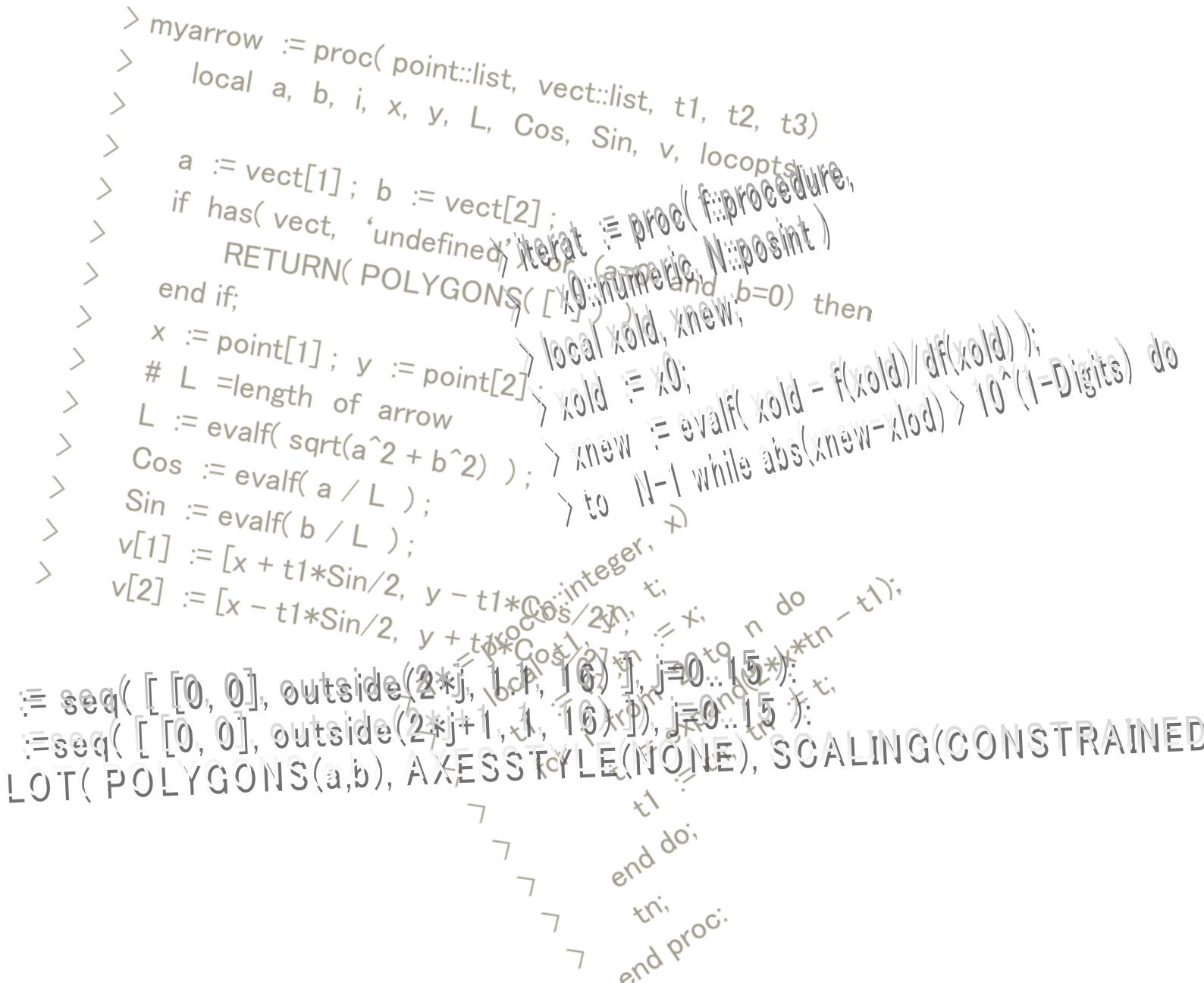
Maple 11

ビギナーズ・チュートリアル

数式処理ソフトウェア



1. はじめに

1.1. 数式処理ソフトウェア Maple の歴史と特徴

『Maple[1]』（メイプル）は、1980 年 11 月にカナダのウォータールー大学記号計算研究グループで生まれた数式処理システムです。そのコンセプトは、『研究者や学生の誰もが手軽に利用でき、可搬性のある数式処理システムを作り上げること』にありました。プロジェクト開始当時の創始メンバーである Keith Geddes 教授は、今もウォータールー大学において数式処理の研究に携わり、最先端の数式処理と数値計算の融合などに貢献しています。

数式処理とは、文字通り数式を変数のまま計算する技術を意味しています。例えば、中学校で習う２次方程式の解の公式も、Maple では以下のようなコマンド（Maple に計算させるための命令）をキーボードからタイプすることで計算することができます。

```
> solve(a*x^2+b*x+c=0,x);
```

$$-\frac{b-\sqrt{b^2-4ac}}{2a}, -\frac{b+\sqrt{b^2-4ac}}{2a}$$

Maple などの数式処理システムでは、このような数式処理の計算以外に、以下の３つの大きな特徴を持っています。

【特徴１】大きな数（長い桁数）の計算ができる

使用するコンピュータのメモリ量にもよりますが、100 の階乗を計算したり、円周率πや無理数などを 100 万桁まで求めるといった任意の桁長の計算が手軽に実現できます。

【特徴２】関数やデータのグラフィックスを手軽に描画できる

教科書に出てくる数学の関数を簡単にグラフとして描画することができます。

【特徴３】自作のプログラムを作ることができる

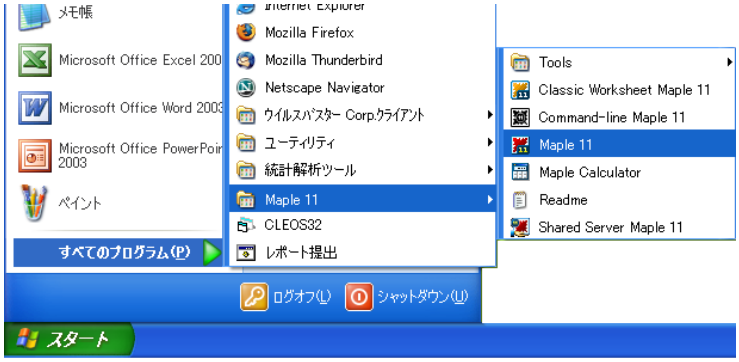
Maple に用意されているコマンドや関数を組み合わせて、ユーザが独自の計算プログラムを作ることができます。

このチュートリアルでは、Maple の使い方を中心に上記３つの特徴についても学んでいきます。

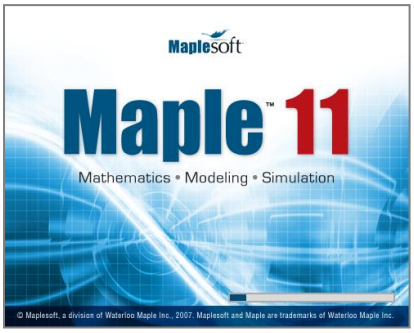
[1] 詳しくは http:// /www.cybernet.co.jp/Maple または http://www.maplesoft.com を参照

1.2. Maple を起動する

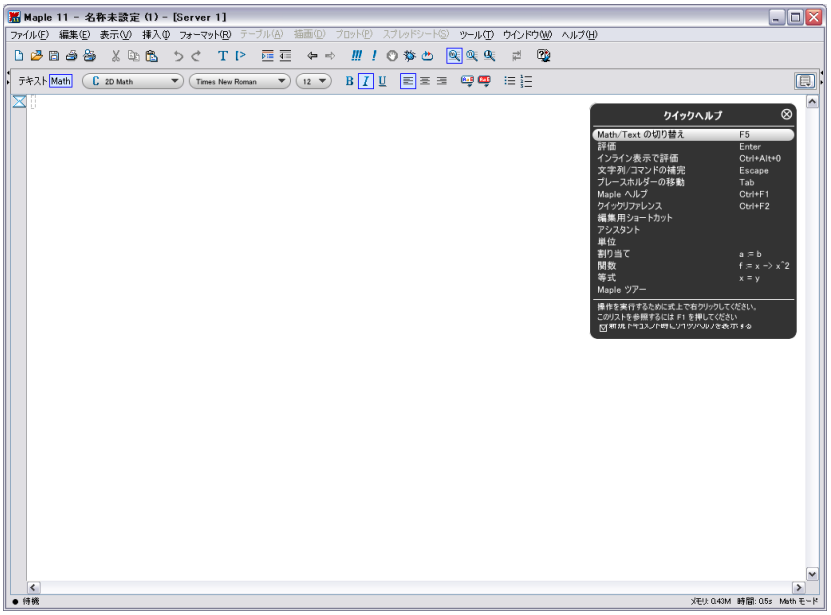
それでは、さっそく Maple を起動してみます。Maple 11 を起動するには、Windows の[スタート]メニューから[Maple 11]のプログラムグループを選択して、[Maple 11]のアイコンをクリックします。



Maple 11 のアイコンをクリックすると、Maple 11 のスプラッシュウィンドウが表示されて、Maple 11 の起動が開始されます。



スプラッシュウィンドウの表示中に、Maple は起動のために必要なファイルを読み込みます。Maple が正しく起動すると、次のように Maple 11 の GUI（グラフィカルユーザインターフェイス）が表示されます。



1.3. ２つのファイルモードと入力形態

Maple で新規のファイルを開く際には、２つのファイルモードが提供されています。ファイルモードのオプション設定は、「本テキストでのオプションの変更方法（pp.4)」を参照してください。

【２つのファイルモード】

① ワークシートモード

旧バージョンの Maple で利用されてきたファイルモードで、Maple の計算コマンドのためのプロンプト（入力記号、通常は赤色の大記号「>」が用いられます）が常に表示されます。Maple 上でコマンドやプログラミングを中心に作業する場合に用いるファイルモードです。

② ドキュメントモード

Maple 10 以降のバージョンで採用されているファイルモードで、コマンド入力のためのプロンプトは表示されません。Maple 上でレポートを書いたり、体裁の整った文書を作成するときに用いるモードです。

また、いずれのファイルモードでも、Maple のコマンドを入力するときには２つの入力形態が提供されています。入力形態のオプション設定は、「本テキストでのオプションの変更方法（pp.4)」を参照してください。

【２つの入力形態】

① Maple Input（テキスト入力）

コマンドを通常の文字と同じ形式で入力する方法です。例えば、2 次方程式の解を求めるときには `solve(a*x^2+b*x+c=0,x);` とタイプします。Maple Input の場合は、コマンドの終端に必ずセミコロン（；）またはコロン（：）を付加してください。

Maple Input のデフォルトの設定では、赤字で表示されます。（計算結果は青色表示です）

【コマンドを入力するときの注意点】

Maple では、コマンドの終端（行端ではないことに注意）に必ず「；」（セミコロン）または「：」（コロン）が必要です。

セミコロンは出力をそのまま表示し、コロンでは出力を抑制する（計算は実行するが表示しない）

```
> 10!;
                    3628800

> 10!:
```

② 2D Math Input （数式入力）

分数や指数、微積分記号などを含む数式を教科書やレポートなどでも見慣れた数式として表示・計算できるための入力形態です。例えば、2 次方程式の解を求めるときには、

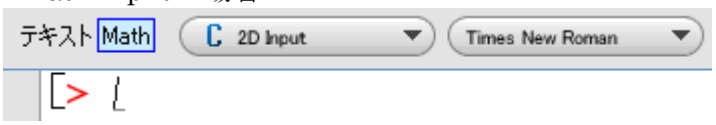
$$solve\left(a \cdot x^2 + b \cdot x + c = 0,\ x\right)$$

という形態で入力できます。Maple のデフォルトの入力形態はこの 2D Math Input モードになっています。2D Math Input では、コマンドの終端にセミコロンは付加しなくても計算が可能です。（付加しても問題はありません）

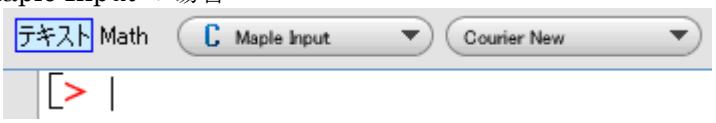
2D Math Input のデフォルトの文字色は黒字です。（出力は青色表示）

これらの入力形態は、F5キーを押すことで相互に切り替えることができます。

2D Math Input の場合



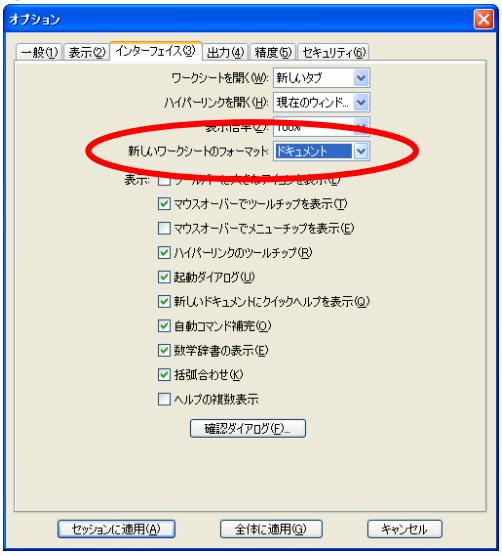
Maple Input の場合



デフォルトの設定では、2D Math Input 時はカーソルが斜体に、Maple Input（テキスト）のときはカーソルが垂直になっています。

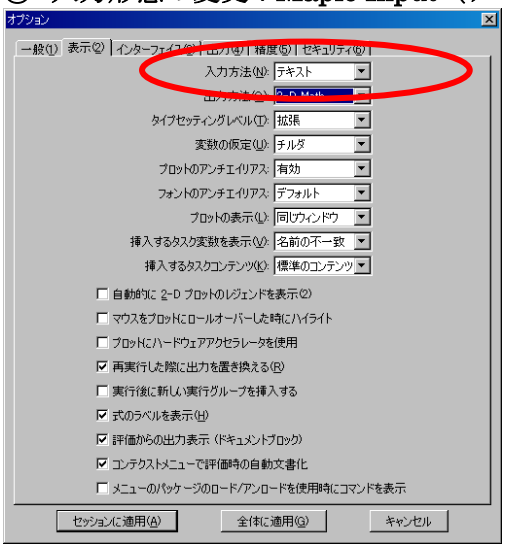
1.4. 本テキストでのオプションの変更方法

① ファイルモードの変更：ワークシートモード



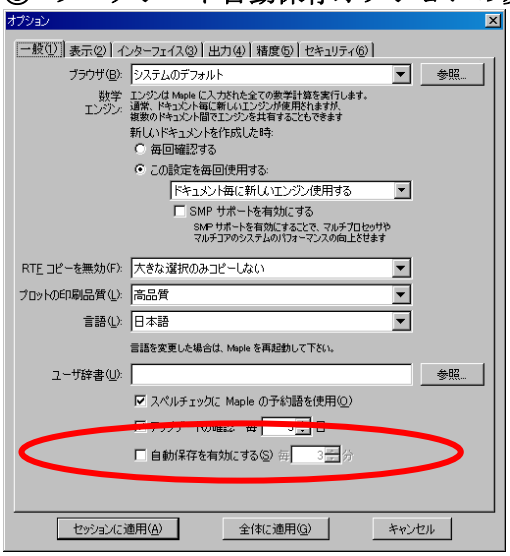
新規ファイル作成時にどちらのファイルモードを利用するかは、[ツール]メニューから[オプション]を選択してオプションダイアログを表示して、[インターフェイス]タブにある**「新しいワークシートのフォーマット」**の設定を変更することで指定できます。

② 入力形態の変更：Maple Input（テキスト）



デフォルトの入力形態は、[ツール]メニューの[オプション]を選択してオプションダイアログを表示し「表示」タブの入力方法の設定を「テキスト」へ変更します。

③ ワークシート自動保存オプションの変更：自動保存なし



最後に、[一般]タブの最下部にある**「自動保存を有効にする」のチェックボックスをはずして**、自動保存設定を無効にしてください。

これは、自動保存を有効にしていると Maple が急に止まってしまったように見えることがあるためです。(特に、大きな出力を必要とする計算を行うときは、自動保存に時間がかかることもあります)

1.5. 新規ファイルをワークシートモードで開く

それでは、ここから実際にコマンドをいろいろとタイプして計算を実施していくために、[ファイル]メニューから[新規作成]をポイントし、**[ワークシートモード]**を選択して新しいファイルを作成してください。

もしくは、ツールバーのファイル新規作成アイコン（  ）をクリックするか、または Ctrl+N キーで開くことも可能です。

1.6. ツールバーの説明

Maple の GUI（グラフィカルユーザインターフェイス）には、Microsoft Word や Excel などと同様なツールバーが用意されています。それぞれのボタンについては以下の説明を参照してください。

ファイル操作と編集

左のボタンから順に以下のような機能が割り当てられています。

① ファイルの新規作成（デフォルトのファイルモード設定によります）
② ファイルを開く
③ 現在開いているファイルを保存する
④ 現在開いているファイルを印刷する
⑤ ファイル印刷前にプレビューする
⑥ 選択した部分を切り取る
⑦ 選択した部分をクリップボードにコピーする
⑧ クリップボードの内容を現在のワークシートに貼り付ける
⑨ 直前の操作を元に戻す（アンドゥ）
⑩ 直前の操作を取り消す

グループの入力とセクション、移動

左のボタンから順に以下のような機能が割り当てられています。

① 現在の実行グループの次にテキストを挿入する
② 現在の実行グループの次に新しい実行グループを挿入する
③ 選択した部分をひとつのセクション（章立て）にする
④ 章立て部分を解除する
⑤ ハイパーリンクの履歴を戻る
⑥ ハイパーリンクをひとつ進む

計算の実行と中断、デバッグ、カーネルの初期化、表示倍率

左のボタンから順に以下のような機能が割り当てられています。

① ワークシート中のコマンドを最初からすべて実行する
② 現在カーソルのある実行グループのコマンドを実行する
③ 実行中の計算を中断する
④ 現在の処理をデバッグする
⑤ 現在の計算カーネルを初期化する
⑥ 100%ズームで表示
⑦ 150%ズームで表示
⑧ 200%ズームで表示

タブ動作の切替、ヘルプ表示

左のボタンから順に以下のような機能が割り当てられています。

① テキストグループでのタブキーの挙動の変更
② ヘルプを起動する

1.7. 計算の実行方法

Maple で計算を実行するには、必要なコマンドや関数をタイプして、**[Enter]キーを押す**ことで計算が実行[2]されます。コマンドを複数行に渡って記述するには、適宜[Shift]キー+[Enter]キーで改行されます。

【コマンドを入力するときの注意点】
Maple では、コマンドの終端（行端ではないことに注意）に必ず「；」（セミコロン）または「：」（コロン）が必要です。

セミコロンは出力をそのまま表示し、コロンでは出力を抑制する（計算は実行するが結果を表示しない）

```
> 10!;
                    3628800

> 10!:
```

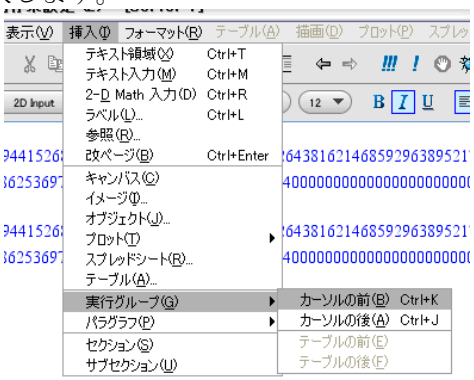
[2] Maple はインタプリタ型のソフトウェアですので、Java や C 言語などのように逐一コンパイルする必要はありません。

1.8. 実行グループについて

実行グループとは、プロンプトの左側に表示されている [記号を指します。実行グループは、入力と出力の１セットで構成されます。

1.9. 新しい実行グループを挿入する

Maple は一度記述した実行グループの前に新しい実行グループを挿入することが可能です。実行グループを挿入するには、現在の実行グループにカーソルを置き、[Ctrl+K]を押します。または[挿入]メニューの[実行グループ]をポイントし「カーソルの前」を選択します。
同様に、現在の実行グループの次に新しい実行グループを挿入するには、Ctrl+J または「カーソルの次」のメニューを選択します。



なお、Maple の実行グループはいつでも編集して再計算に用いることができます。一度計算を実行した後に変数の値を変えたり、追加することが可能です。

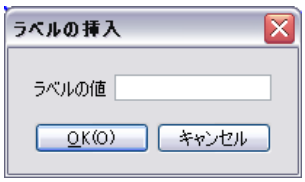
1.10. 実行グループ内の入力・出力単位で削除する

不要な実行グループ内の入力・出力単位で削除するには、実行グループにカーソルを持っていき、Ctrl キー+Del キーを押します。するとカーソルのあった入力単位または出力単位で削除することが可能です。

1.11. 数式ラベルを参照する

Maple 10 以降、出力には必ず数式番号（ラベル）が付加されます。前の計算結果に付いている数式ラベルを参照して計算させることも可能です。

数式番号を参照して入力するには、Ctrl+L キーを押して表示されるダイアログに数式番号を入力します。または、[挿入]メニューから[ラベル]を選択します。



```
> 100!;
933262154439441526816992388562667004907159682643\      (1)
    38162146859296389521759999322991560894146397\
    61565182862536979208272237582511852109168640\
    000000000000000000000000
> ifactor((1));
(2)^97 (3)^48 (5)^24 (7)^16 (11)^9 (13)^7 (17)^5 (19)^5 (23)^4      (2)
    (29)^3 (31)^3 (37)^2 (41)^2 (43)^2 (47)^2 (53) (59)
    (61) (67) (71) (73) (79) (83) (89) (97)
```

例えば、前に計算した 100!の結果を素因数分解するには、以下のような手順で入力します。
① 素因数分解のためのコマンド「**ifactor**」とタイプし、左丸括弧を入力します。
② Ctrl+L キーを押してラベル入力のダイアログを表示します。
③ 数式ラベル 1 番を参照するため「1」と指定します。
④ [OK]ボタンを押します。
⑤ 番号(1)が入力されたら、閉括弧（右丸括弧）を入力し、最後に[Enter]キーを押します。
⑥ 計算結果が表示されます。

なお、ここでは前の計算結果を参照するために数式ラベルを用いましたが、前の計算結果を一旦別の変数に割り当てて、その変数名を用いて計算することも可能です。変数への割り当ては10 ページを参照してください。

2. コマンドの基本的な使い方

Maple には 4000 種類以上のコマンドが提供されています。そのすべてのコマンドを覚えることは不可能ですので、ここではまず基本的なコマンド５種類を覚えます。

Maple のコマンドは以下のような方法で記述します。

◆ Maple Input（テキスト入力）の場合：文字は赤色で表示されます。
```
> コマンド名(引数);
```

ここで「>」記号はプロンプトと呼ばれる入力のための記号です。この記号は Maple 側が表示しているので、ユーザが入力する必要はありません。
コマンドの引数は必ず丸括弧で括ります。角括弧（[]）や波括弧（{}）はそれぞれ別の使い方で用いられます。

2.1. 方程式を解くための solve コマンド

solve コマンドは、引数で与えられた方程式を指定された変数について解きます。

```
solve(方程式, 変数);
```

最初に解きたい方程式を指定して、２番目の引数で解く変数名を指定します。以下の場合、与えられた方程式を変数 x について解いています。

solve コマンドは、１変数の方程式に限らず、多変数の方程式（連立方程式）についても解を求めることが可能です。連立方程式を解く場合は、複数の方程式と解くべき複数の変数を角括弧または波括弧でまとめます。
例えば、次のコマンドは、円 $x^2+y^2=1$ と直線 $x-y=1$ の交点を求めています。

```
> solve([x^2+y^2=1,x-y=1],[x,y]);
```
$$[[x=1, y=0], [x=0, y=-1]]$$

2.2. 関数やデータを描画するための plot コマンド

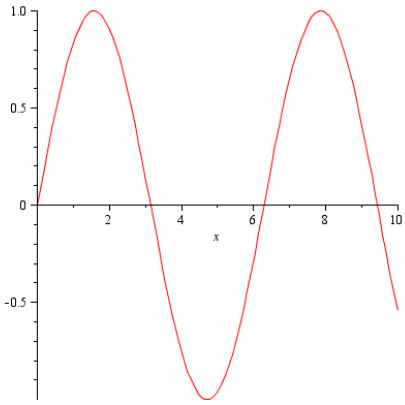
2 次関数や三角関数などを描画するには、**plot** コマンドを用います。**plot** コマンドは 1 変数の関数を描画するためのコマンドです。

```
plot(１変数関数, 変数=範囲..範囲);
```

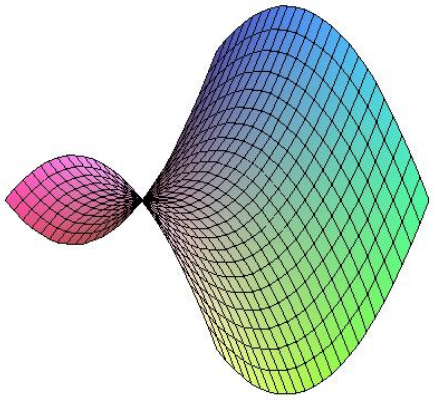
2 番目の引数には描画する関数の範囲を指定します。複数の 1 変数関数を描画する場合は関数を波括弧で指定します。
また、2 変数の関数を描画するときは、**plot3d** コマンドを用います。

```
plot3d(2 変数関数, 変数 1=範囲..範囲, 変数 2=範囲..範囲);
```

```
> plot(sin(x),x=0..10);
```




```
> plot3d(x^2-y^2,x=-2..2,y=-2..2);
```



3 次元のグラフ（曲面）は、マウスで自由に視点を変更することができます。一度グラフをマウスでクリックし選択してから、マウスで回転を行ってみてください。

なお、プロットコマンドには、軸の種類の指定や線・面の色の指定、タイトルやレジェンドの指定などを行うためのオプションを指定することができます。詳細は後述の各章を参照してください。

2.3. 変数への割り当てを行う「：＝」

計算した結果を別の変数に保存する（この処理を Maple では“変数に結果を割り当てる”と言います）を行うには、次の書式でタイプします。

```
変数:=値(式);
```

例えば、変数 a に 2 という値を割り当てるには次のように「変数：＝値（式)」として記述します。一度割り当てた変数は、変数名を使って計算させることが可能です。

```
> a := 2;
```

$$a := 2$$

```
> a^2+a;
```

$$6$$

ここで、単なる等号（＝記号）は、**solve** コマンドの説明の際に使ったように、方程式の等号として用いることに注意してください。

2.4. 変数に値を代入する subs コマンド

一旦、変数に値を割り当ててしまうと、その変数を持つすべての数式に影響があります。そこで、通常は式への変数の代入はその都度行うようにします。**subs** コマンドは、指定した式に対して形式的に変数への代入を行います。代入は値のみだけでなく、別の変数や数式を指定することも可能です。

```
> subs({a=2},a^2+a);
```

$$6$$

```
> subs({x=s,y=2*t},x^2+2*y+y^2);
```

$$s^2 + 4\,t + 4\,t^2$$

2.5. 式を評価する eval コマンド

一方で、数学的な意味での評価のためのコマンドとして **eval** コマンドが用意されています。このコマンドは $x = a$ の点における $f(x)$ の値を評価します。

```
> eval(sin(x),x=2*Pi);
```

$$0$$

eval コマンドも **subs** コマンドと同じように、式への変数の値の代入のために用いることが可能です。しかし、**eval** コマンドは与えられた式に対する数学的な意味での評価である一方、**subs** コマンドは形式的に代入を行います。その違いは以下の例で見ることができます。

```
> subs({x=0},sin(x)/cos(x));
```

$$\frac{\sin(0)}{\cos(0)}$$

```
> eval(sin(x)/cos(x),x=0);
```

$$0$$

subs コマンドの場合、形式的な代入を行った後の計算は適用されていませんが、**eval** コマンドでは sin(x)の値の評価を行った後で得られた値同士の計算まで行われています。

2.6. 計算エンジン（カーネル）を初期化する restart コマンド

Maple の計算エンジンを初期化するには **restart** コマンドを用います。**restart** コマンドを使うとこれまでの計算結果や変数への割り当て情報をすべて消去して最初に起動した状態に戻します。

以下では、変数 x に値を割り当てて計算を行った後に **restart** を行っています。**restart** コマンドの実行後は変数への割り当ては解除されます。

```
> x := 2;
```

$$x := 2$$

```
> y := x^2-x+2;
```

$$y := 4$$

```
> restart;
> x;
```

$$x$$

2.7. コマンドの使い方がわからないとき（ヘルプの参照方法）

先にも述べたように、Maple には 4000 種類以上の関数が用意されています。それぞれの関数には引数の指定の仕方やオプションが用意されていますが、詳しい使い方がわからないときは、Maple ヘルプを参照します。

特定のコマンドのヘルプを参照するには、以下の２通りの方法があります。

- **?**コマンドを使ってヘルプを参照する
  例えば `solve` コマンドの使い方を参照する場合、プロンプト上で **`?solve`** とタイプし実行します。すると、**`solve`** コマンドのヘルプページがヘルプブラウザで表示されます。

- コマンドの文字列中にカーソルを合わせて F2 キーを押す
  この場合も?コマンドを使った場合同様に、該当するコマンドのヘルプページがヘルプブラウザ上で表示されます。

ヘルプの詳しい使い方については第 6 章の「ヘルプブラウザの使い方」を参照してください。

2.8. 練習問題

ここまで学んだ solve コマンド、plot コマンド（または plot3d コマンド）、変数への割り当てを用いて次の練習問題を Maple で行ってみてください。

① 関数 $y=\cos(x)$ の曲線グラフを $x=0$ から $x=10$ の範囲で描画しなさい。
② 関数 $z=\exp(-x^2-y^2)$ の曲面グラフを x, y 共にそれぞれ-1.2 から 1.2 の範囲で描画しなさい。
③ $t=3$ のとき、式 t^3+t^2+t+1 の値を求めなさい。

3. プロットをカスタマイズする

ここでは、プロットのカスタマイズ方法について紹介しています。

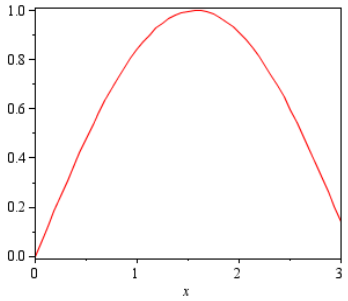
3.1. plot コマンドや plot3d コマンドにオプションを指定する

Maple の plot コマンドや plot3d コマンドでは、軸の指定や曲線の色、グリッド線などを指定するオプションが用意されています。オプションはそれぞれ組み合わせて用いることが可能です。

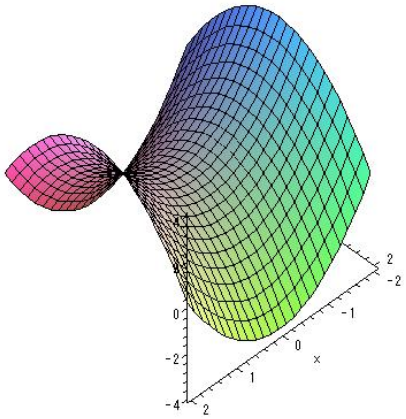
① 軸を描画する

axes オプションを指定します。指定できる値は、**boxed**, **frame**, **none**, **normal** の４つです。

```
> plot(sin(x),x=0..3,axes=boxed);
```



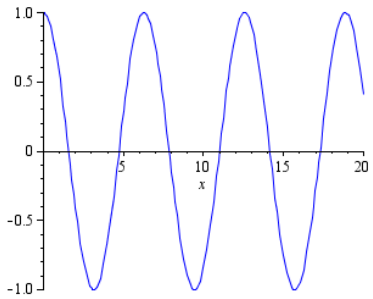

```
> plot3d(x^2-y^2,x=-2..2,y=-2..2,
    axes=frame);
```



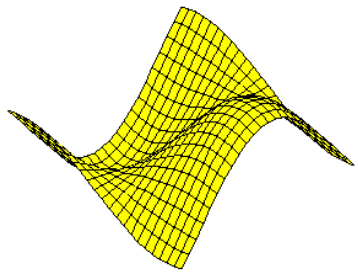
② 線の色を変える（面の色を変える）

color オプションで指定します。

```
> plot(cos(x),x=0..20,
    color=blue);
```




```
> plot3d(cos(x)*sin(y),
    x=-2..2,y=-2..2,
    color=yellow);
```



plot および **plot3d** コマンドで指定できる色の名前のリストは、**?plot,colornames** を実行して表示されるヘルプページを参照してください。

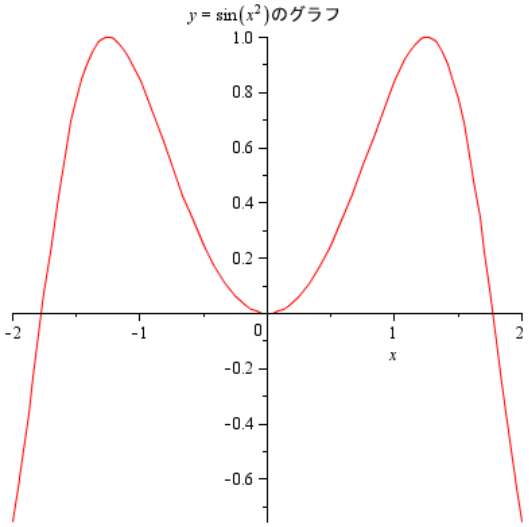
③ グラフにタイトルを付ける

グラフのタイトルは **title** オプションで指定します。この際、**typeset** コマンドで数式を指定することができます。

```
> f := sin(x^2);
```

$$f := \sin\left(x^2\right)$$

```
> plot(f,x=-2..2,title=typeset(y=f,"のグラフ"));
```



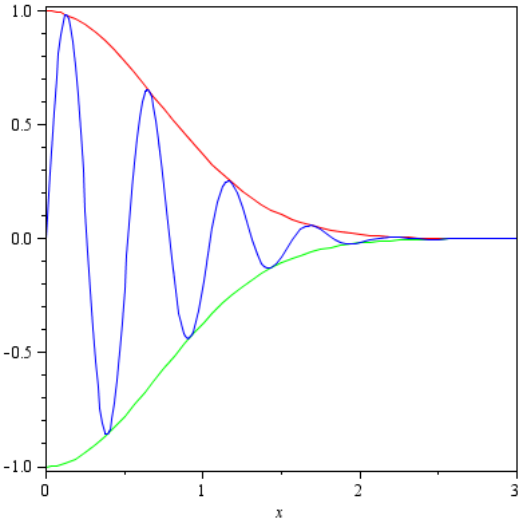
なお、数式が正しく表示されるのは 2 次元グラフに限ります。3 次元グラフでは 1 次元の数式（テキストの表記）になります。

ここで紹介したオプションを組み合わせて、次のようなグラフを描画することもできます。

```
> g := exp(-x^2);
```

$$g := e^{-x^2}$$

```
> plot([g,-g,sin(12*x)*g],x=0..3,
      axes=boxed,color=[red,green,blue]);
```

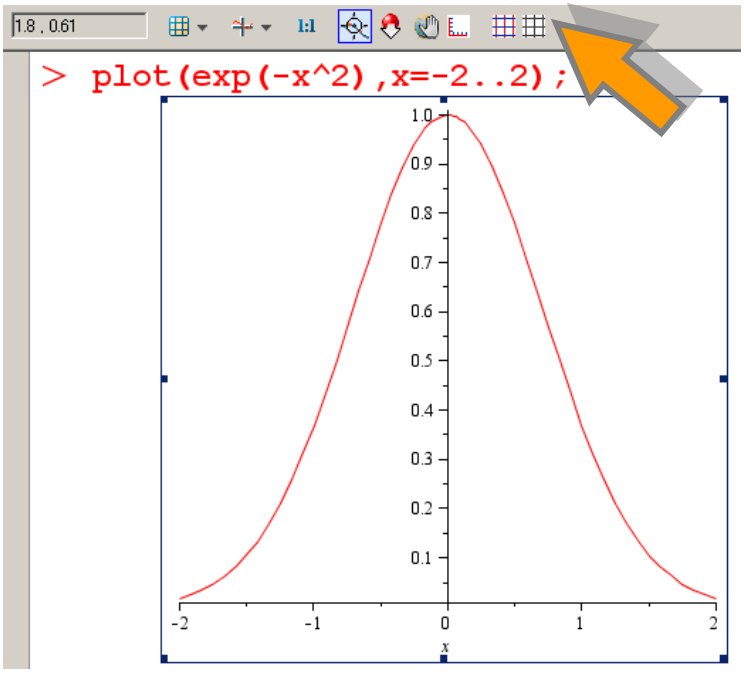


コマンドを複数行に渡って書く場合は、適当なところで改行（Shift+Enter キー）するようにしましょう。

3.2. コマンドを使わずにオプションを変更する

前章では**plot**コマンドや**plot3d**コマンドに**axes**や**title**などのオプションを指定してグラフをカスタマイズしましたが、Maple ではメニュー操作でもグラフのスタイルを変更することができます。

まず、**plot** コマンドまたは **plot3d** コマンドでグラフを描画します。次に描画されたグラフをマウスで選択します。すると、ワークシートウィンドウの上部にあるツールバーがプロット用のツールバーに変更されます。



プロット用ツールバーには以下のようなボタンが用意されています。

左から順に以下の機能が用意されています；

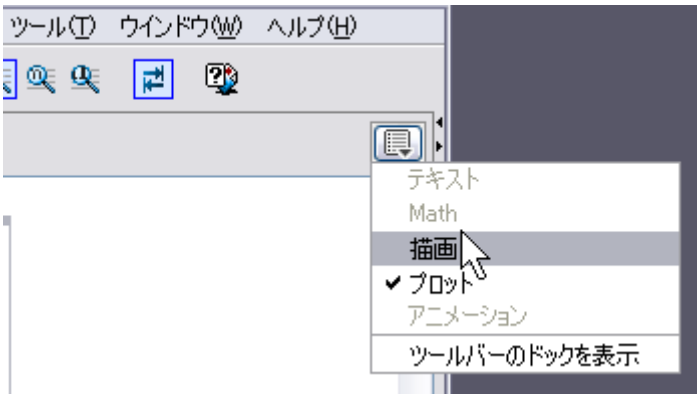
① グラフスタイルの変更：線種や面の描画方法を変更できます。
② 軸スタイルの変更：axes オプションで指定できる軸の設定変更が可能です。
③ 軸の縦横比の設定
④ 座標のプロービング（座標数値の追跡）
⑤ プロットのズーム（大きさ）の変更
⑥ グラフの移動
⑦ 軸のプロパティウィンドウの表示
⑧ グリッド線プロパティウィンドウの表示
⑨ グリッド線の表示切替

これらはすべてマウス操作で変更・設定することが可能です。また、ツールバーに用意されている設定変更機能は、グラフ上でマウス右ボタンをクリックして表示されるメニューでも同様に変更することが可能です。

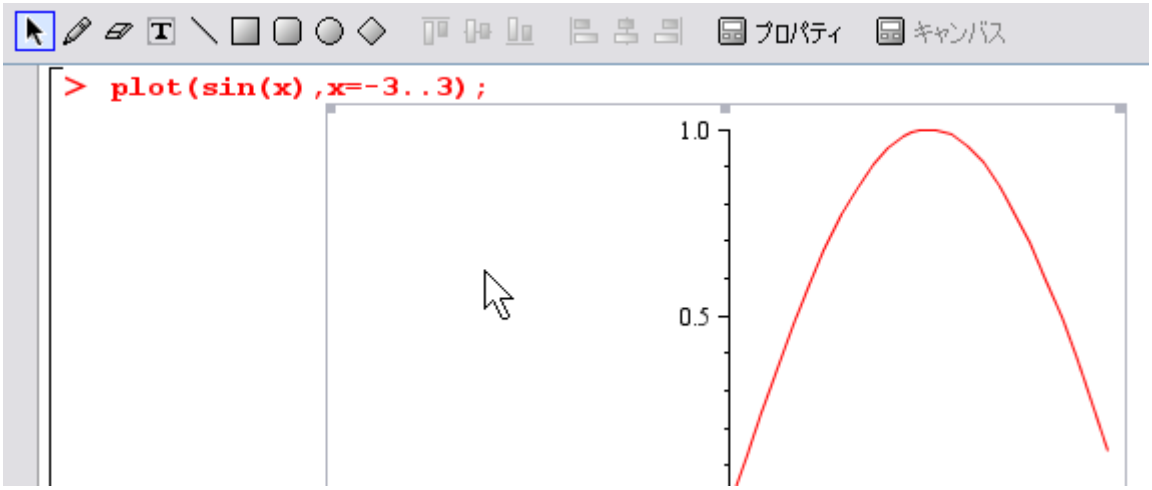
3.3. 描画ツールを使う

Maple 11 以降に用意されている描画ツールを使うことで、コマンドによって描画されたグラフを元にして 2 次元グラフをカスタマイズすることができます。

描画ツールを表示するには、まず適当なグラフを描画し、グラフをマウスで選択してから、ツールバー右横にあるツールバー切替メニューから[描画]を選択します。



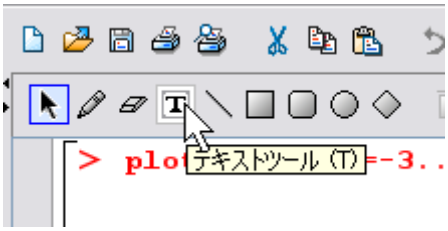
描画ツールを選択すると、以下のようにツールバーが変更されます。(グラフを必ず選択しておいてください)



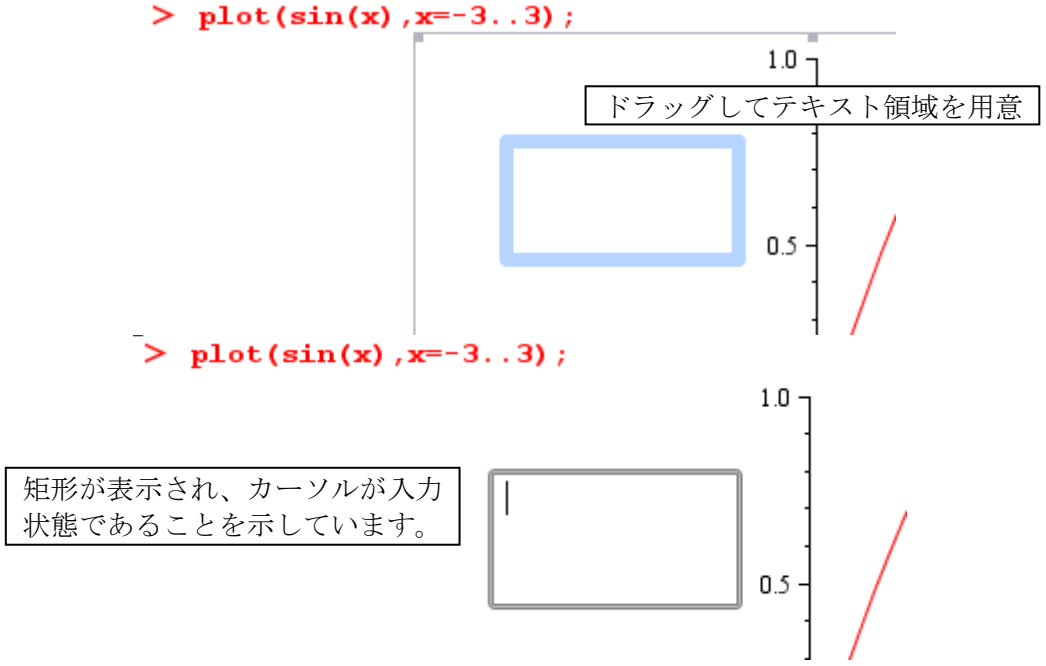
変更されたツールバーは以下のように表示されます。

描画ツールを使って、グラフ上の任意の場所にテキストを挿入し、また矢印や丸を描画してみます。

まず、グラフを選択状態にした後で、[テキストツール]のボタンをクリックします。



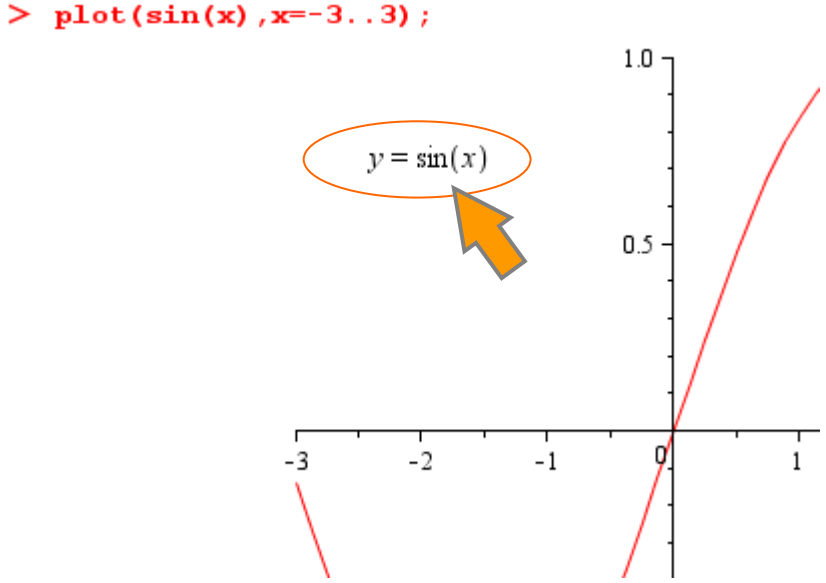
グラフ上の適当な位置で、ドラッグしてテキストを描画する範囲を用意します。



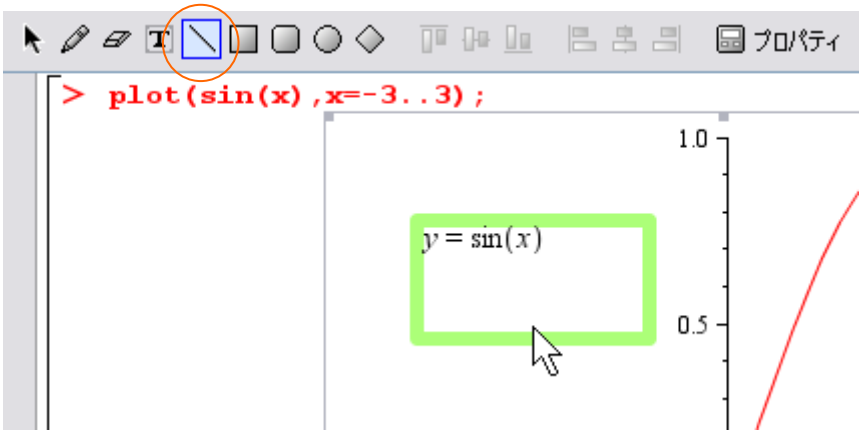
ここでテキストを入力します。テキストには 2D Math モードを利用して任意の数式を入力することもできます。入力状態の切替（テキスト→2D Math）は F5 キーを押します。

ここでは、2D Math モードでの入力形態に変更し、*y=sin(x)* とタイプしています。

テキストや数式を入力して、一旦グラフを選択（クリック）すると、入力された文字列が確定します。

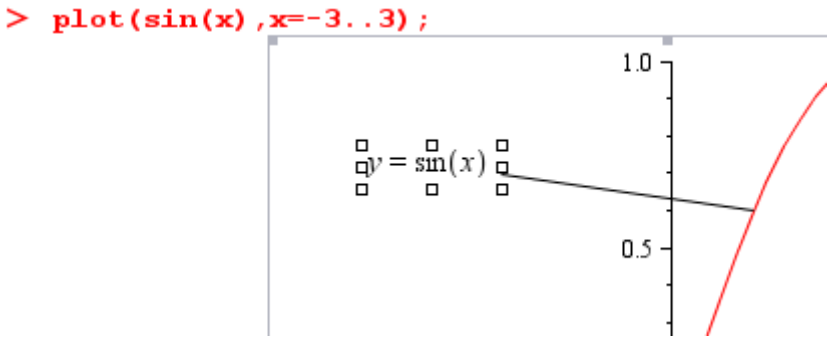


次に、グラフ上に矢印線を加えます。線ツールを選択して、さきほど入力した文字列の近辺にマウスカーソルを合わせます。すると、テキスト領域が緑色の枠で表示されます。

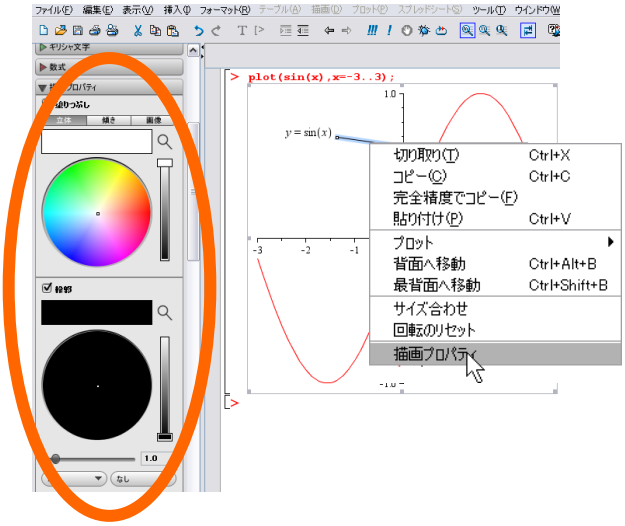


線ツールは直線を描画するためのツールとなります。文字列領域から関数の曲線の適当な場所へマウスを移動（線を描画）します。曲線上の適当な位置でマウスをダブルクリックすると線は閉じます。（ダブルクリックした箇所が終端となります）

現在の文字列領域は大きいので、マウスでテキスト領域をクリックしてアクティブにし、領域サイズを変更します。



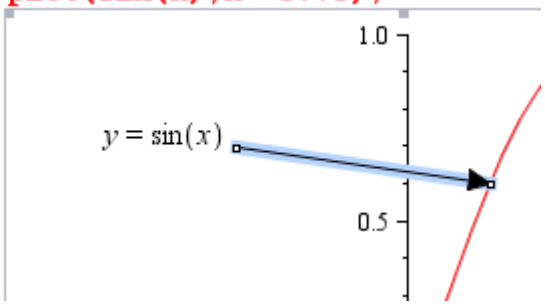
線が描けたら、線の端を矢尻にします。線をクリックして選択し、マウス右ボタンから描画プロパティのメニューを選びます。



すると、画面左側のパレット領域に「描画プロパティ」パレットが表示されます。

左図を参考にして、矢尻を設定します。
すると、下図のように、線の端が矢尻となります。

```
> plot(sin(x),x=-3..3);
```



ここでは、テキストと矢印線を描きましたが、これ以外にも丸や四角などの任意の図形を描画することができます。

3.4. 練習問題

① 三角関数 $\sin(x),\sin(2x),\sin(3x)$ のグラフを同時に描画しなさい。ここでそれぞれの曲線の色を赤、青、緑としなさい。
② ①の問題で作成した３つの曲線がそれぞれどの三角関数を意味するかを表示するため、描画ツールを用いて注釈を付け加え、注釈と曲線を矢印線で結びなさい。
③ コマンド
```
plot3d(sin(sqrt(x^2+y^2))/sqrt(x^2+y^2),x=-6..6,y=-6..6);
```
を実行しなさい。また、オプション numpoints=2000 を与えて再実行し、オプションを指定せずに実行した結果と何が違うかを観察しなさい。

4. Maple を使って問題を解く

この章では、Maple の正しい操作方法を学ぶために、Maple を使って具体的な問題を解く作業を行っていきます。問題を解く作業を行うことで、Maple に用意されているコマンドの使い方やその特徴を学びます。

4.1. 問題１：接線を求める

放物線 $y = x^2 - x + 1$ について、 $x = 2$ の点における接線を求めよ。

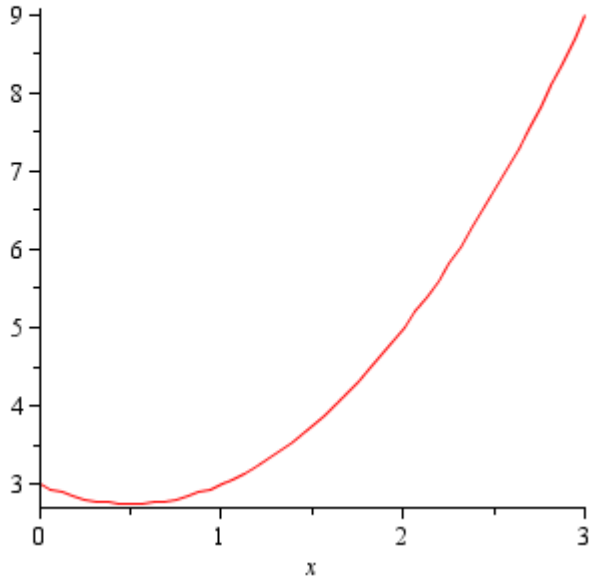
【解答】

まずは問題を図示してみます。新しいワークシートを開いてください。描画ツールを使って点と接線も描いてみます。

```
> restart;
> F := x^2-x+3;
```

$$F := x^2 - x + 3$$

```
> plot(F,x=0..3);
```



接線の定義からその傾きは曲線の微分を計算することで求まります。Maple で微分を計算するには **diff** コマンドを用います。

```
> dF := diff(F,x);
```

$$dF := 2x - 1$$

```
> eval(dF, x=2);
```

$$3$$

これが *a* の値です。

次に *y* 切片である *b* の値を求めます。*x*=2 のときの *y* の値を求めれば *b* の値を計算することができます。

```
> yval := eval(F,x=2);
```

$$yval := 5$$

```
> eq := eval(y=3*x+b, [x=2,y=yval]);
```

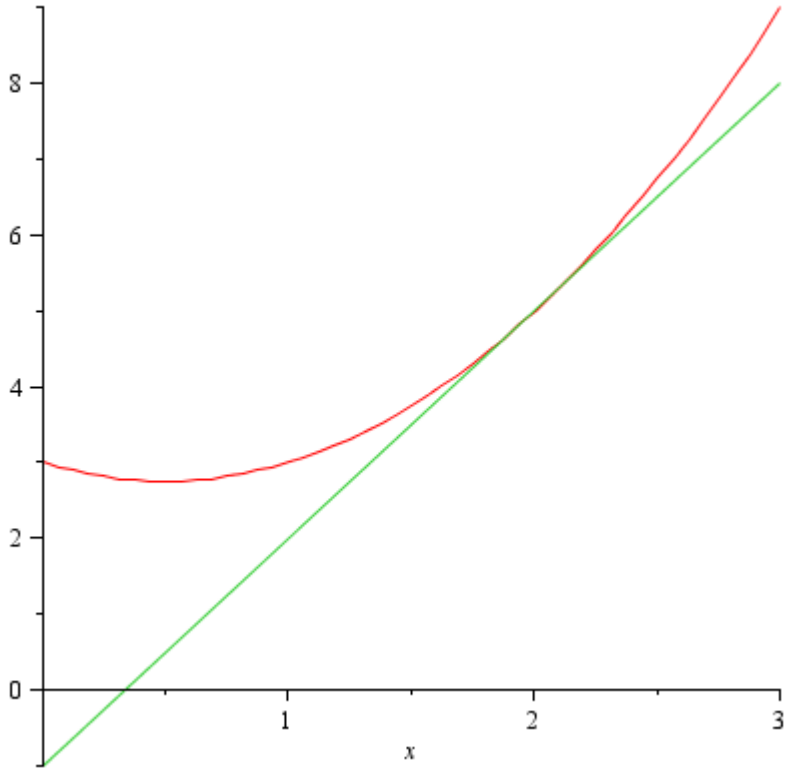
$$eq := 5 = 6 + b$$

```
> solve(eq, b);
```

$$-1$$

従って、求めるべき接線の式は、$y = 3x - 1$となることがわかります。
求めた接線の式と元の式を同時に描画してみます。

```
> plot([F,3*x-1],x=0..3);
```



ここまでは Maple の微分コマンドを使ってすべて手計算同様にひとつひとつ計算してきました。一方で、Maple には接線を計算するといった目的毎のコマンドが別のパッケージとして用意されています。
この問題の接線に関しては、Student パッケージの Calculus1 サブパッケージに用意されている **Tangent** コマンドで求めることが可能です。

```
> with(Student[Calculus1]):
> Tangent(F, x=2);
```

$$3x - 1$$

パッケージは Maple が起動した段階では利用できません。利用する場合に適宜パッケージを読み込ませる必要があります。パッケージを読み込むには **with** コマンドでパッケージ名を指定します。(なお、この例では行末にコロン（：）を付加しています。これは **with** コマンドの出力を表示しないようにするためです)

```
with(パッケージ名);
```

Maple に標準で用意されているパッケージには、線形代数のための LinearAlgebra パッケージや、プロットコマンド類を拡張する plots パッケージなど、分野毎の計算や機能拡張のためのパッケージが用意されています。パッケージの詳細については、35 ページの説明を参照してください。

4.2. 問題２：交点の計算と面積を求める

$a>0$ とする。$y=x^2$ と $y=2\cdot(x-a)^2+1$ が、ただ一点のみ共有するような a の値と、その共有点の座標を求めよ。また、a がその値のときにこの２つの曲線と y 軸とで囲まれる図形の面積を求めよ。（学習院大学入試問題：2003 年経済学部）

【解答】

まず、題意から二つの放物線の式を等号で結び、変数 x と a の方程式として解いてみます。新しいワークシートを開いておいてからコマンドを記述して計算を実行していきます。

方程式を解くには **solve** コマンドを用います。**solve** コマンドの結果は、変数 **sol** に割り当てておきましょう。

```
> restart;
> sol := solve(x^2=2*(x-a)^2+1,x);
```

$$sol := 2a+\sqrt{2a^2-1},\ 2a-\sqrt{2a^2-1} \qquad (1)$$

これが交点の x 座標となります。題意から交点は共有しているので、変数 **sol** に割り当てられた２つの結果を等号で結び、この方程式を変数 a に関して解きます。

```
> asol := solve(sol[1]=sol[2],a);
```

$$asol := \frac{\sqrt{2}}{2},\ -\frac{\sqrt{2}}{2} \qquad (2)$$

solve コマンドによって、a の値が２つ求められました。問題設定から $a>0$ なので、求めるべき a の値は $\frac{\sqrt{2}}{2}$ であることがわかります。

確認をするために、元の方程式系を変数 **eq** として割り当てて、得られた a の値のときのグラフを描画してみます。なお、**asol[1]** としている部分は、**asol** で得られた結果のうち、正である値の番号を指定して取り出していることに注意してください。

```
> eq := [x^2,2*(x-a)^2+1];
```

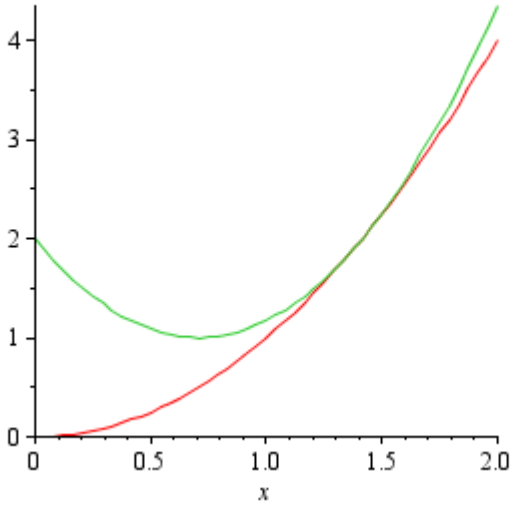
$$eq := \left[x^2,\ 2(x-a)^2+1\right] \qquad (3)$$

```
> funcs := eval(eq, a=asol[1]);
```

$$funcs := \left[x^2,\ 2\left(x-\frac{\sqrt{2}}{2}\right)^2+1\right] \qquad (4)$$

得られた２つの関数について、グラフを描画してみます。

```
> plot(funcs,x=0..2);
```



確かに交点は１点だけとなっていることがわかります。実際に交点の x 座標を求めるには、**funcs** の２つの多項式を等号で結んで方程式を解くか、または変数 **sol** で得られた結果のうち、正であるものに a の結果を代入することで得られます。下記にあるように、結果をいずれも **sqrt(2)** となります。

```
> solve(funcs[1]=funcs[2],x);
```

$$\sqrt{2}, \sqrt{2} \qquad (5)$$

```
> eval(sol[1], a=asol[1]);
```

$$\sqrt{2} \qquad (6)$$

次に、面積を求めます。被積分関数 **g** は、**funcs[2]** から **funcs[1]** を引いたものです。

```
> g := funcs[2]-funcs[1];
```

$$g := 2\left(x - \frac{\sqrt{2}}{2}\right)^2 + 1 - x^2 \qquad (7)$$

この被積分関数を x=0 から先程求めた共有点の x 座標の値まで積分します。Maple では、積分を計算するために **int** コマンドを用います。

```
> int(g, x=0..sqrt(2));
```

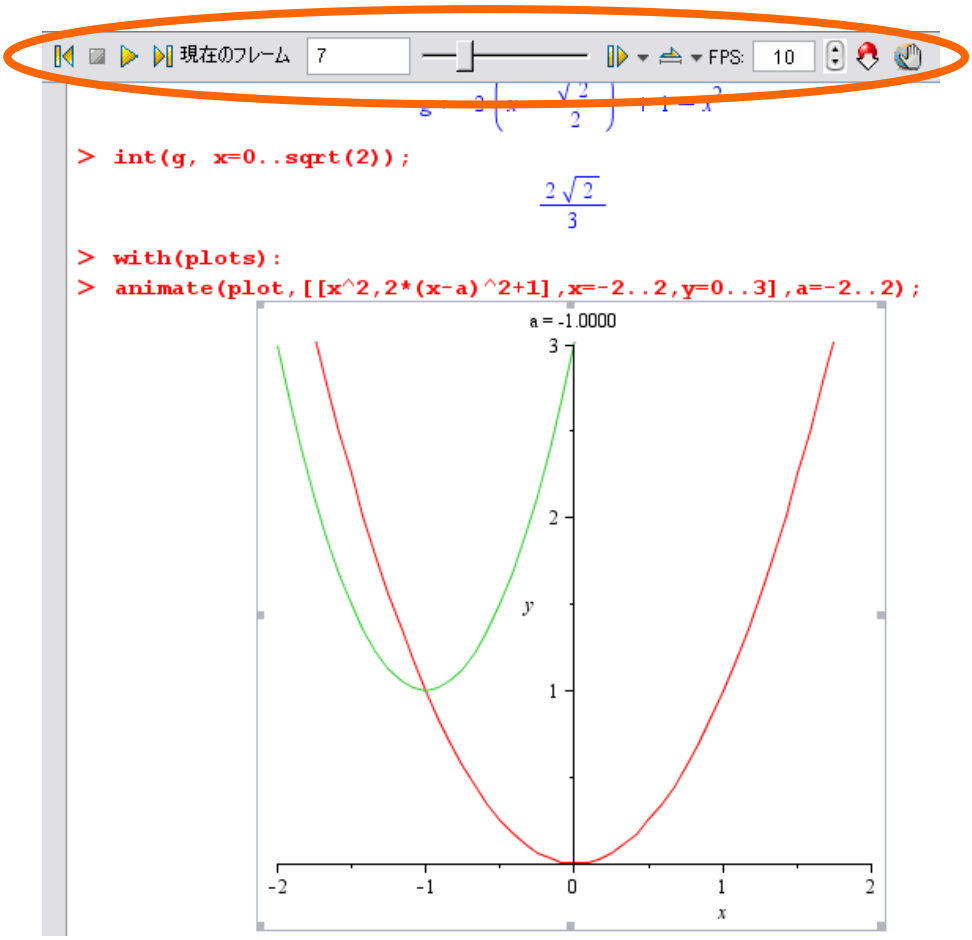
$$\frac{2\sqrt{2}}{3} \qquad (8)$$

なお、a の値が変化したときに、２つのグラフの交点がどのように変化するかを見ることも Maple では簡単に実現できます。plots パッケージに用意されている animate コマンドを用いて、以下のようにコマンドを入力・実行してみてください。これは、2 つのグラフの式で、a の値が-2 から 2 まで変化したときの様子をアニメーションで表示します。

```
> with(plots):
> animate(plot,[[x^2,2*(x-a)^2+1],x=-2..2,y=0..3],a=-2..2);
```

グラフが表示されたら、グラフ部分を１回マウスで選択します。すると、コンテキストバーに

アニメーション用ツールが表示されます。



再生ボタンやスライダーを動かすと、緑色のグラフが変化して交点がどのように動くかを確認できます。

4.3. 問題３：方程式の求解と計算結果の簡単化

$x^2+3\cdot x+8=0$ の２つの解をそれぞれ $a,\ b$ とするとき、$a^2+a\cdot b+b^2, a^4+21\cdot b^3$ の値を求めよ。（学習院大学入試問題：2005 年経済学部）

【解答】

新しいワークシートを開いてください。

与式の解と係数の関係を変数 **eq** に割り当てます。

```
> restart;
> eq := [a+b=-3,a*b=8];
```

$$eq := [a + b = -3, a\,b = 8] \tag{1}$$

eq の条件の下でそれぞれの式の値を求めるには **simplify** コマンドを用います。**simplify** コマンドは式を簡単にするためのコマンドで、2 番目の引数には簡単にするためのルールを指定することができます。

```
> simplify(a^2+a*b+b^2,eq);
```

$$1 \tag{2}$$

```
> simplify(a^4+21*b^3,eq);
```

$$433 \tag{3}$$

simplify コマンドはMapleが内部で保有している公式を参照して数式を簡単にすることもできます。例えば、以下のような三角関数式や根号式を簡単にすることが可能です。

```
> simplify(sin(x)^2+cos(x)^2);
```

$$1 \tag{4}$$

```
> simplify(sqrt(x^2),symbolic);
```

$$x \tag{5}$$

5. Maple でプログラミングする

Maple は、実はそれ自体がひとつのプログラミング言語（Maple 言語）で開発されています。この章では Maple のプログラミングの基本について学習します。

5.1. 関数とプロシージャ

引数を受け取って結果を返すものを関数と呼びますが、Maple ではこのような数学的な関数を次のような記述方法で定義することができます。

```
関数名 := 変数 -> 処理;
```

実際に func という名前の関数を作って試してみます。新しいワークシートを開いてください。

```
> restart;
> func := x -> x^2+x;
```

$$func := x \to x^2 + x \qquad (1)$$

```
> func(2);
```

$$6 \qquad (2)$$

```
> func(-1/2);
```

$$-\frac{1}{4} \qquad (3)$$

ここで、関数定義の際の矢印は **->** をタイプすると自動的に矢印に置き換わることに注意してください。（2D Math Input モードの場合は自動的に矢印の表記になります）

関数の引数を複数指定するときは丸括弧で複数の変数を括って定義します。例えば、x, y を 2 辺とする直角三角形の斜辺の長さを求める関数を定義するには、次のように記述します。

```
> shahen := (x,y) -> sqrt(x^2+y^2);
```

$$shahen := (x, y) \mapsto \sqrt{x^2 + y^2} \qquad (4)$$

```
> shahen(3,4);
```

$$5 \qquad (5)$$

```
> shahen(a,b);
```

$$\sqrt{a^2 + b^2} \qquad (6)$$

関数 **shahen** は二つの引数であれば問題はないので、任意の変数を引数に指定することも可能です。しかし、複数の引数で定義した関数に一つの引数しか渡していない場合はエラーが返されます。

```
> shahen(3);
Error, invalid input: shahen uses a 2nd argument, y, which
is missing
```

関数はこのように単純な処理の計算に適した定義方法です。

一方、複雑な計算や処理を必要とするような場合は、プロシージャ（**proc**）を用います。

```
関数名 := proc(引数)
  local 局所変数;

  ...処理...

  return(戻り値);
end proc;
```

例えば、指定された式 **expr** を指定された引数 **var** で微分・積分し、その結果を返すような処理を行うプロシージャを定義してみます。

```
> restart;
> biseki := proc(expr, var)
    local d,s;

    d := diff(expr,var);
    s := int(expr,var);

    return([d,s]);
  end proc;
```

$$biseki := \mathbf{proc}(expr, var) \qquad (1)$$

```
  local d, s;
  d := diff(expr, var); s := int(expr, var); return [d, s]
end proc
```

```
> biseki(x^3,x);
```

$$\left[3\,x^2,\ \frac{x^4}{4}\right] \qquad (2)$$

5.2. 反復

反復計算が必要なときは、**for-loop** 文を用います。
例えば、1 から数 n までの二乗和を計算する関数 **sum2** を作る場合は次のように記述します。

```
> sum2 := proc(n)
    local t,i;
    t := 0;
    for i from 1 to n do
      t := t+i^2;
    od;
  end proc:
> sum2(10);
```

$$385 \qquad (1)$$

```
> add(k^2,k=1..10);
```

$$385 \qquad (2)$$

組込み関数である add を用いた結果と同じであることが確認できます。

一方、反復と同様に、同じ計算を複数回実行し、そのすべての計算結果が必要なときは **seq** コマンドを用います。**seq** とは sequence の最初の 3 文字を取った名前です。
例えば、以下の例では、引数の関数を 1 階から n 階までの微分を求めるための記述です。

```
> nbibun := proc(expr, var, n)
    local i;

    seq(diff(expr,var$i), i=1..n);
  end proc:
> nbibun(x^6,x,3);
```

$$6x^5, 30x^4, 120x^3 \qquad (3)$$

5.3. 条件分岐

関数やプロシージャでは条件に応じて処理内容を変更することができます。条件分岐は **if-then-else** 文を用います。

例えば、引数 n を偶数か奇数か判定し、メッセージを出力するプロシージャを作ってみます。偶数か奇数かの判定は、2 で割った余りから判断します。余りを求めるには mod 演算子を用います。

```
> restart;
> hantei := proc(n)
    local k;

    k := n mod 2;
    if k=0 then
      printf("%d は偶数です！", n);
    else
      printf("%d は奇数です！", n);
    end if;
  end proc:
> hantei(309);
309 は奇数です！
> hantei(1024);
1024 は偶数です！
```

上記で用いている **printf** 関数は、C 言語の **printf** 関数と同じ動作をします。

5.4. 数に関連するデータ型

関数やプロシージャを定義する際、特定の引数にのみ有効な処理を定義したい場合があります。Maple には、**posint**（正整数）、**complex**（複素数）などの数に関する型や、**algebraic**（代数式）、**list**（リスト）、**listlist**（入れ子リスト）など配列や構造に関するデータ型が用意されています。(データ型の詳細は**?type** とタイプして表示されるヘルプに記述されています)

例えば、先程定義した **shahen** 関数の引数 *x*, *y* を、数値のみに限定した **shahen2** 関数を定義してみます。
型を定義するときは、「**変数名::型名**」と記述します。

```
> restart;
> shahen2 := (x::numeric,y::numeric)->sqrt(x^2+y^2);
```

$$shahen2 := (x::numeric, y::numeric) \mapsto \sqrt{x^2+y^2} \qquad (1)$$

実際に計算に用いてみます。

```
> shahen2(3,4);
```

$$5 \qquad (2)$$

```
> shahen2(1/2,1/3);
```

$$\frac{\sqrt{13}}{6} \qquad (3)$$

数に関しては問題なく機能しています。

一方、先程同様に未知の変数や根号（無理数）を与えて計算してみます。

```
> shahen2(a,b);
Error, invalid input: shahen2 expects its 1st argument, x,
to be of type numeric, but received a
> shahen2(2,3^(1/2));
Error, invalid input: shahen2 expects its 2nd argument, y,
to be of type numeric, but received 3^(1/2)
```

どちらの場合もエラーが返されます。これは、numeric 型が整数（integer 型）、分数（fraction 型）、浮動小数（float 型）のいずれかのときに成り立つからです。（つまり、無理数は numeric 型のデータではないということになります）

5.5. 配列に関連するデータ型

複数のデータを集めた型を一般に「配列」や「構造体」などと呼びますが、Maple にも色々な配列・構造体に関連したデータ型が用意されています。Maple で主に用いる配列・構造に関連するデータ型は以下の通りです。

① 式列

Maple 独自のデータ型で複数の要素を「,（コンマ）」で連結しているデータです。例えば

```
> mydata := 1,3,5,7;
```

$$mydata := 1, 3, 5, 7 \qquad (1)$$

のような形で定義します。なお、式列の任意の番号の要素は[]記号で取り出せます。以下では、**mydata** に割り当てられている 2 番目の値を取り出しています。

```
> mydata[2];
```

$$3 \qquad (2)$$

また、式列は **seq** コマンドでも生成されます。（**seq** コマンドは sequence の略です）
以下の例は、*n* 番目の素数を求める **ithprime** コマンドを使って 1 番目から 10 番目までの素数を生成し、結果を変数 **p** に割り当てています。

```
> p := seq(ithprime(n), n=1..10);
```

$$p := 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 \qquad (3)$$

② list（角括弧[]で表されるデータ）

式列を角括弧で括ったものが **list** となります。

```
> mydata := [1,3,5,7];
```

$$mydata := [1, 3, 5, 7] \quad (4)$$

次は入れ子となった **list** です。

```
> mydata3 := [[1,2,3],[2,3,4],[3,4,5]];
```

$$mydata3 := [[1, 2, 3], [2, 3, 4], [3, 4, 5]] \quad (5)$$

```
> mydata3[2,3];
```

$$4 \quad (6)$$

上記のデータを取り出すときは、**[]**で要素の番号を指定します。例えば上記では 2 番目のリスト**[2,3,4]**の 3 番目の要素を取り出しています。このように、list 型では指定した順番の要素を取り出すことが可能です。

③ 集合（波括弧 {} で表されるデータ）

式列を波括弧**{}**で括ったものが集合です。Maple の集合は数学的な意味での集合と等価な配列となります。

下記の例では、変数**myset1**に{3,3,2,5,1,2}として与えています。すると結果は{1,2,3,5}となります。

```
> myset1 := {3,3,2,5,1,2};
```

$$myset1 := \{1, 2, 3, 5\} \quad (7)$$

（Maple の）集合は以下の特徴を持ちます。

- 重複する要素は省かれます
  上記の集合のように数 3 は複数含まれていますが、戻り値の集合では 1 個です。

- 順序は保存されません
  集合内での要素の並び順序は、入力した順番通りに保存されるとは限りません。

※なお、上で述べた list 型では、順序の保存が保証されていることに注意してください。

④ Matrix 型、Vector 型、Array 型

Matrix 型および Vector 型は、線形代数のための LinearAlgebra パッケージやベクトル解析のための VectorCalculus パッケージで用いるためのデータ型です。Array 型は Matrix や Vector の基本となるデータ型で、任意の次元・サイズの配列となります。

例えば、行列 [[1,2],[3,4]] は以下のように定義します。

```
> m := Matrix([[1,2],[3,4]]);
```

$$m := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad (8)$$

行列と **list** 型は一見すると同じように見えます。しかし、**list** 型は特にメモリに関して何も考慮せずデータを冗長に保持します。一方、Matrix 型や Vector 型は、**datatype** オプションを指定することで、最適なメモリサイズや読み書き制限などの属性を保持しており、配列データへのアクセスも式列、**list** に比べて非常に高速です。従って、大きなデータを扱う処理を行うときには Matrix, Vector, Array などの型でデータを用意することが良いでしょう。

5.6. プログラミングを使った例題

Maple のプログラミング機能を使って簡単な例題を解いてみます。

【問題】ある自然数 m について、m の約数の和が $3m$ となるとき、m を 3 倍型であると呼ぶものとします。例えば自然数 6 の場合だと、その約数は 1,2,3,6 で、1+2+3+6=12 で 3×6＝18 なので、6 は 3 倍型ではありません。（学習院大学入試問題：2005 年理学部）

① 自然数 672 が 3 倍型であることを確かめなさい。

② 与えられた自然数 N について、N 以下の自然数で 3 倍型となる数を計算するプログラムを作成し、実行してみなさい。

【解答】

与えられた数の約数を求めるには、**numtheory** パッケージ（整数論のためのパッケージ）に用意されている **divisors** コマンドを使います。**divisors** コマンドに 672 という整数を与えて計算し、その結果を **yakusu** という名前の変数に割り当てます。

```
> restart;
> with(numtheory):
> yakusu := divisors(672);
```

$$yakusu := \{1, 2, 3, 4, 6, 7, 8, 12, 14, 16, 21, 24, 28, 32, 42, 48, 56, 84, 96, 112, 168, 224, 336, 672\} \quad (1)$$

求まった約数の和を計算します。データの和を計算するときは、**add** コマンドを使います。

```
> add(k,k=yakusu);
```

$$2016 \quad (2)$$

```
> is((2) = 3*672);
```

$$true \quad (3)$$

add コマンドで得られた結果は 2016 でした。その結果が、3 倍型であるかどうかを確認します。672 を 3 倍した数と 2016 が等しいか否かを判定するために、上記では **is** コマンド（英語で使う be の 3 人称単数現在形である is（イズ）と同じ）を用いています。

ここまでの手順をまとめて、プログラムを作成していきます。まず、与えられた数が 3 倍型であるかを判定するための関数を定義しましょう。

ここでは、関数名を **check3** とし、引数 n を受け取ってその約数の和を計算し、和が $3n$ と等しいかどうかを **is** コマンドで判定しています。

```
> check3 := n -> is(add(k,k=divisors(n))=3*n);
```

$$check3 := n \rightarrow is(add(k, k = numtheory:-divisors(n)) = 3\,n) \quad (4)$$

```
> check3(672);
```

$$true \quad (5)$$

```
> check3(11);
```

$$false \quad (6)$$

check3 関数を定義したら、実際に試してみます。確かに 672 のときは **true** を返しているこ

とが確認できます。

さて、**check3** 関数を使って、1 から *N* までの数の中で 3 倍型となる数を探していきます。以下のプロシージャ **find3bai** を記述します。（プロシージャを記述する際は Maple Input モードへ切り替えてください。2D Math Input から Maple Input への切替は F5 キーを押します）

```
> find3bai := proc(N)
    local k, result;

    result := NULL;
    for k from 2 to N do
      if check3(k)=true then
        result := result, k;
      end if;
    end do;
    return(result);
  end proc:
> find3bai(1000);
```

$$120, 672 \tag{7}$$

```
> find3bai(10^4);
```

$$120, 672 \tag{8}$$

プロシージャを定義したら、さっそく計算してみます。上記では 1000 以下の 3 倍型、10,000 以下の 3 倍型を探していますが、結果は 120 と 672 でした。

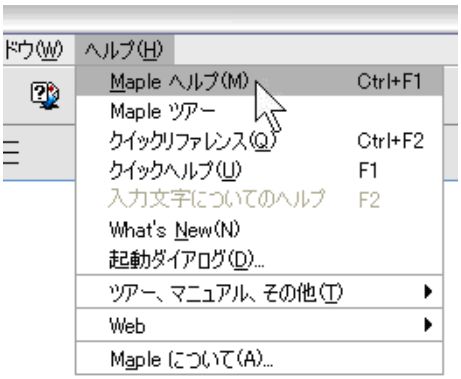
上記のプロシージャでは、局所変数として `result` という名前の変数を定義しています。これは反復計算で見つかった 3 倍型の数を保存しておくための変数です。`result` には見つかった 3 倍型の数が式列として格納されていきます。なお、コード中では、for 文による反復計算を始める前に result 変数に NULL 値を割り当てています。NULL 値は空の値を意味しています。

6. ヘルプブラウザの使い方

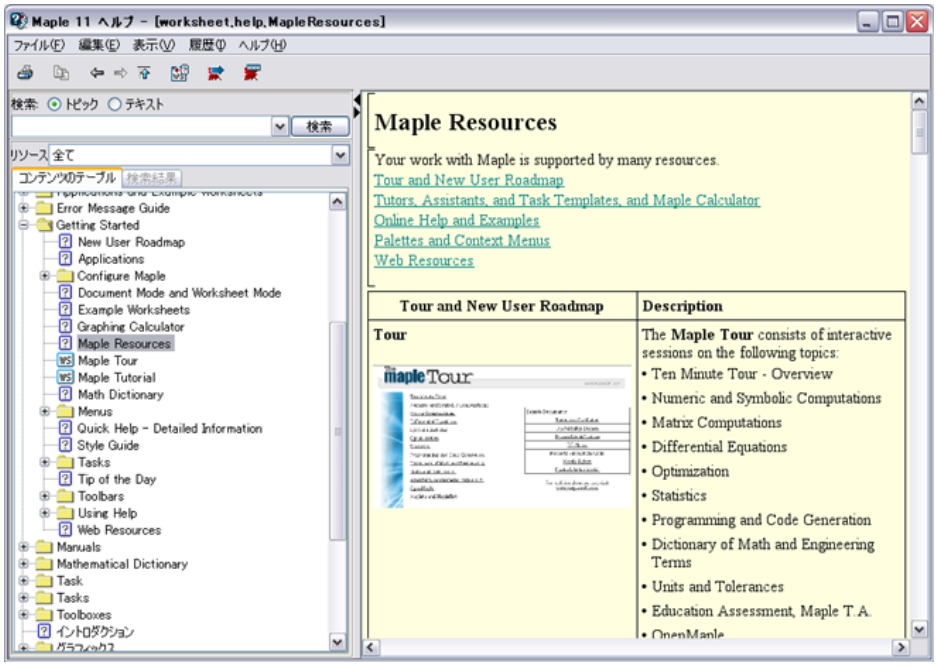
Maple には 4000 種類以上の関数やコマンドが用意されていますが、そのひとつひとつの細かい使い方（コマンドの名前、機能、引数の指定方法・型、オプション）を学ぶには、ヘルプブラウザで該当するページを参照します。

6.1. ヘルプブラウザを表示する

ヘルプブラウザを表示するには、[ヘルプ]メニューから[Maple ヘルプ]を選択するか、またはワークシート上で[Ctrl]キー＋[F1]キーを押します。

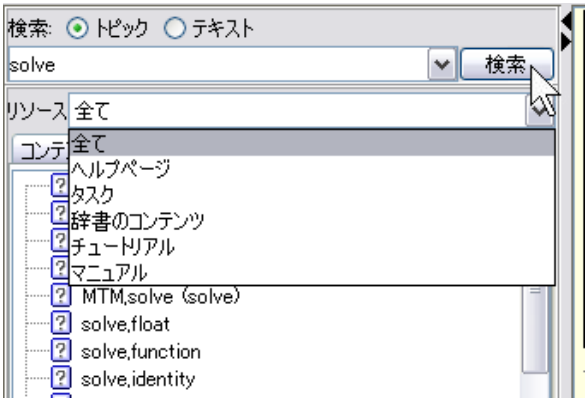


ヘルプブラウザは左側にコンテンツ・ペイン（または検索結果ペイン）が表示され、右側にヘルプコンテンツが表示されます。



6.2. 特定コマンドのヘルプページを参照する

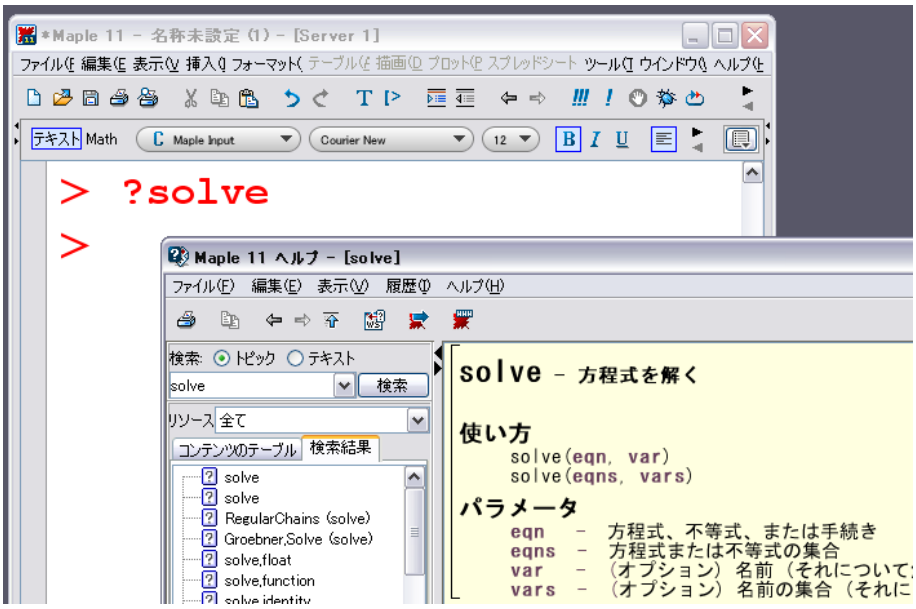
調べたいコマンドのヘルプを参照するには、検索フィールドにコマンド名を入力して[検索]ボタンを押します。例えば **solve** コマンドについて調べたいときは、検索フィールドに「**solve**」とタイプして[検索]ボタンを押します。すると **solve** コマンドに関連したコンテンツ結果リストが検索結果ツリー内に表示されます。



なお、検索の対象はデフォルトでトピック名（コマンドやパッケージ名に指定された文字列が含まれるか否か）で検索されます。ヘルプコンテンツ内のすべての記載内容を対象にして検索を行うには、検索対象を[テキスト]に変更します。
また、検索対象は、ヘルプページ、タスク、辞書、マニュアルなどに絞ることができます。検索対象は[リソース]で選択できます。

【注意】Maple ヘルプブラウザは日本語の検索は受け付けません。コマンド名または検索する文字列は必ず英語で指定してください。

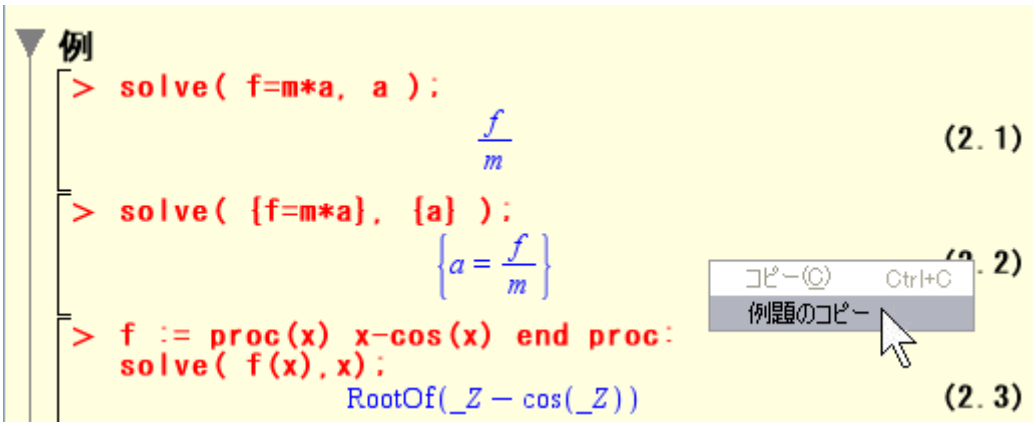
また、特定コマンドのヘルプページを参照する場合の別の方法として、ワークシート上で以下のように **?コマンド名** とタイプしても、同様に対象となるコマンドのヘルプページが参照されます。



6.3. ヘルプコンテンツのコマンド例をコピーして使う

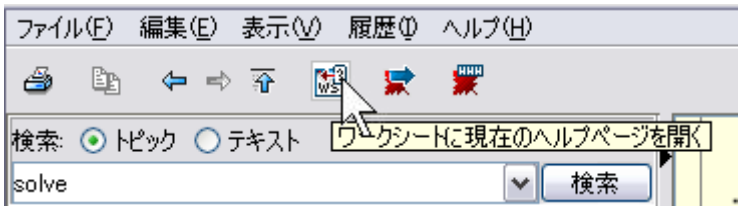
各コマンドやパッケージのヘルプページには、例題が記載されています。例題を自分が作成しているワークシートにコピーすることができます。

ヘルプページのコマンド例をコピーするには、コマンド例の上でマウスの右ボタンを押して[例題のコピー]を選択するか、または[編集]メニューから[例題のコピー]を選択します。



その後、自分が作成しているワークシートにペースト（貼り付け）をすれば、ヘルプに記載されていたコマンド例がすべてコピーされます。

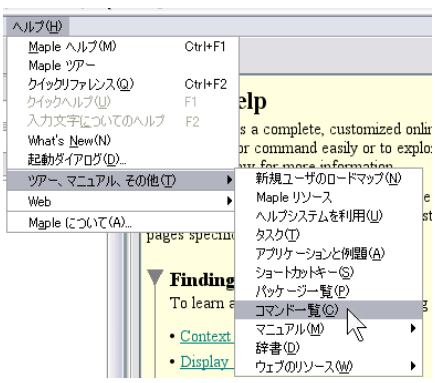
また、ヘルプページ自体をワークシートとして開くための機能も用意されています。



ヘルプブラウザの上図のボタンを押すと、現在開いているヘルプページをワークシートとして開くことができます。ヘルプに記載されているコマンドの引数の指定方法などを参照しながら計算させたい場合などに便利です。

6.4. コマンドやパッケージの一覧を見る

Maple を起動したときに利用可能な組込みコマンドと Maple に標準で用意されているパッケージの一覧を見るには、[ヘルプ]メニューから[ツアー、マニュアル、その他]をポイントし、[コマンド一覧]または[パッケージ一覧]を選択します。



7. もっと Maple を使いこなそう

これまでの章を一通り学習することで、基本的な Maple の操作方法についてマスターしてきたと思います。この章では、Maple を使った様々な応用について挑戦してみましょう。
まだ Maple について十分慣れていない場合でも、紹介されているコマンドをそのまま入力して実行するだけでも構いません。

7.1. 極座標プロットを使った模様作成

これまで私たちが授業などで習ってきた数学上の関数も、極座標プロットで描画すると一味違った形で見ることが出来ます。Maple で極座標プロットを実行するには、**plots** パッケージを読み込みます。（以下のコマンドを実行する前に新しいワークシートを開きましょう）

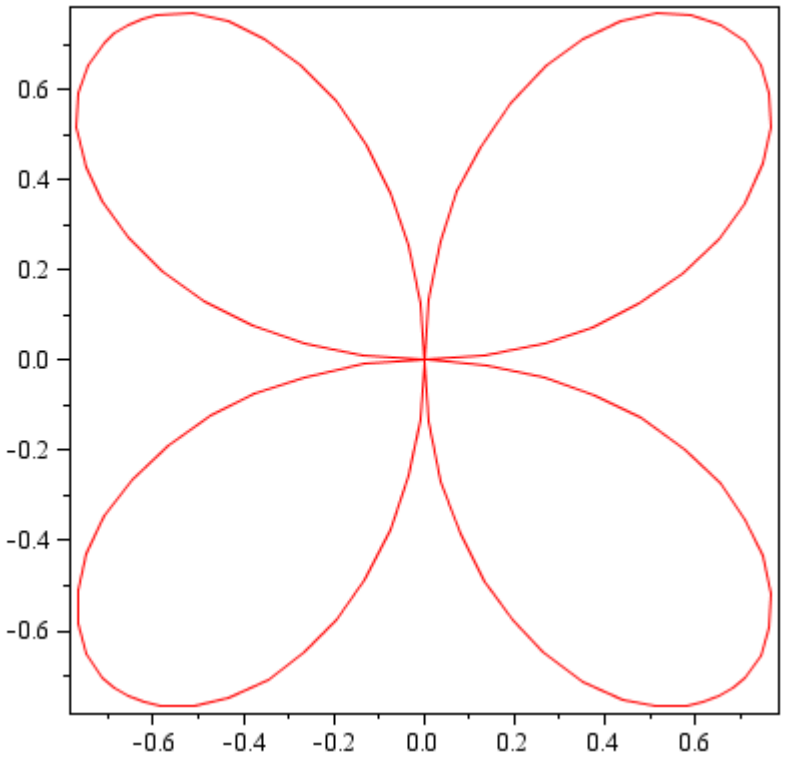
```
> restart;
> with(plots):
```

また、プロットを実行する前に **setoptions** コマンドを用いて、プロットを実行するときに指定するオプションをあらかじめ固定しておきます。こうすることで、毎回コマンド記入時にオプション値を指定する必要がなくなります。

```
> setoptions(scaling=constrained,axes=boxed):
```

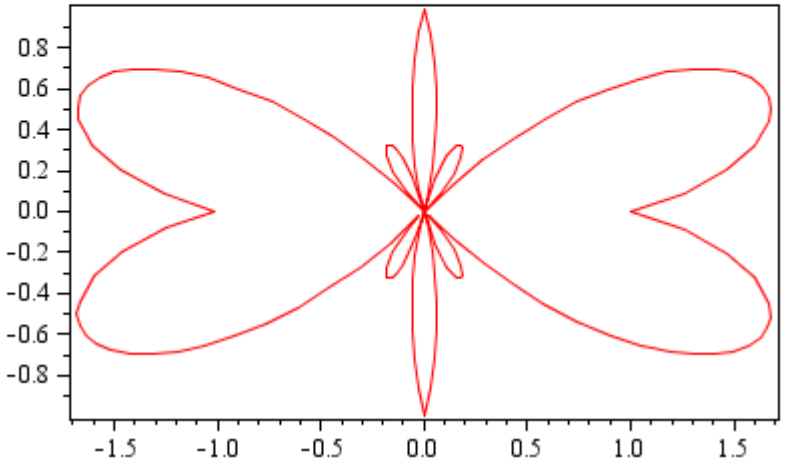
極座標プロットを実行するには **polarplot** コマンドを用います。**polarplot** コマンドの使い方は通常の **plot** コマンドとほとんど同じです。
例えば、sin(2t)というグラフを極座標で描いてみます。

```
> polarplot(sin(2*t),t=0..2*Pi);
```



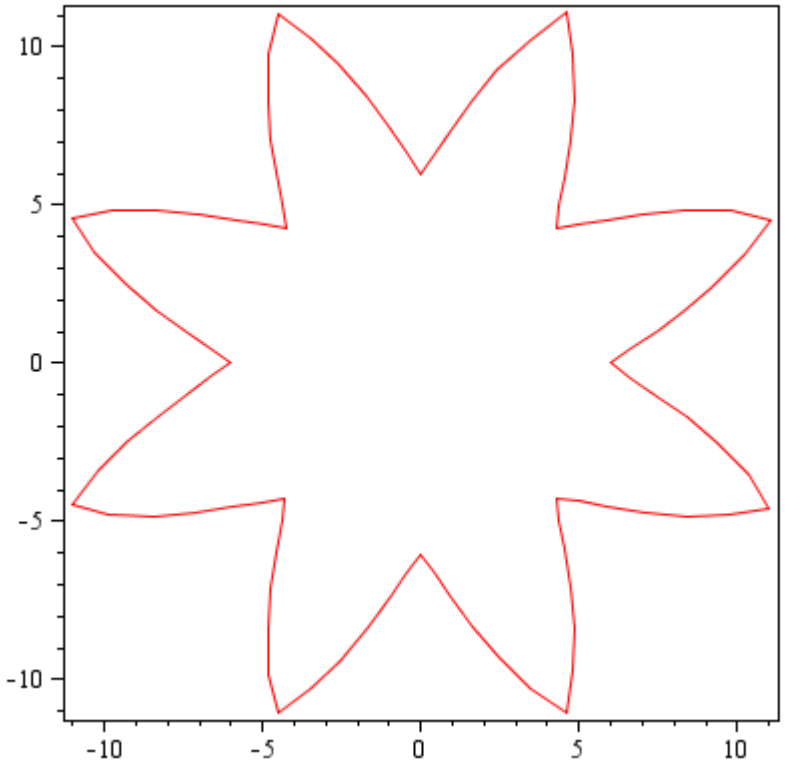
少し数式が長いですが、三角関数を組み合わせることで次のようなグラフが描けます。ここで、abs は絶対値を表す関数です。

```
> polarplot(abs(sin(4*t))+cos(2*t),t=0..2*Pi);
```



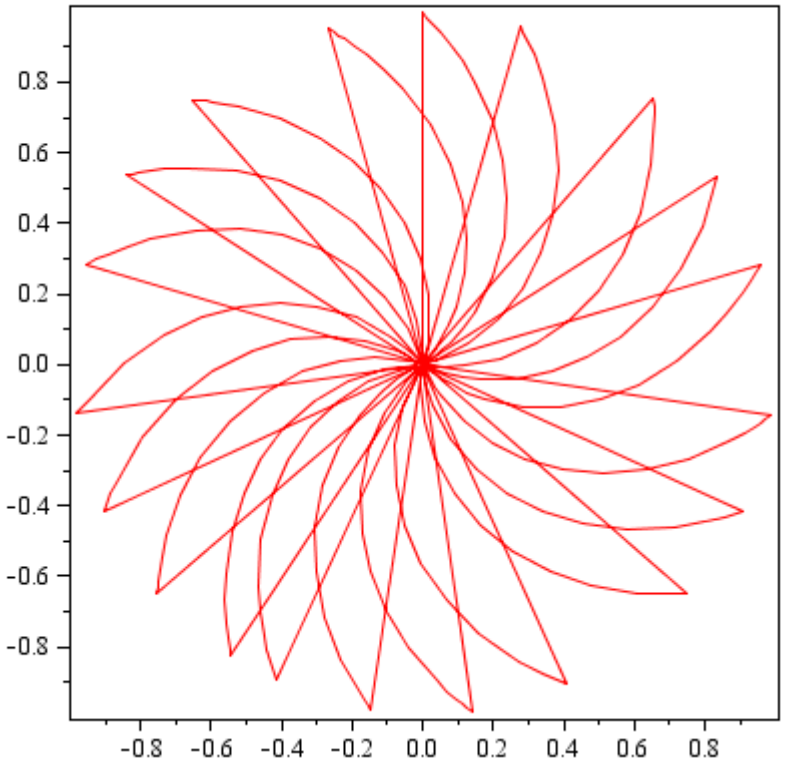
さらに複数の三角関数を組み合わせて、次のような図形も描けます。

```
> polarplot((3+abs(sin(4*t)))*(3-abs(cos(4*t))),t=0..2*Pi);
```



以下の関数は、階段関数と呼ばれている関数です。

```
> polarplot(floor(t)-t,t=0..20);
```



ひとつの関数に対して、パラメータを変化させるだけで様々な図形を作り出すことも出来ます。以下では $a-\sin(t\,k), t=0\sim 2\pi$ という式に対して、a, k を変化させることでいろんな図形を描画するための処理を関数として定義しています。（ただし、f2 では関数の絶対値を描画するようにしています）

```
> f1 := (a,k)->polarplot(a-sin(t*k), t=0..6.3, color=blue);
```

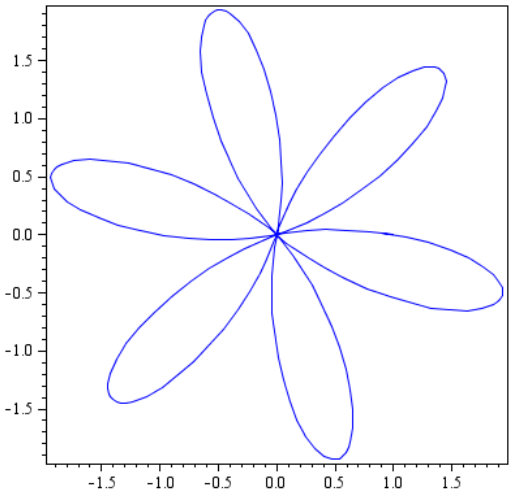
$$f1 := (a,k) \rightarrow plots:\text{-}polarplot(a-\sin(t\,k), t=0..6.3, color=blue) \quad (1)$$

```
> f2 := (a,k)->polarplot(abs(a-sin(t*k)), t=0..6.3, color=
  green);
```

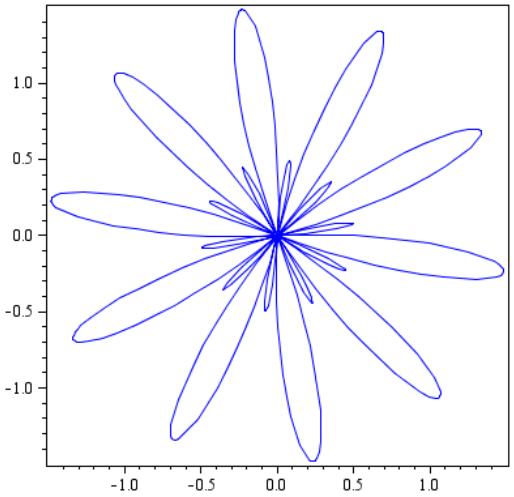
$$f2 := (a,k) \rightarrow plots:\text{-}polarplot(|a-\sin(t\,k)|, t=0..6.3, color=green) \quad (2)$$

ここで定義した f1, f2 を使って図形を描画してみます。

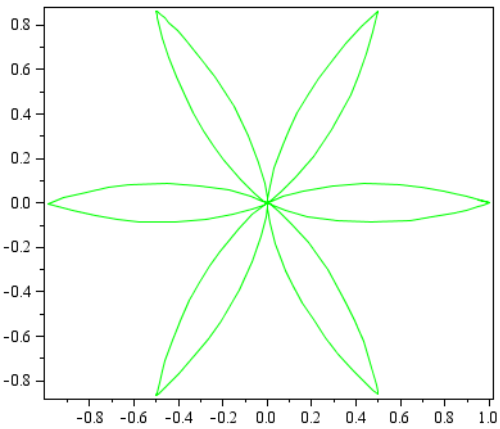
```
> f1(1,6);
```



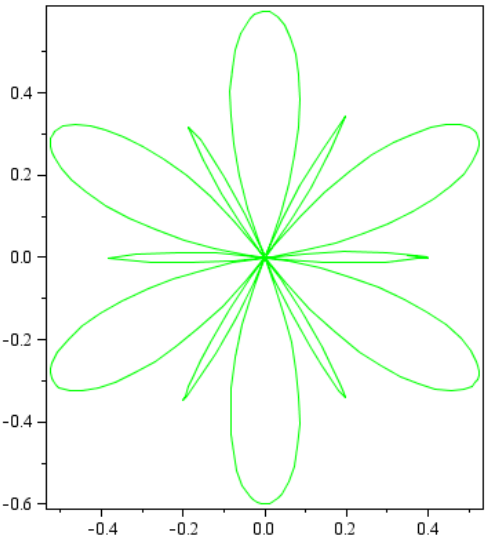
```
> f1(0.5,10);
```



```
> f2(1,3);
```



```
> f2(0.4,3);
```



パラメータ a, k をいろいろと変形させる中で、それぞれのパラメータが図形にどのような変化を与えるか考察してみるとよいでしょう。

7.2. 曲線と積分の応用

グラフを描くだけでも楽しいですが、せっかくなのでもう少し“数学”してみましょう。ここでは、次の関数で与えられる曲線の長さを求めることを考えます。

```
> restart;
> Starr := (a,b,c)->
    [(2+1/2*sin(a*t))*cos(t+sin(b*t)/c),
     (2+1/2*sin(a*t))*sin(t+sin(b*t)/c)]:
```

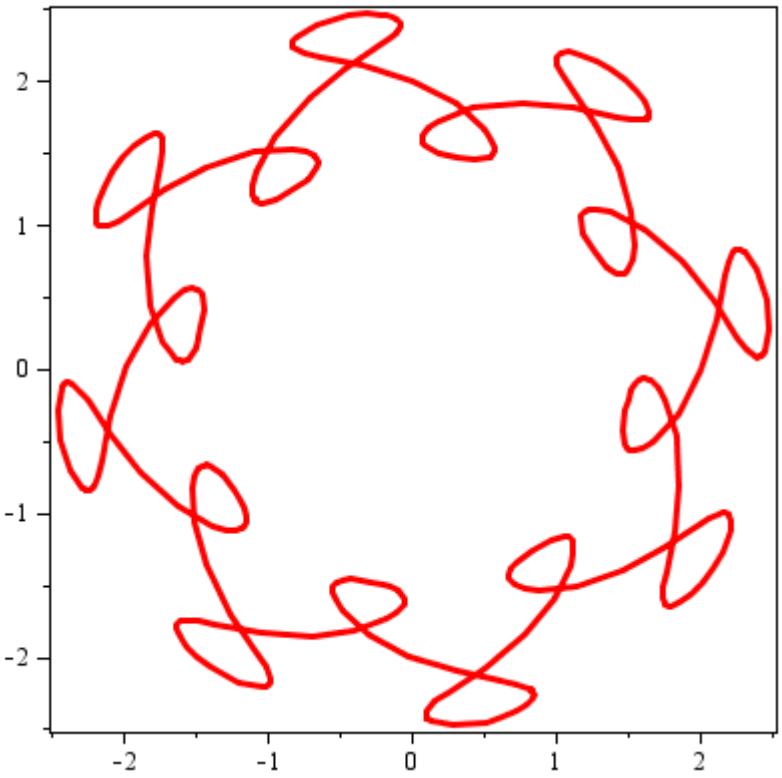
上記は、パラメータ a,b,c を持つ Starr（シュタール）曲線と呼ばれる関数です。このグラフは a,b,c を変化させることで面白い形になります。

まず、定義した Starr 関数を使ってグラフを描画するための関数も定義しておきます。

```
> StarrPlot := (a,b,c)->
    plot([op(Starr(a,b,c)),t=0..2*Pi],
    thickness=3,scaling=constrained,axes=boxed):
```

さっそく描画してみましょう。

```
> StarrPlot(8,16,4);
```



すでに私たちは微積分学により媒介変数で与えられた関数の曲線の長さを求めるための公式を知っています。媒介変数 $x = x(t), y = y(t)$ で与えられる曲線の長さ L の公式は以下で与えられます。

$$L = \int_{t_1}^{t_2} \sqrt{\left(\frac{d}{dt}x(t)\right)^2 + \left(\frac{d}{dt}y(t)\right)^2}\, dt$$

この公式に基づいて、上記で実行した Starr 曲線の長さを計算してみます。

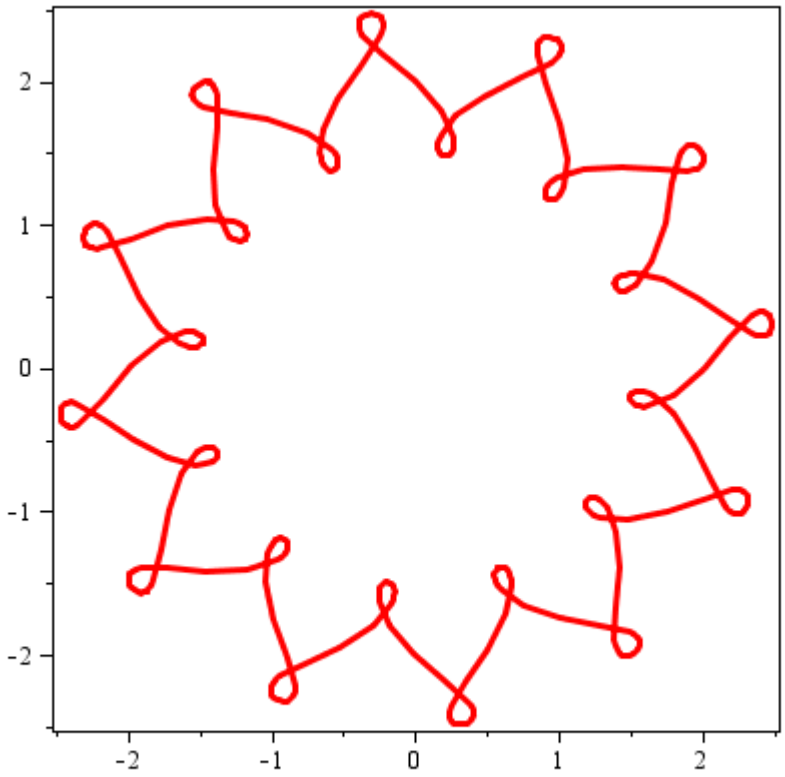
```
> T := Starr(8,16,4):
  evalf(Int(sqrt(diff(T[1],t)^2+diff(T[2],t)^2),t=0..2*Pi));
                              38.10475534
```

このように、積分を用いると上図のような複雑な曲線でもその長さが計算できてしまいます。実際、現実世界の工学計算ではこの公式に基づいて、**必要な材料の量を理論的に計算する**などの目的で用いられています。例えば、この章で用いた模様を刺繍として製作するような場合は、必要となる糸の長さがここでの積分計算により理論的に把握することが出来ます。

なお、ここで、**evalf** コマンドと **Int** コマンドを組合せて使っていることに注意してください。この２つのコマンドの組合せは、指定した関数の数値積分を計算するためのものです。

Starr 曲線はパラメータの値を変えると様々な図形に変化します。いろいろなパラメータで挑戦してみてください。

```
> StarrPlot(12,24,11);
```




```
> T := Starr(12,24,11):
  evalf(Int(sqrt(diff(T[1],t)^2+diff(T[2],t)^2),t=0..2*Pi));
                              31.95159106
```

8. Maple の利用事例

Maple のユーザは、全世界に広がっています。それら世界中の Maple ユーザが作成・開発した例題や目的別のパッケージなどは開発元である Maplesoft 社が運営する Application Center に用意されています。

URL: http://www.maplesoft.com/applications/index.aspx

また、国内代理店であるサイバネットシステム（株）のホームページにもいくつかの利用事例が掲載されています。

URL: http://www.cybernet.co.jp/maple/example/

数式処理ソフトウェア「Maple11」ビギナーズ・チュートリアル

2008 年 5 月

発行　　学習院大学計算機センター
