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

前回までの内容

- 第1回 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
- 第2回 データ表現と表示
  - 自由度
  - 質点系（2次元、3次元）
  - スカラ場（2次元、3次元）

今回の内容

- 第3回 データ表現と表示
  - 講義＋演習（テキスト G20~G32）
    - スカラ場（3次元）
    - ベクトル場（2･3次元）
    - アニメーション（場の運動）
- プリントの演習課題を時間内に提出
- 講義アンケート（20分程度）
- 講義資料
  - http://www.cg.ces.kyutech.ac.jp/lecture/sim/

前回の演習問題の解説

自由度について

- 問題2：3次元の質点系の運動（x,y,z,t）

```
In[1]:= p[t_] := {Exp[-t] Cos[2 t * Pi / 5], Exp[-t] Sin[2 t * Pi / 5], -t / (t * t * t + 1) + 0.3}
```

- 問題4：2次元のスカラ場の運動（x,y,t,s）

```
In[1]:= ft = {-Exp[-x] Sin[x] * Sin[y], -Exp[-x] Sin[2 x] * Sin[2 y], -Exp[-x] Sin[3 x] * Sin[3 y]}
```

4．自由度
上の問題３・４について、次の文章の空欄に当てはまる数字を書きなさい。（ヒント：0 も有り得る）

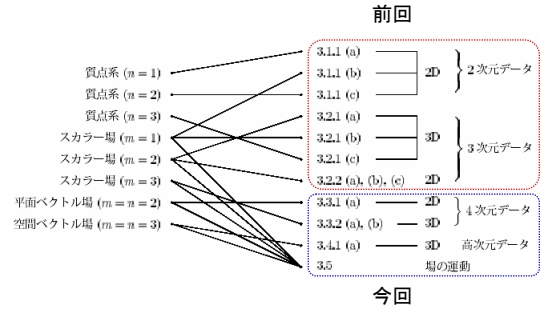
この問題の自由度は （ア） であり、そのうち連続データの数が （イ） 自由度、離散データの数は （ウ） 自由度である。この問題を Mathematica で表現すると、 （エ） 変数を引数とする関数の （オ） 次リストとして表すことができる。

問題2 （ア） 4 （イ） 4 （ウ） 0 （エ） 1 （オ） 0

問題3 （ア） 4 （イ） 3 （ウ） 1 （エ） 2 （オ） 1

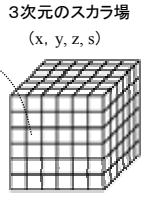
データ量の表現と表示

データ量の種類



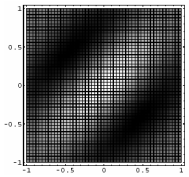
スカラ場

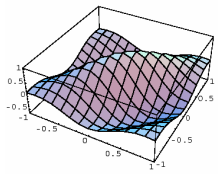
- 3次元のスカラ場
  - (x, y, z, s) 2次元空間の各点がスカラ値を持つ
  - $s = f(x, y, z)$
  - 4自由度
  - 画面に表示するのは困難
- 3次元のスカラ場の変化
  - (x, y, z, t, s) 5自由度
  - アニメーションなどを使わない限り画面に表示できない



2次元のスカラ場の表示(復習)

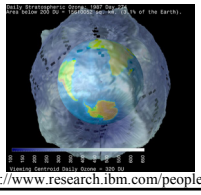
- 2次元空間の密度プロットとして表示
  - スカラ値を濃度(色)として表現
- 3次元空間の平面として表示
  - スカラ値を高さとして表現 (一部が隠れてしまう)



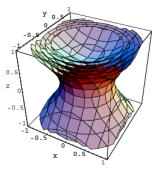


3次元のスカラ場の表示

- 3次元空間の密度プロットとして表示
  - 半透明で各点の濃度を重ねて描画
    - 複雑な場には対応できない、描画方法が複雑
- 3次元空間の等高面として表示
  - スカラ値が等しい点をつないで面として描画



http://www.research.ibm.com/people/l/lloydt/

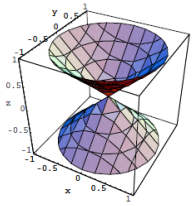


スカラ場の等高面の描画

```
In[1]:= Needs["Graphics`ContourPlot3D`"]

In[2]:= f[x_, y_, z_]:= x^2 + y^2 - z^2

In[3]:= ContourPlot3D[f[x, y, z], {x, -1, 1}, {y, -1, 1}, {z, -1, 1},
          Axes->True, AxesLabel->{x, y, z}];
```

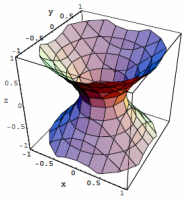


スカラ場の等高面の描画

- f = 0 以外の等高面の描画

ContourPlot3D では, デフォルトで関数の零点集合 ($f = 0$ の等高面) のみが表示される. 他のレベルの表示が欲しい場合には, オプション Contours を指定する.

```
In[4]:= ContourPlot3D[f[x, y, z], {x, -1, 1}, {y, -1, 1}, {z, -1, 1},
        Contours->{0.2}, Axes->True, AxesLabel->{x, y, z}];
```



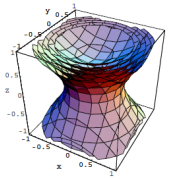
SKIP!!
時間がないので、
この入力は飛ばします

スカラ場の等高面の描画

- 複数の等高面を描画

2次元の ContourPlot の場合のように, 複数のレベル集合 (等高面) が欲しいときにも, Contours で指定できる.

```
In[5]:= ContourPlot3D[f[x, y, z], {x, -1, 1}, {y, -1, 1}, {z, -1, 1},
        Contours->{-0.4, -0.2, 0, 0.2, 0.4}, Axes->True,
        AxesLabel->{x, y, z}];
```

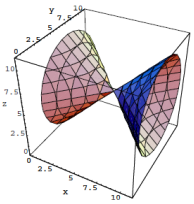


SKIP!!
時間がないので、
この入力は飛ばします

スカラ場の等高面の描画

- 離散データ

```
In[6]:= data = Table[f[j1, j2, j3], {j1, -1, 1, 0.2}, {j2, -1, 1, 0.2},
        {j3, -1, 1, 0.2}];
In[7]:= ListContourPlot3D[data, Axes->True, AxesLabel->{x, y, z}];
```

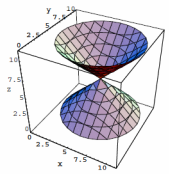


SKIP!!
時間がないので、
この入力は飛ばします

スカラ場の等高面の描画

連続空間の場合と異なった向きのグラフィックスが出てしまった. これは ListContourPlot3D では, 3 次リスト q[[j_1]][[j_2]][[j_3]] の添え字 j_1, j_2, j_3 を $j_1 \to z$, $j_2 \to y$, $j_3 \to x$ と解釈し, 高さ $z = j_1$ に関するレベル集合を描画することによる. 正しくは, 離散化を次のように行わなければならないのであった.

```
In[8]:= data = Table[f[j3, j2, j1], {j1, -1, 1, 0.2}, {j2, -1, 1, 0.2},
        {j3, -1, 1, 0.2}];
In[9]:= ListContourPlot3D[data, Axes->True, AxesLabel->{x, y, z}];
```



SKIP!!
時間がないので、
この入力は飛ばします

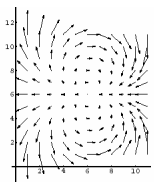
ベクトル場

- 2次元のベクトル場
  - (x, y, u, v) 4自由度
  - 2次元空間の各点(x, y)が2次元のベクトル値(u, v)を持つ

$$u = f(x, y)$$
$$v = g(x, y)$$

- 2次元のベクトル場の変化
  - (x, y, t, u, v) 5自由度

$$u = f(x, y, t) \quad v = g(x, y, t)$$



2次元のベクトル場
(x, y, u, v)

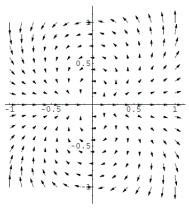
2次元のベクトル場

- 2次元のベクトル場(4自由度)
- 連続値
  - PlotVectorField [関数の1次リスト, 変数と範囲, オプション]
- 離散値
  - ListPlotVectorField[2次リスト, オプション]

2次元のベクトル場の描画

- 連続値

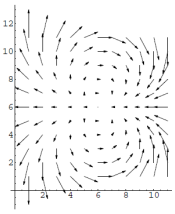
```
In[1]:= Needs["Graphics`PlotField`"]
In[2]:= u[x_, y_]:= x^2 - y^2; v[x_, y_]:= 2x y
In[3]:= PlotVectorField[{u[x, y], v[x, y]}, {x, -1, 1}, {y, -1, 1}, Axes->True];
```



2次元のベクトル場の描画

- 離散値

```
In[4]:= data1 = Table[{u[j1, j2], v[j1, j2]}, {j1, -1, 1, .2},
        {j2, -1, 1, .2}];
In[5]:= ListPlotVectorField[data1, Axes->True];
```

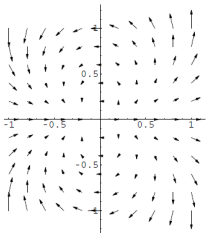


SKIP!!
時間がないので、
この入力は飛ばします

2次元のベクトル場の描画

- 離散値（目盛りを変更）

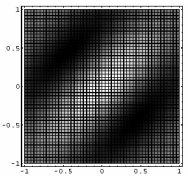
```
In[6]:= data2 = Table[{{j1, j2}, 0.1*{u[j1, j2], v[j1, j2]}},
        {j1, -1, 1, .2}, {j2, -1, 1, .2}];
In[7]:= ListPlotVectorField[Flatten[data2, 1], Axes->True];
```



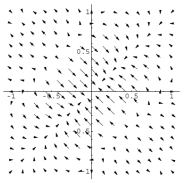
SKIP!!
時間がないので、
この入力は飛ばします

2次元の勾配場の描画

- スカラー場の勾配もまた、ベクトル場になる
  - 各点におけるスカラ値の変化量（周囲の点とのスカラ値の差）



2次元のスカラ場
(x, y, s)

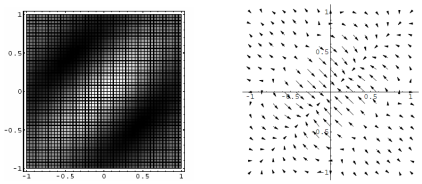


2次元のベクトル場
(x, y, u, v)

2次元の勾配場の描画

- スカラ場の関数から勾配ベクトルを計算

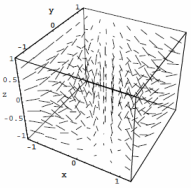
```
In[1]:= f[x_, y_]:= Exp[-(x^2 + y^2)]Cos[Pi(x - y)]
In[2]:= DensityPlot[f[x, y], {x, -1, 1}, {y, -1, 1}, PlotPoints->{50, 50}];
In[7]:= Needs["Graphics`PlotField`"]
In[8]:= PlotGradientField[f[x, y], {x, -1, 1}, {y, -1, 1}, Axes->True];
```



3次元の勾配場の描画

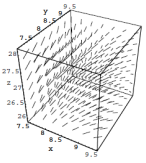
- スカラ場の関数を指定し、勾配ベクトルを描画

```
In[2]:= f[x_, y_, z_]:= x^2 + y^2 - z^2
In[10]:= Needs["Graphics`PlotField3D`"]
In[11]:= PlotGradientField3D[f[x, y, z], {x, -1, 1}, {y, -1, 1},
         {z, -1, 1}, Axes->True, AxesLabel->{x, y, z}];
```



3次元のベクトル場の描画（連続値）

```
In[1]:= u[x_, y_, z_]:= -10 (x - y)
In[2]:= v[x_, y_, z_]:= 24.7368 x - y - x z
In[3]:= w[x_, y_, z_]:= -2.6667 z + x y
In[4]:= Needs["Graphics`PlotFiled3D`"]
In[5]:= PlotVectorField3D[{u[x, y, z], v[x, y, z], w[x, y, z]},
          {x, 6 2^(1/2) - 1, 6 2^(1/2) + 1},
          {y, 6 2^(1/2) - 1, 6 2^(1/2) + 1},
          {z, 27 - 1, 27 + 1}, Axes->True, AxesLabel->{x, y, z}];
```



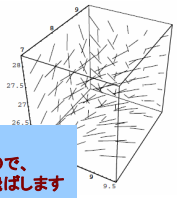
3次元のベクトル場の描画（離散値）

VectorField3D 関数は，入力として，成分が

{{j1, j2, j3}, {u[[j1, j2, j3]], v[[j1, j2, j3]], w[[j1, j2, j3]]}}

の 2 次リスト (2×3 次元) であるようなリスト，すなわち，$J \times 2 \times 3$ 次元の 3 次リストを要求する．ここで定義された離散データ data は $J_1 \times J_2 \times J_3 \times 2 \times 3$ 次元の 5 次リストである[11]．これを $J \times 2 \times 3$ 次元の 3 次リストに直すため，リストの上位 3 レベル ($J_1 \times J_2 \times J_3$ 次元の部分) をつぶして，1 つのレベルにする．今の場合，上位 3 つを 1 つにするので，2 回平坦化すればよい．これは Flatten[*list*, 2] でできる．

```
In[6]:= data = Table[{{j1, j2, j3},
          {u[j1, j2, j3]/40, v[j1, j2, j3]/80, w[j1, j2, j3]/80}}
          {j1, 6 2^(1/2) - 1, 6 2^(1/2) + 1, 0.5},
          {j2, 6 2^(1/2) - 1, 6 2^(1/2) + 1, 0.5},
          {j3, 27 - 1, 27 + 1, 0.5}];
In[7]:= ListPlotVectorField3D[Flatten[data, 2],
          Axes->True, AxesLabel->{x, y, z}];
```



SKIP!!
時間がないので、
この入力は飛ばします

場の運動の描画

- 場の運動（時間変化）の描画
  - ある時刻のスカラ場・ベクトル場は、これまでの方法で描画できる
  - 各時刻の場の描画の組み合わせで、変化を描画
- 主な描画方法

重ね描き 同じグラフィックスに，一連の状態を重ねて描く．

展開 時間軸を独立にとり，異なる時刻の状態を時間軸方向にずらして描く．

並列 一連の状態のグラフィックスを並べて表示する．

アニメーション 一連の状態のグラフィックスをアニメーションで表示する．

1次元のスカラ場の運動の例

スカラー場として

$$u(t,x) = \begin{cases} \delta(x) & (t=0) \\ \dfrac{1}{\sqrt{t}} \exp\left(-\dfrac{x^2}{2t}\right) & (t \neq 0) \end{cases}$$

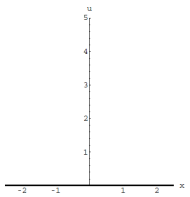
を考える[12]．この関数は「熱伝導方程式」と呼ばれる偏微分方程式の解であって，針金等の 1 次元空間において，最初に原点にインパルス的に与えられた熱に対し，温度分布が時間と共に空間全体に広がっていく様子を表す．このような，場合分けされた関数は *Mathematica* では

```
In[1]:= u[t_, x_]:= DiracDelta[x] /; t == 0
In[2]:= u[t_, x_]:= Exp[-x^2/(2t)]/(t)^(1/2) /; t != 0
```

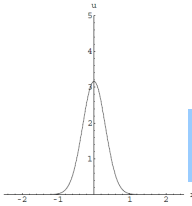
1次元のスカラ場の運動の例

$t = 0$ および $t = 0.1$ での状態をプロットしてみよう．

```
In[3]:= Plot[u[0, x], {x, -2.5, 2.5}, AxesLable->{x, u},
          PlotRange->{{-2.5, 2.5}, {0, 5}}, AspectRatio->Automatic,
          PlotStyle->Thickness[0.01]];
```



```
In[4]:= Plot[u[0.1, x], {x, -2.5, 2.5}, AxesLable->{x, u},
          PlotRange->{{-2.5, 2.5}, {0, 5}}, AspectRatio->Automatic];
```



SKIP!!
時間がないので、
この入力は飛ばします

1次元のスカラ場の運動の例

- 画像のリストを用意する

以後での運動の表示で用いる，一連の状態のグラフィックスをリストで準備する．

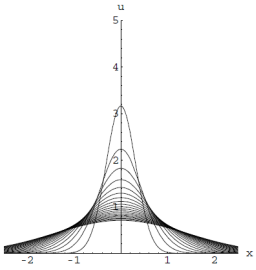
```
In[5]:= gr0 = Table[Plot[u[t, x], {x, -2.5, 2.5},
          Ticks->None, PlotRange->{{-2.5, 2.5}, {0, 5}},
          DisplayFunction->Identity], {t, 0, 2, 0.1}];
```

※ DisplayFunction->Identity は，データの生成段階ではグラフィックスを表示しないようにするためのオプション指定．

重ね描き

複数状態の重ね描きは Show が持っている機能をそのまま利用する.

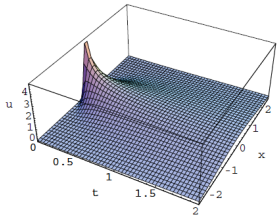
```
In[6]:= Show[gr0, DisplayFunction->$DisplayFunction,
        AspectRatio->Automatic, AxesLabel->{x, u}, Ticks->True];
```



展開

展開は 3 次元グラフィックスを利用し, グラフィックス中に時間軸を取り込んで, 時間方向への発展を連続的に表示する.

```
In[7]:= Plot3D[u[t, x], {t, 0, 2}, {x, -2.5, 2.5},
        PlotPoints->{50, 50}, PlotRange->All, AxesLabel->{t, x, u}];
```



並列

Mathematica では一連のグラフィックスを並べて表示する機能がある. グラフィックスの配列を GraphicsArray 関数で 1 個のグラフィックスオブジェクトに変換し, Show で表示する. 重ね描きでやった要領で, グラフィックスオブジェクトの配列を作り (このままでは 21 枚のグラフィックスなので), GraphicsArray で 1 個のグラフィックスオブジェクトに変換してから, Show で表示する.

```
In[8]:= Show[GraphicsArray[gr0]];
```

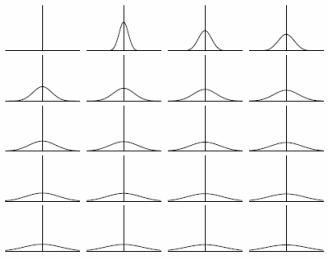

通常は, 一列にたくさん並べても見にくい. このような場合, リストを間引いて表示するか, GraphicsArray の 2 次リストとして, 並列する方法がある.

```
In[9]:= Show[GraphicsArray[Take[gr0, {1, 20, 5}]]];
```

SKIP!!
時間がないので、
この入力は飛ばします

並列

```
In[10]:= Show[GraphicsArray[Table[Take[gr0, {4(j-1) + 1, 4j}], {j, 5}]]];
```



Take[list, {a, b}]
リストから指定範囲
のデータを取り出す

アニメーション

Mathematica では, グラフィックスのリストはそのまま, ShowAnimation でアニメーション表示される. 従って, 必要な手順は途中まで並列と同じである. 但し, この関数を呼び出すためには "Graphics`Animation`" パッケージを事前に読み込んでおく必要がある.

```
In[11]:= Needs["Graphics`Animation`"]

In[12]:= ShowAnimation[gr0, AxesLabel->{x, u}]
```

※このような出力に不満な者は, ShowAnimation の出力セル全体を選択し, Ctr + Y を実行してみよ.

場の運動の描画のまとめ

- 主な描画方法

重ね描き 同じグラフィックスに, 一連の状態を重ねて描く.

展開 時間軸を独立にとり, 異なる時刻の状態を時間軸方向にずらして描く.

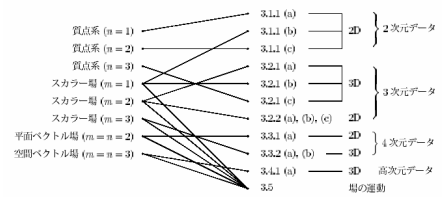
並列 一連の状態のグラフィックスを並べて表示する.

アニメーション 一連の状態のグラフィックスをアニメーションで表示する.

- 可視化したい情報や場の種類に応じて、適切な描画方法を選択することが必要
  - おおまかな場の変化を確認したいか、各時刻の場の様子を詳しく確認したいか、など

データ量のまとめ

- 自由度
- 質点系、スカラ場、ベクトル場
- 連続データ、離散データ



演習

授業評価アンケート

- アンケート
  - 様式1：選択式
  - 様式2：自由記述式

演習問題の解説

まとめ

- 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
  - 自由度
  - 質点系、スカラ場、ベクトル場
  - 連続データ、離散データ
- 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