Mathematica 利用の手引き

TSUBAME Computing Services, Global Scientific Information and Computing Center

2024-06-20

Table of contents

1. はじめに

運用終了

TSUBAME3 は既に運用を終了しています。 TSUBAME4 のマニュアル類はこちら

本書は、Mathematicaを東京工業大学学術国際情報センターのTSUBAMEで利用する方法について説明しています。 また、TSUBAMEを利用するにあたっては、「TSUBAME利用の手引き」もご覧下さい。サーバの利用環境や注意事項などが詳細に記述されていますので、よく読んでください。

Mathematicaの開発・販売元のウルフラム リサーチではMathematicaに関するWebページを公開しています。 次のアドレスを参照してください。

http://www.wolfram.com/mathematica/

1.1. 利用できるバージョン

TSUBAME3で利用可能な最新バージョンについてはTSUBAME計算サービスWebサイトの アプリケーション ページをご確認下さい。
研究に支障がない限り、バグ修正の入っている最新版をご利用下さい。

1.1.1. バージョンの切り替え

module コマンドでmodule ファイルを読み込むことでバージョンの切り替えが可能です。

コマンド例

```
# 11.3.0(デフォルト)を利用する場合
$ module load mathematica/11.3.0
12.0.0を利用する場合
$ module load mathematica/12.0.0
既に別バージョンを読み込んでいた場合は下記コマンドを実行してからmodule loadを実行して下さい
$ module purge
moduleオプションの詳細についてはman moduleもしくはmoduleのman page をご確認ください。
```

1.2. 概要

Mathematicaは、数値計算と数式処理のエンジン、グラフィックスのシステム、プログラミング言語、ドキュメントシステム、他のアプリケーションとの高度な接続性をシームレスに統合しています。
表面的なレベルでは、Mathematicaは使いやすい計算機です。数学、化学、工学、金融関数の世界で最も包括的なセットが、ほとんどの場合マウスをクリックする、あるいはコマンド1つですぐに使えます。しかし、Mathematicaの関数はあらゆるサイズまたは精度の数を扱うことができ、記号計算が可能で、簡単にグラフィックスを表示し、最適な答を返すために自動的にアルゴリズムを切り換え、結果の精度を検証・調整します。たとえ計算している問題の構造をよく知らないユーザでも、常に返された答を信頼することができるのは、このような精巧さのためです。計算を続けている間、ノートブックドキュメントはその完全な記録　インタラクティブでありながらタイプセットも施した形式での入力、出力、グラフィックスを取り続けています。テキストや見出し、教科書からの公式、インターフェースの要素でさえも直接加えることができ、オリジナルの素材を使ってスライドショーやWeb・XML・印刷等による発表用資料を即座に作成することができます。事実、ノートブックドキュメントの技術を使えば、受け手がコンテンツとインタラクトできるような完全にカスタマイズされたインターフェースが簡単に提供できます。
直接的な計算からプログラムされた計算への移行は進化的に進めることができます。Mathematicaではたった1行で意味あるプログラムを書くことができます。方法論、シンタックス、入出力に使うドキュメントは直接計算するときのままで構いません。
Mathematicaは強力なソフトウェア開発環境でもあります。Mathematicaのパッケージは、デバッグしカプセル化してカスタムユーザインターフェースでラップすることができ、しかもこの過程のすべてをMathematicaのシステムの中から行うことができます。また、Java、C、あるいは私有のシステムへのリンクからMathematicaのパワーを使うことも可能です。
Mathematicaに他を寄せ付けない幅広さを与えるもとになっている技術は記号プログラミングです。記号プログラミングによって、データ、関数、グラフィックス、プログラム、完全なドキュメント等のあらゆるタイプのオブジェクトとあらゆる操作を、記号式という一律の方法で表すことができるようになりました。このような一本化は、簡単に学習できるということから各関数の適用範囲が広がるということまで、多くの実際的なメリットを生みました。Mathematicaの生のアルゴリズムパワーがこのような統一原則でさらに強力に拡張されているのです

1.2.1. 用途

- 数十万、数百万個の項を含むことがよくある複雑な記号計算の処理
- データのロード、解析、ビジュアル化
- 方程式、微分方程式、最小化の問題の数値的あるいは記号的な解の探究
- 単純な管理システムから銀河系における衝突、金融デリバティブ、複雑な生物系、化学反応、環境影響の研究、粒子加速器における磁場にいたるまでの数値的モデル化とシミュレーション
- 技術系の企業や金融機関における高速アプリケーション開発(RAD)の促進
- 電子・印刷媒体用のプロ級の品質を備えたインタラクティブな技術報告書や文書の作成
- 幼稚園児から大学院生向けまでの数学・科学的コンセプトの解説
- アメリカ合衆国の特許のような技術情報用のタイプセット
- 技術的な内容のプレゼンテーションやセミナーの実施

1.3. マニュアル

Wolfram Mathematica ドキュメントセンター (wolfram.com)

2. TSUBAME3での利用方法

運用終了

TSUBAME3 は既に運用を終了しています。 TSUBAME4 のマニュアル類はこちら

2.1. Mathematicaの実行

2.1.1. インタラクティブノードでのMathematicaの起動/実行

ログインノードは計算ノードとは別構成となっており、ログインノード上でアプリケーションを実行することは想定されておりません。
ログイン方法を参考にログインノードにログイン後、インタラクティブノードを利用したX転送を参考にノードをX転送付きで確保して下さい。
以下以降の例では、全て計算ノードにログインした状態で行います。

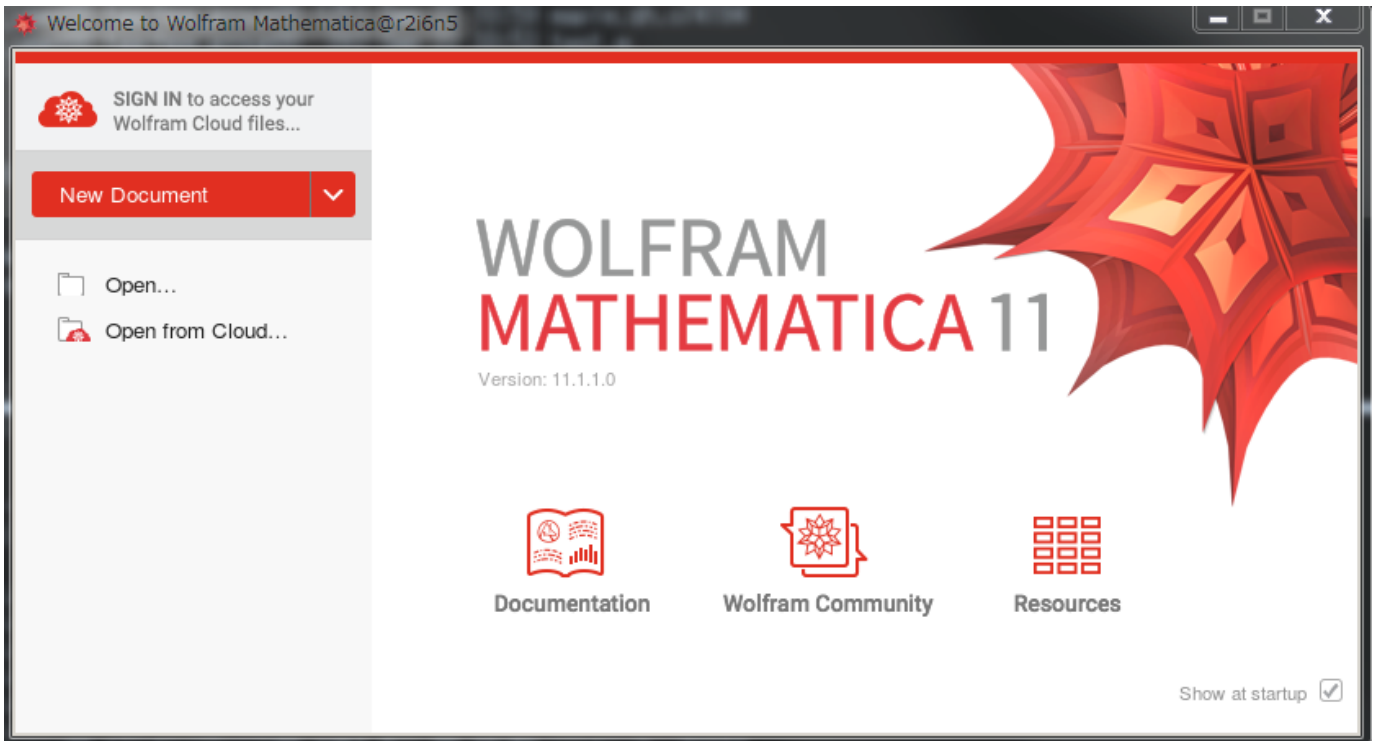
CUIの場合は以下のコマンドを実行してください。

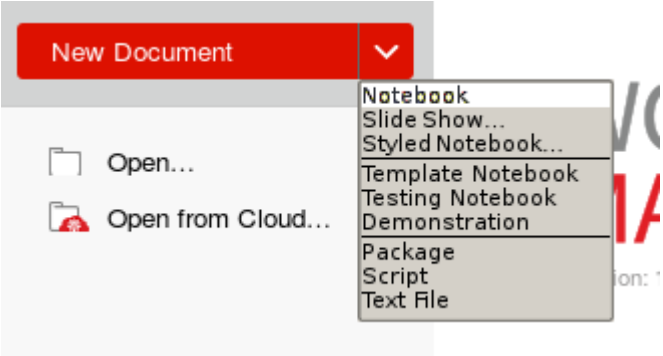
```
$ cd <利用したいディレクトリ>
$ module load mathematica/11.1.1
$ math
Mathematica 11.1.1 Kernel for Linux x86 (64-bit)
Copyright 1988-2017 Wolfram Research, Inc.
In[.1]:=
「Quit」コマンドにより、終了することが出来ます。
> In[1]:=Quit
```

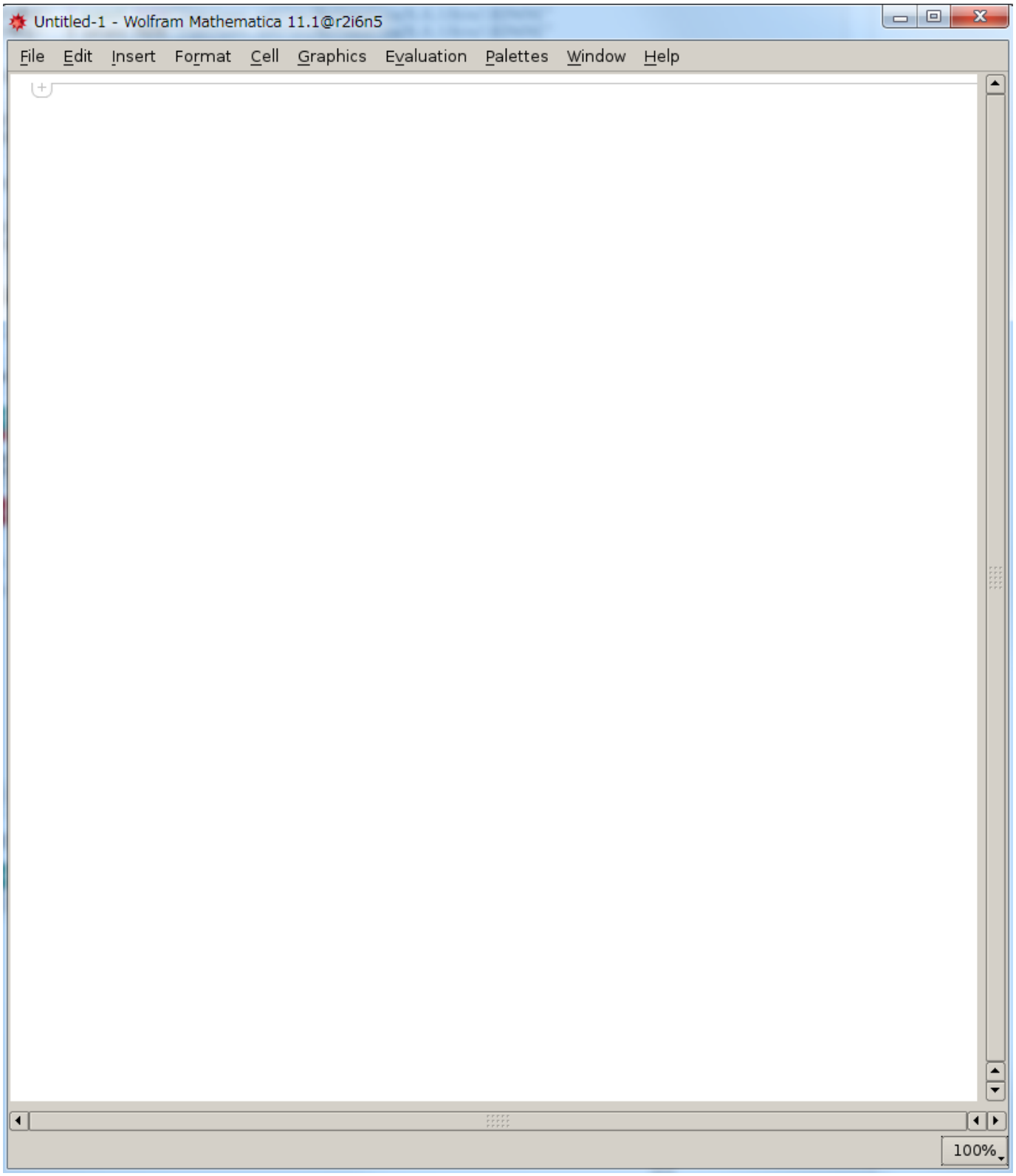
GUIの場合は以下の方法で実施してください。

コマンド実行例
例では割り当てノードとしてr0i0n0が割り当てられた場合を想定しております。
割り当てノードはコマンド実行時に空いているノードですので、明示的にノードを指定することはできません。

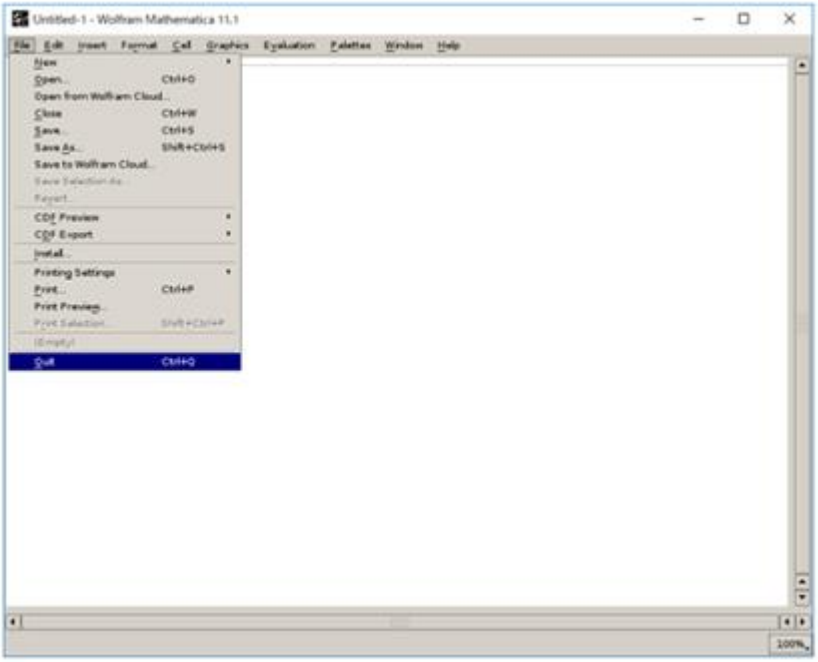
```
r0i0n0:~> module load mathematica/11.1.1
r0i0n0:~> mathematica
```




New Document
Open...
Open from Cloud...
Notebook
Slide Show...
Styled Notebook...
Template Notebook
Testing Notebook
Demonstration
Package
Script
Text File



画面左部の「New Document >Notebook」をクリックすることで、ノートブックが起動します。
終了する場合は、ノートブックのメニューバーから [File]を選択し「Exitをクリックします」


Untitled-1 - Wolfram Mathematica 11.1
File Edit Insert Format Cell Graphics Evaluation Palettes Window Help
New
Open... Ctrl+O
Open from Wolfram Cloud...
Close Ctrl+W
Save... Ctrl+S
Save As... Shift+Ctrl+S
Save to Wolfram Cloud...
Save Selection As...
Export...
CDF Preview
CDF Export
Install...
Printing Settings
Print... Ctrl+P
Print Preview...
Print Selection... Shift+Ctrl+P
(Empty)
Quit Ctrl+Q
100%

3. Mathematica の基本的な使用方法

運用終了

TSUBAME3 は既に運用を終了しています。 TSUBAME4 のマニュアル類はこちら

3.1. Mathematicaを利用するにあたって

3.1.1. 入力コマンドの実行方法

入力コマンドを実行するには、「Shift」キーを押しながら、「Enter」キーを押します。「Enter」キーを押しただけでは、改行しかされません。

3.1.2. 関数名の大文字と小文字

Mathematicaで定義されている記号(例えば、Plot、Sin、Precision、Pi、Iなど)は大文字で始まります。 何らかのユーザ関数を定義する場合は、小文字にした方が良いです。 それは、Mathematica内部で定義された記号とユーザ定義関数を混同しないようにするためです。 Mathematicaは1000個近くの内部記号をもち、また外部パッケージではそれ以上の記号が定義されています。
もし、ユーザ定義関数に大文字で始まる名前をつけたら、おそらく同じ名前が出現すると思います。

3.1.3. 括弧の使用方法

・丸括弧「()」は普通の数学計算と同じで、式をグループ化させます

```
○  x + (1 - y)/2
×  Sin()
```

・中括弧「{}」はリスト、範囲を定義します

```
○  {x, y, z}
×  Sin{}
```

・角括弧「[]」は関数専用で、他には使用できません

```
○  Sin[x^2]
×  [1 + y]/2
```

3.1.4. 「=」の使用方法

・等号「=」は、ある変数にある値を与える際に使われます

```
x=6
y=1 + m/2
```

・コロンと等号の組み合わせ記号「:=」は、関数定義に使われます

```
f[x_]:=x^2 - 1
```

・二重等号「==」は、方程式を定義する時、　つの表現式を比較する時に使われます

例　連立一次方程式

```
{x + y==2, 4x - 2y == 5}
```

例　表現式を比較する

```
In[1]:= x = 2
Out[1]= 2

In[2]:= x == 1
Out[2]= False


In[3]:= x == 2
Out[3]= True
```

3.1.5. ヘルプ機能について

・「?」で、ある関数の情報が分かります

```
In[1]:= ?Plot
  Plot[f,{x,Subscript[x, min],Subscript[x, max]}] generates a plot of f as a function of x from Subscript[x, min] to Subscript[x, max].
  Plot[{Subscript[f, 1],Subscript[f, 2],...},{x,Subscript[x, min],Subscript[x, max]}] plots several functions Subscript[f, i]. >>
```

・ある特定の文字を含むすべての記号を知ることもできます「*」は任意の文字列を意味します

```
In[2]:= ?Plot*
Plot              PlotJoined        PlotPoints        PlotRangePadding
Plot3D            PlotLabel         PlotRange         PlotRegion
Plot3Matrix       PlotLayout        PlotRangeClipping PlotStyle
PlotDivision      PlotMarkers
```

・「??」を付けると、使用方法に加えてオプションと属性を知ることができます

```
In[3]:= ??Plot
  Plot[f,{x,Subscript[x, min],Subscript[x, max]}] generates a plot of f as a function of x from Subscript[x, min] to Subscript[x, max].
  Plot[{Subscript[f, 1],Subscript[f, 2],...},{x,Subscript[x, min],Subscript[x, max]}] plots several functions Subscript[f, i]. >>

  Attributes[Plot] = {HoldAll, Protected}

  Options[Plot] = {AlignmentPoint -> Center, AspectRatio -> GoldenRatio^(-1),
      Axes -> True, AxesLabel -> None, AxesOrigin -> Automatic,
      AxesStyle -> {}, Background -> None, BaselinePosition -> Automatic,
      BaseStyle -> {}, ClippingStyle -> None, ColorFunction -> Automatic,
      ColorFunctionScaling -> True, ColorOutput -> Automatic,
      ContentSelectable -> Automatic, CoordinatesToolOptions -> Automatic,
      DisplayFunction :> $DisplayFunction, Epilog -> {},
      Evaluated -> System`Private`$Evaluated, EvaluationMonitor -> None,
      Exclusions -> Automatic, ExclusionsStyle -> None, Filling -> None,
      FillingStyle -> Automatic, FormatType :> TraditionalForm, Frame -> False,
      FrameLabel -> None, FrameStyle -> {}, FrameTicks -> Automatic,
      FrameTicksStyle -> {}, GridLines -> None, GridLinesStyle -> {},
      ImageMargins -> 0., ImagePadding -> All, ImageSize -> Automatic,
      ImageSizeRaw -> Automatic, LabelStyle -> {}, MaxRecursion -> Automatic,
      Mesh -> None, MeshFunctions -> {#1 & }, MeshShading -> None,
      MeshStyle -> Automatic, Method -> Automatic,
      PerformanceGoal :> $PerformanceGoal, PlotLabel -> None,
      PlotPoints -> Automatic, PlotRange -> {Full, Automatic},
      PlotRangeClipping -> True, PlotRangePadding -> Automatic,
      PlotRegion -> Automatic, PlotStyle -> Automatic,
      PreserveImageOptions -> Automatic, Prolog -> {},
      RegionFunction -> (True & ), RotateLabel -> True, Ticks -> Automatic,
      TicksStyle -> {}, WorkingPrecision -> MachinePrecision
```

3.2. 数値計算

3.2.1. 電卓として使う

・足し算、引き算

```
In[1]:= 2 + 3
Out[1]= 5
In[2]:= 0.17 - 1.5
Out[2]= -1.33
```

・掛け算 アスタリスク「*」とスペースは掛け算を表します

```
In[3]:= 32 * 5 7
Out[3]= 1120
```

・割り算

```
In[4]:= 7/2
        7
Out[4]= -
        2
```

・べき乗

```
In[5]:= 3^12
Out[5]= 531441
```

3.2.2. 数の型と数学定数

・整数、有理数、実数

有理数に関する処理は、デフォルトでは、計算されずにそのままの形で表示されます

```
In[1]:= 1/2 - 1/3 + 1/5 -1/7
        47
Out[1]= ---
        210
```

入力データに小数点数が混合している場合は、小数で表示されます

```
In[2]:= 0.5 - 1/3 + 1/5 -1/7
Out[2]= 0.22381
```

有理数から実数へ変換する場合は、関数「N」を使用します

```
In[3]:= 3/5
        3
Out[3]= -
        5

In[4]:= N[3/5]
Out[4]= 0.6
```

・複素数

「-1」の平方根(虚数単位)を「I」(アルファベット大文字のアイ)で表します

```
In[5]:= Sqrt[-1]
Out[5]= I
```

また、複素数は実部と虚部に分かれて表示されます

```
In[6]:= (3 + 2I)/(5 - 3I)
        9    19 I
Out[6]= -- + ----
        34    34

In[7]:= (3.0 + 2I)/(5 - 3I)
Out[7]= 0.264706 + 0.558824 I
```

3.2.3. 数学関数

・三角関数

すべての三角関数が利用できます

```
In[1]:= Sin[Pi/2]
Out[1]= 1

In[2]:= Cos[Pi/2]
Out[2]= 0

In[3]:= Sin[-Pi/3]
        -Sqrt[3]
Out[3]= --------
           2

In[4]:= N[Sin[-Pi/3]]
Out[4]= -0.866025
```

```
In[5]:= ArcCos[0]
        Pi
Out[5]= --
        2

In[6]:= N[Sinh[Pi/3]]
Out[6]= 1.24937
```

3.2.4. 行列

・マトリクスの作成 マトリクスを作成するには、Table関数[]を使用します

```
In[1]:= ?Table
  Table[expr,{Subscript[i, max]}] generates a list of Subscript[i, max] copies of expr.
  Table[expr,{i,Subscript[i, max]}] generates a list of the values of expr when i runs from 1 to Subscript[i, max].
  Table[expr,{i,Subscript[i, min],Subscript[i, max]}] starts with i=Subscript[i, min].
  Table[expr,{i,Subscript[i, min],Subscript[i, max],di}] uses steps di.
  Table[expr,{i,{Subscript[i, 1],Subscript[i, 2],...}}] uses the successive values Subscript[i, 1], Subscript[i, 2], ....
  Table[expr,{i,Subscript[i, min],Subscript[i, max]},{j,Subscript[j, min],Subscript[j, max]},...] gives a nested list. The list associated with i is
outermost.>>

In[2]:= Table[i, {i, 1, 10}]
Out[2]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In[3]:= Table[a, {i, 1, 5}]
Out[3]= {a, a, a, a, a}

In[4]:= Table[i^2, {i, 1, 10, 0.5}]
Out[4]= {1., 2.25, 4., 6.25, 9., 12.25, 16., 20.25, 25., 30.25, 36., 42.25,
>    49., 56.25, 64., 72.25, 81., 90.25, 100.}
```

・マトリクスの表示

行列を一般的な形で表示するには、MatrixForm[]関数を使用します

```
In[1]:= ?MatrixForm
  MatrixForm[list] prints with the elements of list arranged in a regular array. >>

In[2]:= matrix = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} }
Out[2]= {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}

In[3]:= MatrixForm[matrix]
Out[3]//MatrixForm= 1   2   3

                    4   5   6

                    7   8   9
```

・行列の処理

```
In[4]:= matrix2 = Table[ 1/(i + j), {i, 1, 3}, {j, 1, 3} ]

          1  1  1    1  1  1    1  1  1
Out[4]= {{-, -, -}, {-, -, -}, {-, -, -}}
          2  3  4    3  4  5    4  5  6

In[5]:= MatrixForm[matrix2]

Out[5]//MatrixForm= 1   1   1
                    -   -   -
                    2   3   4

                    1   1   1
                    -   -   -
                    3   4   5

                    1   1   1
                    -   -   -
                    4   5   6
```

・逆行列を求める

逆行列を求めるには、Inverse[]関数を使用します

```
In[6]:= Inverse[matrix2]
Out[6]= {{72, -240, 180}, {-240, 900, -720}, {180, -720, 600}}
```

・行列式を求める

行列式を求めるには、Det[]関数を使用します

```
In[7]:= Det[matrix2]
          1
Out[7]= -----
        43200
```

・行列の固有値を求める

固有値を求めるには、Eigenvalues[]関数を使用します

```
In[9]:= Eigenvalues[matrix]

         3 (5 + Sqrt[33])  3 (5 - Sqrt[33])
Out[9]= {----------------, ----------------, 0}
                2                 2
```

3.2.5. 方程式

・一元方程式

方程式を解くには、Solve[]関数を使用します。Solve[]関数は、記号的(例　ルート、分数表示する)に方程式の解を生成します。方程式は「==」(二重等号)で表します。Solve[]関数の第二引数は、変数を意味し、これについて方程式を解きます。

```
In[1]:= Solve[a x^2 + b x + c == 0, x]
                                2                                2
                -b - Sqrt[b  - 4 a c]          -b + Sqrt[b  - 4 a c]
Out[1]= {{x -> ---------------------}, {x -> ---------------------}}
                         2 a                            2 a
```

NSolve[]関数は、数値的(例　ルート、分数表示しない)に方程式の解を生成します

```
In[2]:= Solve[3x^2 + 4x + 1 == 0, x]
                                 1
Out[2]= {{x -> -1}, {x -> -(-)}}
                                 3

In[3]:= NSolve[3x^2 + 4x + 1 == 0, x]
Out[3]= {{x -> -1.}, {x -> -0.333333}}
```

・連立方程式

連立方程式を解くには、Solve[]関数、NSolve[]関数を使用します

```
In[4]:= Solve[ { x + y == 2, x - 3y + z ==3, x - y + z ==0 }, { x, y ,z } ]
                7          3
Out[4]= {{x -> -, y -> -(-), z -> -5}}
                2          2
In[5]:= NSolve[ { x + y == 2, x -3y + z ==3, x -y + z ==0 }, { x, y ,z } ]
Out[5]= {{x -> 3.5, y -> -1.5, z -> -5.}}
```

・特殊な一元方程式

以下のような方程式を解くには、FindRoot[]関数が使用できます

```
In[1]:= FindRoot[ 3 Cos[x] == Log[x], {x, 1} ]
Out[1]= {x -> 1.44726}
```

・微分方程式

微分方程式を解くには、DSolve[]関数、NDSolve[]関数を使用します 以下に初期条件を伴う簡単な微分方程式の記号解の例を示します

```
In[1]:= DSolve [{y'[x] == a y[x] + 1, y[0] == 0}, y[x], x]
                          a x
                    -1 + E
Out[1]= {{y[x] -> ---------}}
                        a
```

以下に非線形微分方程式の数値解の例を示します。結果は関数 y の規則です InterpolatingFunction は <> で示される数値データを含む数値関数を表します

```
Out[1]= {{y -> InterpolatingFunction[{{0., 50.}}, <>]}}
```

3.3. 数式処理

3.3.1. 代数

・基本的な式の操作

記号演算の式では、指定しない限り演算は行いません。次式を実行しても自動的に昇順に並べかえるだけです。

```
In[1]:= 2 x + x ^2 + 5
                  2
Out[1]= 5 + 2 x + x

In[2]:= (x + z) + y + x^2 - a y^2 + b z^3
            2          2          3
Out[2]= x + x  + y - a y  + z + b z
```

同類項はただ纏められます。

```
In[3]:= (1 + 3x) + (5x + 2) - (4y + 5) + (2y + 3)
Out[3]= 1 + 8 x - 2 y

In[4]:= E^((3x - 4x) * (z + 5z))
         -6 x z
Out[4]= E
```

基本的な単純化をし、まとめられる数学関数はまとめます

```
In[5]:= Sec[x]/(Cot[x]Cos[x])
             2
Out[5]= Sec[x]  Tan[x]

In[6]:= BesselY[5/2, x]
             2             3 Cos[x]   3 Sin[x]
        Sqrt[--] (Cos[x] - -------- - --------)
             Pi                2         x
                              x
Out[6]= ----------------------------------------
                       Sqrt[x]
```

数式を展開するには、その旨の指定をする必要があります。数式を展開するには、Expand[]関数を使用します

```
In[7]:= (a x + b y + c z)^3
                          3
Out[7]= (a x + b y + c z)

In[8]:= Expand[%]
         3  3      2    2          2    2    3  3      2    2
Out[8]= a  x  + 3 a  b x  y + 3 a b  x y  + b  y  + 3 a  c x  z +
                   2    2          2    2        2    2    3  3
>    6 a b c x y z + 3 b  c y  z + 3 a c  x z  + 3 b c  y z  + c  z
```

数式の単純化は自動的には実行されません。数式の単純化を行うには、主にSimplify[]関数を使用しますこの関数は、入力された数式を、同等で最も単純な形で出力します

```
In[9]:= (1 -x^3)(1 + x^3 + x^6)
              3        3    6
Out[9]= (1 - x ) (1 + x  + x )

In[10]:= Simplify[(1 - x^3)(1 + x^3 + x^6)]
             9
Out[10]= 1 - x
```

数式の単純化には、Simplify[]関数よりもパワフルなFullSimplify[]関数があります しかし、実行時間は長くなります

```
In[11]:= FullSimplify[Gamma[1 - z] Gamma[z]]
Out[11]= Pi Csc[Pi z]
```

数式の因数分解であれば、Factor[]関数があります

```
In[12]:= Factor[1 - 2x + x^2]
                  2
Out[12]= (-1 + x)
```

・オプションの設定

Mathematicaは、各関数について様々なオプションが設定されています。 オプションを修正することで、さらに多様な処理ができます。
例として、Factor[]関数を取り上げますデフォルトの設定では、因数分解は整数の範囲で行われます

```
In[13]:= Factor[x^4 - 1]
                                  2
Out[13]= (-1 + x) (1 + x) (1 + x )
```

複素数を含む整数レベルまで拡張して行う場合、Factor[]関数のオプション設定を調べます

```
In[14]:= Options[Factor]
Out[14]= {Extension -> None, GaussianIntegers -> False, Modulus -> 0,
>    Trig -> False}
```

GaussianIntegersをTrueにすれば、複素数レベルまで処理することが分かります

```
In[15]:= Factor[ x^4 -1, GaussianIntegrts -> True]
Factor::optx: Unknown option GaussianIntegrts in
                  4
    Factor[-1 + x , GaussianIntegrts -> True].
                    4
Out[15]= Factor[-1 + x , GaussianIntegrts -> True]
```

3.3.2. 変数の代入について

ある数式に変数を代入するには(数式) /. (変数)->(値)と入力します

```
In[1]:= x + y + 2z /. z -> 5
Out[1]= 10 + x + y

In[2]:= x + y + 2z /. z -> 1 + d
Out[2]= 2 (1 + d) + x + y

In[3]:= ex = Log[(1 - x)/x]
            1 - x
Out[3]= Log[-----]
               x

In[4]:= ex /. x -> 0.3
Out[4]= 0.847298

In[5]:= Sqrt[x^2 + 2x + y^3 + 6y + 1] /. {x -> 2, y-> 3}
Out[5]= 3 Sqrt[6]

In[6]:= Sqrt[x^2 + 2x + y^3 + 6y + 1] /. {{x -> 2, y-> 3}, {x -> 3, y -> 5}}
Out[6]= {3 Sqrt[6], 3 Sqrt[19]}
```

3.3.3. 方程式の求解

Solve[]関数を用いると、方程式、連立方程式の一般解を求められます

```
In[1]:= Solve[a x^2 + b x + c == 0, x]
                          2                              2
               -b - Sqrt[b  - 4 a c]          -b + Sqrt[b  - 4 a c]
Out[1]= {{x -> ---------------------}, {x -> ---------------------}}
                       2 a                            2 a
```

Solve[]関数では、連立方程式を方程式のリストとして入力し、その後に変数をリストとして指定します

```
In[2]:= Solve[{x + 3y ==a, 5x + 2y == b}, {x, y}]
                -2 a + 3 b       5 a   b
Out[2]= {{x -> ----------, y -> --- - --}}
                    13           13    13
```

また、LinearSolve[]関数を用いることで、行列により連立方程式を解くことができます

```
In[3]:= coef = {{1, 3}, {5, 2}}
Out[3]= {{1, 3}, {5, 2}}

In[4]:= LinearSolve[coef, {a, b}]
         -2 a   3 b  5 a - b
Out[4]= {---- + ---, -------}
          13    13     13
```

3.3.4. 解析学

・微分

微分を行うには、D[]関数を使用します

```
In[1]:= D[x^5 - 3x + E^(2x + 1), x]
                1 + 2 x      4
Out[1]= -3 + 2 E        + 5 x

In[2]:= D[Cos[n Pi x], x]
Out[2]= -(n Pi Sin[n Pi x])
```

・積分

積分を行うには、Integrate[]関数を使用します

```
In[1]:= Integrate[Cos[n Pi x], x]
        Sin[n Pi x]
Out[1]= -----------
           n Pi
```

・テイラー展開

テイラー展開を行うには、Series[]関数を使用します

```
In[1]:= Series[Cos[n Pi x], {x, 0, 7}]
             2   2  2    4   4  4    6   6  6
            n  Pi  x    n  Pi  x    n  Pi  x        8
Out[1]= 1 - --------- + --------- - --------- + O[x]
                2          24          720
```

O[x]^8は、余剰項が8次から始まることを示します この項を除くためには、Normal[]関数を使用します

```
In[2]:= Normal[%]
             2   2  2    4   4  4    6   6  6
            n  Pi  x    n  Pi  x    n  Pi  x
Out[2]= 1 - --------- + --------- - ---------
                2          24          720
```

・極限

極限を求めるには、Limit[]関数を使用します

```
In[1]:= Limit[(x^2 - 4)/(x - 2), x -> 2]
Out[1]= 4

In[2]:= Limit[Sin[x]/x, x -> 0]
Out[2]= 1
```

3.4. グラフィックス

3.4.1. 2次元グラフィックス

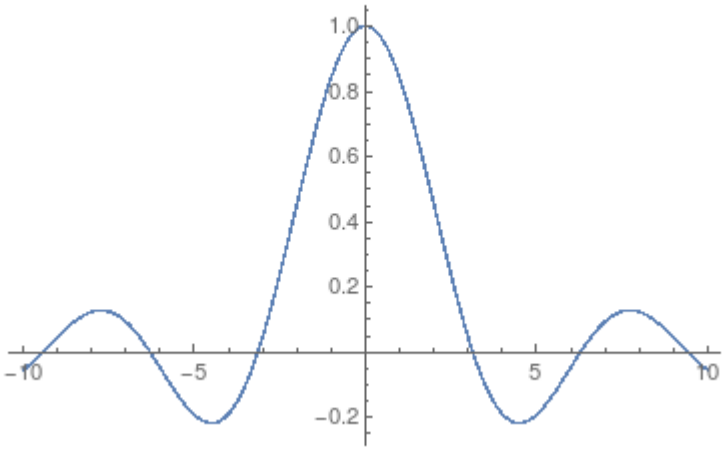
・標準的な2次元作図

標準的な2次元作図を行う場合、Plot[]関数を使用します
図の出力の際はSHIFT+ENTERを押してください。

```
Plot[Sin[x]/x, {x, -10, 10}]
```

```
In[1]:= Plot[Sin[x] / x, {x, -10, 10}]
```

Out[1]=



Plot[]関数には、様々なオプションがあります

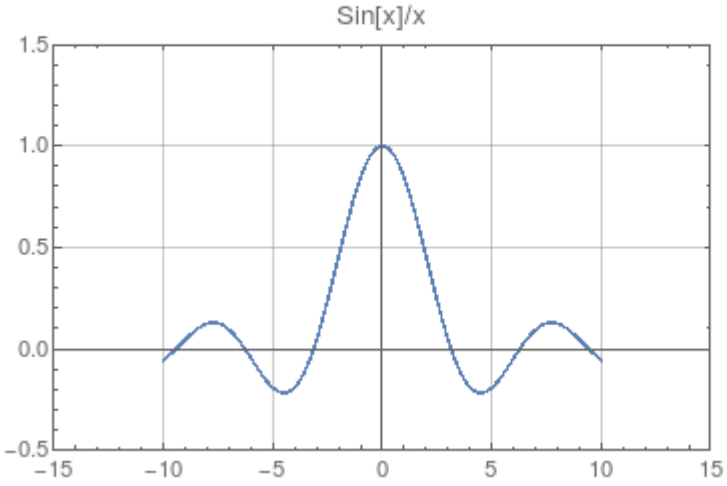
```
In[1]:= Options[Plot]

Out[1]= {AlignmentPoint -> Center, AspectRatio -> -----------, Axes -> True,
                                                 GoldenRatio

>    AxesLabel -> None, AxesOrigin -> Automatic, AxesStyle -> {},
>    Background -> None, BaselinePosition -> Automatic, BaseStyle -> {},
>    ClippingStyle -> None, ColorFunction -> Automatic,
>    ColorFunctionScaling -> True, ColorOutput -> Automatic,
>    ContentSelectable -> Automatic, CoordinatesToolOptions -> Automatic,
>    DisplayFunction :> $DisplayFunction, Epilog -> {},
>    Evaluated -> Automatic, EvaluationMonitor -> None,
>    Exclusions -> Automatic, ExclusionsStyle -> None, Filling -> None,
>    FillingStyle -> Automatic, FormatType :> TraditionalForm,
>    Frame -> False, FrameLabel -> None, FrameStyle -> {},
>    FrameTicks -> Automatic, FrameTicksStyle -> {}, GridLines -> None,
>    GridLinesStyle -> {}, ImageMargins -> 0., ImagePadding -> All,
>    ImageSize -> Automatic, ImageSizeRaw -> Automatic, LabelStyle -> {},
>    MaxRecursion -> Automatic, Mesh -> None, MeshFunctions -> {#1 & },
>    MeshShading -> None, MeshStyle -> Automatic, Method -> Automatic,
>    PerformanceGoal :> $PerformanceGoal, PlotLabel -> None,
>    PlotPoints -> Automatic, PlotRange -> {Full, Automatic},
>    PlotRangeClipping -> True, PlotRangePadding -> Automatic,
>    PlotRegion -> Automatic, PlotStyle -> Automatic,
>    PreserveImageOptions -> Automatic, Prolog -> {},
>    RegionFunction -> (True & ), RotateLabel -> True, Ticks -> Automatic,
>    TicksStyle -> {}, WorkingPrecision -> MachinePrecision}
```

上のグラフに名前、フレーム、表示範囲を設定して、表示します。

```
Plot[Sin[x]/x, {x, -10, 10}, Frame -> True, PlotLabel -> "Sin[x]/x",
  GridLines -> Automatic, PlotRange -> {{-15, 15}, {-0.5, 1.5}}]
```

```
In[1]:= Plot[Sin[x] / x, {x, -10, 10}, Frame → True, PlotLabel → "Sin[x]/x",
  GridLines → Automatic, PlotRange → {{-15, 15}, {-0.5, 1.5}}]
```

Out[1]=

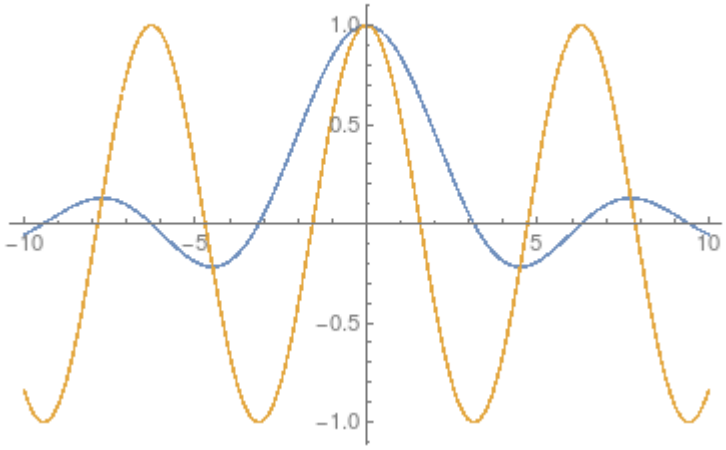


・複数の関数を重ね合わせて表示させる

```
Plot[{Sin[x]/x, Cos[x]}, {x, -10, 10}]
```

```
In[2]:= Plot[{Sin[x] / x, Cos[x]}, {x, -10, 10}]
```

Out[2]=

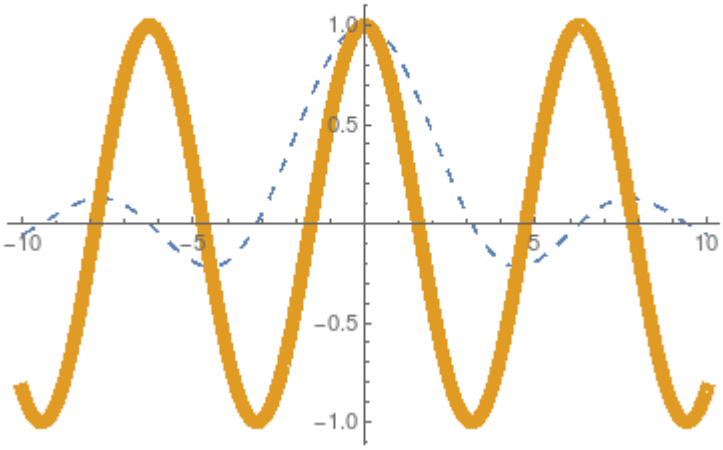


デフォルトではどちらも同じスタイルでグラフが描かれるので、区別できるように別々のスタイルで定義します

```
Plot[{Sin[x]/x, Cos[x]}, {x, -10, 10},
  PlotStyle -> {{Dashing[{0.02}]}, {Thickness[0.02]}}]
```

```
In[3]:= Plot[{Sin[x] / x, Cos[x]}, {x, -10, 10},
          PlotStyle → {{Dashing[{0.02}]}, {Thickness[0.02]}}]
```

Out[3]=



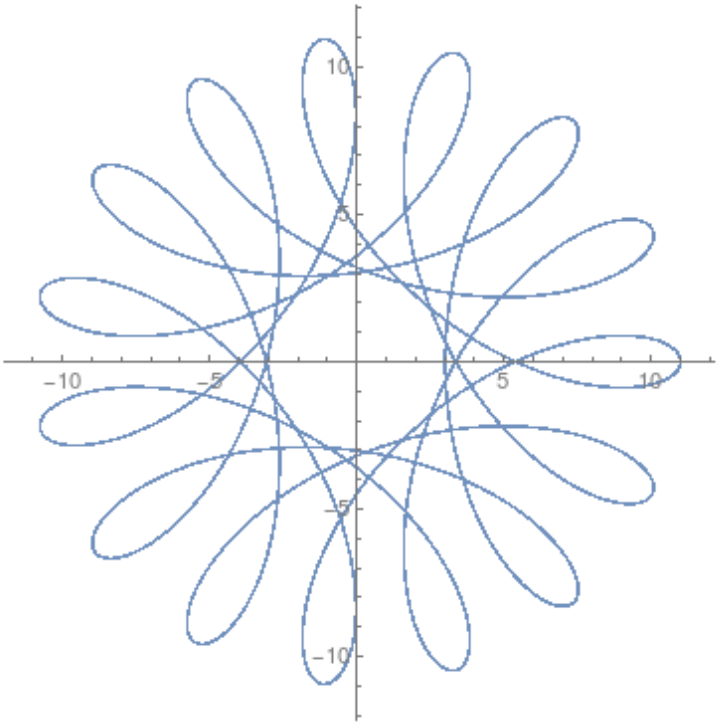
・2次元曲線

簡単な関数で表せない2次元曲線(例　サイクロイド曲線、アステロイド曲線)の作図する場合、ParametricPlot[]関数を使用します

```
ParametricPlot[{4Cos[-11t/4] + 7Cos[t], 4Sin[-11t/4]+7Sin[t]}, {t, 0, 8Pi},
  AspectRatio -> Automatic]
```

```
In[5]:= ParametricPlot[{4 Cos[-11 t / 4] + 7 Cos[t], 4 Sin[-11 t / 4] + 7 Sin[t]}, {t, 0, 8 Pi},
          AspectRatio → Automatic]
```

Out[5]=



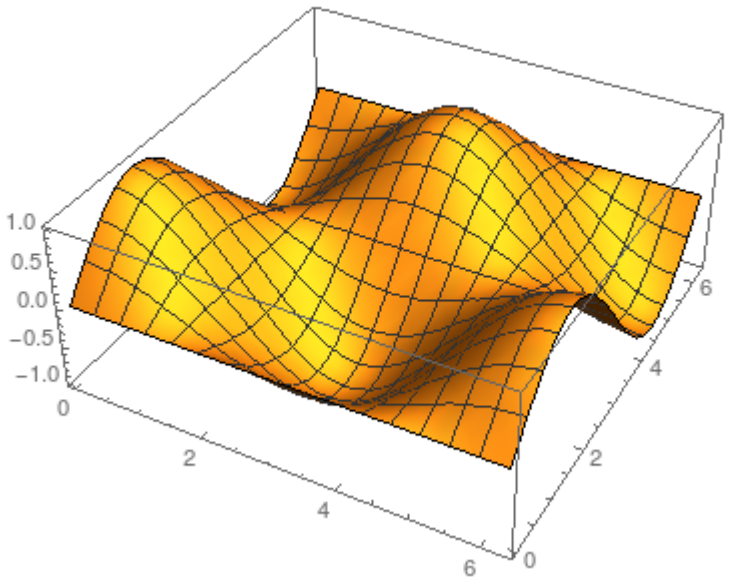
3.4.2. 3次元グラフィックス

・標準的な3次元作図 標準的な　次元作図を行う場合、Plot3D[]関数を使用します

```
Plot3D[Cos[x]Sin[y], {x, 0, 2Pi}, {y, 0, 2Pi}]
```

In[6]:= `Plot3D[Cos[x] Sin[y], {x, 0, 2 Pi}, {y, 0, 2 Pi}]`

Out[6]=



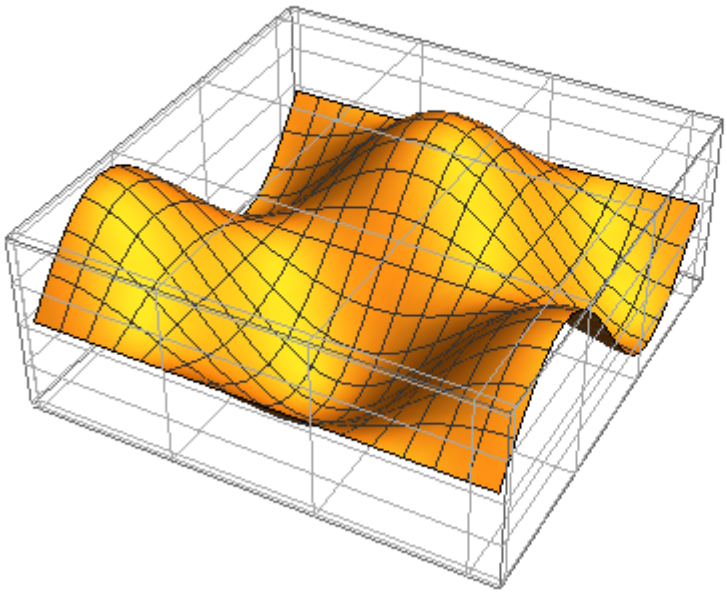
Plot3D[]関数には、様々なオプションがあります

```
In[1]:= Options[Plot3D]
Out[1]= {AlignmentPoint -> Center, AspectRatio -> Automatic,
>    AutomaticImageSize -> False, Axes -> True, AxesEdge -> Automatic,
>    Background -> None, BaselinePosition -> Automatic, BaseStyle -> {},
>    BoundaryStyle -> GrayLevel[0], Boxed -> True, BoxRatios -> {1, 1, 0.4},
>    BoxStyle -> {}, ClippingStyle -> Automatic, ColorFunction -> Automatic,
>    ColorFunctionScaling -> True, ColorOutput -> Automatic,
>    ContentSelectable -> Automatic, ControllerLinking -> Automatic,
>    ControllerMethod -> Automatic, ControllerPath -> Automatic,
>    CoordinatesToolOptions -> Automatic,
>    DisplayFunction :> $DisplayFunction, Epilog -> {},
>    Evaluated -> Automatic, EvaluationMonitor -> None,
>    Exclusions -> Automatic, ExclusionsStyle -> None, FaceGrids -> None,
>    FaceGridsStyle -> {}, Filling -> None, FillingStyle -> Opacity[0.5],
>    FormatType :> TraditionalForm, ImageMargins -> 0., ImagePadding -> All,
>    ImageSize -> Automatic, LabelStyle -> {}, Lighting -> Automatic,
>    MaxRecursion -> Automatic, Mesh -> Automatic,
>    MeshFunctions -> {#1 & , #2 & }, MeshShading -> None,
>    MeshStyle -> Automatic, Method -> Automatic,
>    NormalsFunction -> Automatic, PerformanceGoal :> $PerformanceGoal,
>    PlotLabel -> None, PlotPoints -> Automatic,
>    PlotRange -> {Full, Full, Automatic}, PlotRangePadding -> Automatic,
>    PlotRegion -> Automatic, PlotStyle -> Automatic,
>    PreserveImageOptions -> Automatic, Prolog -> {},
>    RegionFunction -> (True & ), RotationAction -> Fit,
>    SphericalRegion -> False, TextureCoordinateFunction -> Automatic,
>    TextureCoordinateScaling -> Automatic, Ticks -> Automatic,
>    TicksStyle -> {}, ViewAngle -> Automatic, ViewCenter -> Automatic,
>    ViewMatrix -> Automatic, ViewPoint -> {1.3, -2.4, 2.}, ViewRange -> All,
>    ViewVector -> Automatic, ViewVertical -> {0, 0, 1},
>    WorkingPrecision -> MachinePrecision}
```

```
Plot3D[Cos[x]Sin[y], {x, 0, 2Pi}, {y, 0, 2Pi},
  Axes -> False, FaceGrids -> All, PlotPoints -> 25]
```

In[7]:=
```
Plot3D[Cos[x] Sin[y], {x, 0, 2 Pi}, {y, 0, 2 Pi}, Axes → False, FaceGrids → All,
 PlotPoints → 25]
```

Out[7]=



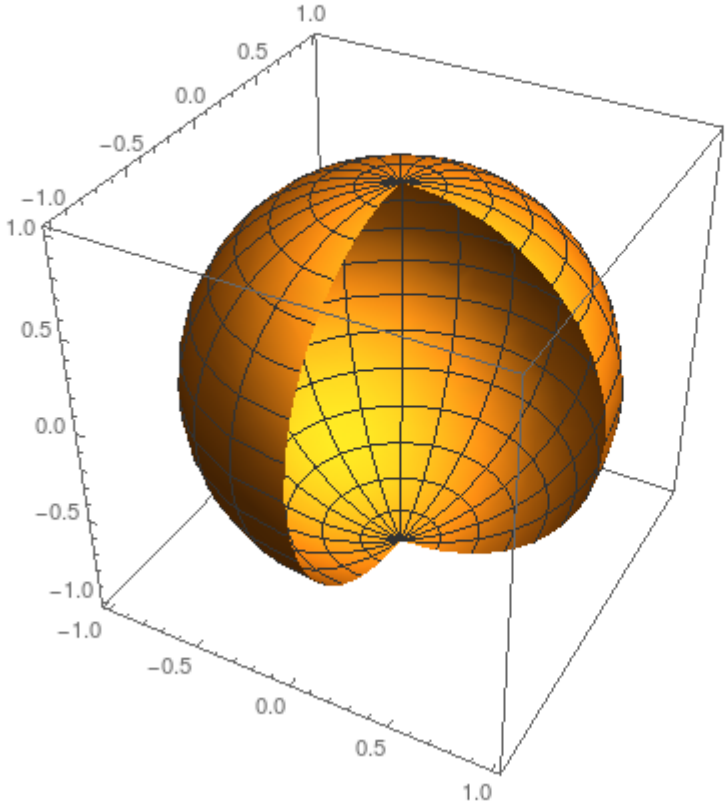
・3次元曲面

簡単な関数で表せない　次元曲面の作図をする場合、ParametricPlot3D[]関数を使用します

```
ParametricPlot3D[{Sin[v] Cos[u], Sin[v] Sin[u], Cos[v]},
  {u, 0, 3Pi/2}, {v, 0, Pi}]
```

In[8]:=
```
ParametricPlot3D[{Sin[v] Cos[u], Sin[v] Sin[u], Cos[v]}, {u, 0, 3 Pi / 2},
 {v, 0, Pi}]
```

Out[8]=



4. CUDALink の使用方法

運用終了

TSUBAME3 は既に運用を終了しています。 TSUBAME4 のマニュアル類はこちら

4.1. CUDALink について

Warning

現在mathematica/13.1.0でのみCUDALinkの動作を確認しております。

CUDALink を用いることで、Mathematicaにて GPU を利用できるようになります。
CUDALinkについての詳細な内容は、Mathematicaドキュメントセンターの以下のページをご参照ください。
CUDALink ユーザガイド
CUDALink
GPU計算

4.2. CUDALinkの読み込みと動作確認

```
GSICUSER@r0i0n0:~> module load cuda mathematica/13.1.0
GSICUSER@r0i0n0:~> math
Mathematica 11.2.0 Kernel for Linux x86 (64-bit)
Copyright 1988-2017 Wolfram Research, Inc.

In[1]:= Needs["CUDALink`"]
In[2]:= CUDAQ[]

Out[2]= True

In[3]:= CUDADot[Table[i, {i, 10}, {j, 10}],Table[i, {i, 10}, {j, 10}]]
Out[3]= {{55, 55, 55, 55, 55, 55, 55, 55, 55, 55}, {110, 110, 110, 110, 110, 110, 110, 110, 110, 110},

>    {165, 165, 165, 165, 165, 165, 165, 165, 165, 165}, {220, 220, 220, 220, 220, 220, 220, 220, 220, 220},

>    {275, 275, 275, 275, 275, 275, 275, 275, 275, 275}, {330, 330, 330, 330, 330, 330, 330, 330, 330, 330},

>    {385, 385, 385, 385, 385, 385, 385, 385, 385, 385}, {440, 440, 440, 440, 440, 440, 440, 440, 440, 440},

>    {495, 495, 495, 495, 495, 495, 495, 495, 495, 495}, {550, 550, 550, 550, 550, 550, 550, 550, 550, 550}}

In[4]:= lst = RandomReal[1., {10}];
In[5]:= CUDAFourier[lst]
Out[5]= {2.00511 + 0. I, 0.170165 + 0.153382 I, 0.313523 + 0.0756414 I, -0.0606903 + 0.209749 I, 0.178923 - 0.522008 I,

>    -0.200097 + 0. I, 0.178923 + 0.522008 I, -0.0606903 - 0.209749 I, 0.313523 - 0.0756414 I, 0.170165 - 0.153382 I}

In[6]:= Fourier[lst]
Out[6]= {2.00511 + 0. I, 0.170165 + 0.153382 I, 0.313523 + 0.0756414 I, -0.0606903 + 0.209749 I, 0.178923 - 0.522008 I,

>    -0.200097 + 0. I, 0.178923 + 0.522008 I, -0.0606903 - 0.209749 I, 0.313523 - 0.0756414 I, 0.170165 - 0.153382 I}
```

改訂履歴

改定日付	内容
2022/09/26	「4. CUDALink の使用方法」の修正
2019/08/01	mkdocs版作成
2018/03/14	「4.1CUDALink について」を追加
2017/09/25	初版