[{x[t2], y[t2]} /.
   NDSolve[{x'[t] == y[t], y'[t] == -Sin[x[t]], x[0] == x0,
     y[0] == 0}, {x, y}, {t, 0, 20}]], {t2, 0, 20},
  PlotRange -> {{-1.2 Pi, 1.2 Pi}, {-Pi, Pi}}]
Show[Table[my[a Pi], {a, 0.1, 0.9, 0.1}]]
```

とか

```
Remove[t,x,y,a,b]
sol = Table[{x[t], y[t]} /.
       NDSolve[{x'[t] == y[t], y'[t] == -Sin[x[t]], x[0]==a*Pi, y[0]==b},
        {x, y}, {t, -20, 20}], {b, -2.4, 2.4, 0.2}, {a, -2, 2, 2}];
g = ParametricPlot[sol, {t, -20, 20}, PlotRange -> {{-3 Pi, 3 Pi}, {-Pi, Pi}}]
```

ちなみにエネルギー保存則の式で描くと

```
ContourPlot[
 1/2 y^2 - 9.8 Cos[x] - 9.8, {x, -2 Pi, 2 Pi}, {y, -10, 10},
 Contours -> Table[x, {x, -40, 40, 5}], AspectRatio -> 0.5]
```

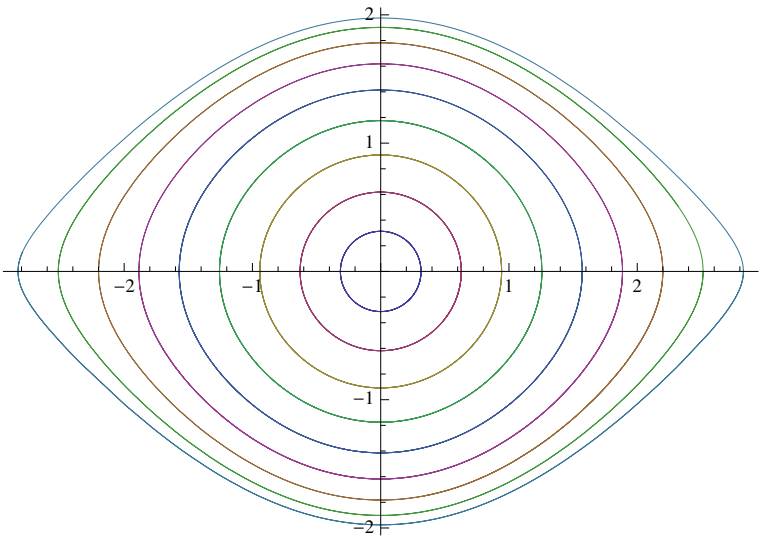


図 9: 振り子の運動の相図: $\theta(0) = 0.1\pi, \cdots, 0.9\pi$, $\theta'(0) = 0$ (そっと手放す)

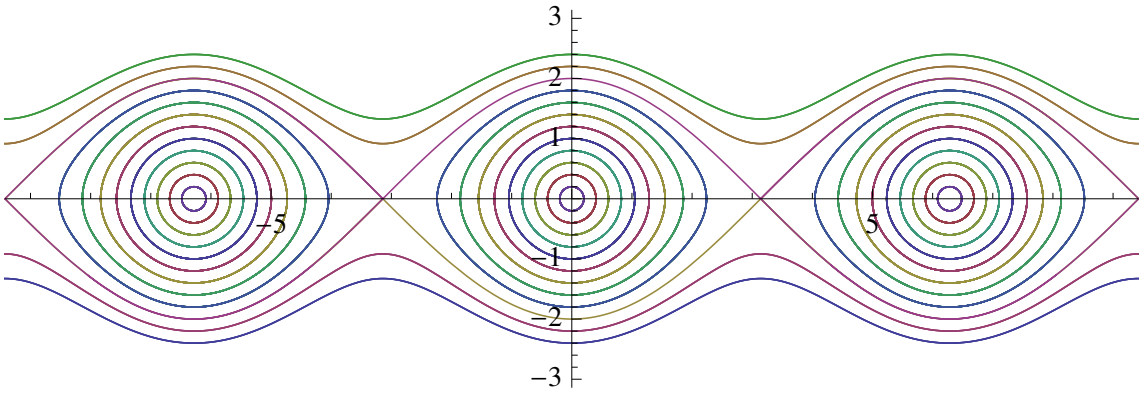


図 10: 良く本に載っている図

おまけ 2: 放物運動

```
?? Global‘*

Remove["Global‘*"]

v=20;
theta=45;
ParametricPlot[Evaluate[{x[t2], y[t2]} /.
  NDSolve[{x’ ’[t] == 0, y’ ’[t] == -9.8,
    x[0]==0, y[0]==0, x’[0]==v*Cos[theta Degree], y’[0]==v*Sin[theta Degree]},
     {x, y}, {t, 0, 5}]],
   {t2, 0, 4}]
```

もちろん

```
DSolve[{x’ ’[t] == 0, y’ ’[t] == -g,
  x[0] == 0, y[0] == 0, x’[0] == v Cos[theta], y’[0] == v Sin[theta]},
  {x, y}, t]
```

として、式変形で解くことも出来る。

5.22 乱数

乱数関係

```
RandomInteger[]
RandomReal[]
RandomChoice[]
RandomPrime[]
```

5.23　仮定を加えて

```
Integrate[1/(1 + e Cos[x])^2, {x, 0, 2 Pi},  Assumptions -> e > 0 && e < 1]
```

あるいは

```
Assuming[e > 0 && e < 1, Integrate[1/(1 + e Cos[x])^2, {x, 0, 2 Pi}]]
```

(結果は $\frac{2\pi}{(1-e^2)^{3/2}}$ となる。$0 < e < 1$ という仮定をつけないと、場合分けを含んだ複雑な式が返ってくる。あるいは、非力なマシンだと、なかなか実行が終わらない。)

仮定を加えるだけでなく、簡単にするよう指定しないといけない場合もある。

```

```

(`Simplify[]` しないと単に $\cos n\pi$ という結果を返すだけ。`Simplify[]` して $(-1)^n$ という結果を返してくれる。)

6　基本的なプログラミング機能、特に制御構造

Mathematica は、これまでに述べてきたような電卓的な使用法、つまり一つ一つの計算の指示を人間が手で入力する仕方でも十分役立ちますが、通常のプログラミング言語にひけを取らないプログラミングの機能を持っています。これまでに説明していないもので、プログラミングに必須 or 役立つものを説明します。

- 複数の文を並べること
- 条件判断 (真偽の判定)
- 条件判断に基づく分岐
- 繰り返し処理用の専用メカニズム
- まとまった処理に名前をつけて一つの単位とする (関数を作る)

このうちの最後のものは節を改めて説明することにして、ここでは最初の 4 つについて説明する。

6.1　複数の文を並べる

(1) `";"` で区切って並べることができる。

(2) “(”, “)” で括ることもできる。

6.2　条件判断 (論理式)

関係演算子、論理演算子は C 言語のそれに良く似ている:

`a == b`	等しいか？
`a != b`	等しくないか？
`a < b`	a は b より小さいか？
`a > b`	a は b より大きいか？
`a <= b`	a は b より小さいか、等しい？
`a >= b`	a は b より大きいか？
`&&`	「かつ」.
`||`	「または」.
`!`	「否定」.
`Positive[]`	正かどうか
`Negative[]`	負かどうか

論理式の値は、False = 偽, True= 真, である。次の例を試してみること。

```
1+1 == 2
2^3 == 7
1 < 2 < 3                                    (こういう 3 数の比較が出来るプログラミング言語は珍し
2 != 3
LogicalExpand[(p || q) && !(r || s)]
Remove[p,q,r,s]
```

実は x1<x2<x3 のような式は、普通の数学に現れる式 $x_1 < x_2$ かつ $x_2 < x_3$ という意味に解釈されるので、本当は C 言語風ではない。

6.3 条件判断による分岐

If[*test*, *then-statement*, *else-statement*] とすると分岐が実現できる。例えば

```
If [1+1==2, Print["Yes, you are right."], Print["No, you are wrong."]]
```

たくさんの場合に分けるには、Which[] が便利である。

```
myfunc[t_]:=Which[t < -1, Sqrt[-(t + 1)],
                  t < 0, Sqrt[t + 1],
                  t < 1, Sqrt[1 - t],
                  True, Sqrt[t - 1]]
```

このような関数定義をする場合、式 /; 条件式 を使うという手もあるが、自分の使用経験では Which[] の方が良い場合が多かった。例えば

```
a0[x_] := Which[x > 0, x, True, -x]
a1[x_] := x /; x > 0
a1[x_] := -x /; x <= 0
Plot[a0[x], {x, -1, 1}]
Plot[a1[x], {x, -1, 1}]
Plot[a0'[x], {x, -1, 1}]
Plot[a1'[x], {x, -1, 1}]
```

6.4 繰り返し構文

計算機のプログラムで広い意味の繰り返しは重要である。それを実現するために、再帰的関数 (手続き) 定義や、繰り返しの構文が用意されている。ここでは繰り返しの構文の紹介をする。

Do 関数 使い方は "`Do[`*statement*, *iterator*`]`". Fortran の do 文、C の for 文、BASIC の FOR NEXT 構文、Pascal の for 文に似ている。*iterator* (繰り返し指定) で指定しただけ *statement* (文) を繰り返して実行する。

```
Do[Print[i!], {i,5}]              i=1,2,3,4,5
Do[Print[2^i], {i,0,5}]           i=0,1,2,3,4,5
Do[Print[I^i], {i,0,10,3}]        i=0,3,6,9
r=1; Do[r=1/(1+r), {100}]; r      100 回
Do[Print[i], {i,2a,4a,a}]         i=2a,3a,4a
Do[Print[r], {r,0.0,3.5}]         r=0.0,1.0,2.0,3.0
Do[Print[{i,j}], {i,3}, {j,i}]    do i=1,3
                                    do j=1,i
                                      Print {i,j}
                                    end do
                                  end do
```

While 関数 使い方は "`While[`*test*, *statement*`]`". C や Pascal の while 文に似ている。*test* が真である間だけ *statement* を繰り返す。

```
i=1; While[i <= 10, Print[i]; i++]
```

この例では "`i <= 10`" が *test*, "`Print[i]; i++`" が *statement* になっている。

For 関数 使い方は "`For[`*statement1*, *test*, *statement2*, *statement*`]`". C の for 文に似ている。

```
For[i=1, i <=10, i++, Print[i, " ", i^2]]
```

繰り返しの指定 Do 文で使われている iterator は、あちこちで使われる。和 $\sum$ を計算する `Sum[]` や乗積 Π を計算する `Product[]`、表 (値を並べたリスト) を作る `Table[]` など。

`Sum[i, {i,1,10}]`	$i = 1, 2, \cdots, 10$
`Product[x-i, {i,0,5}]`	$i = 0, 1, \cdots, 5$
`Product[e, {e,x,x-5,-1}]`	$e = x, x-1, x-2, x-3, x-4, x-5$
`Table[i!, {i,5}]`	$i = 1, 2, 3, 4, 5$

`Print[]` の結果は表示されるだけで、後には残らないので、残して解析したい場合は、`Table[]` を使うのがお勧めである (このあたりは C 言語のプログラミングの発想とは異なる)。

7　簡単なユーザー関数の定義の仕方と応用例

実は Mathematica のプログラムは (ユーザーが定義する) 関数の集合という形になる。

7.1　関数定義

百聞は一見にしかず。$f(x) = x^2$ なる関数 f は、

```
f[x_] := x^2
```

または

```
f[x_] = x^2
```

で定義出来る。つまり、

関数を定義する文法

関数名 [引数名_] := 式
または
関数名 [引数名_] = 式

ということである。アンダースコア (下線) '_' が必要である (また "f[名前_ 型名] := 式" のように型名を指定することも出来る)。

:= はいわゆる遅延評価をすることの指定である。左辺 := 右辺 とすると、右辺の式を評価しないまま、左辺に代入し、それ以降、左辺が出現して評価されるごとに (関数定義に使った場合は、関数が呼び出されるごとに)、右辺に置き換えられて、それから評価される。(これについては、??:= として、Mathematica のヘルプを読むことを勧める。あるいはチュートリアルの中の「即時的な定義と遅延的な定義」)

関数を再定義すると古い定義は消えてしまう。

上のように定義しておくと、以下のように使える (細かい説明は不要であろう)。

```
f[4]
f[a+1]
f[3x+x^2]
```

もちろん関数 "f" の定義が見たい場合は、"?f", または "??f" とする。

```
??f
```

多変数の関数も定義できる。

```
f[x_,y_] := (x^2 - y^2) / (x^2 + y^2)
```

7.2 例 1: 普通の関数らしい使い方

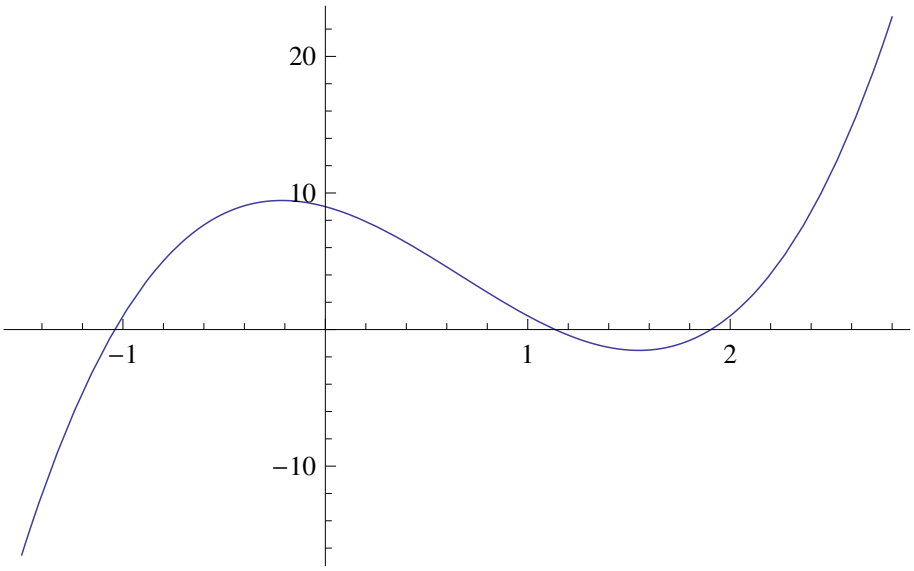
関数 $f(x) = 4x^3 - 8x^2 - 4x + 9$ の増減を調べよう。

高校数学？

```
f[x_]:=4x^3-8x^2-4x+9
Plot[f[x],{x,-4,4}]                {-1.5,2.5} くらいが良いか？
s=Solve[f[x]==0,x]
N[s,20]
sp=Solve[f'[x]==0,x]
f[x] /. sp
Simplify[%]
Remove[f,s,sp,x,y]
```

x 軸との交点の x 座標は、(3 次方程式の 3 つの相異なる実根なので、不還元の場合で、Cardano の公式を使って解くと虚数の 3 乗根が現れる) $-1.0403\cdots, 1.13540\cdots, 1.90489\cdots$.

$x = \dfrac{2-\sqrt{7}}{3}$ で極大で、極大値は $\dfrac{107+56\sqrt{7}}{27} \fallingdotseq 9.45045$. $x = \dfrac{2+\sqrt{7}}{3}$ で極小で、極小値は $\dfrac{107-56\sqrt{7}}{27} \fallingdotseq -1.52452$.



(2 変数関数の増減を調べるのは研究課題としたい。付録 C.2 にプログラム例がある。)

7.3 例 2: 数列

うっかりする人がいるかもしれないが、数列というのは、変数が自然数 (あるいは整数) の関数である。

次の例では、

$$a_0 = 1, \quad a_1 = 1, \quad a_n = a_{n-1} + a_{n-2} \quad (n \geq 2)$$

で定義される Fibonacci 数列を第 100 項まで表示している。

```
a[0]=1;a[1]=1
a[n_]:=a[n]=a[n-1]+a[n-2]                    ここまで（2 行）が関数定義

a[10]                                        a[10] を計算して結果を表示
??a                                          a の中身を見る
Table[a[n],{n,100}]                          a[1],...,a[100] を表示
```

(細かい工夫であるが…) 2 行目は、単に a[n_]:=a[n-1]+a[n-2] としても正しく計算する関数になるが、計算の効率が非常に低い。a[n_]:=a[n]=a[n-1]+a[n-2]] とすることによって、計算結果を記憶しておくと、効率が向上する。

> a[-1] を計算させようとすると、以前の Mathematica では暴走した。最近では滅多に暴走しなくなったが、万一暴走させた (あるいは計算がなかなか終らない) 場合は、[評価] → [評価を放棄 (A)] で強制終了させる (英語では [Kernel] → [Abort Evaluation(A)])

この例では、遅延評価 := を使うことが必須である (a[n_]=a[n]=a[n-1]+a[n-2] は通らないし、a[n_]=a[n-1]+a[n-2] では遅くて使い物にならない)。

また、(当然のことであるが) 0 以上の整数でないものを引数に与えると (暴走はしなくなったが) 再起呼び出しが止まらず、エラーになる。

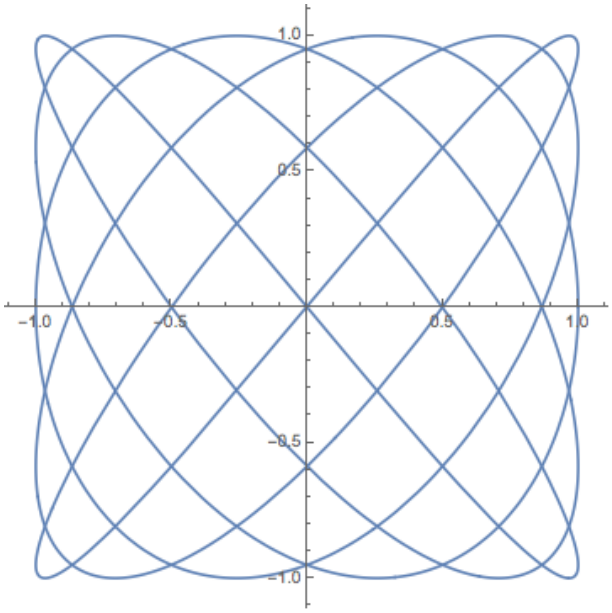
7.4 例 3: 手続き風の使い方

Mathematica の文法的には「関数」ということになるが、返す値に直接興味を持たれない “手続き” として利用することもある。

いわゆるリサージュ (Lissajous) 図形を描くために、関数を定義して、それを利用した例をあげる。

```
lis[m_,n_]:=ParametricPlot[{Cos[m t],Sin[n t]},{t,0,2Pi}]
lis[1,1]
lis[2,1]
lis[5,6]
```

(これも遅延評価 := を用いることが必須である。)



もう一つ引数を増やすべきか？

```
lis2[m_,n_,ph_]:=ParametricPlot[{Cos[m t],Sin[n t+ph]},{t,0,2Pi}]
lis2[3,4,Pi/2]
```

```
Remove[lis,lis2,m,n,ph,t]
```

7.5 Block[], Module[]

(工事中)

関数の中である程度複雑な計算をする場合、変数を使用する必要が生じるでしょうが、局所的な変数を使用するようにしないと思わぬ副作用が生じます。

次の例は、局所変数 i と p (初期値として 1 を代入) を用いて、階乗 $n!$ を計算するための関数を定義したものです (もちろん Mathematica には、階乗を計算する演算子 ! があるので、こういう関数を作る必要はありませんが)。

```
fact[n_]:=Module[{i,p=1}, For[i=1, i<=n, i++, p=p*i]; p]
```

C 言語だったら次のように書くところです。

```
int fact(int n)
{
  int i, p = 1;
  for (i = 1; i <= n; i++)
    p *= i;
  return p;
}
```

7.6 少し凝った例: 円周率の計算

「円周率の計算の歴史」[13] という文書もある。興味があれば見てみよう。

arctan の Maclaurin 展開

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{2n+1}$$

を用いて円周率 π を表す色々な無限級数を作ることができる。

有名なのが $x = 1$ とおいてできるマーダヴァ・グレゴリー・ライプニッツ級数である:

$$\frac{\pi}{4} = \arctan 1 \quad \text{より} \quad \pi = 4 \arctan 1 = 4 \left(\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots \right).$$

これは印象的であるが、収束は極端に遅く、π を計算する目的にはまったく使い物にならない。

絶対値の小さい x を選ぶと実用的な公式が得られる。例えば $\tan \pi/6 = 1/\sqrt{3}$ より

$$\begin{aligned} \pi &= 6 \arctan \frac{1}{\sqrt{3}} = 6 \sum_{n=0}^{\infty} \frac{(-1)^n}{\left(\sqrt{3}\right)^{2n+1} (2n+1)} = 2\sqrt{3} \sum_{n=0}^{\infty} \frac{1}{(-3)^n (2n+1)} \\ &= 2\sqrt{3} \left(1 - \frac{1}{3 \cdot 3} + \frac{1}{3^2 \cdot 5} - \frac{1}{3^3 \cdot 7} + \frac{1}{3^4 \cdot 9} - \cdots \right) \end{aligned}$$

Abraham Sharp は 210 項まで足し合わせて、小数点以下 100 桁以上の円周率の値を求めたという。

$$s_n := 2\sqrt{3} \sum_{k=0}^{n} \frac{1}{(-3)^k (2k+1)}$$

とおき、s_n を計算する関数を作ってこのことを確かめよ。

[13] http://nalab.mind.meiji.ac.jp/~mk/syori2-2011/jouhousyori2-2011-06/node10.html

```
s[n_]:=2 Sqrt[3]Sum[1/((-3)^k*(2k+1)),{k,0,n}]
s210=s[210]
N[s210,200]
s210-Pi

ns[n_]:=N[s[n],1000]
match[n_]:=-1.0*Log[10,Abs[ns[n]-Pi]]
ListPlot[Table[match[n],{n,210}]]
Remove[k,match,n,ns,s]
```

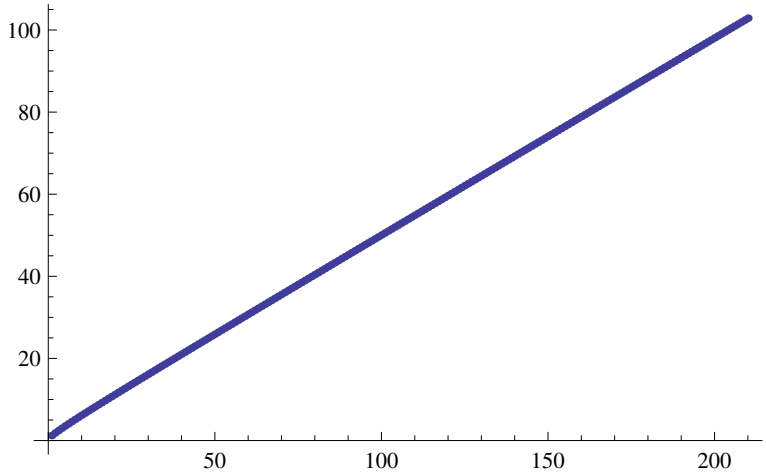


図 11: Sharp の方法．何項取れば何桁合うか．

L. Euler (超有名数学者) は次の公式を 1737 年に得た。

$$\frac{\pi}{4} = \arctan\frac{1}{2} + \arctan\frac{1}{3}, \quad \pi = 20\arctan\frac{1}{7} + 8\arctan\frac{3}{79}.$$

John Machin (1680–1752, ロンドン大学天文学教授) は

$$\frac{\pi}{4} = 4\arctan\frac{1}{5} - \arctan\frac{1}{239}$$

を用いて 100 桁の値を計算した。この公式は以後多くの人達に採用され続ける。人手での π の計算の記録としては、William Shanks (1812–1882) が 1873 年に 707 桁計算した (527 桁までが正しかった) のが最高だが、彼もこの公式を使った。

C. F. Gauss (数学界の巨人) は 1863 年に以下の公式を得た。

$$\frac{\pi}{4} = 12\arctan\frac{1}{18} + 8\arctan\frac{1}{57} - 5\arctan\frac{1}{239},$$

$$\frac{\pi}{4} = 12\arctan\frac{1}{38} + 20\arctan\frac{1}{57} + 7\arctan\frac{1}{239} + 24\arctan\frac{1}{268}.$$

前者はおそらく 3 項の arctan で表される公式のうちで最も効率が高い (らしい)。

2005 年現在の最高記録は、2002 年 12 月、金田康正、後保範（うしろやすのり）等のグループが達成した 1 兆 2400 億桁というものだが、それは高野喜久雄の公式

$$\frac{\pi}{4} = 12\arctan\frac{1}{49} + 32\arctan\frac{1}{57} - 5\arctan\frac{1}{239} + 12\arctan\frac{1}{110443}$$

による (この話は WWW で検索すれば色々ヒットするので省略)。

高野喜久雄氏は詩人として有名で、ウィキペディアにも載っている (それによると高校で数学の教員をしていたとか。合唱曲「水のいのち」の作詞者。)。2002 年の計算については、ご自身の WWW サイトで短いコメントを残している。今はそのサイトはなくなっているが、Internet Archive には残っている (「気になる談話室」[14])。

7.7 遅延評価を使うか使わないか

個人的には、普段は関数定義をするときあまり考えず、:= を使っている。
:= を使わないとまずい場合として、すぐ思い浮かぶのは

- `lis[m_,n_]:=ParametricPlot[{Cos[m t],Sin[n t]},{t,0,2Pi}]` のようなグラフィックス描画を目的とした関数
  — 呼び出したときに描いてもらわないと意味がない。
- 再起呼び出しをするような関数

:= が使えない場合として、Mathematica のチュートリアルには、次のようなことがあげられている。計算の結果として得られた式の中のパターンに、何かを「代入」してどうなるか調べる場合、/. を使った一時的代入で済むけれど、関数を定義する場合には := でなく = を使う。要するに =% は通るけれど、:=% は通らない、ということだよね。

チュートリアルに載っている例

```
D[Log[Sin[x]]^2, x]
dlog[x_] = %
```

まあ、こんなのは `D[Log[Sin[x]]^2, x]` の結果 (`2 Cot[x] Log[Sin[x]]`) をコピペして、

```
dlog[x_] := 2 Cot[x] Log[Sin[x]]
```

とすれば良いだけのような気もするけど。

個人的な経験では、重い計算が必要だが、それは 1 回だけやっておけば済むような場合に、:= を使った関数定義の右辺に書いてしまうと、効率が落ちて困った経験がある。でもそれは、:= と = の問題というよりは、もっと別の問題かもしれない。

[14] https://web.archive.org/web/20010802023531/http://www.asahi-net.or.jp/~yp5k-tkn/danwa.html

8　Mathematica のグラフィックス機能

お急ぎの方は、8.1「まずはやってみよう」 だけでも。

8.1　まずはやってみよう

8.1.1　1 変数関数のグラフ Plot[]

1 変数関数のグラフを描くのに Plot[ ] という関数が使える。

```
Plot[4x^3-8x^2-4x+9,{x,-2,3}]

Plot[4x^3-8x^2-4x+9,{x,-2,3},PlotRange->Full]

Plot[4x^3-8x^2-4x+9,{x,-2,3},PlotRange->Full,AspectRatio->1]

Plot[4x^3-8x^2-4x+9,{x,-2,3},PlotRange->Full,AspectRatio->Automatic]
```

Option の選び方は案外難しい。余裕のあるときに、Plot[] のヘルプを参考に色々試してみることを勧める。

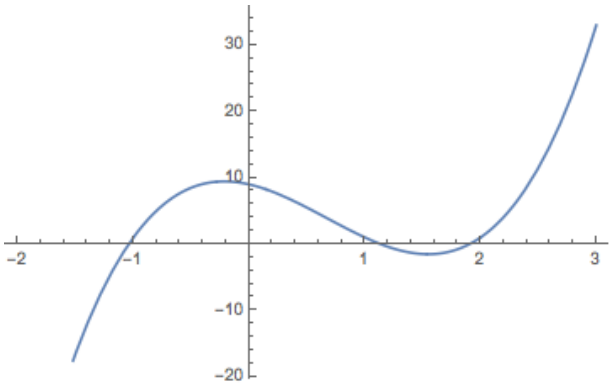


図 12: $f(x) = 4x^3 - 8x^2 - 4x + 9$ $(-2 \leq x \leq 3)$ のグラフ

グラフィックス関係のオプションについては、8.3.2 を参照すること。

8.1.2　2 変数関数のグラフ Plot3D[]

2 変数関数のグラフを描くのに Plot3D[ ] という関数が使える。

```
Plot3D[x^2-y^2,{x,-1,1}, {y, -1, 1}]
```

グラフをマウスでドラッグして動かすことができる。ぜひ試してみること。

複数の関数のグラフを同時に描くことも出来ます (関数のリストを渡す)。

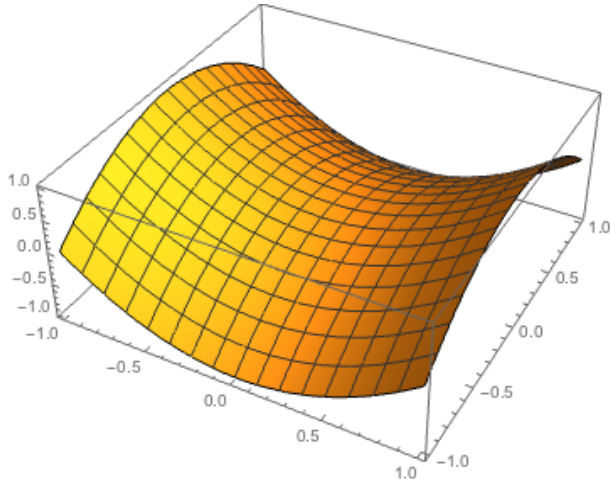


図 13: $f(x,y) = x^2 - y^2$ $(-1 \leq x \leq 1,\ -1 \leq y \leq 1)$ のグラフ

```
Plot3D[{x^3+y^3-3x y,0},{x,-2,2},{y,-2,2}]
```

(平面 $z = 0$ を同時に描画することで、関数 $z = x^3 + y^3 - 3xy$ のグラフが分かりやすくなる。)

$z = \sqrt{1 - x^2 - y^2}$ $(x^2 + y^2 \leq 1)$ に基づき球面を描く。負数の $\sqrt{\ }$ が出て来ないように注意が必要である。以下の 2 つの例は少し工夫している (これは以前の Mathematica 用で、Version 7 は工夫をする必要がない？)。

```
Plot3D[Sqrt[Max[0, 1 - x^2 - y^2]], {x, -1, 1}, {y, -1, 1}]
```

(`Max[a,b]` は `a` と `b` の大きい方)

```
Plot3D[Sqrt[Boole[x^2+y^2<1]*(1-x^2-y^2)], {x, -1, 1}, {y, -1, 1}]
```

(`Boole[]` は真ならば 1, 偽ならば 0 を返す関数です。)

```
Plot3D[Sqrt[1-x^2-y^2], {x, -1, 1}, {y, -1, 1},
  RegionFunction->Function[{x,y,z},x^2+y^2<=1]]
```

極座標で表された関数のグラフ $z = f(r,\theta)$ を描く場合、$r = \sqrt{x^2 + y^2}$, $\theta = \arg(x + iy)$ という関係を用いると良い。例えば $f(r,\theta) = r^2 \cos 2\theta$ (実は $x^2 - y^2$) のグラフを描くには、以下のようにすれば良い。

```
  Plot3D[(x^2+y^2)Cos[2Arg[x+I y]],{x,-1,1},{y,-1,1}]
```

(こちらがいいのか？)

```
  Plot3D[(x^2+y^2)Cos[2ArcTan[x,y]],{x,-1,1},{y,-1,1}]
```

(実は $\cos n\theta$, $\sin n\theta$ が欲しいことが多いように思う。

```
c[x_, y_, n_] := Re[ComplexExpand[Re[(x + I  y)^n]]]/Sqrt[x^2 + y^2]^n
s[x_, y_, n_] := Re[ComplexExpand[Im[(x + I  y)^n]]]/Sqrt[x^2 + y^2]^n
```

とするのだろうか。)

2 次元グラフィックスの AspectRatio に相当するオプションとして、BoxRatios がある。BoxRatios -> {1,2,3} のように 3 つの数からなるリストを指定すると、3D グラフィックスの境界である直方体の辺の長さの比を与えることになる。

(Plot3D[ ], ListPlot3D[ ], ListPointPlot3D[ ] ではデフォールトで、{1,1,0.4} を使う。だからあんなに寸詰まりなんだ。ContourPlot3D[ ], ListContourPlot3D[ ] ではデフォールトで、{1,1,1} を使う。)

BoxRatios -> Automatic とすると、実際の座標の値に対応する直方体の長さの比を与える。例えば球を球として表示したいときなどに使う。

8.1.3 パラメーター曲線 ParametricPlot[]

平面内のパラメーター曲線を描くのに ParametricPlot[ ] という関数が使える。

```
  ParametricPlot[{Cos[3t], Sin[2t]}, {t, 0, 2Pi}]
```

空間内のパラメーター曲線を描くのに ParametricPlot3D[ ] という関数が使える。

```
  ParametricPlot3D[{Sin[t], Cos[t], t/4}, {t, 0, 4Pi}]
```

極形式の方程式 $r = f(\theta)$ を描く命令もある (自分で簡単に普通のパラメーター曲線に変換できるけれども)。

```
eps=0.8
PolarPlot[1/(1 + eps Cos[t]), {t, 0, 2 Pi}]
```

この例では、別にパラメーター ε が含まれているが、こういう場合にパラメーターを変えて描画するテクニックとして、Manipulate[] を紹介する。

```
Manipulate[PolarPlot[1/(1 + eps Cos[t]), {t, 0, 2 Pi}], {eps, 0, 2, 0.01}]
```

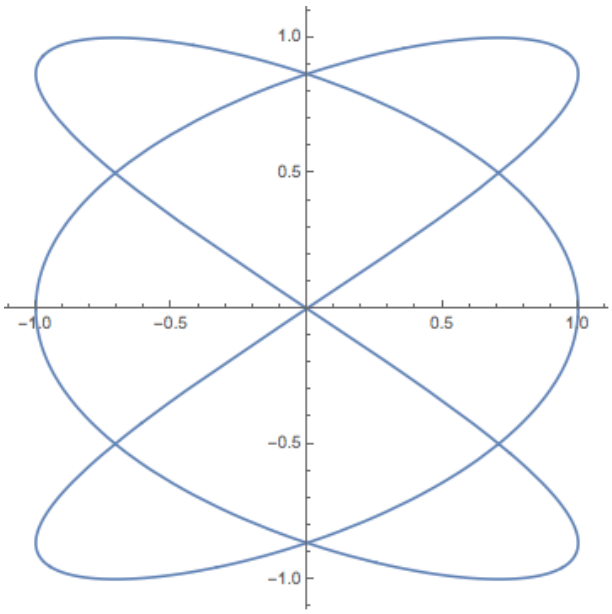


図 14: パラメーター曲線 $x = \cos 3t,\ y = \sin 2t\ (t \in [0, 2\pi])$

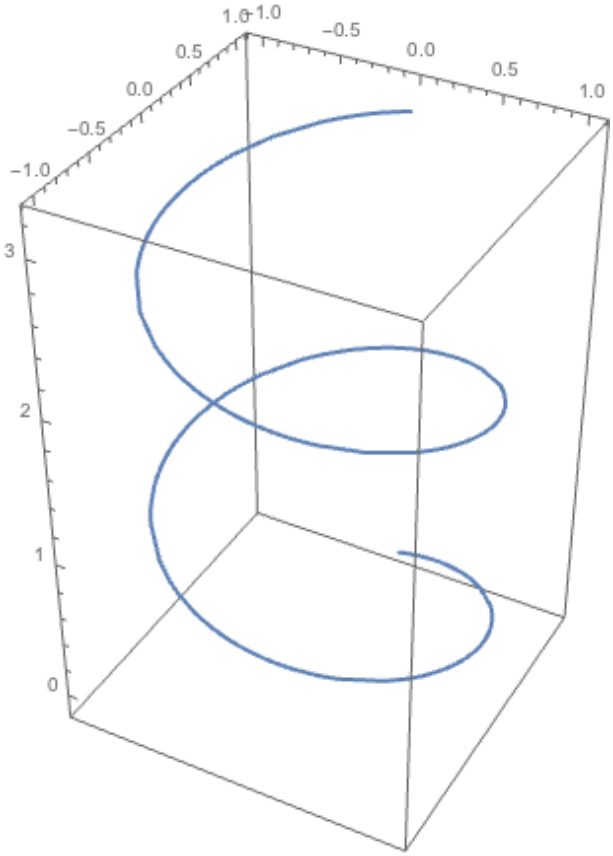


図 15: パラメーター曲線 $x = \sin t,\ y = \cos t,\ z = t/4\ (t \in [0, 4\pi])$

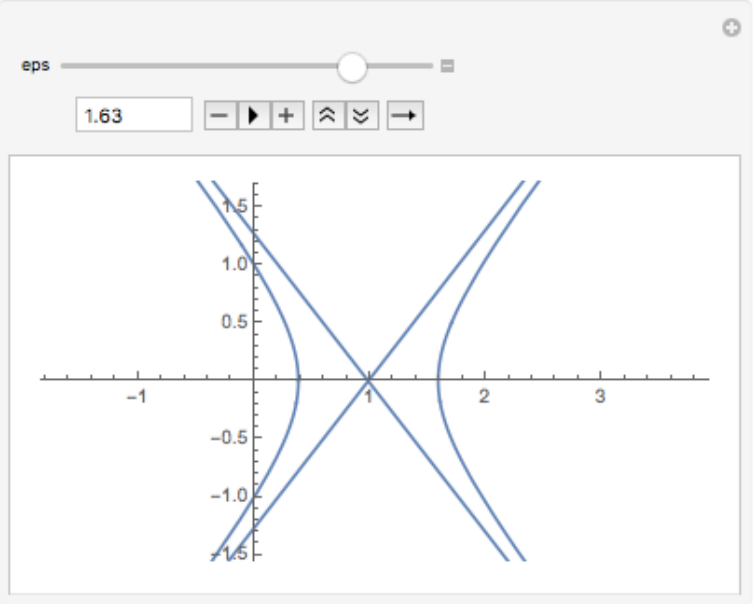


図 16: 円錐曲線 $r = \dfrac{1}{1 + e\cos\theta}$, $e = 1.63$

8.1.4 陰関数曲線 (平面内の) ContourPlot[]

2 変数関数のレベルセット ($f(x,y)=c$ を満たす (x,y) の集合) として与えられる陰関数曲線を描くには、ContourPlot[方程式, 範囲] が使える。

(以前は ImplicitPlot[] という命令を使っていた。)

```
ContourPlot[x^2 - y^2 == 1, {x, -2, 2}, {y, -2, 2}]
```

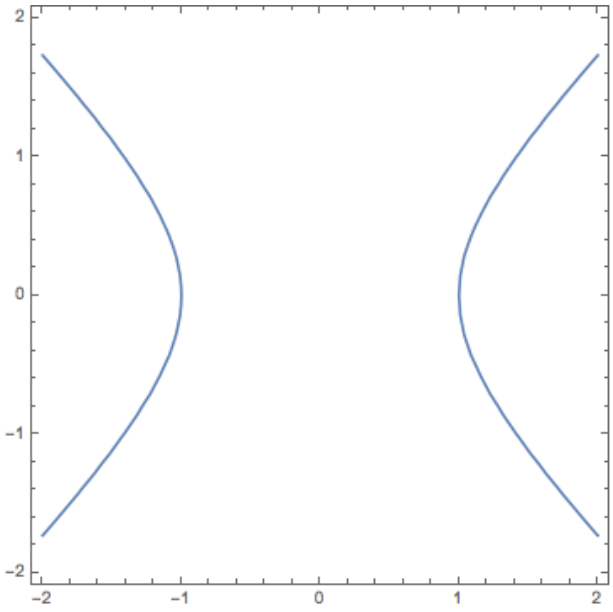


図 17: 方程式 $x^2-y^2=1$ で定まる曲線 (双曲線)

福岡大学の濱田先生に教わった (http://mathworld.wolfram.com/HeartCurve.html)

```
f[x_,y_]:=(x^2+y^2-1)^3-x^2y^3

ContourPlot[f[x,y]==0,{x,-2,2},{y,-2,2}]

ContourPlot[f[x,y],{x,-2,2},{y,-2,2}]

Plot3D[f[x,y],{x,-2,2},{y,-2,2}]

ContourPlot3D[(x^2+9y^2/4+z^2-1)^3-x^2 z^3-9y^2z^3/80==0,
              {x,-1.5,1.5}, {y,-1.5,1.5}, {z,-1,1.5}]
```

(結果は見てのお楽しみ)

なお等高線を太くしたり、色を指定するには、ContourStyle->Thick や ContourStyle->{Thick,Blue} のようにオプションを指定する。

バッドノウハウ？ 次の例では，$\cos m\pi x \cos n\pi y + \cos n\pi x \cos m\pi y = 0$ を描こうとしていて，本当は

```
plus[m_, n_] :=
 ContourPlot[
  Cos[m*Pi*x]* Cos[n*Pi*y] + Cos[n*Pi*x]*Cos[m*Pi*y]==0,
  {x, 0, 1}, {y, 0, 1},
  BoundaryStyle -> Black, Frame -> None, PlotPoints -> 100]
```

で出来るかと考えたが，なぜか境界 (正方形) が描かれなかったので，$\cos m\pi x \cos n\pi y + \cos n\pi x \cos m\pi y$ の等高線 (高さ 0 のもののみ) を描かせている。

```
plus[m_, n_] :=
 ContourPlot[
  Cos[m*Pi*x]* Cos[n*Pi*y] + Cos[n*Pi*x]*Cos[m*Pi*y],
  {x, 0, 1}, {y, 0, 1},
  BoundaryStyle -> Black, Frame -> None, Contours -> {0},
  ContourShading -> None, ContourStyle -> Thickness[0.002],
  PlotPoints -> 100]
```

(これは某学生の修士論文に掲載する図を作成するプログラムなので，オプションの指定が結構細かい。検討すると得るところがあると思う。)

8.1.5 パラメーター曲面 ParametricPlot3D[]

(2 年生は後期の「多変数の微分積分学 2」や「曲線と曲面」で詳しく学ぶことになりますが、2 つのパラメーターを持つ 3 つの関数の組 $x = \varphi_1(u, v)$, $y = \varphi_2(u, v)$, $z = \varphi_3(u, v)$ は、ふつう空間内の曲面を表わします。)

空間極座標 (球座標) を学んだ人には、次の例は理解できるでしょう。

```
ParametricPlot3D[{Sin[u]Cos[v], Sin[u]Sin[v], Cos[u]},
                 {u, 0, Pi}, {v, 0, 2Pi}]
```

次の例は是非試してみよう。結果が事前に想像できますか？

```
ParametricPlot3D[{Cos[t],Sin[t],u}, {t,0,2Pi}, {u,0,4}]
ParametricPlot3D[{Cos[t](3+Cos[u]),Sin[t](3+Cos[u]),Sin[u]},
        {t,0,2Pi}, {u,0,2Pi}]
```

ListPlot3D[] 等もある。

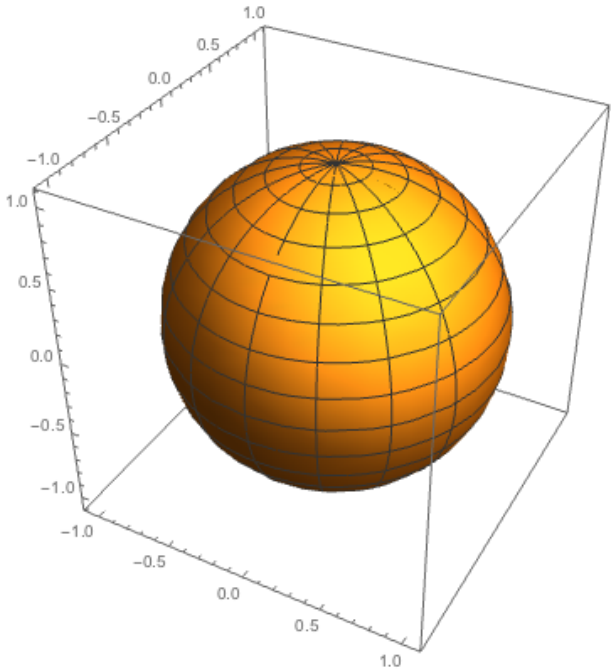


図 18: $x = \sin u \cos v,\ y = \sin u \sin v,\ z = \cos u\ (u \in [0, \pi],\ v \in [0, 2\pi])$

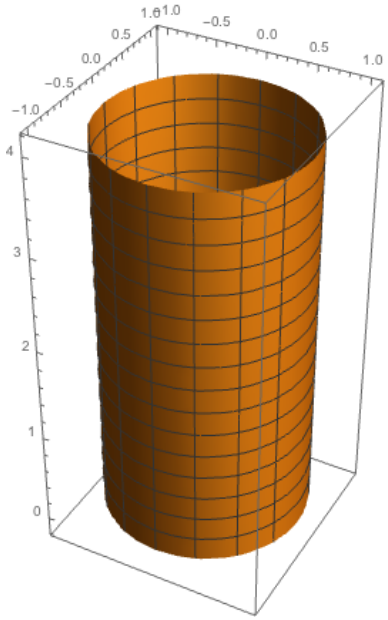


図 19: $x = \cos t,\ y = \sin t,\ z = u\ (t \in [0, 2\pi],\ u \in [0, 4])$

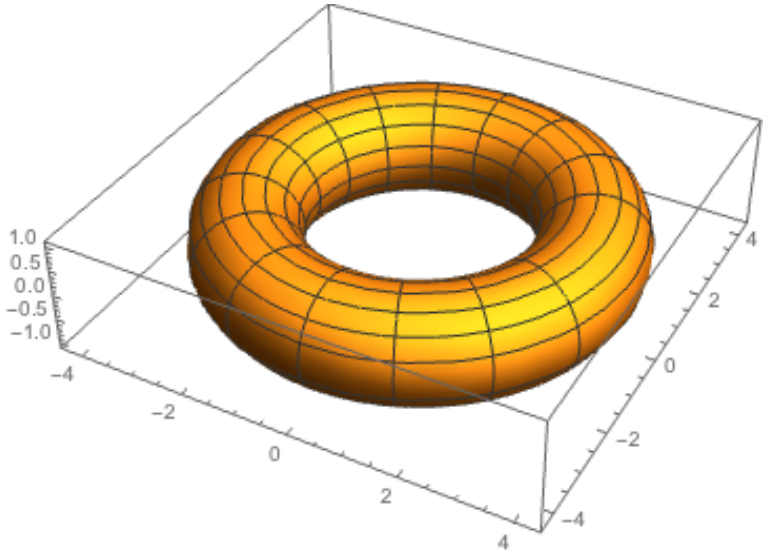


図 20: トーラス

```
Plot3D[Sin[Pi*x] + y, {x, -1, 1}, {y, 0, 1}]
d = Table[Sin[Pi x] + y, {y, 0, 1, 0.1}, {x, -1, 1, 0.1}];
ListPlot3D[d]
ListPlot3D[d,DataRange->{{-1,1},{0.1}}]
```

(x と y の範囲指定の順番に注意。)

8.1.6 3 変数関数の等値面 ContourPlot3D[]

(工事中)

3 変数関数 $F = F(x, y, z)$ のレベルセット (等値面) $F(x, y, z) = c$ は、曲面であると期待できるが、それを描くための関数 ContourPlot3D[] がある。

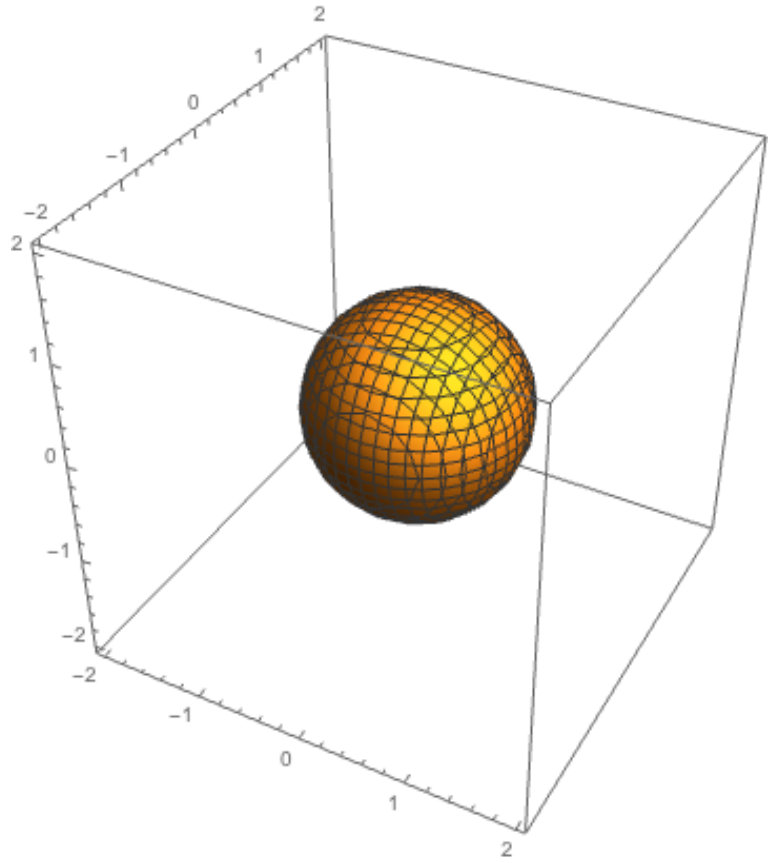
```
ContourPlot3D[x^2+y^2+ z^2==1,{x,-2,2},{y,-2,2},{z,-2,2}]
```

8.1.7 グラフィックスのファイルへの保存 Export[]

(2019/11/3 ここに書いてあることは古いので、書き直しが必要。Windows は使っていないし、今更 PostScript でもないし。)

2013 年度の環境では、以下の SetDirectory[] をしないでも、マイドキュメントに保存される。

関数 Export[] を Export["ファイル名", グラフィックス] のように使って、グラフィックスをファイルに出力できる。ファイル名の末尾を .eps にすると、自動的に EPS フォーマットが選択される。



論より run

```
g=Plot[4x^3-8x^2-4x+9,{x,-4,4}]
```

として描画したグラフィックス (g という変数に記憶された) を EPS ファイルとして保存するには、

```
Export["z:\\.windows2000\\graph1.eps", g]
```

とするか、

```
SetDirectory["Z:\\.windows2000"]    (一度やっておくと)
Export["graph1.eps", g]             (後の Export[] が簡単になる)
```

とする。

注意: Mathematica の文字列中で、バックスラッシュ \ を表わすには、二つ続けた \\ とする必要がある。そのため、パス名は本来 "Z:\.windows2000\graph1.eps" であるところを、上の例のようにしなければならない。これは C 言語などと同じである。なお、\ の代りに / を用いて、Export["Z:/.windows2000/graph1.eps",g] とすることも出来る。

- 使い方は、Export[ファイル名，グラフィックス] またはExport[ファイル名，グラフィックス，オプション] とする。

- グラフィックスの保存用の形式としては、EPS (`.eps`), GIF (`.gif`), JPEG (`.jpeg`, `.jpg`), PNG (`.png`), PDF (`.pdf`), WMF (`.wmf`) 等がある。
- ファイル名の拡張子を「慣習」に従って選ぶと、形式が自動的に選択される。例えば "`graph.eps`" というファイル名にすると、EPS (Encapsulated PostScript) 形式となる。
- ファイル名に完全パス名 ("`Z:¥.windows2000¥graph.eps`" のような、ドライブ名 (ここでは `Z:` のこと) から始まる長い名前) をつけることも出来るが、事前にカレント・ディレクトリィを適当にセットしておけば、短い相対パス名で済ませることができる。たくさんのグラフィックスを保存する場合には、例えばマイドキュメント (情報処理教室の Windows 環境の場合 `Z:¥.windows2000`) をカレント・ディレクトリィにするとよい。

EPS ファイルは LaTeX 文書に取り込むのに便利であるが、Windows 環境で印刷する場合は、`Export["torus.jpg", g]` のようにして、JPEG のようなイメージ・フォーマットにした方が簡単かも知れない。

Version 6 以降の Mathematica では、それ以前と比べてグラフィックス機能が大幅に拡張された影響で、`Export[]` で PostScript データを生成すると、ファイルのサイズが巨大になってしまう (あえて断言)。TeX 文書などに取り込む場合、障害になることがある。根本的な解決ではないが、二つ応急処置を示す。

1. イメージ形式で出力してから、EPS に変換する。例えば

   ```
   Export["graph.jpg", g]
   ```

   として、コマンド・プロンプトまたはターミナルで

   ```
   jpeg2ps graph.jpg > graph.eps (あるいは wjpeg2ps を使う)
   ```

   とする (2013 年度情報処理教室の Windows 環境には、Cygwin の中に jpeg2ps が入っていて、コマンド・プロンプトを開けば jpeg2ps コマンドが使える。)。`Export[]` を実行する際、

   ```
   Export["graph.jpg", g, ImageResolution->1200]
   ```

   のように `ImageResolution` (解像度、単位は dpi) を指定することで、画像の品質を選ぶことが出来る。

2. Version 5 互換のグラフィックス機能を使う。

   ```
   <<Version5`Graphics`
   ```

   とすると、Version 5 と同等の処理をする。例えば `g=Plot3D[x^2-y^2,{x,-1,1}, {y,-1,1}]` の結果を `Export["graph.eps",g]` とする場合で、61KB と 1MB という 10 倍以上の差が生じた。(参考: http://groups.google.com/group/comp.soft-sys.math.mathematica/browse_thread/thread/a669fd00915cbbf5/6b0387ea4f8732d4?pli=1)

あるいは PNG 形式や PDF 形式で出力して Export["graph.png", g])、適当な方法で graph.bb のようなファイルを生成し (コマンドプロンプトで ebb graph.png とする？情報処理教室の Window 環境に ebb コマンドがあるかどうか未確認)

```
\usepackage[dvipdfm]{graphicx}% これまでは dvips オプションを指定していた。
...
\includegraphics[width=10cm]{graph.png}
```

のように取り込む、というやり方も出来るかもしれません (このやり方をすると単独の dviout では表示できないので、PDF を作って Adobe Acrobat で表示して確認することになります)。

8.1.8 今後の加筆計画

- ベクトル解析関係の描画

8.2 数学を少し

$\mathbf{R}^n$ 内の (パラメーター) 曲線とは、$\mathbf{R}$ の閉区間 $I=[a,b]$ から $\mathbf{R}^n$ への連続写像 $\varphi\colon I\to\mathbf{R}^n$ として定義するのが「普通」である。$n=2$ の場合は ParametricPlot[] で、$n=3$ の場合は ParametricPlot3D[] で描くことが出来る。

平面曲線は、パラメーター曲線として表す以外に、

(i) 1 変数関数のグラフ

$$y=f(x)\quad(x\in I)\quad\text{つまり}\quad\{(x,f(x));x\in I\}.$$

(ii) 2 変数関数の零点集合

$$f(x,y)=0\quad((x,y)\in\Omega)\quad\text{つまり}\quad\{(x,y)\in\Omega;f(x,y)=0\}.$$

あるいはより一般にレベルセット $f(x,y)=L$.

例えば双曲線

$$\frac{x^2}{a^2}-\frac{y^2}{b^2}=1$$

は「2 変数関数のレベルセット」として表現してあるが、これを

$$y=\pm b\sqrt{\frac{x^2}{a^2}-1}\quad(|x|>a)$$

とすると「1 変数関数のグラフ」による表現になるし、

$$\begin{cases}x=\pm a\cosh t\\ y=b\sinh t\end{cases}\qquad(t\in\mathbf{R})$$

のような「パラメーター曲線」による表現が得られる。それぞれ ContourPlot[], Plot[], ParametricPlot[] で描画できる。

同様に、$\mathbf{R}^3$ 内の曲面を表すためには、次の方法がある。

(a) パラメーター曲面

$$x = \varphi_1(u,v), \quad y = \varphi_2(u,v), \quad z = \varphi_3(u,v) \quad ((u,v) \in D \subset \mathbf{R}^2)$$

(b) 2 変数関数のグラフ

$$z = f(x,y) \quad ((x,y) \in D) \quad \text{つまり} \quad \{(x,y,f(x,y)); (x,y) \in D\}.$$

(c) 3 変数関数の零点集合

$$f(x,y,z) = 0 \quad \text{つまり} \quad \{(x,y,z) \in \Omega; f(x,y,z) = 0\}.$$

(a) は ParametricPlot3D[] で、(b) は Plot3D[] で、(c) は ContourPlot3D[] で描くことができるわけである。

8.3 もう少し詳しく

8.3.1 グラフィックス・オブジェクト, Show[] で再表示・合成

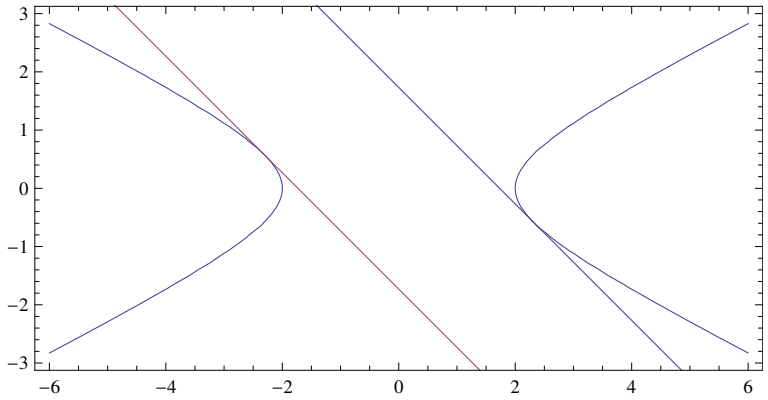
グラフィックスの関数では、対応するグラフィックス・オブジェクトを生成して、画面に表示する。グラフィックス・オブジェクトは変数に代入可能である。オブジェクトを表示するには関数 Show[] を用いる。またオブジェクトがどう表現されているかは InputForm[] で読むことが出来る。

次の例では、変数 g1, g2 に代入しておいて、後から再表示している。

```
g1 = Plot[Sin[x],{x,0,2Pi}]   描画と同時に変数に代入
g2 = Plot[Cos[x],{x,0,2Pi}]   (同上)
Show[g1]                      再表示
Show[g1,g2]                   二つまとめて表示
InputForm[g1]                 グラフィックス・オブジェクトをのぞく
Remove[g1,g2]
```

双曲線と接線を別々に描いて合成。

```
g=ContourPlot[x^2/4-y^2==1,{x,-6,6},{y,-3,3},AspectRatio->1/2]
t=Plot[{-x+Sqrt[3],-x-Sqrt[3]},{x,-6,6}]
Show[g,t]
```



Show[ ] の描画範囲は、最初の引数のグラフィックスで決まるようなので、大きいものを最初に持って来るのが良い。

すべてのグラフィックスを見ることを明示するには、PlotRange->All とする。

複数のグラフィックスを並べる手段に、Grid[], GraphicsGrid[], GraphicsRow[], GraphicsColumn[] などがある。

8.3.2 グラフィックスの関数の引数、特にオプション

グラフィックスの関数が、引数として取るものは、順に

(1) 関数あるいは関数のリスト or 数値データによるリスト

(2) 描画する範囲

(3) 描画を制御するオプション

このうちオプションはその名前の通り省略可能である。

(オプションとは？実際にどうやって描画するかについて、自由度があり、それを一々指定するのはとても面倒なので、普段はそれを適当に決めているが、そのデフォールト値で満足できない場合に、ユーザーが自分の求めるものを指定することが出来るようにしたもの。)

関数がどういうオプションを持つかは “??関数名” で調べられる。例えば、関数 Plot[] のオプションを調べたければ、次のようにすればよい。

Plot[] のオプションは？

```
??Plot
```

オプションの指定の仕方は、” オプション名 -> 値” である。例えば

AspectRatio -> 数値	縦横比。デフォールト値は 1/GoldenRatio Automatic とすると縦横の縮尺を揃える。
Axes -> 真偽値	軸を描くかどうか
AxesLabel -> {"x", "y=f(x)"}	
BoxRatios -> {X,Y,Z}	Automatic とすると 3D グラフィックスにおける実際の座標の値に対応するボック

スの長さの辺の比

`PlotLabel -> "Graph of f"`	
`AxesOrigin -> {0,0}`	座標軸の交点は（0,0）
`Compiled -> False`	デフォールト値は True
`Frame -> True`	枠を描くかどうか（デフォールト値は False）
`GridLines -> Automatic`	
`PlotRange -> {zmin,zmax}`	All というのもある
`PlotPoints->100`	使用するサンプル点の個数（多いと忠実）
`PlotStyle->Thick`	太い線で
`PlotStyle->Red`	赤い線で
`PlotStyle->{Thick,Red}`	太い赤い線で
`PlotStyle->Thickness[0.005]`	太さを数値で指定

(アスペクト比 (AspectRatio) は縦横比 (= 縦 ÷ 横) である。また黄金比 (golden ratio) は $\frac{1+\sqrt{5}}{2} = 1.61\cdots$ である。`AspectRatio -> Automatic` は、縦軸と横軸の縮尺が同じになるようにアスペクト比を調節する。

しばらくの間、オプションを変更したまま使いたい場合は、`SetOptions[]` を使う。次の例では、関数 Plot3D を使用する時に、PlotPoints を 100 にするよう設定している。

```
SetOptions[Plot3D, PlotPoints->100]
```

新しい Mathematica には、Exclusions というオプションがある。

```
Plot[Tan[x],{x,-10,10},Exclusions->{Cos[x]==0}]
```

8.3.3 Plot[] — 1 変数関数のグラフを描く

1 変数関数のグラフを描くには `Plot[]` を使う。基本的な使い方は

```
Plot[関数, 描画範囲を示すリスト]
Plot[{関数 1, 関数 2, ..., 関数 n}, 描画範囲を示すリスト]
```

描画範囲を示すリストから、関数値を計算するための変数の適当な値を選びだして、そこでの関数値を評価して、そうして得た点 (標本点) を順に結んでグラフを描く。標本点の個数を多くするには PlotPoints オプションを指定する。

```
Plot[Sin[x], {x,0,2Pi}]
Plot[{Sin[x], Sin[2x], Sin[3x]}, {x,0,2Pi}]
Plot[Sin[50x],{x,0,Pi},PlotPoints->1000]
```

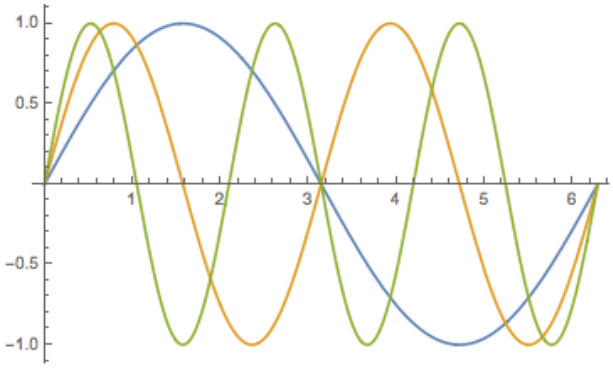


図 21: 3 つの関数のグラフを描く

8.3.4 ListPlot[] — リストの形で与えられた平面上の点を結んで折れ線を描く

数値データのリストから曲線を描く。これを用いると、例えば外部のプログラムで計算したデータを表示できる。次の例では Table[] で生成したリストから曲線を描いている。

```
fp = Table[{t,N[Sin[Pi t]]}, {t,0,0.5,0.025}]
ListPlot[fp]
ListPlot[fp, PlotJoined -> True]
Remove[fp]
```

```
p[z_, alpha_, maxn_] :=
  Module[{r, t, w}, r = Abs[z]; t = Arg[z]; w = r^alpha*Exp[I alpha t];
         Table[{Re[w Exp[I n alpha 2 Pi]], Im[w Exp[I n alpha 2 Pi]]},{n,maxn}]
        ]
g8 = ListPlot[p[1, 1/8, 8],
              AspectRatio -> Automatic, PlotStyle -> {Red, PointSize[0.03]}]
```

ひまわりの種 (黄金角が大事って本当？)

```
a=N[GoldenAngle]
ps[a_] := Table[N[Sqrt[n] {Cos[a*n], Sin[a*n]}], {n, 1, 1000}];
ListPlot[ps[a], AspectRatio -> 1, PlotStyle -> Thick]
Manipulate[ListPlot[ps[a + eps], AspectRatio -> 1, PlotStyle -> Thick],
 {eps, -0.1, 0.1, 1/1000}]
```

8.3.5 NDSolve[] の結果を描く

少し脱線するが、微分方程式を数値的に解く NDSolve[] の結果を描く方法を説明しておく。

```
NDSolve[{y'[x]==Sin[y[x]],y[0]==1}, y, {x,0,4}]
Plot[Evaluate[y[x] /.  %], {x,0,4}]
```

(この NDSolve[] は数値的に微分方程式を解くコマンドであり、結果は離散的な点での関数値を近似的に求めたものである。計算していない点での値は補間により簡略計算する。こういうものを InterpolatingFunction という。)

```
    y[1.5] /. %%
```

8.3.6 ContourPlot[], DensityPlot[] — 2 変数関数の等高線、濃淡図、陰関数

2 変数関数 $(x, y) \mapsto f(x, y)$ を可視化するために、等高線を描く ContourPlot[][15], 濃淡図を描く DensityPlot[] が用意されている[16]。 使い方は

```
ContourPlot[f[x,y], {x,xmin,xmax}, {y,ymin,ymax}]
DensityPlot[f[x,y], {x,xmin,xmax}, {y,ymin,ymax}]
```

例として

```
ContourPlot[Sin[x]Sin[y], {x,-2,2}, {y,-2,2}]
DensityPlot[Sin[x]Sin[y], {x,-2,2}, {y,-2,2}]
```

$f(x, y) = c$ で表される曲線を描くために、

```
ContourPlot[f[x,y]==c, {x,xmin,xmax}, {y,ymin,ymax}]
```

という使い方が出来る。

```
ContourPlot[Sin[x]Sin[y]==0, {x,-2,2}, {y,-2,2}]
```

オプションとして

Contours -> 整数（等高線の本数）
Contours -> 数値のリスト（等高線のレベル）
PlotRange -> {zmin,zmax} または Automatic
PlotPoints -> 整数 （デフォールトが 15。小さい！非力さが出て来る。）

などがある。

$f(x, y) = x^2 - y^2$ と $g(x, y) = 2xy$ は座標系を回転すると一致する。そのことを見るために、等高線を描いてみる。

[15]等高線のことを contour (line) と呼ぶ。
[16]密度、濃度を density と呼ぶ。

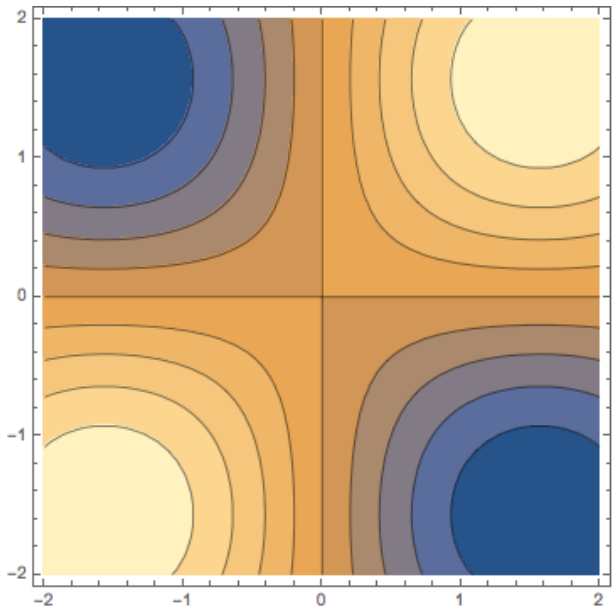


図 22: ContourPlot[] の例

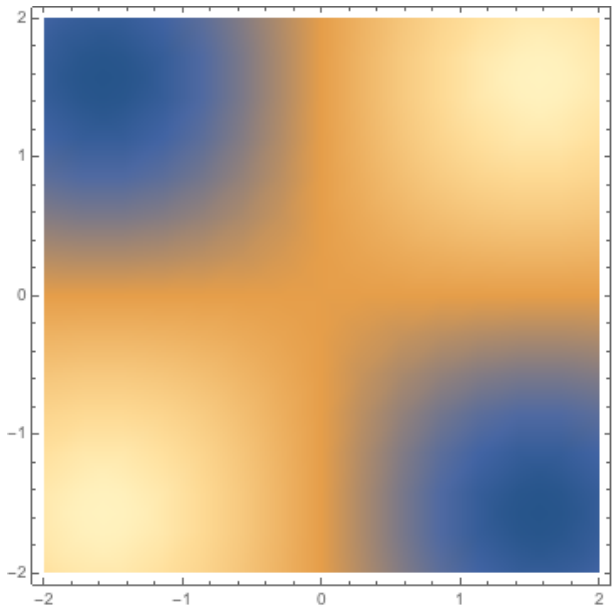


図 23: DensityPlot[] の例

```
fc=ContourPlot[x^2 - y^2, {x, -1, 1}, {y, -1, 1},
 RegionFunction -> Function[{x, y, z}, x^2 + y^2 < 1]]

gc=ContourPlot[2 x y, {x, -1, 1}, {y, -1, 1},
 RegionFunction -> Function[{x, y, z}, x^2 + y^2 < 1],
 Contours -> Table[h, {h, -1, 1, 0.2}]]
```

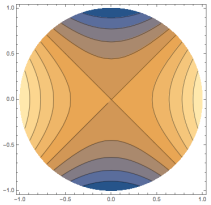


図 24: $f(x,y) = x^2 - y^2$ の等高線

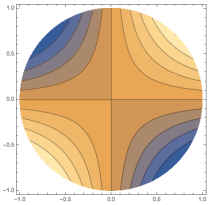


図 25: $g(x,y) = 2xy$ の等高線

8.3.7 Plot3D[]

2 変数関数のグラフを描く Plot3D[] は

```
Plot3D[f, {x,xmin,xmax}, {y,ymin,ymax}]
```

が基本的な使い方。例えば

```
Plot3D[Sin[x y], {x,0,3}, {y,0,3}]
```

オプションとして以下のようなものがある。

HiddenSurface -> False	隠面消去をしない
PlotPoints -> 個数	大きくすると細かい図を書く。
ViewPoint -> {x,y,z}	視点の指定

視点を色々変えてみる

```
my[a_,b_,c_]:= Plot3D[Sin[x y],{x,0,3},{y,0,3}, ViewPoint->{a,b,c}]]
my[1,1,1]
my[1,-1,1]
Table[my[1,t,1],{t,1,-1,-0.2}]
```

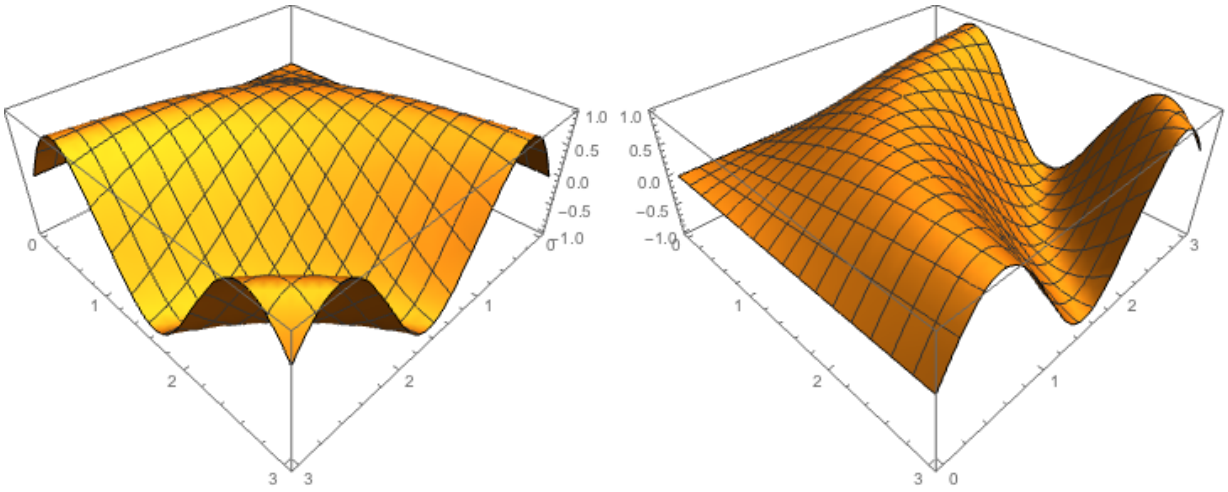


図 26: $z = \sin xy$ を $(1, 1, 1)$ から見たときと、$(1, -1, 1)$ から見たとき

8.4 Manipulate[]

(工事中)
熱方程式の基本解

$$G(x,t) = \frac{1}{\sqrt{4\pi t}} \exp\left(-\frac{x^2}{4t}\right)$$

について、t を固定するごとに $x \mapsto G(x,t)$ のグラフを描いてみる。

```
G[x_, t_] := Exp[-x^2/(4 t)]/(2*Sqrt[Pi*t])
g=Plot[Table[G[x, t], {t, 0.1, 1.0, 0.1}], {x, -5, 5}, PlotRange -> All]
Manipulate[Plot[G[x, t], {x, -5, 5}, PlotRange -> {0, 3}], {t, 0.01, 2}]
```

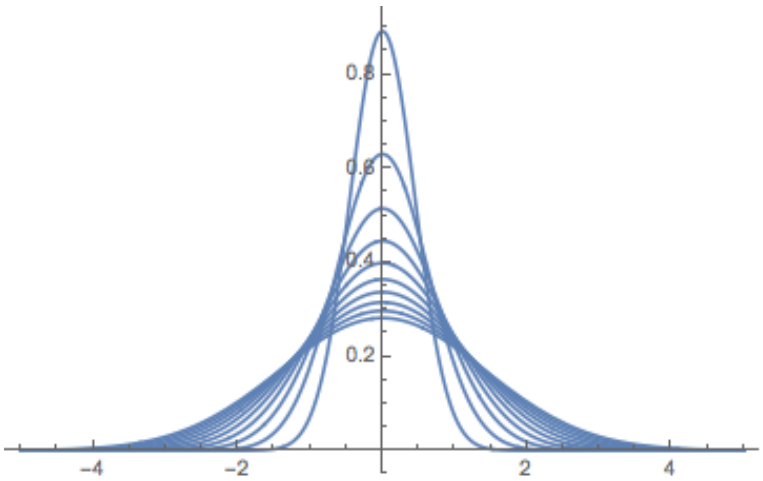


図 27: 一度に全部描いてしまうと分かりにくい…

離心率が e の円錐曲線は適当な座標系で

$$r = \frac{1}{1 + e\cos\theta}$$

と表すことが出来る。

```
Manipulate[g=PolarPlot[1/(1 + eps Cos[t]), {t, 0, 2 Pi}], {eps, 0, 2, 0.01}]
```

8.5 RegionPlot3D[]

(準備中)
学生のレポートから。

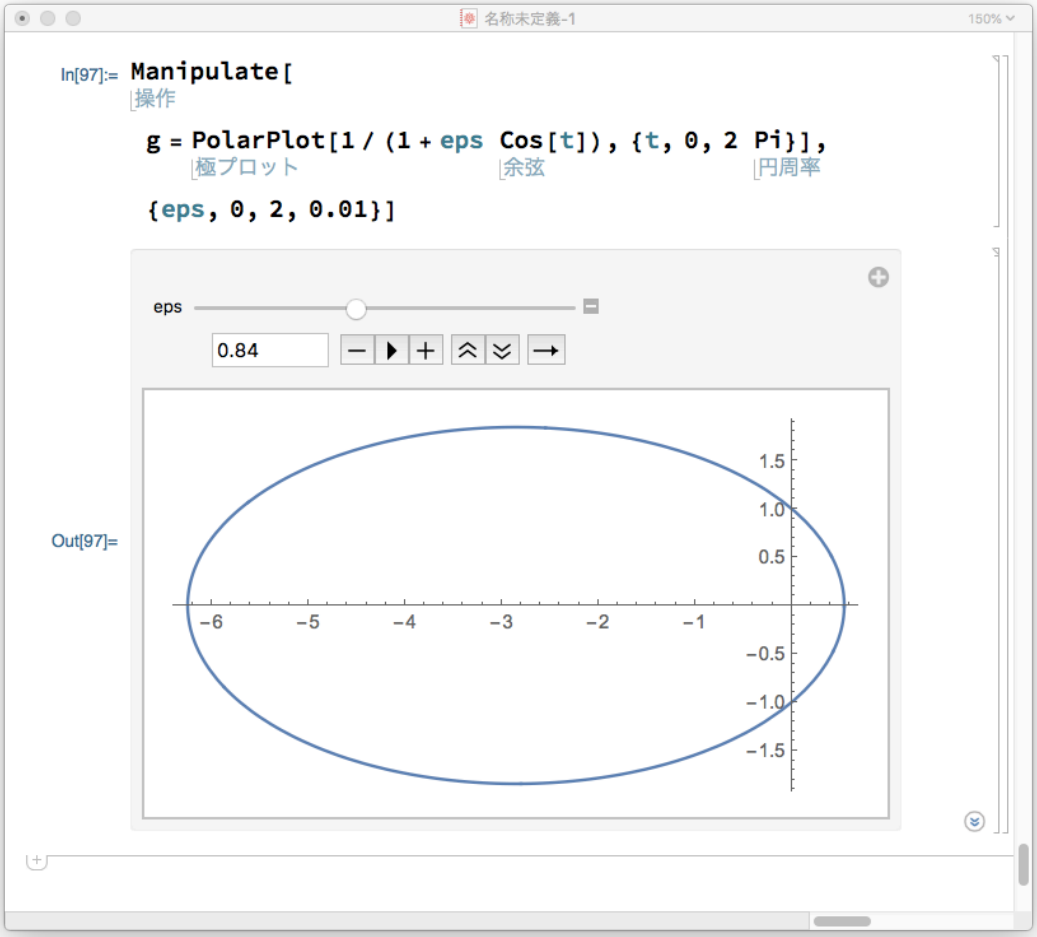
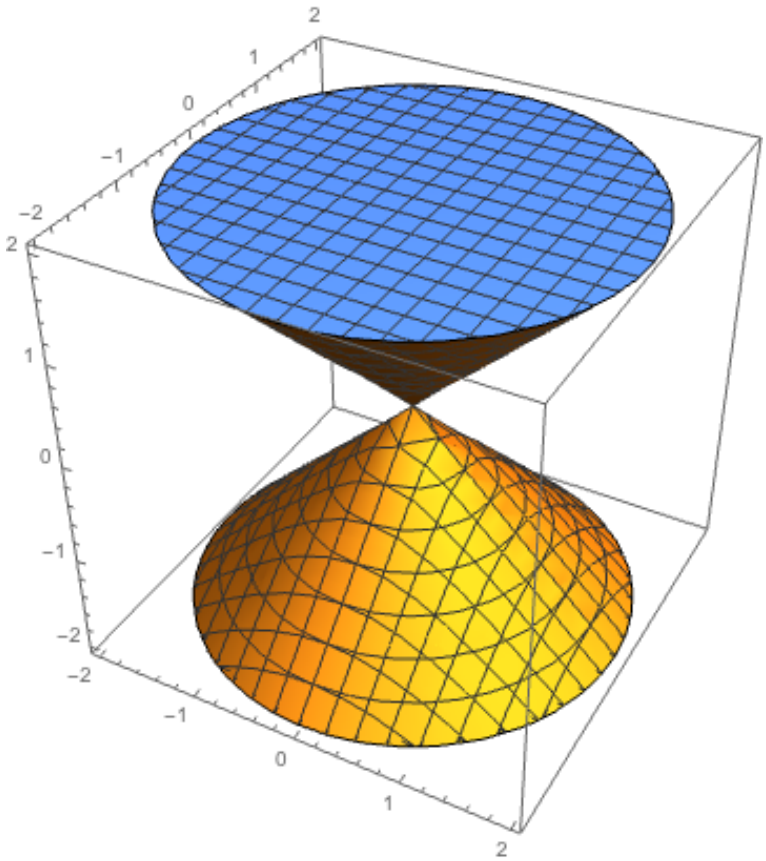

名称未定義-1
150%
In[97]:=
Manipulate[
操作
g = PolarPlot[1 / (1 + eps Cos[t]), {t, 0, 2 Pi}],
極プロット
余弦
円周率
{eps, 0, 2, 0.01}]
eps
0.84
Out[97]=
1.5
1.0
0.5
−6
−5
−4
−3
−2
−1
−0.5
−1.0
−1.5

図 28: $e = 0.84$ は楕円

```
RegionPlot3D[x^2+y^2-z^2<0,{x,-2,2},{y,-2,2},{z,-2,2}]

RegionPlot3D[x^2+y^2-z^2<1,{x,-4,4},{y,-4,4},{z,-4,4}]

RegionPlot3D[x^2+y^2-z^2<-1/2,{x,-3,3},{y,-3,3},{z,-3,3}]
```

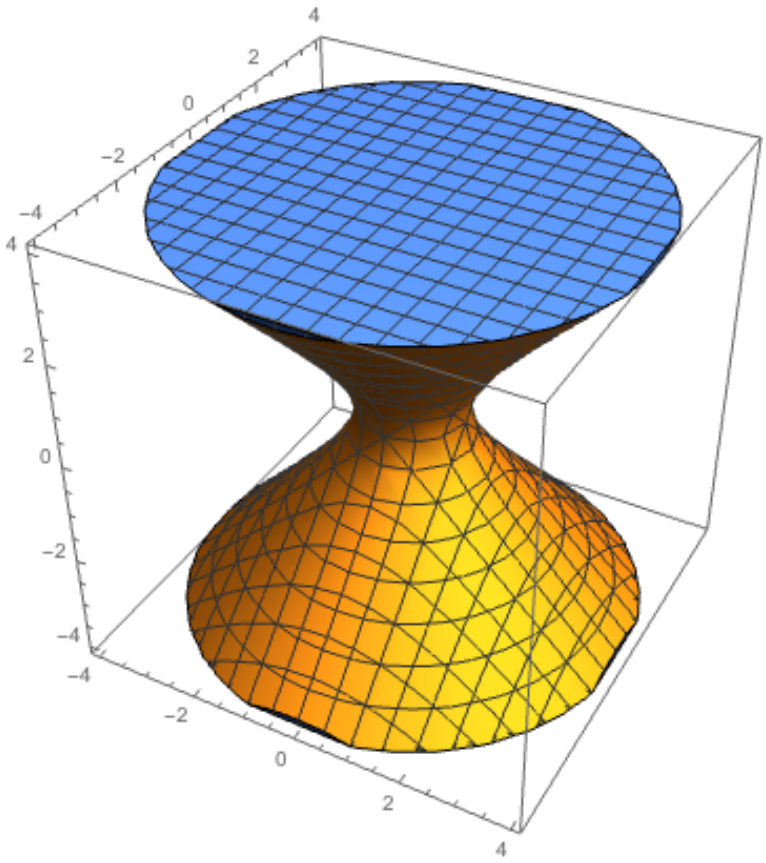


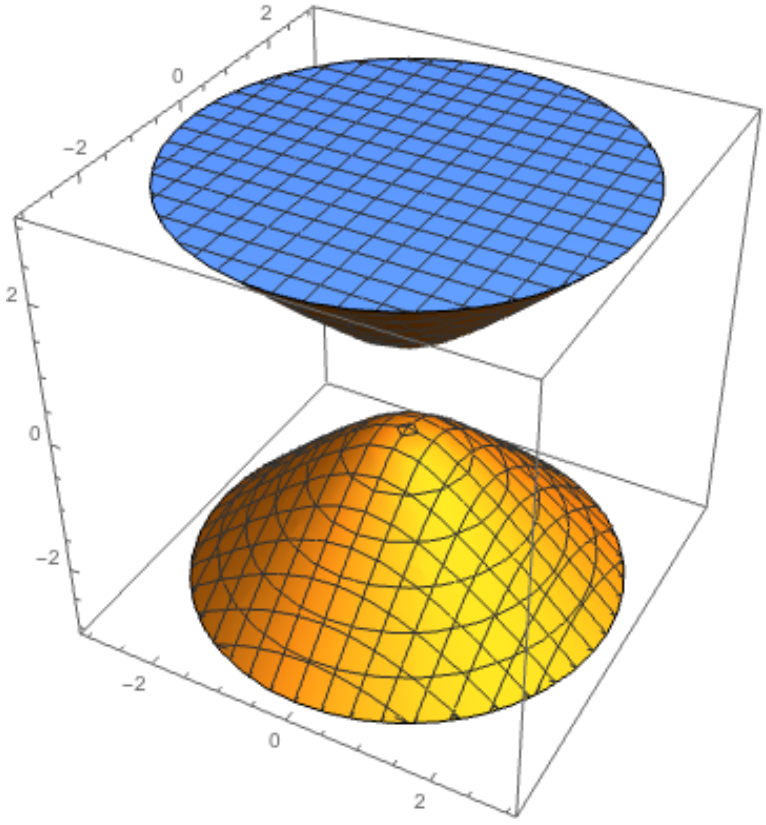
9 Mathematica のサウンド機能

(準備中？以前、卒研で学生に http://nalab.mind.meiji.ac.jp/~mk/labo/report/open/2003-matsuyama.pdf というレポートを書いてもらって以来、まとめておきたいと考えているけれど…)

『音の取り扱いに関するメモ』[17] に少しだけ書いた。

[17] http://nalab.mind.meiji.ac.jp/~mk/labo/text/memo-sound/


4
2
0
−2
−4
4
2
0
−2
−4
−4
−2
0
2
4


2
0
−2
2
0
−2
−2
0
2

10　ある年度の情報処理 II レポート問題

問題 1

高校での数学の計算問題、大学の微分積分や線形代数の計算問題を解かせてみる。どこまで解けるか、どういう問題が解けないか (ごくまれに結果がおかしいこともある)、色々試し、また解けないのは何故か推測する。(問題によっては工夫が必要かもしれない。簡単に Mathematica では解けないと決めつけないで、考えること。色々考えた経過をレポートする。)

問題 2

1 変数、または 2 変数の関数 f を定め、グラフ、等高線、微分が 0 になる点、またその点での関数値等を計算する。多変数関数の極値問題を解く際にどのように利用できるか、試して見よ。(この方向に発展させると、極値問題を解くプログラムが作れるかも知れない? これまで完全なプログラムを書いた学生はいない。挑戦を期待する。

11　工事計画

- ノートブックの使い方
- 画像の印刷など
- 音声の取り扱い
  とりあえず 松山周五郎「音の Fourier 解析」(2003 年度卒業研究レポート)[18]
- レポートの作り方

12　書き足すべきもの

12.1　線形代数

- `NullSpace[]`
- `MatrixRank[]`
- `LinearSolve[]`
- `RowReduce[]`

[18] `http://nalab.mind.meiji.ac.jp/~mk/labo/report/open/2003-matsuyama.pdf`

12.2 Mathematica 6 以降の機能について

(いつか書きたそう。)

- Manipulate[]
- ContourPlot3D[]

A その他使う可能性の高いリスト処理用の関数

リストの取り扱いをするための関数をいくつか紹介する (これらは音声データの取り扱いに限ったものではないが、Mathematica の入門的解説にはあまり詳しく解説されていない)。

A.1 Take[] リストの要素の取り出し

Take[a,n] で、n が正整数のときは、a の先頭 n 要素からなるリスト、n が負整数のときは、a の末尾 n 要素からなるリストを返す。

Take[a,{n1,n2}] で、a の第 n1 要素から第 n2 要素までからなるリストを返す。

A.2 Drop[] リストの要素の削除

Drop[a,n] で、n が正整数のときは、a の先頭 n 要素を削除したリスト、n が負整数のときは、a の末尾 n 要素を削除したリストを返す。

Drop[a,{n1,n2}] で、a の第 n1 要素から第 n2 要素までを削除したリストを返す。

Drop[a,{n}] で、a の第 n 要素を削除したリストを返す。

似た機能を持つ関数に Delete[] がある。の第 1 要素と第 3 要素を削除するには、Delete[a,{{1},{3}}] とする。

A.3 Sort[] リストを大きさの順に並べる

a の要素を小さい順に並べたリストを得るには Sort[a] とする。

a の要素を大きい順に並べたリストを得るには Sort[a,Greater] とする。

比較の方法を式で指定できる。例えば要素を大きい順に並べるには、Sort[a, #1>#2 &] とすることも出来る。次の例では、リストの第 1 要素同士を比較して、大きいかどうかを判定して、ソートさせている。

よく使いそうな例 (リストのソート)

```
In[169]:= d = {{2, c}, {3, b}, {1, a}}

Out[169]= {{2, c}, {3, b}, {1, a}}

In[170]:= Sort[d, #1[[1]] > #2[[1]] &]

Out[170]= {{3, b}, {2, c}, {1, a}}
```

A.4 Ordering[] リストの要素から大きいもの小さいものを取り出す

Ordering[a] で、a の要素を小さい順に並べた場合の、要素のもとの位置からなるリストを返す。例えば、b=a[[Ordering[a]]] とすると、b は a を小さい順に並べたリストとなる (もちろん、そうするためだけならば、b=Sort[a] とするのが簡単である)。

Ordering[a,n] で、n が正整数のときは、a の要素の小さい順に n 番目までの要素の位置からなるリスト、n が負整数のときは、a の要素の大きい順に n 番目までの要素の位置からなるリストを返す。例えば、a 最小要素の位置は Orderine[a,1][[1]], 最大要素の位置は Ordering[a,-1][[1]] で求められる。

遊び半分、色々な関数の紹介を兼ねて: a を大きい順に並べるには、Sort[a, Greater], Reverse[Sort[a]], a[[Ordering[a,-Length[a]]]] などなど色々やり方がある。

リストの最大要素と、それがどこにあるかを返す関数。

```
maxAndIndex[l_] := Module[{id = Ordering[l, -1][[1]]}, {l[[id]], id}]
```

A.5 Select[] リストの要素のうち条件に合致するものを選ぶ

以下の例では、3 より大きい要素を選んでリストを作っている。

```
In[1]:= Select[{0, 6, 1, 4, 5, 3, 7}, # > 3 &]
Out[1]= {6,4,5,7}
```

5 個の数 $0, 1, \ldots, 4$ の順列のうち、先頭が 0 でないもののリスト

```
Select[Permutations[{0,1,2,3,4}],#[[1]] != 0&]
```

引数の末尾 (# と & を含む部分) は、純関数というものになっている。

リストに含まれる要素すべてがある条件式を満たすかどうかのチェック。

```
forAll[list_, cond_] := Select[list, ! cond@# &, 1] === {};
```

(これと同じことをする関数は絶対必要だと考えるのだけど、何で Mathematica に入っていない

のかな?ちなみにhttp://stackoverflow.com/questions/8512658/how-to-do-logical-tests-for-al
というページで読んだ。)

B　式の評価順序の話 — Evaluate[]

```
Table[BesselJ[n,x],{n,5}]
```

とすると、{BesselJ[1,x],BesselJ[2,x],BesselJ[3,x],BesselJ[4,x],BesselJ[5,x]} という 5 つの関数を含んだリストができる。一方、

```
Plot[{BesselJ[1,x],BesselJ[2,x],BesselJ[3,x],BesselJ[4,x],
BesselJ[5,x]},{x,0.0,10.0}]
```

とすると、その 5 つの関数のグラフが表示される。すると、

```
Plot[Table[BesselJ[n,x],n,5],{x,0.0,10.0}]
```

とすれば 5 つの関数のグラフが描けるような気がするが、実はうまく行かない。Lisper だったら、その理由は分かるでしょう。そして、

```
Plot[Evaluate[Table[BesselJ[n,x],{n,5}]],{x,0,10}]
```

とすると、うまく行くというのにも納得できるでしょう。一件落着。
　同様に

```
f[x_]:=Sin[x]
```

としたとき、D[f[x],x] は Cos[x] となるが、

```
Plot[D[f[x],x],{x,0,2Pi}]
```

としても Cos[x] のグラフは描けない。

```
Plot[Evaluate[D[f[x],x]],{x,0,2Pi}]
```

あるいは、

```
Plot[f'[x],{x,0,2Pi}]
```

とする。

C　プログラム例

C.1　2 次元の Newton 法

```
f[x_,y_]:={x^2-y^2+x+1,2 x y +y}
Df[a_,b_]:=Module[
            {x,y},
            Transpose[{D[f[x,y],x],D[f[x,y],y]}] /. {x->a,y->b}
           ]
{x,y}={1,1}
Do[{x,y}={x,y}-Inverse[Df[x,y]].f[x,y];
   Print[{x,y},"=",N[{p,q},20]],
   {6}
  ]
```

あるいは、変数の方もベクトル的にリストを使って書いて、

```
f[{x_,y_}]:={x^2-y^2+x+1,2 x y +y}
Df[{a_,b_}]:=Module[
            {x,y},
            Transpose[{D[f[{x,y}],x],D[f[{x,y}],y]}] /. {x->a,y->b}
           ]
xk={1,1}
Do[xk=xk-Inverse[Df[xk]].f[xk]; Print[xk,"=",N[xk,20]], {6}]
```

C.2 二変数関数の極値問題 kyokuchi.m

```
(* 二変数関数 f の停留点を求める（よう努力する）*)
teiryuuten[f_]:=
Module[
  {fx,fy},
  fx=Simplify[D[f[x,y],x]];
  fy=Simplify[D[f[x,y],y]];
  Solve[{fx==0,fy==0},{x,y}]
]

(* 二変数関数 f とその停留点のリスト s を分析し、極値の判定をする *)
bunseki[s_,f_]:=
Module[
  {ff,HesseXY,aSolution,restSolutions,valf,l1,l2},
  ff=f[x,y];
  HesseXY = {{D[ff,x,x],D[ff,x,y]},
             {D[ff,y,x],D[ff,y,y]}};
  restSolutions = s;
  While [(restSolutions != {}),
         aSolution = First[restSolutions];
         restSolutions = Rest[restSolutions];
         valf = ff /. aSolution;
         {l1,l2} = Eigenvalues[HesseXY /. aSolution];
         If [l1 > 0 && l2 > 0,
            Print[aSolution, ", 極小 f(x,y)=", valf]];
         If [l1 < 0 && l2 < 0,
            Print[aSolution, ", 極大 f(x,y)=", valf]];
         If [(l1 l2 < 0),
            Print[aSolution, ", 極値でない"]];
         If [(l1 l2 == 0),
            Print[aSolution, ", 極値であるかどうか分からない。"]];
      ]
]
```

これで良い？

```
  st[f_]:=Solve[{D[f[x,y],x]==0,D[f[x,y],y]==0},{x,y}]
```

使用例

```
oyabun% math
Mathematica 4.0 for Solaris
Copyright 1988-1999 Wolfram Research, Inc.
 -- Motif graphics initialized --

In[1]:= << /home/syori2/kyokuchi.m

In[2]:= f[x_,y_]:=x y(x^2+y^2-4)

In[3]:= s=teiryuuten[f]

Out[3]= {{x -> -2, y -> 0}, {x -> -1, y -> -1}, {x -> -1, y -> 1},

>    {x -> 0, y -> 0}, {x -> 1, y -> -1}, {x -> 1, y -> 1}, {x -> 2, y -> 0},

>    {y -> -2, x -> 0}, {y -> 2, x -> 0}}

In[4]:= bunseki[s,f]
{x -> -2, y -> 0}, 極値でない
{x -> -1, y -> -1}, 極小 f(x,y)=-2
{x -> -1, y -> 1}, 極大 f(x,y)=2
{x -> 0, y -> 0}, 極値でない
{x -> 1, y -> -1}, 極大 f(x,y)=2
{x -> 1, y -> 1}, 極小 f(x,y)=-2
{x -> 2, y -> 0}, 極値でない
{y -> -2, x -> 0}, 極値でない
{y -> 2, x -> 0}, 極値でない

In[5]:=
```

D　Mathematicaの多変数の微積分への応用

D.1　はじめに

関数のグラフや等値面、接線・接平面を描くのは良くある話だが (2012 年度情報処理 2「レポート課題 9 について」の (4)[19])、連立方程式を解くのに使えるのは案外便利である。実は微積分の問題を作成する際に、計算のチェックに Mathematica を時々使っている。

[19] http://nalab.mind.meiji.ac.jp/~mk/syori2/jouhousyori2-2012-12/node3.html

D.2 問題 2

$\mathbf{R}^n$ の開集合 Ω で定義された関数 $f\colon \Omega \to \mathbf{R}^m$ について、(a) f は連続, (b) f は各変数につき偏微分可能, (c) f は C^1 級, (d) f は全微分可能, という 4 つの条件を考える。

(1) 条件 (d) が成り立つとはどういうことか定義を述べよ。

(2) 条件 (a), (b), (c), (d) 間の関係について説明せよ。

(3) 次式で定義される $f\colon \mathbf{R}^2 \to \mathbf{R}$ について以下の (i), (ii) に答えよ。(i) $\mathbf{R}^2 \setminus \{(0,0)\}$ で C^∞ 級であることを示せ。(ii) $\mathbf{R}^2$ で条件 (a), (b), (c), (d) を満たすかどうか調べよ。

$$f(x,y) := \begin{cases} \dfrac{x^2+xy+x^2y+y^2+y^3}{x^2+y^2} & ((x,y) \in \mathbf{R}^2 \setminus \{(0,0)\}) \\ 1 & ((x,y) = (0,0)). \end{cases}$$

D.2.1 Mathematica でやってみる

まず $(0,0)$ での連続性から調べてみよう。

```
In[1] := f[x_,y_]:=(x^2+x y+x^2 y+y^2+y^3)/(x^2+y^2)
In[2] := f[0,0]=1
In[3] := Simplify[f[x,y]-f[0,0]]
```

これから

$$f(x,y) - f(0,0) = \frac{y\,(x^2+x+y^2)}{x^2+y^2}.$$

分子の高次の項を無視すると $\dfrac{xy}{x^2+y^2}$ に等しく、分母分子ともに 2 次同次なので、0 には収束しなさそうと見当がつく。証明するには「$y = mx$ 作戦」を試してみよう。

```
In[4] := % /. y-> m x
In[5] := Limit[%, x->0]
```

として、

$$f(x,mx) - f(0,0) = \frac{m(1+x+m^2x)}{1+m^2},$$

$$\lim_{\substack{y=mx \\ x\to 0}} (f(x,y) - f(0,0)) = \frac{m}{1+m^2}$$

が得られる。この結果は m に依存するので、特に

$$\lim_{(x,y)\to(0,0)} f(x,y) \neq f(0,0).$$

ゆえに f は $(0,0)$ で連続ではない (ゆえに C^1 級でもないし、全微分可能でもない)。

次に $(0,0)$ での偏微分可能性を調べる。

```
In[6] := Simplify[(f[h,0]-f[0,0])/h]
In[7] := Simplify[(f[0,h]-f[0,0])/h]
```

これから

$$\frac{f(h,0)-f(0,0)}{h}=0,\quad \frac{f(0,h)-f(0,0)}{h}=1$$

が得られる。ゆえに (`Limit[%,h->0]` とするまでもなく)

$$f_x(0,0)=\lim_{h\to 0}\frac{f(h,0)-f(0,0)}{h}=0,\quad f_y(0,0)=\lim_{h\to 0}\frac{f(0,h)-f(0,0)}{h}=1.$$

ゆえに f は $(0,0)$ で、x についても y についても偏微分可能である。

D.3 問題 3

C^∞ 級の $f\colon \mathbf{R}^3\ni(x,y,z)\mapsto f(x,y,z)\in\mathbf{R}$ と、$\vec{a}=(a,b,c)$, $\vec{h}=(p,q,r)\in\mathbf{R}^3$ に対して

$$F(t):=f(\vec{a}+t\vec{h})=f(a+tp,b+tq,c+tr)\quad (t\in\mathbf{R})$$

とおくとき、以下の問に答えよ。

(1) $F'(t)$, $F''(t)$ を計算せよ (f を使って表せ)。

(2) 自然数 m に対して $F^{(m)}(t)$ を f を用いて表す式を推定し、帰納法を用いて証明せよ。

D.3.1 Mathematica でやってみる

Taylor の定理の導出に必要な合成関数の高階導関数の計算。

$$F(t):=f(a+tp,b+tq,c+tr)$$

で定義される F の 2 階微分までを計算しよう。

```
In[1] := F[t_]:=f[a+t p,b+t q,c+t r]
In[2] := F'[t]
In[3] := F''[t]
In[4] := Simplify[%]
```

$$F'(t)=pf_x(a+tp,b+tq,c+tr)+qf_y(a+tp,b+tq,c+tr)+rf_z(a+tp,b+tq,c+tr).$$

書くのが面倒なので $d:=(a+tp,b+tq,c+tr)$ とおくことにする (TEX だとコピーするだけなので大丈夫だけれど)。

$$F''(t)=p^2f_{xx}(d)+q^2f_{yy}(d)+r^2f_{zz}(d)+2pqf_{xy}(d)+2qrf_{yz}(d)+2rpf_{zx}(d).$$

D.4 問題 4

$f(x,y) := 2x^3 + 6xy^2 - 2x$ について、以下の問に答えよ。

(1) f のヤコビ行列 $f'(x,y)$ と Hesse 行列 $H(x,y)$ を求めよ。

(2) f の極値を求めよ。

(3) f のグラフ $z = f(x,y)$ の $(x,y) = (1,1)$ での接平面の方程式を求めよ。

D.4.1 Mathematica でやってみよう

典型的な極値問題。微分係数が 0 となる点を求めて、そこで Hesse 行列を求めて、符号 (正値？負値？不定符号？) を調べて、極値かどうかを判定する。

$$f(x,y) := 2x^3 + 6xy^2 - 2x$$

```
In[1] := f[x_,y_]:=2x^3+6x y^2-2x
In[2] := jf[x_,y_]:=D[f[a,b],{{a,b}}]/.{a->x,b->y}
In[3] := jf[x,y]
In[4] := H[x_,y_]:=D[f[a,b],{{a,b},2}]/.{a->x,b->y}
In[5] := MatrixForm[H[x,y]]
```

(ヤコビ行列を表す記号として、昔風の Jf を思い出して使ったけれど、案外良いかもしれない。合わせるために Hesse 行列を Hf と表すか？)

注意 D.1 *(1)* 単に f のヤコビ行列を計算するだけならば

2 変数関数のヤコビ行列 *(*全微分係数*)*

```
In[2] := D[f[x,y],{{x,y}}]
```

で良い。この微分の指定 {{*x,y*}} は覚えておくと良い。後のためには、ヤコビ行列を計算する *jf[]* を定義しておく方が便利であろう。そのために

これはダメ

```
In[2] := jf[x_,y_]:=D[f[x,y],{{x,y}}]
```

とするのはうまく動かない。たとえ話をすると、

$$f'(a) \neq \frac{d}{dx} f(a), \quad f'(a) = \frac{d}{dx} f(x)\Big|_{x=a}$$

の違いであろうか *(*代入と微分の順序の問題*)*。上で *jf[]* の引数の名前を *x, y* として、内部で *a, b* という変数を使ったが、もちろんこれは何でも良くて、逆にした

```
In[2] := jf[a_,b_]:=D[f[x,y],{{x,y}}] /. {x->a,y->b}
```

でも

```
In[2] := jf[imo_,kuri_]:=D[f[x,y],{{x,y}}] /. {x->imo,y->kuri}
```

でも同じことになる。

(2) 微積分は早い段階で学ぶので、難しめの概念は避けたカリキュラムが採用されているが、多変数関数の高階導関数はテンソルとみなすべきである。*Mathematica* の `D[]` はそういう仕様になっているようで、*2*階の導関数を求めるには

```
In[] := D[f[x,y],{{x,y},2}]
```

とすれば良い。f が実数値なので結果は「行列」になるが、ベクトル値であったり、微分の階数が 3 以上であっても同様に計算出来ることに注目しよう。■

$$f'(x,y)=\begin{pmatrix}6x^2+6y^2-2 & 12xy\end{pmatrix},\quad H(x,y)=12\begin{pmatrix}x & y\\ y & x\end{pmatrix}$$

停留点を求めて、Hesse 行列を計算して、符号を調べてみる。

```
In[6] := sp=Solve[jf[x,y]=={0,0},{x,y}]
In[7] := H[x,y]/. sp
In[8] := Eigenvalues[H[x,y]]/. sp
In[9] := f[x,y]/.sp
```

$f'(x,y)=0$ となるのは

$$(x,y)=\left(0,\frac{1}{\sqrt{3}}\right),\left(0,-\frac{1}{\sqrt{3}}\right),\left(\frac{1}{\sqrt{3}},0\right),\left(-\frac{1}{\sqrt{3}},0\right).$$

$$H\left(0,\frac{1}{\sqrt{3}}\right)=\begin{pmatrix}0 & -4\sqrt{3}\\ -4\sqrt{3} & 0\end{pmatrix},\quad H\left(0,-\frac{1}{\sqrt{3}}\right)=\begin{pmatrix}0 & 4\sqrt{3}\\ 4\sqrt{3} & 0\end{pmatrix}$$

はともに (固有値が $\pm4\sqrt{3}$ なので) 不定符号であるから、$(x,y)=(0,\pm1/\sqrt{3})$ では f は極値を取らない。

$$H\left(\frac{1}{\sqrt{3}},0\right)=\begin{pmatrix}-4\sqrt{3} & 0\\ 0 & -4\sqrt{3}\end{pmatrix}$$

の固有値は $-4\sqrt{3}<0$ (重根) なので、$H\left(\frac{1}{\sqrt{3}},0\right)$ は負値である。ゆえに、$(x,y)=\left(\frac{1}{\sqrt{3}},0\right)$ で f は極大となり、極大値は $f\left(\frac{1}{\sqrt{3}},0\right)=\dfrac{4\sqrt{3}}{9}$.

$$H\left(-\frac{1}{\sqrt{3}},0\right)=\begin{pmatrix}4\sqrt{3} & 0\\ 0 & 4\sqrt{3}\end{pmatrix}.$$

の固有値は $4\sqrt{3}>0$ (重根) なので、$H\left(\frac{1}{\sqrt{3}},0\right)$ は正値である。ゆえに、$(x,y)=\left(-\frac{1}{\sqrt{3}},0\right)$ で f は極小となり、極小値は $f\left(-\frac{1}{\sqrt{3}},0\right)=-\dfrac{4\sqrt{3}}{9}$.

$(1,1)$ での接平面は $z = f'(1,1)\begin{pmatrix} x-1 \\ y-1 \end{pmatrix} + f(1,1)$ であるから、

```
In[10] := Simplify[jf[1,1].{x-1,y-1}+f[1,1]]
```

と計算して、これから

$$z = 2(5x + 6y - 8).$$

D.5 問題 5

どちらか一方を選んで解答せよ。(1) $f(r,\phi,\lambda) = \begin{pmatrix} r\cos\phi\cos\lambda \\ r\cos\phi\sin\lambda \\ r\sin\phi \end{pmatrix}$ に対して、$\det f'(r,\phi,\lambda)$ を求めよ。(2) $f(x) := \|x\|^{2-n}$ $(x \in \mathbf{R}^n \setminus \{0\})$ とおくとき、$\dfrac{\partial^2 f}{\partial x_1^2} + \cdots + \dfrac{\partial^2 f}{\partial x_n^2} = 0$ を示せ。

D.5.1 Mathematica でやってみる

(1) 緯度経度形式の極座標変換のヤコビアンを求めよう、という問題である。

$$f(r,\phi,\lambda) = \begin{pmatrix} r\cos\phi\cos\lambda \\ r\cos\phi\sin\lambda \\ r\sin\phi \end{pmatrix}$$

```
In[1] := f[r_,p_,l_]:={r Cos[p]Cos[l],r Cos[p]Sin[l], r Sin[p]}
In[2] := jf[a_,b_,c_]:=D[f[r,p,l],{{r,p,l}}] /. {r->a,p->b,l->c}
In[3] := MatrixForm[jf[r,p,l]]
In[4] := Det[jf[r,p,l]]
In[5] := Simplify[%]
```

$$f'(r,\phi,\lambda) = \begin{pmatrix} \cos\phi\cos\lambda & -r\sin\phi\cos\lambda & -r\cos\phi\sin\lambda \\ \cos\phi\sin\lambda & -r\sin\phi\sin\lambda & r\cos\phi\cos\lambda \\ \sin\phi & r\cos\phi & 0 \end{pmatrix},$$

$$\det f'(r,\phi,\lambda) = -r^2\cos\phi.\blacksquare$$

D.6 問題 6

$f, g\colon \mathbf{R}^2 \to \mathbf{R}$ を $f(x,y) := x^2 + y^2$, $g(x,y) := x^2 + 2xy + 3y^2 - 4$ で定め、$N_g := \{(x,y) \in \mathbf{R}^2; g(x,y) = 0\}$ とおく。
(1) N_g は $\mathbf{R}^2$ の有界閉集合であることを示せ。(2) 条件 $g(x,y) = 0$ のもとでの $f(x,y)$ の最大値と最小値を求めよ。

D.6.1 Mathematica でやってみる

Lagrange の未定乗数法の典型的な問題。

```
In[1] := g[x_,y_]:=x^2+2x y+3y^2-4
In[2] := Ng=ContourPlot[g[x,y]==0,{x,-3,3},{y,-3,3},ContourStyle->Green]
```

これで $N_g := \{(x,y) \in \mathbf{R}^2; g(x,y) = 0\}$ が楕円であることが分かる (もちろん判別式 D の符号を $D/4 = 1^2 - 1 \cdot 3 = -2 < 0$ と調べても良い)。

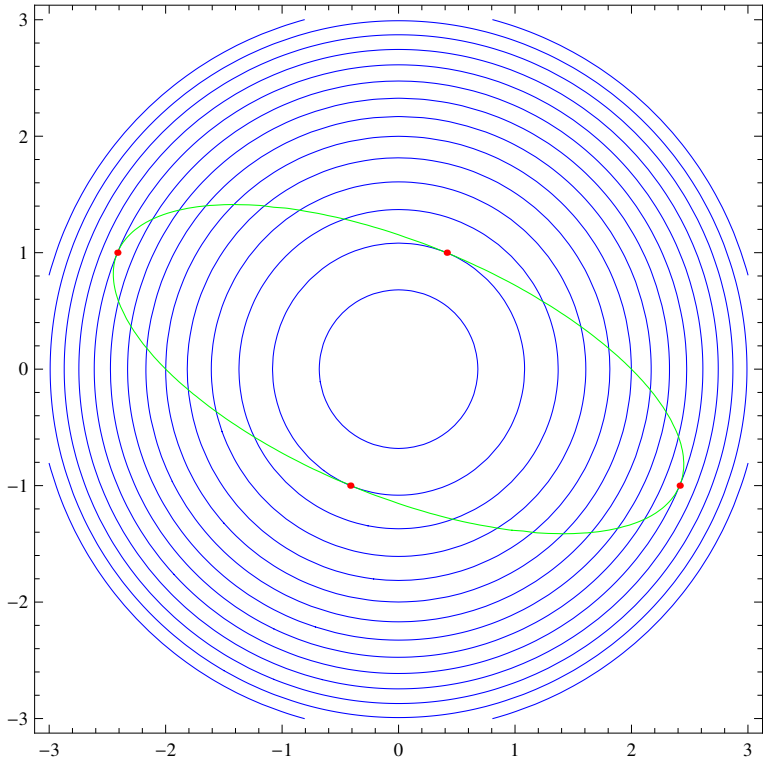


図 29: $g(x,y) = 0$ (緑) と f の等高線 (青) と極値点 (赤)

$\nabla g \neq 0$ on N_g が分かるので (省略する)、極値点は Lagrange の未定乗数法で見つかるはずである。そこで未定乗数法で極値点の候補を求め、それらの点における f の値を計算する。

```
In[3] := f[x_,y_]:=x^2+y^2
In[4] := s=Solve[{D[f[x,y]-L g[x,y],{{x,y}}]=={0,0}, g[x,y]==0},{x,y,L}]
In[5] := f[x,y]/.s
```

$\nabla f(x,y) = \lambda \nabla g(x,y)$, $g(x,y) = 0$ の解は

$$(x,y,\lambda) = \left(-1-\sqrt{2}, 1, 1+\frac{1}{\sqrt{2}}\right), \left(1-\sqrt{2}, -1, 1-\frac{1}{\sqrt{2}}\right),$$
$$\left(-1+\sqrt{2}, 1, 1-\frac{1}{\sqrt{2}}\right), \left(1+\sqrt{2}, -1, 1+\frac{1}{\sqrt{2}}\right).$$

これら極値点の候補での f の値は

$$f\left(-1-\sqrt{2},1\right)=4+2\sqrt{2},\quad f\left(1-\sqrt{2},-1\right)=4-2\sqrt{2},$$
$$f\left(-1+\sqrt{2},1\right)=4-2\sqrt{2},\quad f\left(1+\sqrt{2},-1\right)=4+2\sqrt{2}.$$

ゆえに $(x,y)=\pm(1+\sqrt{2},-1)$ のとき最大値 $4+2\sqrt{2}$, $(x,y)=\pm(-1+\sqrt{2},1)$ のとき最小値 $4-2\sqrt{2}$ を取る。

```
In[6] := fc=ContourPlot[f[x,y],{x,-3,3},{y,-3,3},
              ContourShading->False,ContourStyle->Blue,
              Contours->Table[4+2Sqrt[2]/4*i,{i,-8,8}]]
In[7] := sp = ListPlot[{x,y} /. s, PlotStyle->Red]
In[8] := gr=Show[fc,Ng,sp]
```

D.7 追加の問題

行列 $\begin{pmatrix}1&2&3\\2&2&0\\3&0&3\end{pmatrix}$ が正値であるか、負値であるか、不定符号であるか、そのどれでもないか、判定せよ。

D.7.1 Mathematica でやってみる

出題者の意向は、授業で説明した主座小行列式の符号から判定する方法を使ってもらう、ということである。

```
In[1] := a={{1,2,3},{2,2,0},{3,0,3}}
In[2] := p[n_]:=Take[a,n,n]
In[3] := Table[MatrixForm[p[i]],{i,1,3}]
In[4] := Table[Det[p[i]],{i,1,3}]
```

a の n 次の主座小行列を Take[a,n,n] で求めるのが気づきにくいかもしれない。最後の結果だけ欲しければ、

```
In[2] := Table[Det[Take[a,i,i]], {i,1,3}]
```

とすれば良い。

$$\det A_1=1,\quad \det A_2=-2,\quad \det A_3=-24.$$

$\det A_3<0$ より、A の 3 つの固有値の積が負であるから、A の固有値は、「負が 3 つ」か「正 2 つ、負 1 つ」のどちらかである。前者であれば A が負値で、その場合 $\det A_k$ は $k=1,2,3$ の順に、負、正、負となるはずであるが、そうなっていない。ゆえに A の固有値は「正 2 つ、負 1 つ」であり、正負いずれも存在することから、A は 不定符号である。

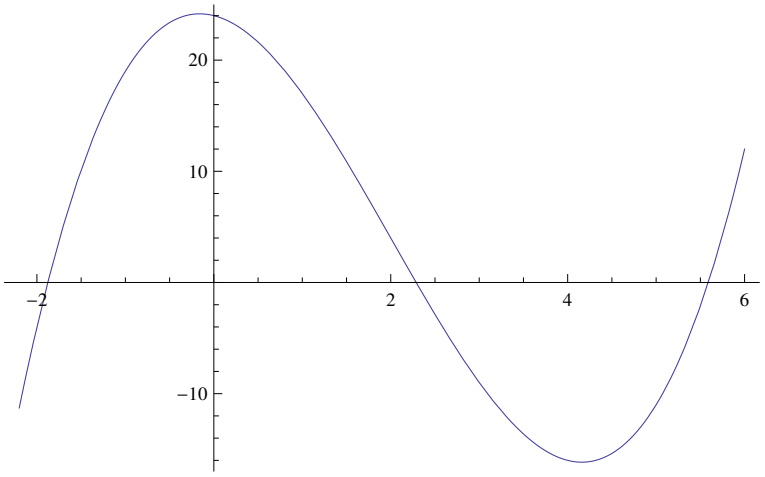
なお、

```
In[5] := Eigenvalues[a]
```

としても Root[$24-2\sharp 1-6\sharp 1^2+\sharp^3$&,1] のような結果が返ってきて、N[%] で近似値を得ることは出来るが、厳密な結果を得るのはそう簡単ではない。行列の特性多項式を調べてみよう。

```
In[6] := f[L_]:=Det[L*IdentityMatrix[3]-a]
In[7] := Plot[f[x],{x,-2.2,6}]
```

これが分かれば、中間値の定理を利用して、簡単かつ厳密な証明が可能である ($f(-2)=-4<0$,



$f(0)=24>0$, $f(4)=-16<0$, $f(6)=12>0$)。

D.8 追加の問題 1

次の各集合が、開集合であるかどうか、閉集合であるかどうか、有界であるかどうかを判定せよ (例えば開集合であれば開集合であることを証明せよ)。

(1) $A=\{(x,y)\in\mathbf{R}^2; x^2-y^2-1=0\}$ (2) $B=\{(x,y)\in\mathbf{R}^2; x^2+2xy+3y^2<1\}$

D.8.1 Mathematica でやってみよう

(1) が閉集合であり、(2) が開集合であることは、それぞれ等式と (等号抜きの) 不等式で定義されていることから分かる。有界非有界はどういう形であるか調べることが出来れば簡単で、Mathematica で描いてみることは良いアプローチであろう。

```
In[1] := g1=ContourPlot[x^2-y^2-1==0,{x,-4,4},{y,-4,4}]
In[2] := g2=RegionPlot[x^2+2x y+3y^2<1,{x,-2,2},{y,-2,2}]
```

そういえば

$$\frac{x^2}{a^2}-\frac{y^2}{b^2}=1$$

は双曲線の方程式であったことに気付くだろうか。

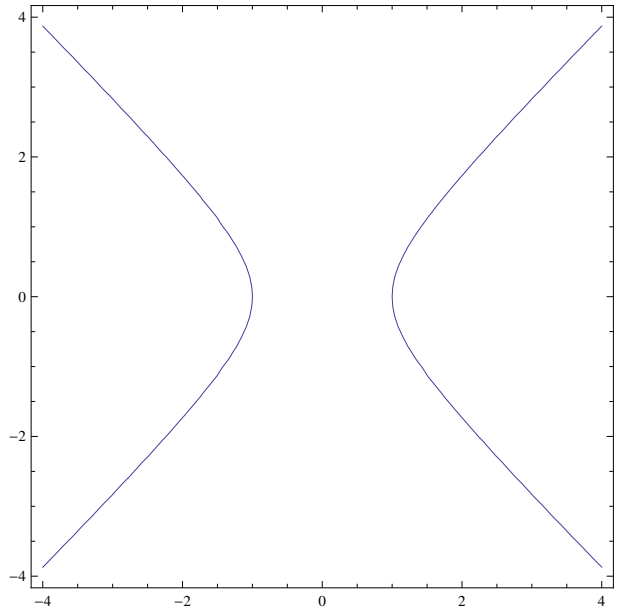


図 30: $x^2 - y^2 = 1$

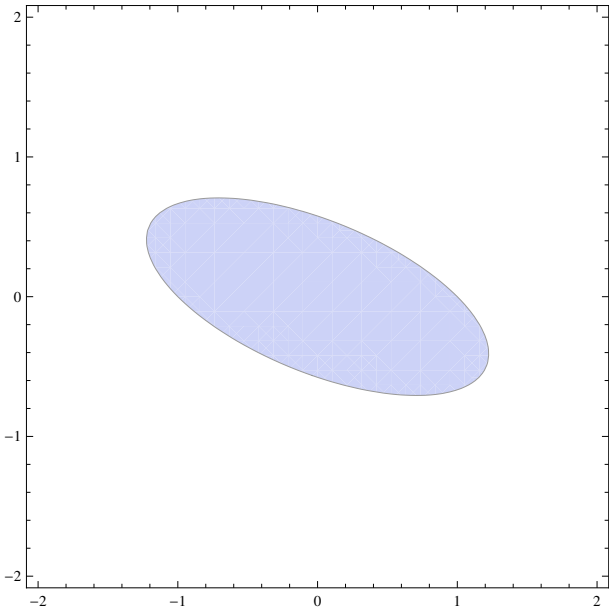


図 31: $x^2 + 2xy + 3y^2 < 1$

D.9　追加の問題 3

C^2 級の関数 $u\colon \mathbf{R}^2 \ni (x,t) \mapsto u(x,t) \in \mathbf{R}$ と正定数 c があるとき、

$$\xi = x - ct, \quad \eta = x + ct, \quad v(\xi,\eta) = u(x,t), \quad \text{すなわち} \quad v(\xi,\eta) := u\left(\frac{\xi+\eta}{2}, \frac{\eta-\xi}{2c}\right)$$

とおくとき、$\dfrac{1}{c^2}u_{tt} - u_{xx}$ を v の偏導関数を用いて表せ。

D.9.1　Mathematica でやってみる

```
In[1] := u[x_,t_]:=v[x-c t,x+c t]
In[2] := D[u[x,t],{t,2}]/c^2-D[u[x,t],{x,2}]
In[3] := Simplify[%]
```

E　行列の固有値問題の験算

(某月某日)「対称行列

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix}$$

を実直交行列によって対角化せよ」という問題の答えの験算をする。

行列を入力して確認する。

```
a={{0, 1, 1}, {1, 0, -1}, {1, -1, 0}}
MatrixForm[a]
```

固有値を求める。Eigenvalues[] を使えば一発で求まるが、特性多項式も計算しておこう。

```
Eigenvalues[a]

f[x_]:=Det[x IdentityMatrix[3] -a]
f[x]

Solve[f[x]==0,x]
```

固有値は $-2, 1$ (重根) であることが分かる。

固有ベクトルは

```
{v1,v2,v3}=Eigenvectors[a]
```

これで

$$\boldsymbol{v}_1 = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}, \quad \boldsymbol{v}_2 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{v}_3 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

が求まるが、念のために固有値がそれぞれ $-2, 1, 1$ であることを確認する。

```
a.v1

a.v1-(-2)v1

a.v2-1 v2

a.v3-1 v3
```

後の 3 個のベクトルはいずれも $\mathbf{0}$.

Gram-Schmidt の直交化法によって正規直交基底を求める。

```
u1=v1/Norm[v1]

u2=v2/Norm[v2]
```

$\boldsymbol{v}_2$ は (異なる固有値に属する固有ベクトルなので) $\boldsymbol{v}_1$ とは直交しているので、単に正規化す

れば良い。$\boldsymbol{v}_3$ からは $\boldsymbol{v}_2$ 成分を削除してから正規化する必要がある。

$$\boldsymbol{v}_3' = \boldsymbol{v}_3 - (\boldsymbol{v}_3, \boldsymbol{u}_2)\boldsymbol{u}_2,$$
$$\boldsymbol{u}_3 = \frac{1}{\|\boldsymbol{v}_3'\|}\boldsymbol{v}_3'$$

とするわけである。

```
vv3=v3-v3.u2 u2
v2.vv3
u3=vv3/Nomr[vv3]
```

$$\boldsymbol{u}_1 = \begin{pmatrix} -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{pmatrix}, \quad \boldsymbol{u}_2 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{3}} \end{pmatrix}, \quad \boldsymbol{u}_3 = \begin{pmatrix} \frac{1}{\sqrt{6}} \\ \sqrt{\frac{2}{3}} \\ -\frac{1}{\sqrt{6}} \end{pmatrix}.$$

これを並べた行列 U を作って表示し、$U^T AU$ を計算して固有値が対角線に並んだ対角行列になっていることを確認する。

```
u=Transpose[{u1,u2,u3}]
MatrixForm[u]

d=Simplify[Transpose[u].a.u]
MatrixForm[d]
```

結果は無事 $\begin{pmatrix} -2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ になった。

F　初等幾何の図を描く

(工事中。そのうち他との調整をはかる。)

F.1　はじめに

- 自分でソフト (例えば Mathematica) を使って描く
- 自分で手を使って描いたものを電子化する
- 誰かが描いたものを電子化する

電子化の方法としては

- スキャナー

- デジカメ

既に電子化されて画像になっているものから、適当な部分を切り取るには、プレビューの [ファイル] の「スクリーンショットを撮る」を使うと良い。

F.2 TEX への取り込み

(ここの記述は古いので、あまり参考にしないで下さい。)

F.2.1 PostScript データ

伝統的に、TEX に取り込む図は PostScript 形式で作ると良いとされてきた。
プリアンブルに

```
\usepakage[dvips]{graphicx}
```

と書いておいて、

```
\includegraphics[width=10cm]{eps/mygraph.eps}
```

mygraph.eps を取り込む TEX 文書の例

```
\documentclass[12pt]{jarticle}
\usepackage[dvips]{graphicx}
\begin{document}

\begin{figure}[htbp]
\centering
\includegraphics[width=10cm]{eps/mygraph.eps}
\caption{三角形}% 図の説明
\end{figure}

\end{document}
```

JPEG 画像は PostScript に変換して取り込むのが簡単である。

```
jpeg2ps mygraph.jpg > mygraph.eps
```

ここで用いた jpeg2ps は MacPorts がインストールされていれば、次のコマンドで簡単にインストールできる。

```
$ sudo port install jpeg2ps
```

F.2.2 画像ファイル

画像ファイルには、PDF, PNG, JPEG など色々ある。(デジカメのデータは JPEG が普通だと思われるが、Mathematica で描いた図を画像ファイルにするときは PDF が良いように感じている。)

```
\usepackage[dvipdfmx]{graphicx}
```

と書いておいて、

```
\includegraphics[width=10cm]{eps/mygraph.pdf}
```

とすれば良い。

蛇足的な注意 \usepackage[dvips]{graphicx} を暗黙の仮定としている伝統がある。他の設定とバッティングしないように調整が必要なことがある。以下は私がひっかかったもの。

1. \usepackage[dvips]{color} は、\usepackage[dvipdfmx]{color} に変更する (これはとても有名な話)。
2. (これは個人的な事情でマイナーすぎるかもしれないが参考まで) gouji.sty は内部で

   ```
   \RequirePackage[dvips]{graphicx}
   ```

   としているので、矛盾が生じる。dvips を dvipdfmx にすれば良い？

F.3 Mathematica メモ

曲線などは、Plot[] や ParametricPlot[] などで描けば良い。
点や線分、円などはグラフィックス・プリミティブを用いて描くのが便利かも。
点の座標はリストとして定義する。

```
f1 = {-Sqrt[3], 0}; f2 = {Sqrt[3], 0}; p = {2 Cos[Pi/4], Sin[Pi/4]};
o = {0, 0};
```

点、線分、円などは、Point[], Line[], Circle[] などで定義する。「先頭の文字は大文字を使わないように」と言われているので、名前の付け方が悩ましい。ここでは $ を使ってみた (悪趣味かも知れない)。

```
$F1 = Point[f1]; $F2 = Point[f2]; $P = Point[p];
$F1P = Line[{f1, p}]; $F2P = Line[{f2, p}];
r = 2; $C = Circle[o, r];
```

ここまで定義した点 F_1, F_2, P, 線分 F_1P, F_2P, 円 C からグラフィックス g1 を作る。

```
g1 = Graphics[{$F1,
               $F2,
               $P,
               $F1P, $F2P, $C}, Axes -> True]
```

(最後に Axes->True とすることで座標軸を描いています。これは好みの問題かも。)

点の説明などの文字列をつけておくのが良いかも。そうするためには、Text[] を使います。

```
g1 = Graphics[{$F1, Text["F1", f1 - {0, 0.1}],
               $F2, Text["F2", f2 - {0, 0.1}],
               $P, Text["P(x,y)", p + {0.1, 0.1}],
               $F1P, $F2P, $C}, Axes -> True]
```

何だか楕円を描くことの方が楽ですね。

```
g2 = ParametricPlot[{2 Cos[t], Sin[t]}, {t, 0, 2 Pi}];
```

g1, g2 を合わせて描きます。

```
g = Show[g1, g2, PlotRange -> All]
```

PlotRange->All は、指定したグラフィックスをすべて描画できる範囲を確保する指示ですが、今の場合は実は必要がありません (g1 の範囲は g2 の範囲よりも大きいので)。

これをファイルに出力するには

```
Export["mygraph.eps", g]

あるいは

Export["mygraph.pdf", g]
```

のようにして出力します。

残念ながら、現在の Mathematica で出力した PostScript データは、しばしば巨大なサイズになってしまい、TEX では扱い辛くなることがあります。代わりに PDF のようなフォーマットを採用するか、一度 JPEG で出力してから PostScript に変換するなどの工夫が必要です。

Mathematica で変数 g に記憶されたグラフィックスを出力

```
In[] := Export["mygraph.jpg", g]

ターミナルで次のようにして変換します。

jpeg2ps mygraph.jpg > mygraph.eps
```

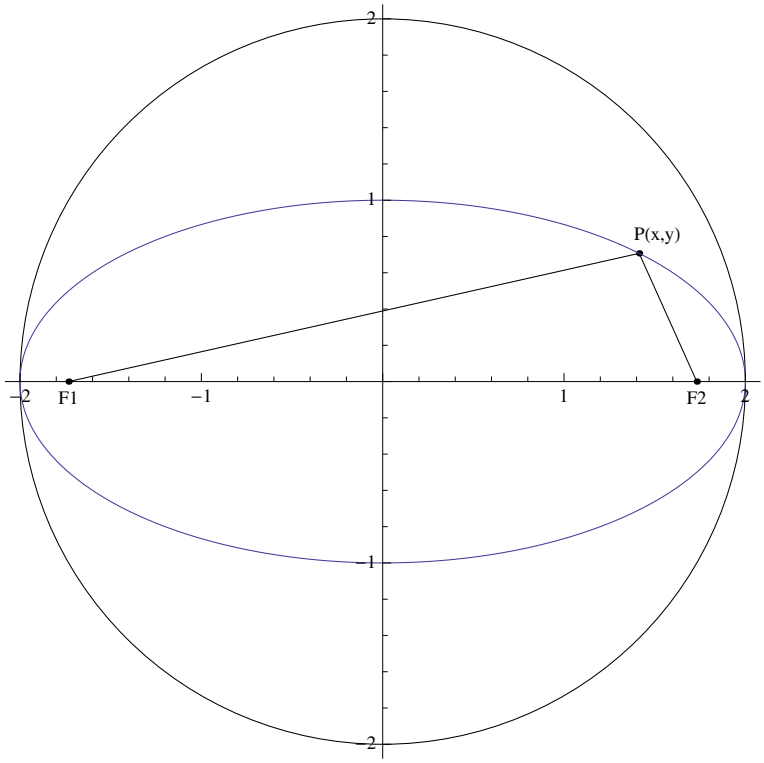


図 32: eps (昔はこれ一択だったのですが)

実は Text[] はかなり高機能である。Text[Style[テキスト，大きさ]，座標，オフセット] とか、色々工夫が出来る。「テキスト」にも単なる文字列でなく、Mathematica の式が書ける。下付き文字のある F_1 は、F の後に Control+_ で入力出来る (Mathematica の [ヘルプ] メニューの検索で、「二次元式の入力」を見てみよう))。$P(x, y)$ を出力したい場合は P[x,y] とするとか。

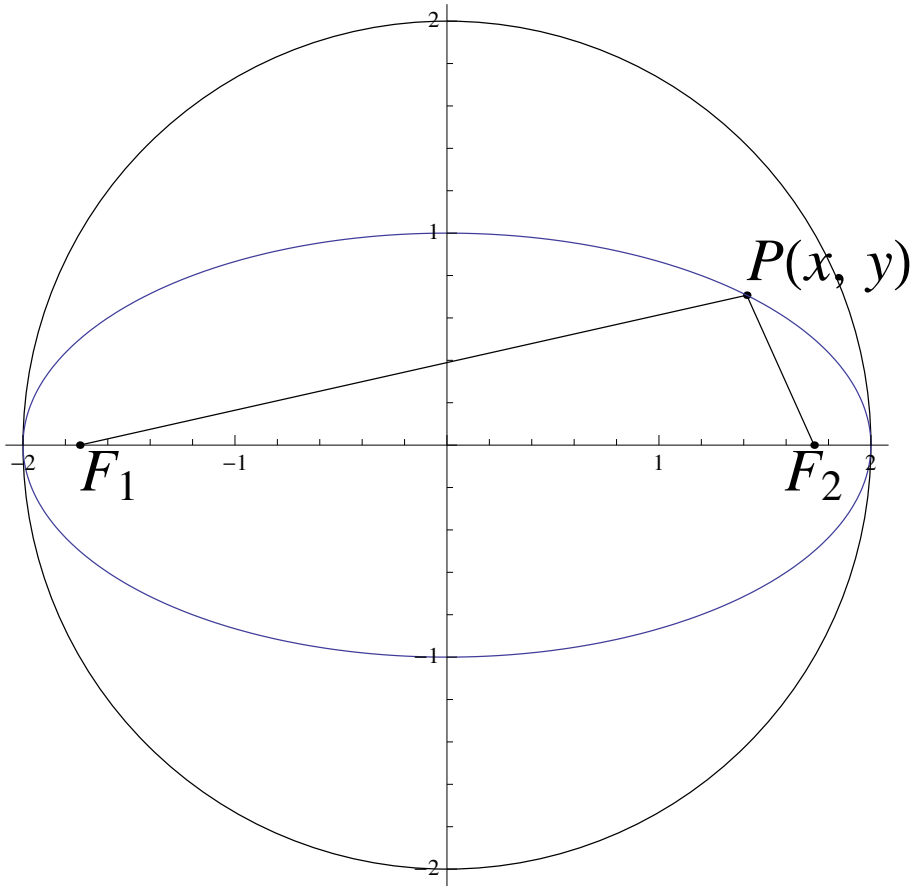


図 33: ちょっとお色直し

- 点を大きくするには、事前に PointSize[数値] をつける。
- 色をつけるには、Red, Green, Blue, Black, White, Cyan, Magenta, Yellow, Brown, Orange, Pink, Purple などのように名前を指定するか、Hue[] などの関数を使う。Hue[] の使い方はオンライン・マニュアルを見ること。
- 線の太さは Thin, Thick という簡単な指定以外に、Thickness[数値] で細かく指定できる。

```
Graphics[{Red, Thin, Line[{{0, 0}, {0, 1}}],
          Blue, Thick, Dotted, Line[{{1, 0}, {1, 1}}],
          Green, Thickness[0.02], Line[{{2, 0}, {2, 1}}]}]
```

- 例えば Plot[] で PlotStyle->{ } で、色の指定、線の太さ、線種の指定が出来るけれど、それと同様のことが指定できるわけだ。

G　イメージ・ファイルを扱う (工事中)

(gazou.tex というのを書いている。そちらと同期をとること。)

G.1　目標

MATLAB を使って画像処理をしている人が多いように思う。ここでは Mathematica で真似をしてみる。

実験に用いた画像データ “lena.tiff” については、「レナ（画像データ）[20] や Lenna[21], The Lenna Story[22] を見よ。

G.2　イメージファイルを読み込んでグラフィックスとする

```
In[] := img = Import["work/gazou/lena.tiff"];
In[] := Show[img]
```

G.3　グラフィックスの画素値を得る

```
In[] := data = ImageData[img];
```

大きさを調べてみる。正攻法は

```
In[] := ImageDimension[img];
```

であろう。

```
In[] := Length[data]
Out[] = 512
In[] := Table[Length[data[[i]]],{i,512}]
Out[] = {512,512,...,512}
In[] := Table[Length[data[[i,j]]], {i, 10}, {j, 10}]
Out[] = {{3,3,..,3},{3,3,...,3},...,{3,3,...,3}}
```

つまり data は、大きさが $512 \times 512 \times 3$ の 3 次元配列である。

[20] http://ja.wikipedia.org/wiki/レナ_(画像データ)
[21] http://en.wikipedia.org/wiki/Lenna
[22] http://www.cs.cmu.edu/~chuck/lennapg/

G.4　カラーをグレースケールに変換

カラー画像をグレースケールにする

```
In[] := data2 = Table[Table[(Sum[data[[i,j,k]],{k,1,3}])/3,{j,512}],{i,512}];
In[] := gry=Image[data2]
```

G.5　R,G,B 画像の取り出し

RGB 画像の取り出し

```
In[] := {r,g,b} = Table[Table[Table[data[[i,j,k]], {j,512}], {i,512}],{k,3}];
In[] := Table[Image[x],{x,{r,g,b}}]
```

今後の予定

- ヒストグラム (度数分布グラフ) → コントラストが分かる
- エッジ検出 (微分フィルタを利用)
- ノイズ除去 (移動平均フィルタを利用, メディアン・フィルタ)

H misc

H.1 Mathematica でプレゼン

Mathematica の計算の様子を画面に移す場合、フォントを大きいものにしないと見づらい。

Mac の場合は、Mathematica の [環境設定] の [詳細] タブで、[オプションインスペクタを開く] ボタンを押し、「オプションの設定」で、[ノートブック設定] の [表示設定] で、“ScreenStyleEnvironment” を “Working(* 作業用 *)” から “Presentation” に変える。

H.2 Mathematica の Floor[]

授業の課題で学生が発見したのだが (こういうのは結構嬉しい)、Floor[] の極限が Mathematica 7 で計算できなかったのが、Mathematica 9 では出来るようになったとか。

```
Limit[Floor[x],x->Infinity}

Limit[(1+1/x)^Floor[x],x->Infinity]
```

H.3 Mathematica の間違える例

これも学生の発見。

高木貞治『解析概論』にある収束する広義積分

$$\int_0^\infty \frac{x}{1+x^6\sin^2 x}dx$$

を、Mathematica が発散すると答える (もちろん Mathematica の間違い) ことを発見した (最新の Mathmatica 9 で試してみても,「…の積分は $(0,\infty)$ で収束しません」と応答が返って来る)。

201x 年 7 月某日の SNS

昨日の情報処理 2 で学生から質問あり。
レポート課題で、Mathematica に教科書の問題を解かせるなりして、Mathematica が間違えた例を見つけて、その理由を考えてみなさい、というのを出してあるのだけど (この課題は毎年出しているのですが、最近はやってくれる人が少ないです)、その流れで

$$\int_0^\infty \frac{x}{1+x^6\sin^2 x}dx$$

という広義積分について、本には (ちらと見たら高木貞治「解析概論」ぽかった…後で確認出来ました) 収束すると書いてあるけれど、Mathematica は発散すると答えを出します、これが Mathematica の間違えている例になるでしょうか、との質問。
忙しかったので、Mathematica おかしいみたいですね、時間がないのでまた今度、と言ったのですが、今日になって気がつきました。このちょっと難し目の積分、どうやって計算するか、昨年研究発表を聴いたことがあるのでした。

大浦拓哉, ある非有界無限区間積分の高速高精度計算

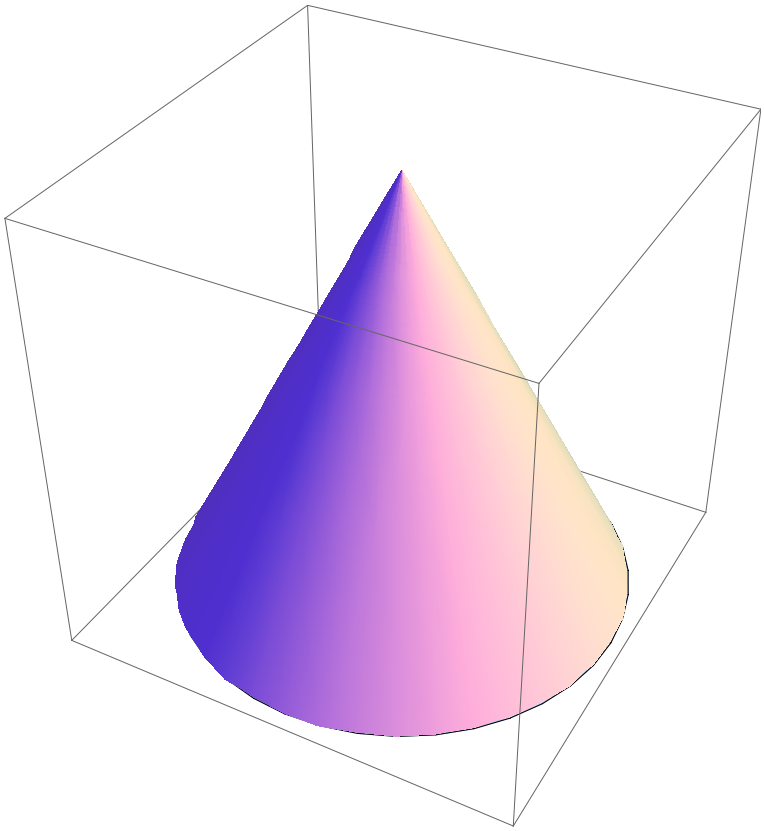
http://www.kurims.kyoto-u.ac.jp/~ooura/papers/toda53a.pdf

被積分関数ぱっと見は分母に x^6 があって、遠方で小さくなりそうですが、sin があるせいで、分母が $x=n\pi$ のところで 1 になるので、関数値自体は $n\pi$ になって全然小さくなってくれない。というわけで積分の収束を証明するだけでも大変です。上の論文には収束の確認は、Goursat と G. H. Hardy による、とあります。偉い人二人の名前が出て来る由緒正しい積分だったわけですね。それが高木先生の解析概論に載っていた、と。
高木先生の放ったボールを学生がキャッチ。ちょっと楽しい出来事でした。
ちなみに、この収束証明もやさしくない広義積分、実は 100 万桁以上計算した、というのが上の論文の内容です（かなりすごい)。

H.4 円錐を描く

まず、Mathematica のグラフィックス・プリミティブ `Cone[]` を使って描いてみます。

```
Graphics3D[Cone[]]
```



Mathematica のグラフィックス・プリミティブに慣れるというのも有意義なのですが，ここは円錐を式で表現して描いてみます。

$z = \sqrt{x^2 + y^2}$ で OK のはず。

```
Plot3D[x^2+y^2,{x,-1,1},{y,-1,1}]
```

イメージと違う。いや、まあ正しいんだけど。

ひっくり返して、高くしましょう。ついでに描画範囲を円盤にすると、ミッキーの帽子風になります。

```
Plot3D[-5Sqrt[x^2+y^2],{x,-1,1},{y,-1,1},
       RegionFunction->Function[{x,y,z},x^2+y^2<1],BoxRatios->Automatic]
```

パラメーター曲面

$$x = r\cos\theta, \quad y = r\sin\theta, \quad z = -5r \quad (r \geq 0,\ \theta \in [0, 2\pi])$$

としても表せます。

```
ParametricPlot3D[{r Cos[t], r Sin[t], -5 r},
   {r,0,2}, {t,0,2Pi}, BoxRatios->Automatic]
```

ContourPlot3D[] や RegionPlot3D[] も面白い。

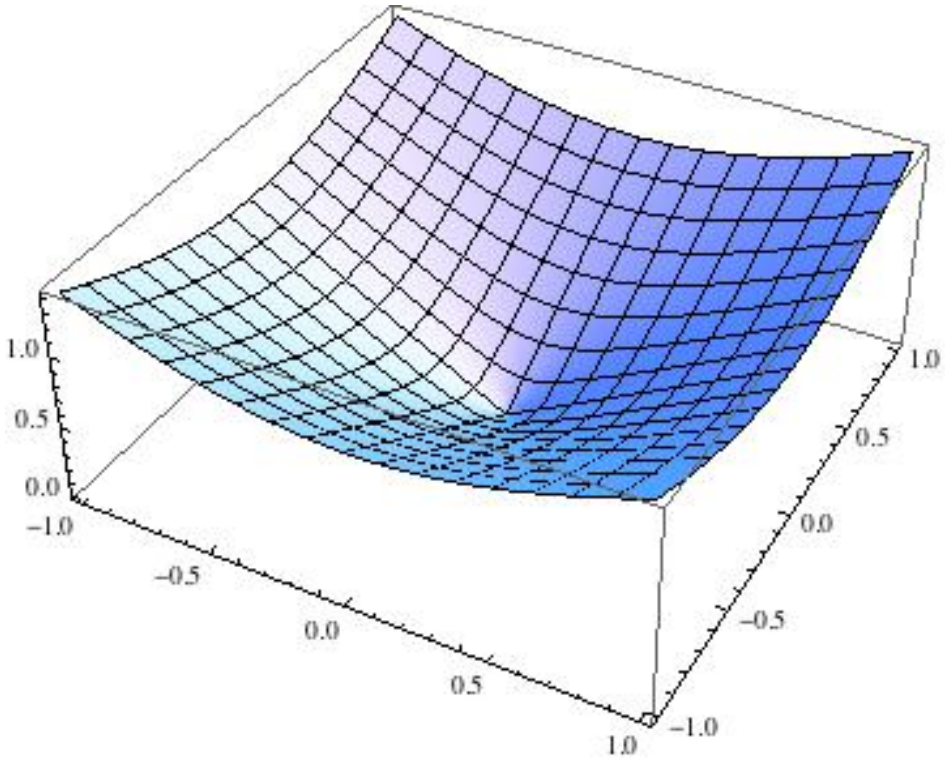


図 34: $z = \sqrt{z^2 + y^2}$

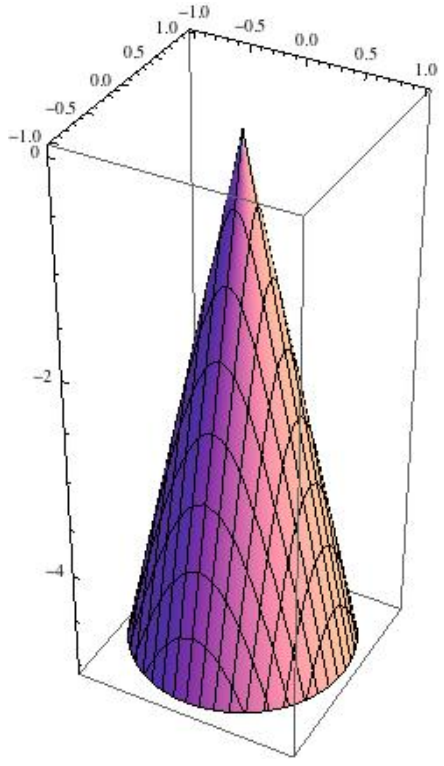


図 35: $z = -5\sqrt{z^2 + y^2}$

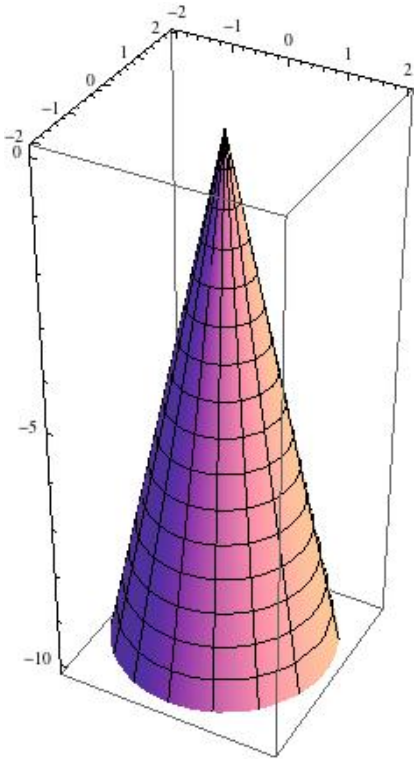


図 36: $z = r\cos\theta,\ y = r\sin\theta,\ z = -5r$

```
g=ContourPlot3D[z^2 - x^2 - y^2 == 0, {x, -1, 1}, {y, -1, 1}, {z, -1, 1}]

g2=RegionPlot3D[z^2 - x^2 - y^2 > 0, {x, -1, 1}, {y, -1, 1}, {z, -1, 1}]
```

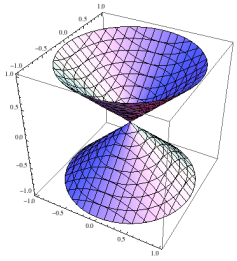


図 37: ContourPlot3D[]

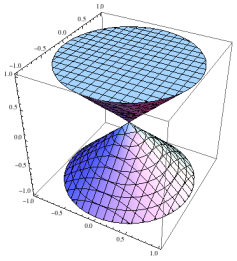


図 38: RegionPlot3D[]