シミュレーション演習

G. 総合演習（*Mathematica*演習）

システム創成情報工学科

テキスト作成：藤尾 光彦
講義担当：尾下 真樹

本演習の目的

- さまざまな次元のデータ量を計算機で扱うための基本的な考え方を学習する
  - 1次元、2次元、3次元
  - 質点系、スカラ場、ベクトル場
  - 連続値、離散値
- Mathematica の基本的な使い方を学習する
  - Mathematica とは何か？
  - Mathematica を使ってデータ量を表現する
  - Mathematica を使ってデータ量を可視化する

前回の内容

- Mathematica の概要と使い方
  - Mathematicaの特徴
  - 数値解と解析解（無理数やπなどをそのまま扱える）
  - 記号計算（Σ、方程式の解、因数分解）
  - リスト操作、行列演算
- 各自、プリントの演習課題を行う
  - 時間内に課題を終えて提出

今回・次回の内容

- 計算機でのデータ量の表現
  - 講義（テキスト G1~G9）
- Mathematica を使ったデータ表現と表示
  - 講義＋演習（テキスト G9~G32）
- 各自、プリントの演習課題
  - 時間内に課題を終えて提出
- 講義資料
  - http://www.cg.ces.kyutech.ac.jp/lecture/sim/

前回の演習問題の解説

1．シグマの計算

(a) $1+2+3+4+....+100$ を計算せよ。
解答には Mathematica に入力したコマンドと出力結果の両方を記述すること

```
In[1]:= Sum[i, {i, 1, 100}]
Out[1]= 5050
```

(b) $1+2+3+4+....+n$ を表す関数 $S(n)$ を定義し、$S(50)$ を計算せよ。

```
In[2]:= S[n_] := Sum[i, {i, 1, n}]
In[3]:= S[50]
Out[3]= 1275
```

(b) $1+2+3+4+....+n=1128$ となるような n を求めよ。

```
In[4]:= Solve[S[x] == 1128, x]
Out[4]= {{x → -48}, {x → 47}}
```

２．数式の計算

(a) Mathematica の機能を使って、式 $x^2+2xy+x-y^2=0$ を y について解きなさい。

```
In[5]:= Solve[x^2+2 x*y+x-y^2 == 0, y]
Out[5]= {{y→ x-√x √1+2 x}, {y→ x+√x √1+2 x}}
```

(b) 上の式に、$x=4$ を代入したときの y の値を求めよ。

```
In[6]:= x = 4;
        Solve[x^2+2 x*y+x-y^2 == 0, y]
Out[7]= {{y→ -2}, {y→ 10}}
```

(b) 上の式に、$x=1$ を代入したときの y の実数値を小数点以下 20 桁まで求めよ。

```
In[10]:= NSolve[x^2+2 x*y+x-y^2 == 0, y, 20]
Out[10]= {{y→ -0.73205080756887729352}, {y→ 2.73205080756887729352}}
```
or
```
In[11]:= N[%%, 20]
```

３．因数分解

(a) $x^4+2x^3-13x^2-14x+24=0$ の素因数分解を計算せよ。

```
In[13]:= Factor[x^4+2 x^3-13 x^2-14 x+24]
Out[13]= (-3+x) (-1+x) (2+x) (4+x)
```

４．リストを使った行列演算

(a) 行列 $\mathbf{A}$ は 3×3 行列であり、の i 行 j 列要素 $\mathbf{A}_{ij}$ は、$\mathbf{A}_{ij}=i(-1)^{i+j}$ で表されるとする。このとき、Mathmatica の Table コマンドを使って、行列 A を作成せよ。

```
In[15]:= a = Table[i*(-1)^(i+j), {i, 1, 3}, {j, 1, 3}]
Out[15]= {{1, -1, 1}, {-2, 2, -2}, {3, -3, 3}}
```

(b) 行列 $\mathbf{B}=\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix}$ であるとき、同じく Table コマンドを使って、行列 B を作成せよ。

```
In[16]:= b = Table[i+j, {i, 0, 6, 3}, {j, 1, 3}]
Out[16]= {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}
```

(c) 2 つの行列の積 $\mathbf{AB}$ を計算せよ。

```
In[17]:= c = a.b
Out[17]= {{4, 5, 6}, {-8, -10, -12}, {12, 15, 18}}
```

(d) 上の(c)で求めた行列を C とおき、行列 C の転置行列 D を、Table コマンドを使い作成せよ。(ヒント： $\mathbf{C}_{ij}=\mathbf{D}_{ji}$)

```
In[18]:= d = Table[c[[j]][[i]], {i, 1, 3}, {j, 1, 3}]
Out[18]= {{4, -8, 12}, {5, -10, 15}, {6, -12, 18}}
```

データ量の扱い

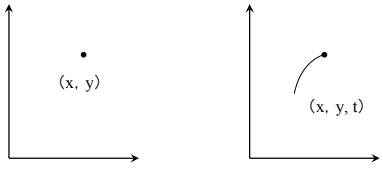
データ量を扱うための考え方

- 自由度
- 質点系と場（スカラ場、ベクトル場）
- 連続値、離散値

- Mathematicaでの多次元データの扱い

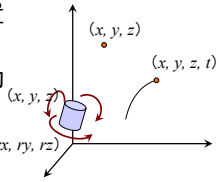
自由度

- 自由度（Degrees of Freedom: DOF）
  - 状態を表すために必要な数値の数
    - 例：2次元空間での質点の状態は（x,y）の2つの変数で表される → 2自由
      運動する質点は（x,y,t）の3自由度



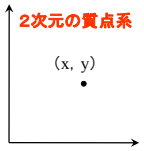
自由度

- 次元が上がると自由度も高くなる
  - 運動を扱うと、時間が加わり1自由度増える
- 3次元の質点系
  - 3次元空間での質点の位置
    → (x, y, z) 3自由度
  - 3次元空間での質点の運動
    →(x, y, z, t) 4自由度
  - 3次元空間での剛体の位置・向き
    → (x, y, z, rx, ry, rz) 6自由度

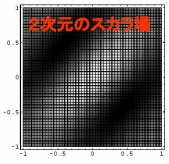


データ空間の種類

- 質点系
  - 空間内にある質点の状態



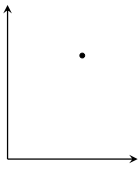
- 場（スカラ場、ベクトル場）
  - 空間内の各点が状態をもつ



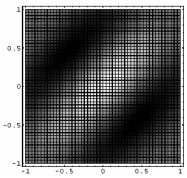
(x, y, s)

データ空間の種類

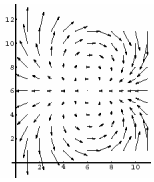
- 質点系 と 場（スカラ場、ベクトル場）



2次元の質点系
(x, y)



2次元のスカラ場
(x, y, s)

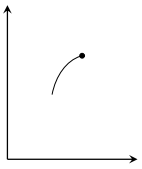


2次元のベクトル場
(x, y, vx, vy)

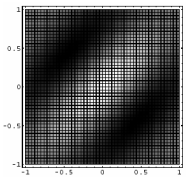
次元が上がれば自由度も増える

データ空間の種類

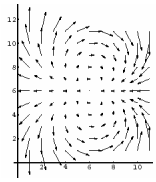
- 運動（時間変化）が加わると1次元増える



(x, y, t)



(x, y, s, t)

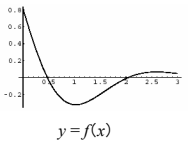


(x, y, vx, vy, t)

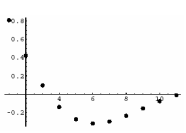
- 位置、スカラ値、ベクトル、時間など、すべてひっくるめて自由度として考えることができる

連続値と離散値

- 連続値
  - 関数によって与えられる連続的な値
    - データ量が何らかの数式により求められるとき

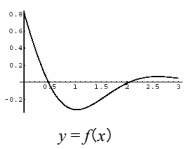


- 離散値
  - 離散的な計測点での値として与えられる値
    - 実際には連続的に変化した値であっても、数式などで表すことができないとき

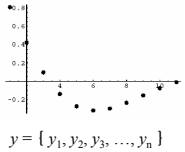


連続値と離散値

- Mathematicaでの連続値と離散値
- 連続値
  - 関数として定義できる
  - f [引数] でアクセス



- 離散値
  - リストとして定義できる
  - f [[データ番号]] でアクセス



連続値と離散値

- Mathematica では、連続値や離散値を同じように扱い、リストに格納できる

関数のリスト（リストを返す関数として扱われる）

$q[t\_]:=\{Q_1[t], Q_2[t], \cdots, Q_n[t]\}$

リストのリスト（2次元配列的に扱われる）

$q=\{\{Q_{1,1}, Q_{1,2}, \cdots, Q_{1,n}\}, \{Q_{2,1}, Q_{2,2}, \cdots, Q_{2,n}\}, \cdots\}$

– どちらも1次のリストを返す
– 記述に注意 q[?]（関数）と q[[?]]（リスト要素）

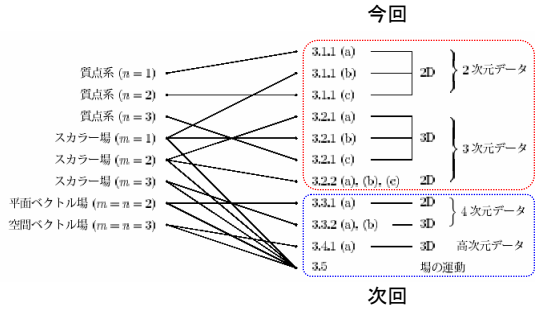
多次元データの扱い

- Mathematica では、リストや関数を組み合わせることで、多次元のデータを表現できる
  – 資料末尾の「関数とリストの混在」も参照
- Javaなどの一般的なプログラミング言語では、関数と変数は別のものなので、混在して扱うためには特別な工夫が必要
  – 「関数」はメソッドという形でしか定義できず、配列などに格納することもできない
  – C++では、[]関数のオーバーライドの機能を使えば、一部は実現できる

データ量の表現と表示

- Mathematicaで、さまざまな自由度のデータ量を扱うやり方を学ぶ
  – リストや関数を混在して扱えるので都合が良い
- データ量を扱うときの考え方は、他のプログラミング環境でも同様
- 今まで行ってきた演習のデータ量も今回学ぶ考え方でとらえなおすことができる
  → 総合演習

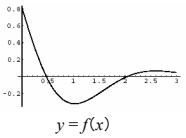
データ量の種類



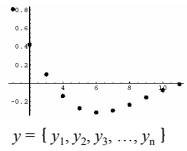
データ量の表現と表示

1次元の質点系の運動

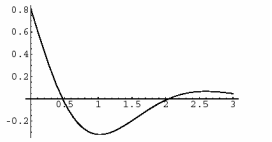
- 1次元の質点系の運動（2自由度）
- 連続値
- Plot [関数, 変数と範囲, オプション]



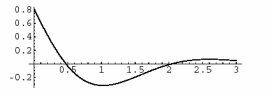
- 離散値
- ListPlot[リスト, オプション]



```
In[1]:= f[t_]:= Exp[-t]Cos[2 t + Pi/5]
In[2]:= Plot[f[t], {t, 0, 3}];
```

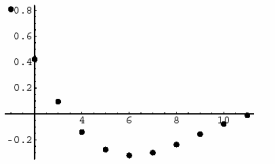


```
In[3]:= Plot[f[t], {t, 0, 3}, AspectRatio->Automatic];
```



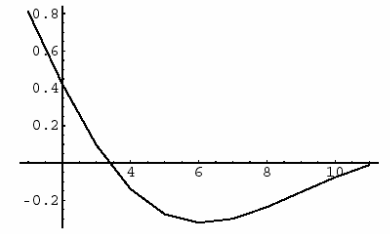
離散値の描画

```
In[4]:= data = Table[N[f[i]], {i, 0, 2, 0.2}]
Out[4]= {0.809017, 0.422678, 0.0951829, -0.139774, -0.274611, -0.320475,
         -0.299264, -0.23653, -0.156132, -0.0769278, -0.0113643}
In[5]:= g1 = ListPlot[data, PlotStyle->PointSize[0.02]];
```



折れ線で描画

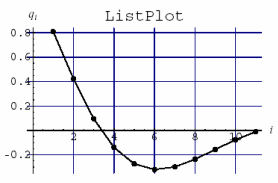
```
In[7]:= g2 = ListPlot[data, PlotJoined->True];
```



- Show[{複数のグラフィックス}, オプション]
  - グラフィックスを重ねて描画

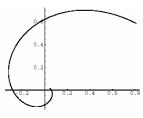
```
In[7]:= g3 = Show[{g1, g2}, GridLines->Automatic];
```

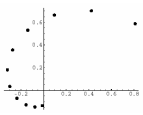
```
In[8]:= Show[g3,
        PlotLabel->StyleForm["ListPlot", FontSize->14],
        AxesLabel->
          {StyleForm["i", FontFamily->"Times", FontSlant->"Italic"],
           StyleForm["q_i", FontFamily->"Times", FontSlant->"Italic"]}]
```



2次元の質点系の運動

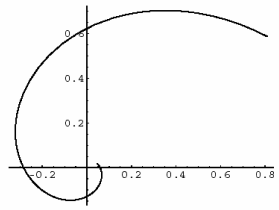
- 2次元の質点系の運動（3自由度）
- 連続値
- ParametricPlot［関数, 変数と範囲, オプション ］
- 離散値
- ListPlot[リスト, オプション ］
  - 1次元の運動と同じ





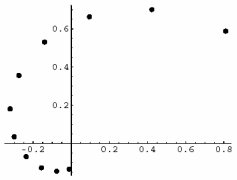
連続値の描画

```
In[1]:= c[t_]:= {Exp[-t]Cos[2 t + Pi/5], Exp[-t]Sin[2 t + Pi/5]}
In[2]:= ParametricPlot[c[t], {t, 0, 3}, AspectRatio->Automatic];
```



離散値の描画

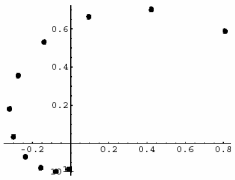
```
In[3]:= data = Table[c[i], {i, 0.0, 2.0, 0.2}]
Out[3]= {{0.809017, 0.587785}, ···, {-0.0113643, -0.134857}}
In[4]:= g1 = ListPlot[data, AspectRatio->Automatic, PlotStyle->PointSize[0.02]];
```



テキストの表示

- Text[表示文字, 表示座標, オフセット座標]

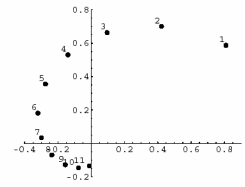
```
In[5]:= txt = Table[Text[i, data[[i]]], {i, 11}]
Out[5]= {Text[1, {0.809017, 0.587785}], ···, Text[11, {-0.0113643, -0.134857}]}
In[6]:= Show[g1, Graphics[txt]];
```



オフセットと描画範囲

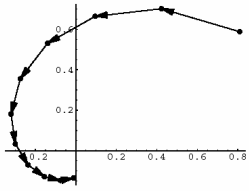
```
In[7]:= txt2 = Table[Text[i, data[[i]], {2, -2}], {i, 11}]
Out[7]= {Text[1, {0.809017, 0.587785}, {2, -2}],
                ··· Text[11, {-0.0113643, -0.134857}, {2, -2}]}
In[8]:= Show[g1, Graphics[txt2]];

In[9]:= Show[g1, Graphics[txt2], PlotRange->{{-0.4, 0.9}, {-0.2, 0.8}}];
```



矢印で描画

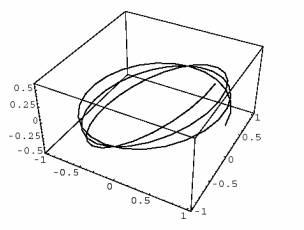
```
In[10]:= Needs["Graphics`Arrow`"]
In[11]:= arr = Table[Arrow[data[[i]], data[[i + 1]]], {i, 10}];
In[12]:= Show[{g1, Graphics[arr]}, PlotRange->{{-0.4, 0.9}, {-0.2, 0.8}}]
```



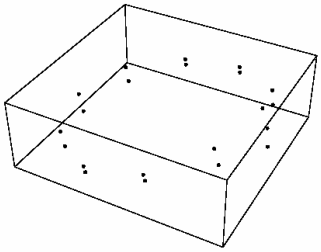
3次元の質点系の運動

- 2次元の質点系の運動（3自由度）
  - テキストG16
- 連続値
  - ParametricPlot3D［関数, 変数と範囲, オプション］
- 離散値
  - 点オブジェクトとして表示
  - Point［リスト, オプション］

```
In[1]:= c[t_]:= {Cos[.2t]Cos[2Pi t], Sin[2Pi t], Sin[.2t]Cos[2Pi t]}
In[2]:= ParametricPlot3D[c[t], {t, 0, 3}]
```

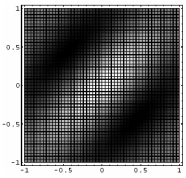


```
In[3]:= data = Table[c[i], {i, 0, 2, 0.1}];
In[4]:= pt = Map[Point, data];
In[5]:= g1 = Show[Graphics3D[pt]];
```



スカラ場

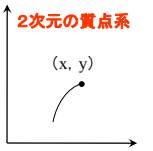
- 2次元のスカラ場
  - (x, y, s) 2次元空間の各点がスカラ値を持つ
  - 3自由度
- 2次元のスカラ場の変化
  - (x, y, s, t) 4自由度
  - アニメーションなどを使わない限り画面に表示できない
  - 来週扱う

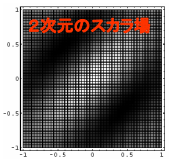


2次元のスカラ場
(x, y, s)

質点系とスカラ場の表現の違い

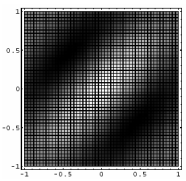
- 2次元の質点系の運動
  - (x, y, t)
    $$(x, y) = f(t)$$
  - (x,y) は関数の返す値になる
- 2次元のスカラ場の運動
  - (x, y, s, t)
    $$s = f(t, x, y)$$
  - (x,y) は関数の引数になる

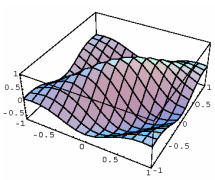




2次元のスカラ場の表示

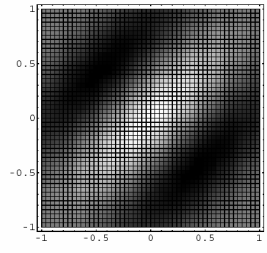
- 2次元空間の密度プロットとして表示
  - スカラ値を濃度(色)として表現
- 3次元空間の平面として表示
  - スカラ値を高さとして表現 (一部が隠れてしまう)



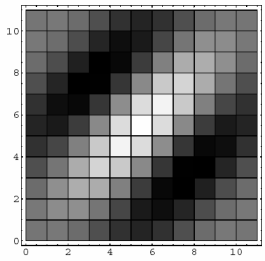


2次元空間の密度プロットとして表示

```
In[1]:= f[x_, y_]:= Exp[-(x^2 + y^2)]Cos[Pi(x - y)]
In[2]:= DensityPlot[f[x, y], {x, -1, 1}, {y, -1, 1}, PlotPoints->{50, 50}];
```

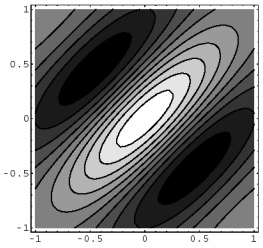


```
In[3]:= data = Table[f[j1, j2], {j1, -1, 1, 0.2}, {j2, -1, 1, 0.2}];
In[4]:= ListDensityPlot[data];
```

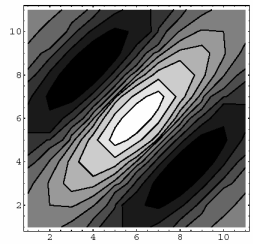


演習 **3.1** data が 11×11 次元の2次リストであることを確かめよ.

```
In[5]:= ContourPlot[f[x, y], {x, -1, 1}, {y, -1, 1}, PlotPoints->{50, 50}];
```

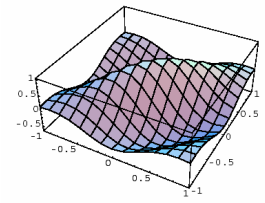


```
In[6]:= ListContourPlot[data];
```

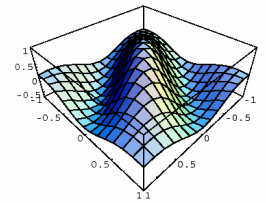


3次元空間の平面として描画

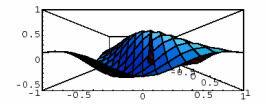
```
In[1]:= f[x_, y_]:= Exp[-(x^2 + y^2)]Cos[Pi(x - y)]
In[2]:= g1 = Plot3D[f[x, y], {x, -1, 1}, {y, -1, 1}, PlotRange->All];
```



```
In[3]:= Show[g1, ViewPoint->{1, 1, 1}];
```

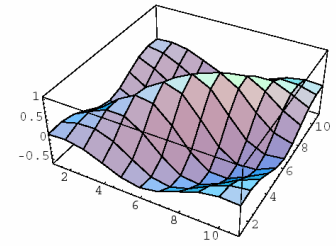


```
In[4]:= Show[g1, ViewPoint->{1, 0, 0}];
```



離散値の描画

```
In[6]:= data = Table[f[j1, j2], {j1, -1, 1, 0.2}, {j2, -1, 1, 0.2}];
In[7]:= ListPlot3D[data];
```



演習