パラメータ推定解説

数理モデルに対する パラメータ推定

長山 雅晴
北海道大学 電子科学研究所
附属社会創造数学研究センター 人間数理研究分野

松永哲
北海道大学 理学院 数学専攻

RIES
Hokkaido University
Research Institute
for
Electronic Science

JST MATH CREST
since 2010 Nagayama

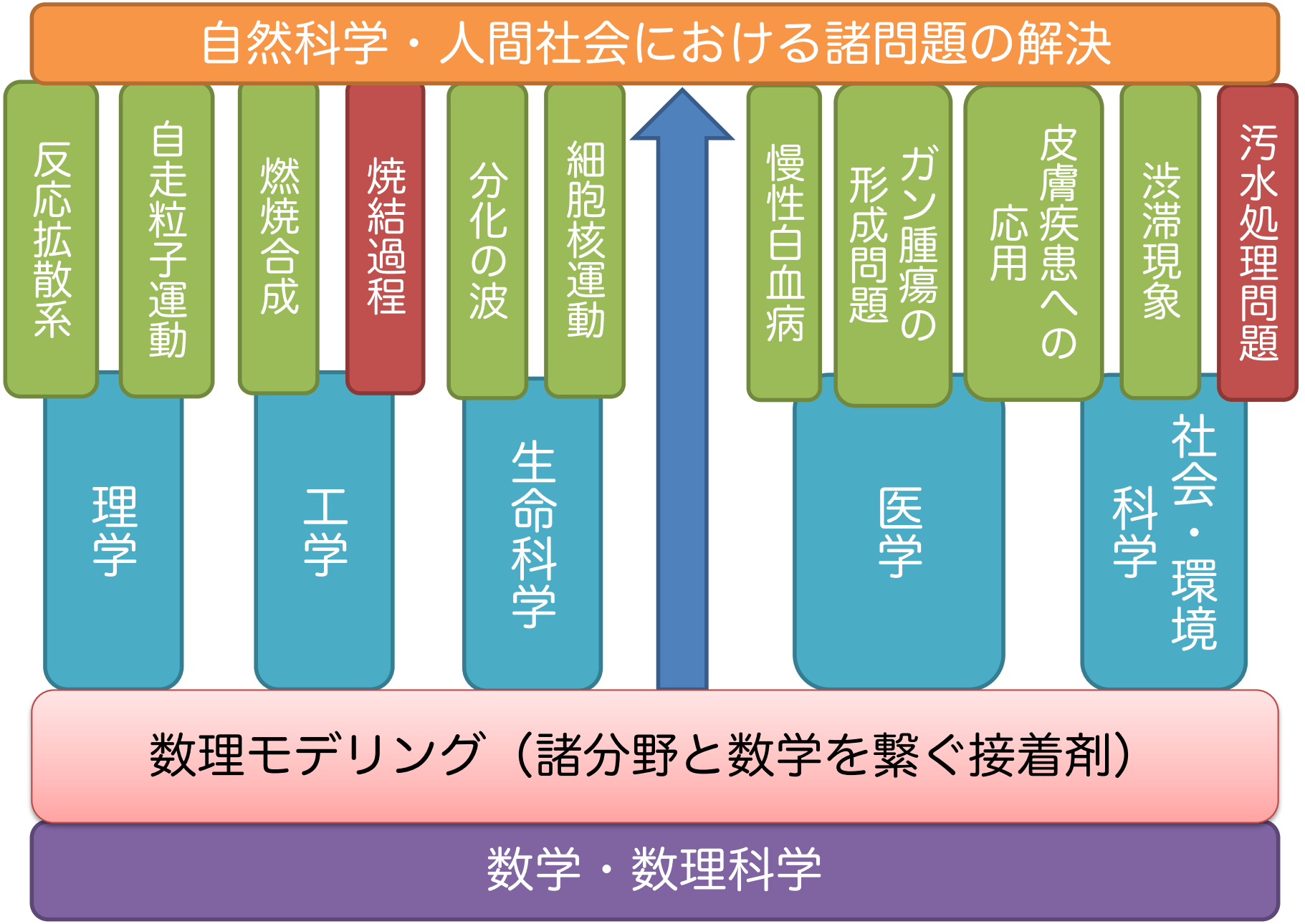
CREST

自己紹介

所属：北海道大学電子科学研究所
　　附属社会創造数学研究センター 人間数理研究分野
専門：応用数学
キーワード：
現象を数式であらわす数理モデリングとその数値シミュレーション
化学反応や自然界に現れるパターン形成の数理解析
数理モデルの解析をおこなうための数値分岐計算

1988年：徳島大学・総合科学部，卒業
1992年：広島大学・理学研究科数学専攻，博士前期課程，修了
1994年：東京大学・数理科学研究科，博士課程後期，修了
1998年：龍谷大学・ハイテクリサーチセンター，博士研究員
1999年：京都大学・数理解析研究所，助手
2004年：金沢大学・大学院自然科学研究科，助教授
2009年：金沢大学・理工研究域数物科学域，教授
2012年：現在

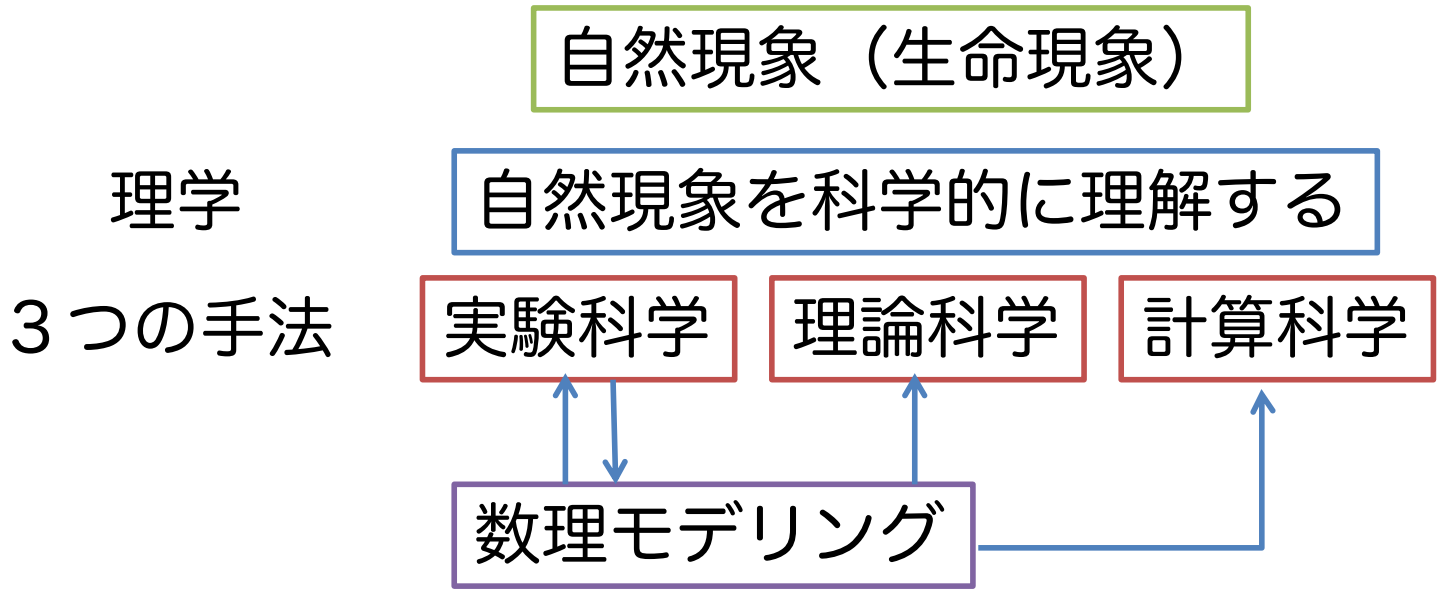
数理モデリング


自然科学・人間社会における諸問題の解決
反応拡散系
自走粒子運動
燃焼合成
焼結過程
分化の波
細胞核運動
慢性白血病
ガン腫瘍の形成問題
皮膚疾患への応用
渋滞現象
汚水処理問題
理学
工学
生命科学
医学
社会・環境科学
数理モデリング（諸分野と数学を繋ぐ接着剤）
数学・数理科学

講義の予定

1. 数理モデリングとは何か？
2. 数理モデリングの評価
    1. 定性的な評価
    2. 定量的な評価
3. パラメータ推定（1点推定）基礎編
    1. 解説
    2. 演習
4. ベイズ推定を用いたパラメータ推定
    1. 解説
    2. 演習

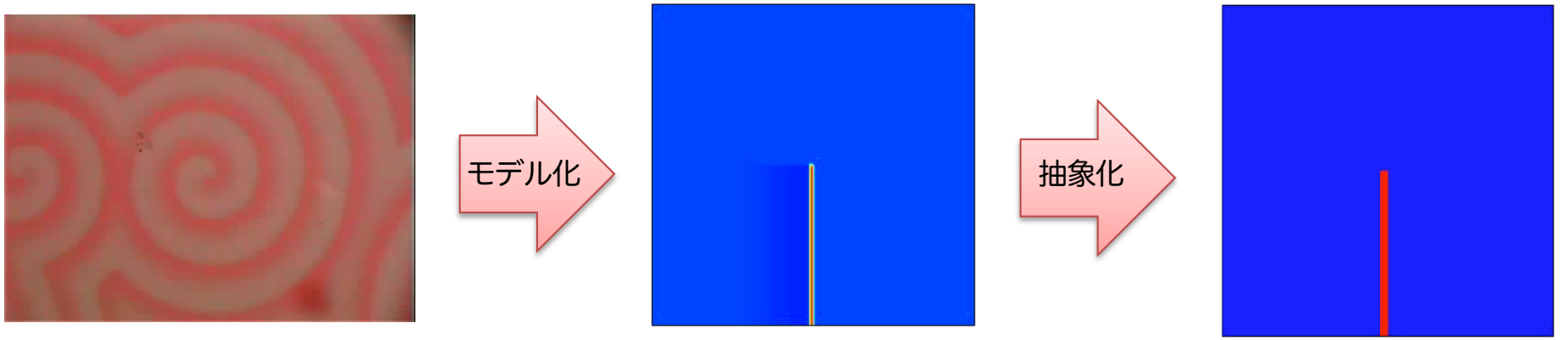
数理モデリングとは何か？



数理モデリングは理論的な理解を与えるための道具である.
現象の定性的な再現は必要条件である.
理論的方法によって未知の部分が理解できなければ意味がない.
数理モデルに正解はない（40点～80点程度）
理解できれば，実験に対する予測ができる.
よい数理モデルができれば，数学の問題へと定式化ができる.

数理モデルとは何か？

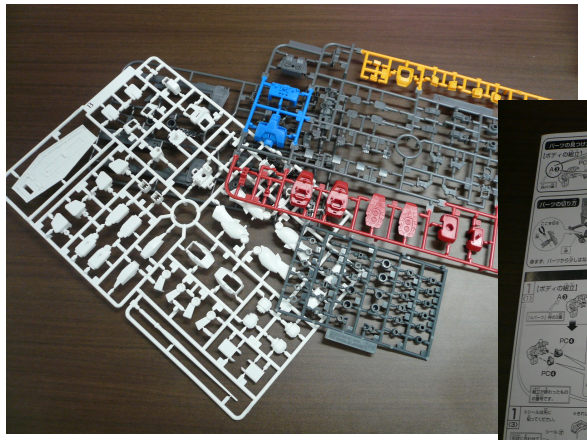
数理モデルとは現象の数式による模倣と抽象化である.
抽象化し現象を模倣することによって，「現象の法則」を明らかにする道具

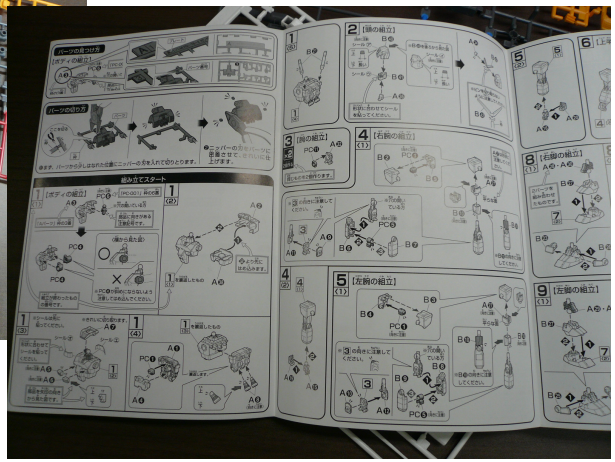


BZ反応（現象）　　モデルから得られた結果
法則の発見　　抽象化したモデル

どのようなモデルを考えればよいか？（立場によるが）
現象の定性的性質を保持してる程度まで抽象化した数理モデルを考えたい.
そのような数理モデルは現象の法則を記述してハズ（信念！）
数理モデルは現象を数学で取り扱えるようにする道具ともいえる.

数理モデルはどのように作るか？





$\log(a_n)\, c_{n+1}$ $\sin(a_n)$ $\cos(b_n)$ $b_n c_n$ a_{n+1} $f(a_n)$ a_n $a_n c_n$ $g(a_n, b_n)$ $h(a_n, b_n)$ b_{n+1} b_n $\log(b_n)$

数理モデル（漸化式，微分方程式）

$$\begin{cases} a_{n+1} - a_n = f(a_n, b_n), \\ \quad - b_n = g(a_n, b_n). \end{cases}$$

$$= 0, 1, 2, \cdots$$

数理モデルは
どのように作るの？

私とっての数理モデルを作る基本原理

重要

単位時間当たりの（量，もの）の変化量
＝　流入量 － 流出量
＝　入ってくる（もの）－　出て行く（もの）
＝　生成されるもの － 消費されるもの

(もの) ＝ 化学物質濃度，密度，温度，力　etc.

数理モデルはこの基本原理によって記述されていると思ってよい．
「流入するもの」と「流出するもの」をいかにして見つけるのかが重要！
いいかえると
「“ルール”をいかにして見つけ出すか」がポイント

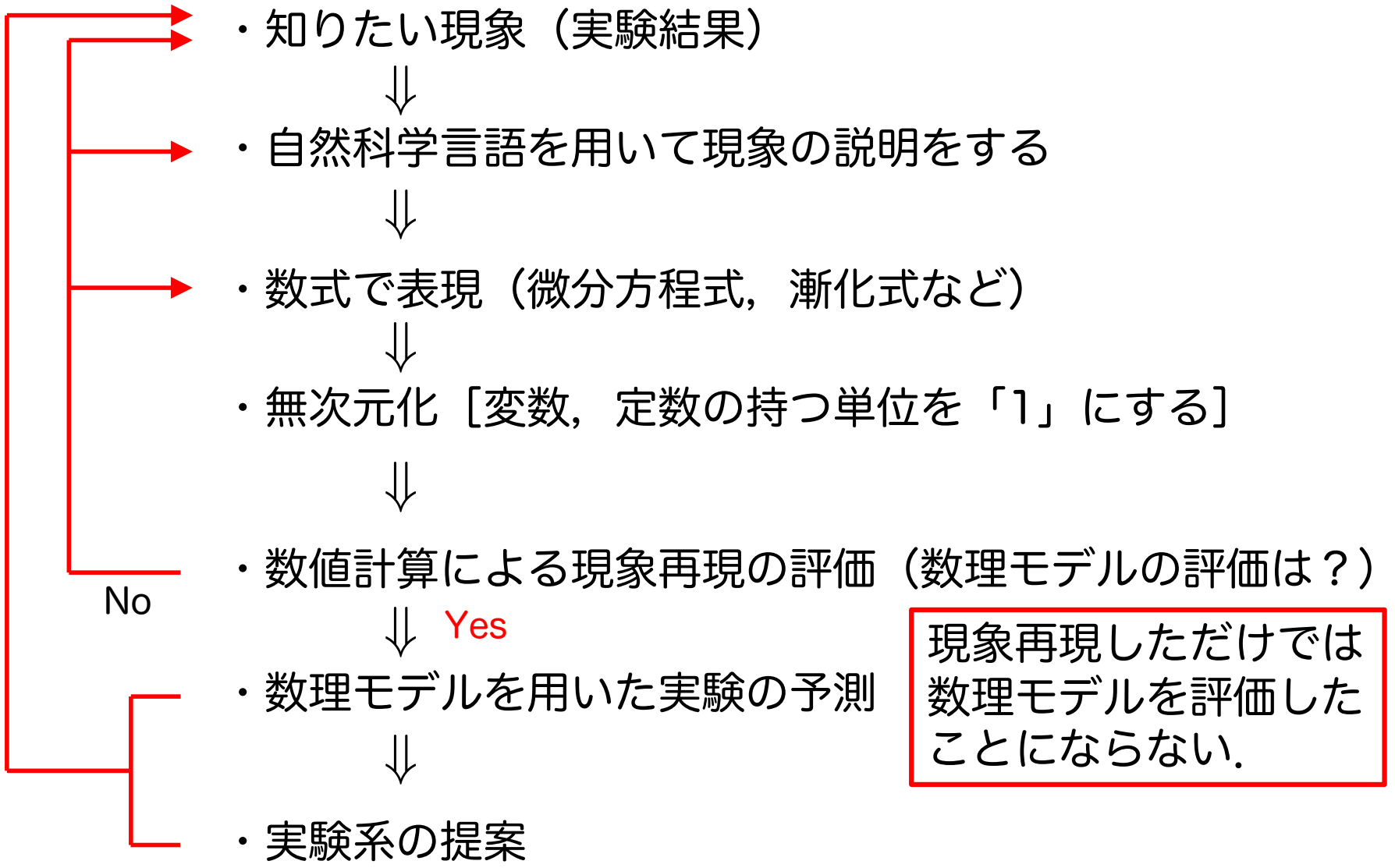
数理モデリングに必要な道具

力学系の基礎知識
　常微分方程式系の平衡解の線形化安定性解析
　相図
　簡単な分岐現象

数値計算法の基礎（水藤先生，斎藤先生）
　常微分方程式の数値計算法
　熱方程式の数値計算法
　（できれば）反応拡散系の数値計算

現象に応じて，物理学，化学，生物学，その他

数理モデルの構成


・知りたい現象（実験結果）
⇓
・自然科学言語を用いて現象の説明をする
⇓
・数式で表現（微分方程式，漸化式など）
⇓
・無次元化［変数，定数の持つ単位を「1」にする］
⇓
・数値計算による現象再現の評価（数理モデルの評価は？）
No
⇓ Yes
・数理モデルを用いた実験の予測
⇓
・実験系の提案
現象再現しただけでは
数理モデルを評価した
ことにならない．

数理モデルの評価とは

現象の数理モデリングを評価する方法は？

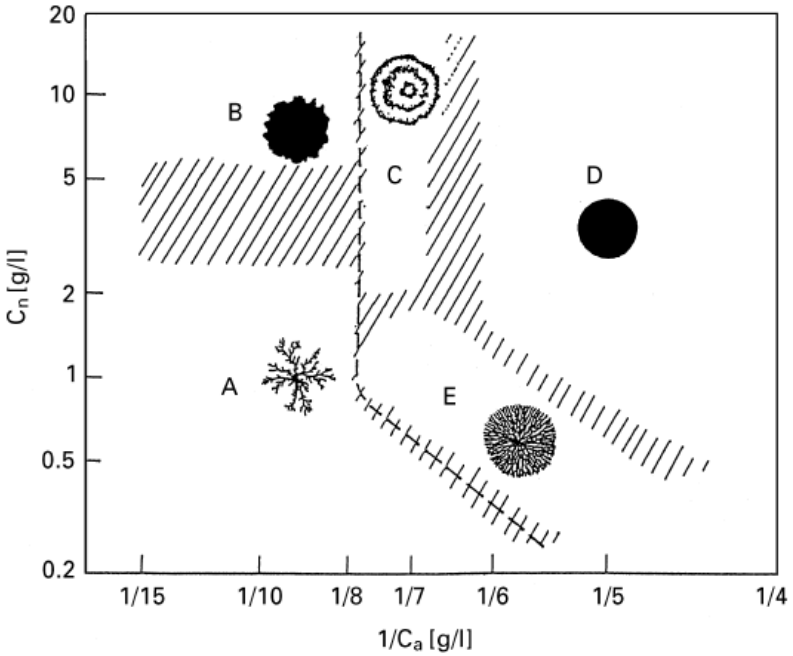
現象を再現しているだけでは不十分
（似て非なる数理モデルを作っているだけの可能性あり）

現象を矛盾なく説明できているならば新しい現象を予測できるのではないか？
モデルからの予測と実験によるその再現性から数理モデルを評価できないか？

定量性の観点から数理モデルを評価することはできないか？
定量性はどのようにあたえることができるのか？
定量的再現性の視点から数理モデルを評価してみたい…

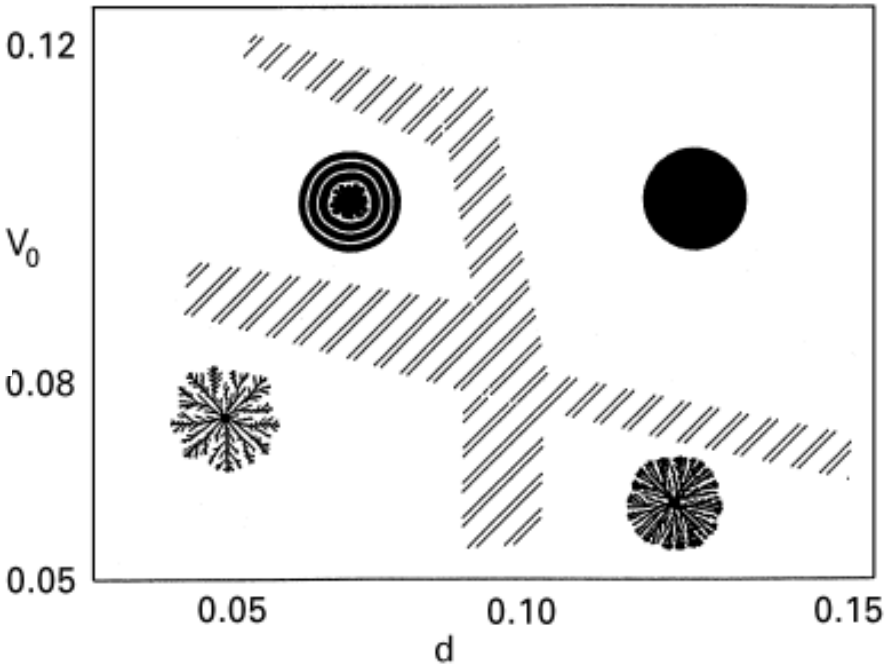
似て非なるもの（リングパターン）

生物実験

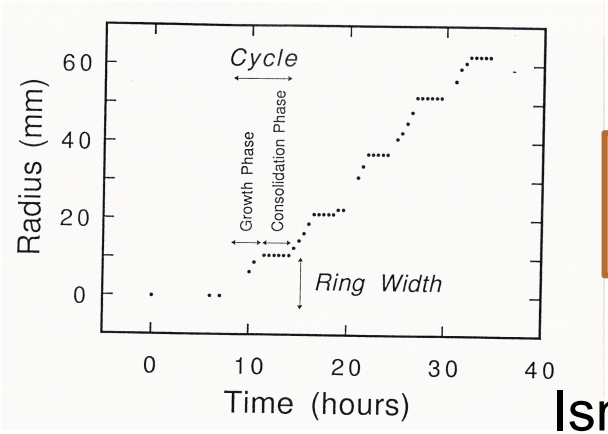


Wakita et al., 1994.

数値実験

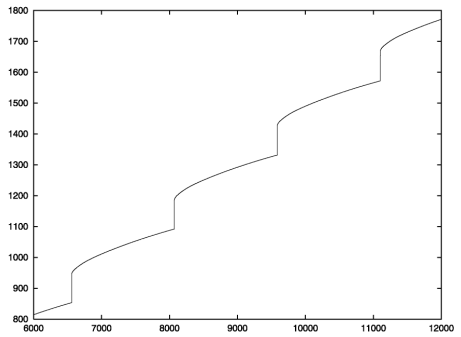


Mimura et al., 2000.



Ismael Ráfols

リングパターンの
成長様式が異なる

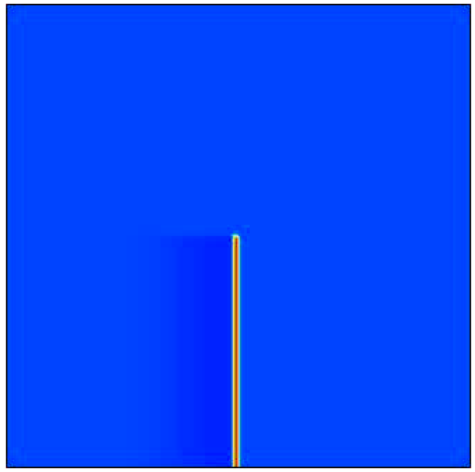


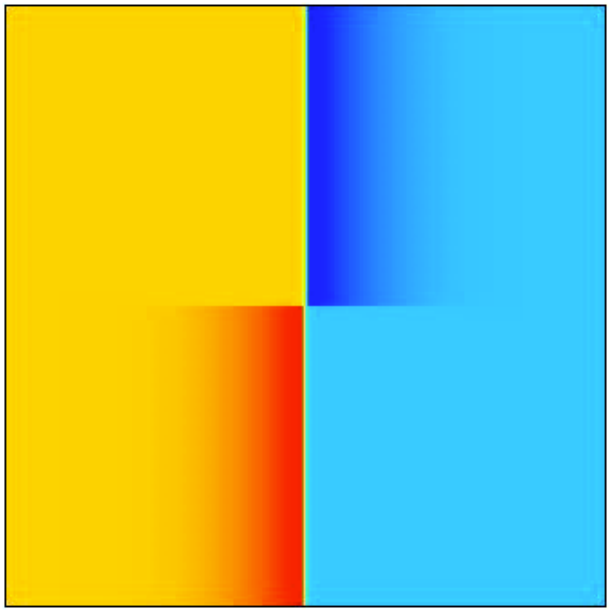
似て非なるもの（スパイラルパターン）

単安定反応拡散系モデル

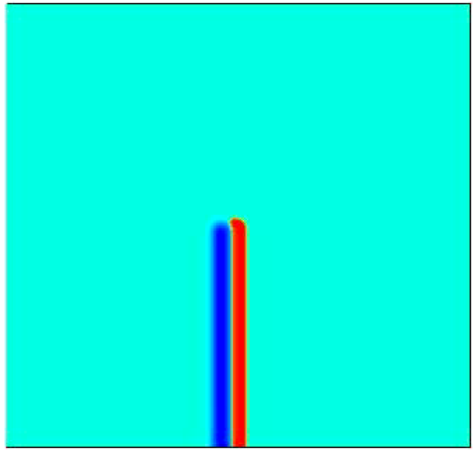
双安定反応拡散系モデル

BZ反応モデル



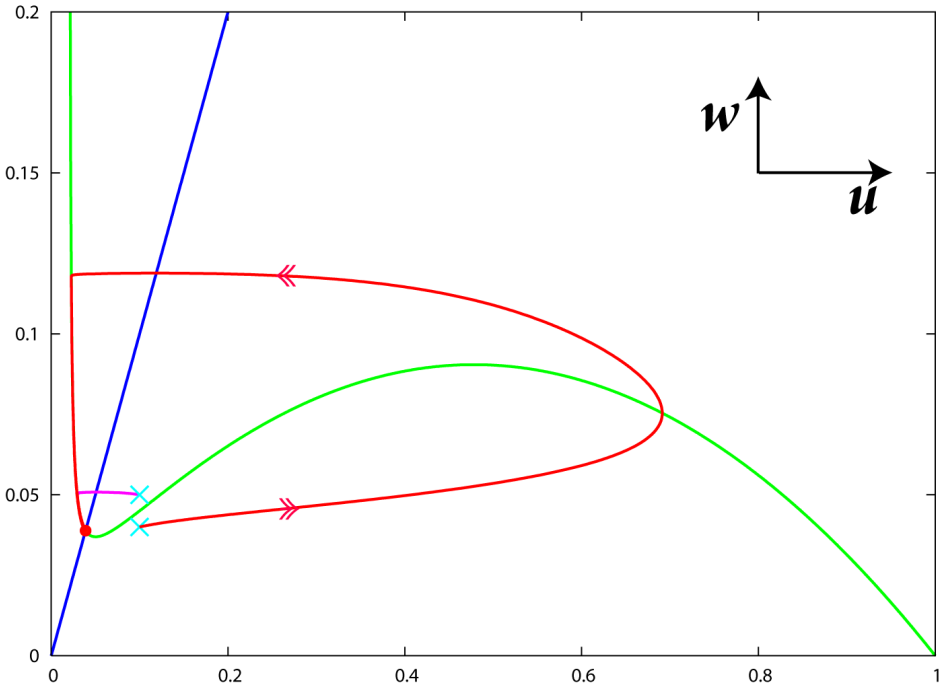


燃焼反応モデル

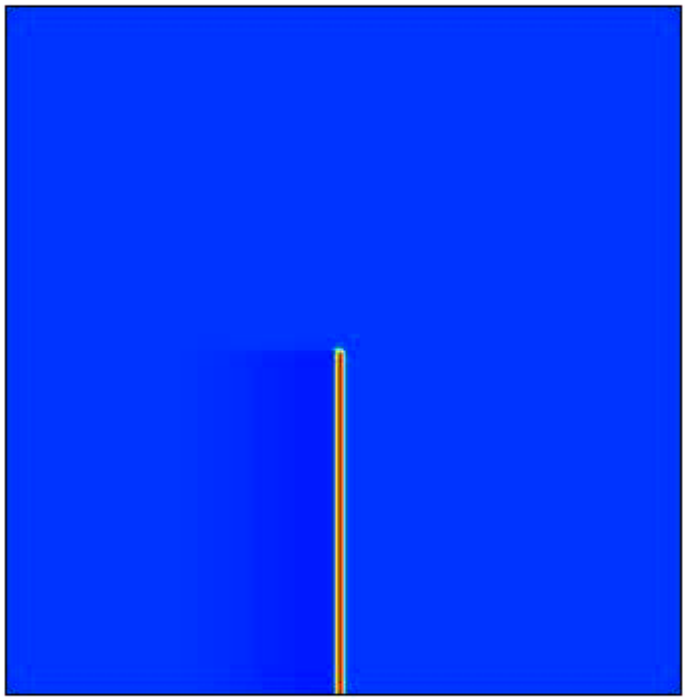


BZ反応モデル

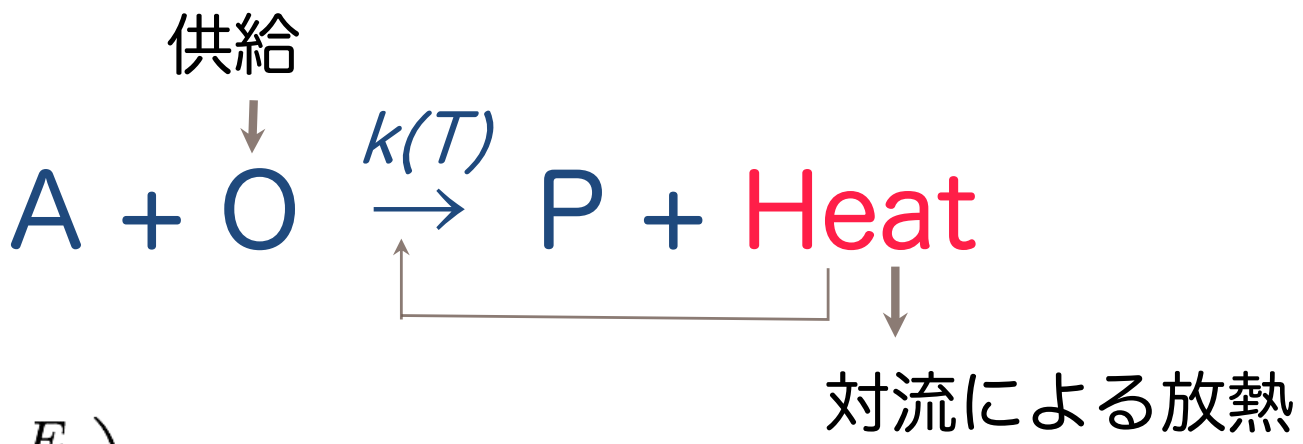
$$\begin{cases} \dfrac{\partial u}{\partial t} = D_u \Delta u + \dfrac{1}{\varepsilon}\left(f v \dfrac{a-u}{a+u} + u(1-u) \right), \\ \dfrac{\partial v}{\partial t} = D_v \Delta v + u - v. \end{cases}$$



単安定興奮系



発熱反応拡散系

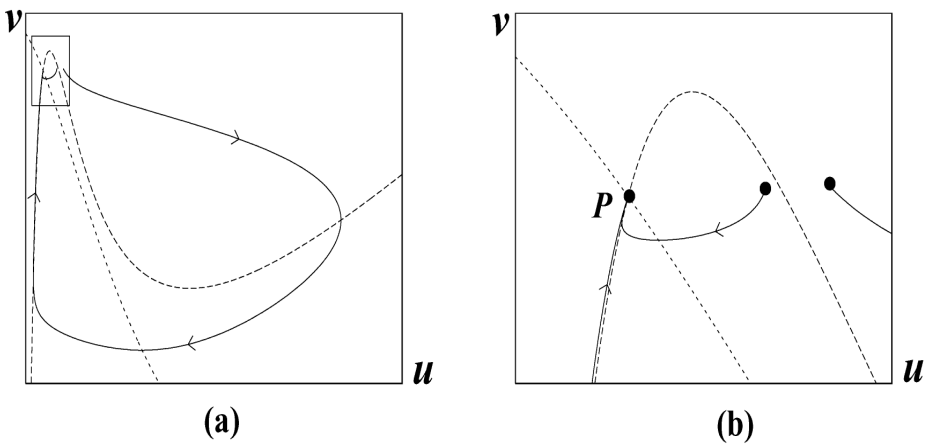


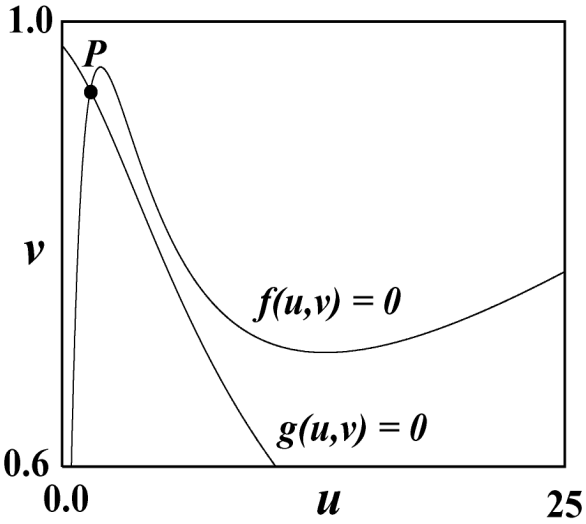
$$k(T) = \exp\left(-\frac{E}{RT}\right) : \text{Arrhenius reaction rate}$$

$$\begin{cases} \dfrac{\partial u}{\partial t} - u_{xx} = \dfrac{1}{\varepsilon}\left(-au + v\exp\left(\dfrac{u}{1+u/c}\right)\right) \equiv \dfrac{1}{\varepsilon} f(u,v) \\ \dfrac{\partial v}{\partial t} - dv_{xx} = h(1-v) - v\exp\left(\dfrac{u}{1+u/c}\right) \equiv g(u,v) \end{cases}$$

発熱反応モデルのダイナミクス

$$\begin{cases} \dfrac{du}{dt} = \dfrac{1}{\varepsilon} f(u,v) \\ \dfrac{dv}{dt} = g(u,v) \end{cases}$$

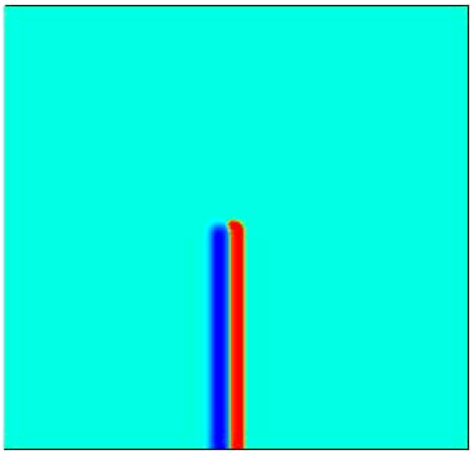




$$\boldsymbol{u}_0 (= P) = (u_0, v_0)$$

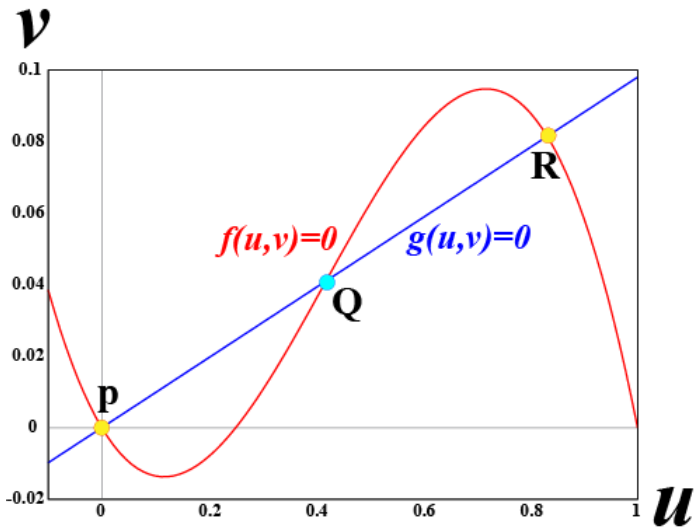
Parameter : a=2.0, c=5.0, h=45.0, ε =0.001

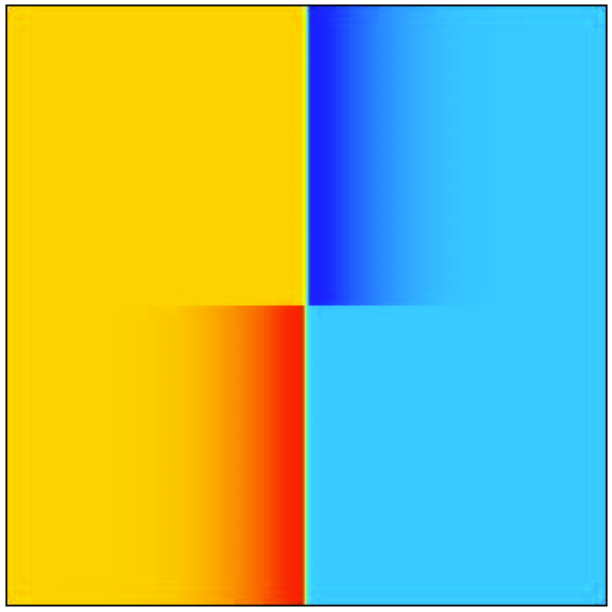
単安定興奮系



双安定系反应拡散方程式

$$\begin{cases} \dfrac{\partial u}{\partial t} = d_u \Delta u + \dfrac{1}{\varepsilon}\left(u(1-u)(u-a) - v\right), \\ \dfrac{\partial v}{\partial t} = d_v \Delta v + u - \gamma v \end{cases}$$



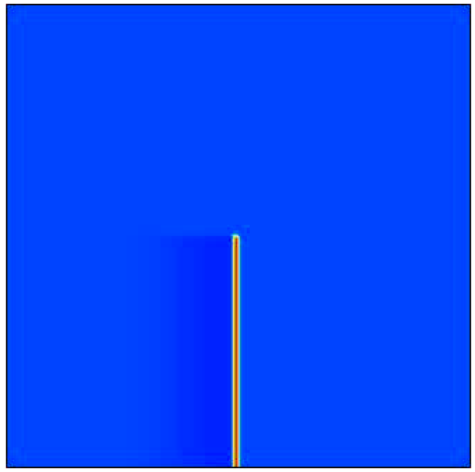


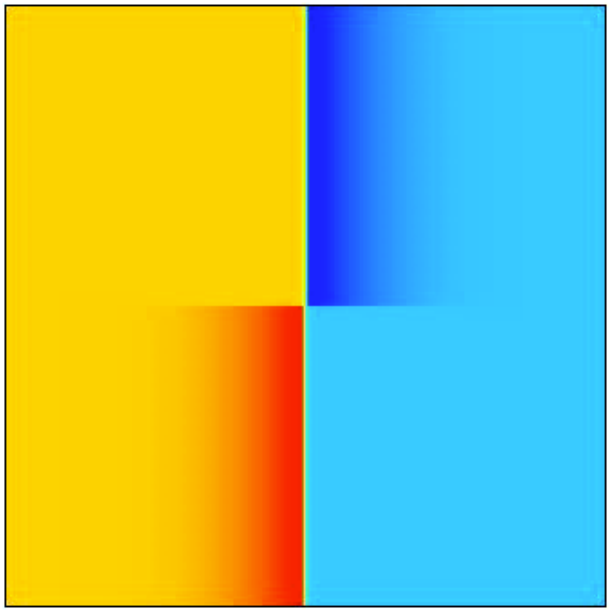
似て非なるもの（スパイラルパターン）

単安定反応拡散系モデル

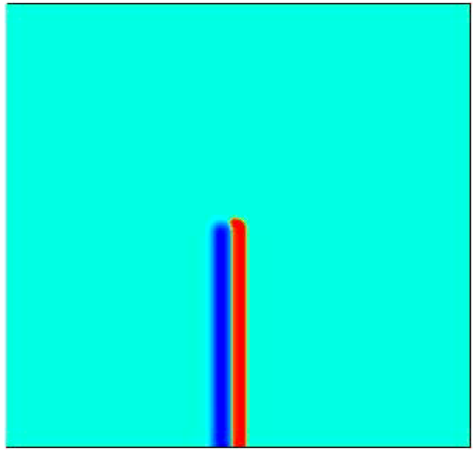
双安定反応拡散系モデル

BZ反応モデル





燃焼反応モデル



パターンが定性的に似ているからといって
メカニズムが同じであるという保証は全くない

数理モデルの評価とは

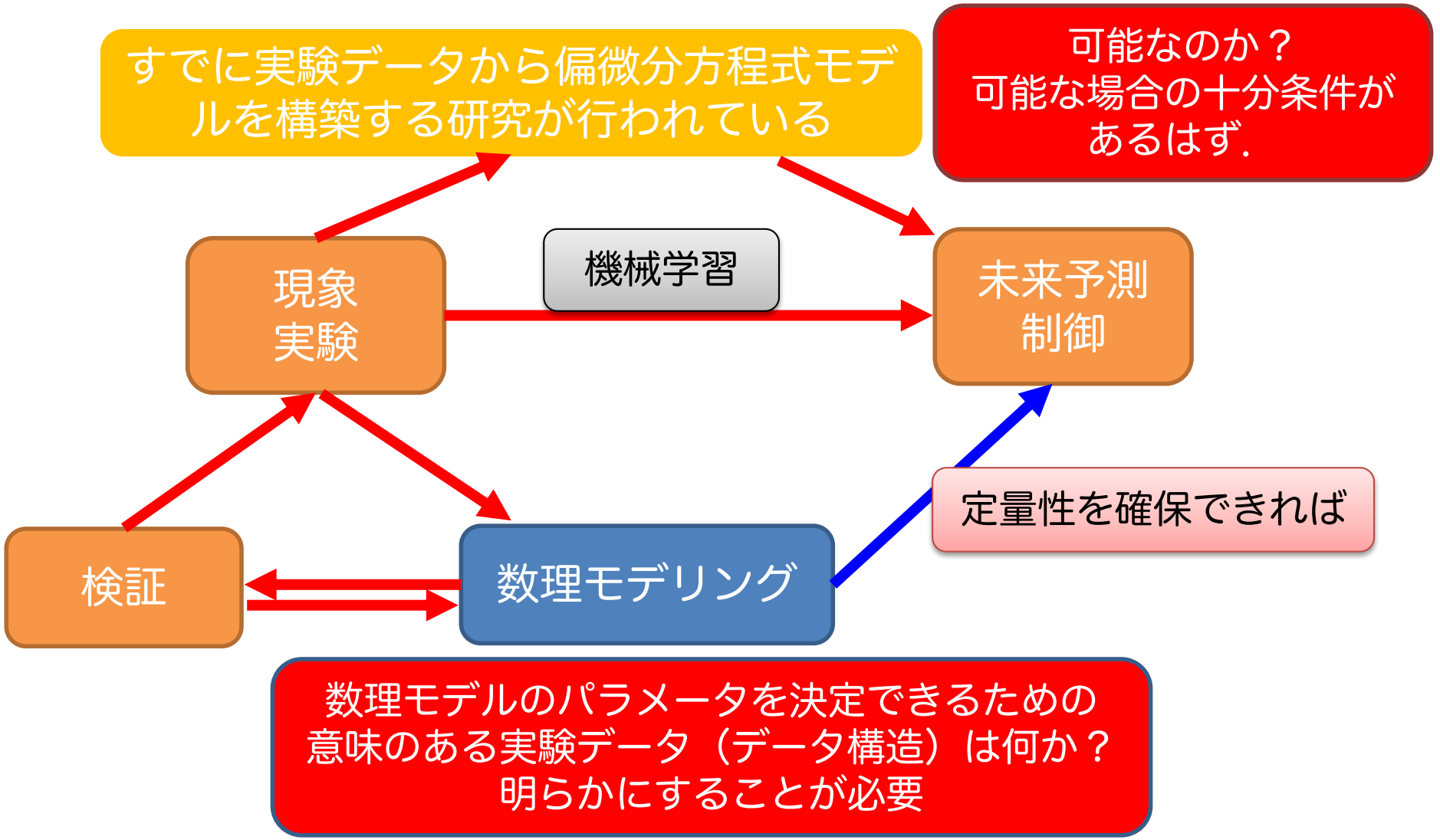
現象の数理モデリングを評価する方法は？

現象を再現しているだけでは不十分
（似て非なる数理モデルを作っているだけの可能性あり）

現象を矛盾なく説明できているならば新しい現象を予測できるのではないか？
モデルからの予測と実験によるその再現性から数理モデルを評価できないか？

定量性の観点から数理モデルを評価することはできないか？
定量性はどのようにあたえることができるのか？
定量的再現性の視点から数理モデルを評価してみたい…

定量的再現性のある数理モデリングを目指して



定量性のある数理モデリング

実験データが与えられていると仮定する.
このときのある数理モデリングを行うためには2つの問題がある

（1）数理モデルは現象を記述しているのか？

（2）数理モデルはデータを再現しているが，定量性はあるのか？

(1) に関しては現象の数理モデリングの定性的評価を行う必要がある.

ここでは (2) について評価方法を考えてみる.
(2) についての問題点：与えられたデータから
(2.1) モデル方程式のパラメータを見つけることができるのか？
(2.2) モデル方程式のパラメータを決定できるのか？
←どのようなデータならばパラメータを決定できるのか？

(2.2) は谷口（やぐち）さん（神戸大学）のCRESTで研究されています.

問題設定（パラメータ推定）

数理モデリングは職人芸なので対象としない.
（本当は数理モデリングの話もしたいが…）
定性的に評価された数理モデルを仮定する.
すなわち，モデル方程式

$$\begin{cases} \dfrac{d\boldsymbol{u}}{dt} = \boldsymbol{f}(\boldsymbol{u};\boldsymbol{a}), \\ \boldsymbol{u}(t) = \boldsymbol{u}_0. \end{cases}$$

に対して， 初期値を込めたパラメータセット $\boldsymbol{\theta}_s = (\boldsymbol{u}_s, \boldsymbol{a}_s)$ を与えて
数値計算から数値データを求め，この数値データ（にノイズを加えたデータ）

$$\boldsymbol{Y}(t_i), \quad (i = 1, 2, \cdots, N)$$

を与えられたデータとする.

問題

このとき，数値データ $\boldsymbol{Y}(t_i)$ を近似するパラメータセット $\boldsymbol{\theta} = (\boldsymbol{u}, \boldsymbol{a})$ を
モデル方程式から求めることができるか？（推定できるか？）

最適化問題として定式化

モデル方程式

$$\begin{cases} \dfrac{d\boldsymbol{u}}{dt} = \boldsymbol{f}(\boldsymbol{u};\boldsymbol{a}), \\ \boldsymbol{u}(t) = \boldsymbol{u}_0. \end{cases}$$

に対して

パラメータ $\boldsymbol{\theta} = (\boldsymbol{u}_0, \boldsymbol{a})$ を与えて求まる数値解を $\boldsymbol{\Phi}(t_i;\boldsymbol{\theta})$ とする

このとき，次の目的関数を与える：

$$f(\boldsymbol{\theta}) = \sum_{i=1}^{N} ||\boldsymbol{\Phi}(t_i;\boldsymbol{\theta}) - \boldsymbol{Y}(t_i)||_2^2 + \alpha||\boldsymbol{\theta}||_2^2$$

この目的関数の最小化から正しくパラメータを推定することができるか？

$$||\boldsymbol{x}||_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

パラメータ推定方法　勾配法(1)

$\boldsymbol{\theta}$ をパラメータとして，次式を最小化する $\boldsymbol{\theta}$ を求める

目的関数 $f(\boldsymbol{\theta}) = ||\boldsymbol{\Phi}(\boldsymbol{\theta}) - \boldsymbol{Y}||_2^2 + a||\boldsymbol{\theta}||_2^2$

数値計算結果 $\boldsymbol{\Phi}(\boldsymbol{\theta})$ と実験値 $\boldsymbol{Y}$ の差の2乗ノルム

観測誤差の過剰学習(オーバーフィッティング)を
防ぐ役割(a ：正則化パラメータ)

パラメータ推定方法　勾配法(2)

適当なパラメータの初期点 $\boldsymbol{\theta}_0$ から出発して目的関数 f の値を

下げるように点列 $\{\boldsymbol{\theta}_k\}$ を生成し，最適解に収束させる(点推定).

パラメータ $\boldsymbol{\theta}_k$ から次のパラメータ $\boldsymbol{\theta}_{k+1}$ に以下のように更新する

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha_k \boldsymbol{d}_k$$

ステップ幅　　探索方向

探索方向として勾配 $\boldsymbol{d}_k = -\nabla f(\boldsymbol{\theta}_k)$ を用いる方法を最急降下法と呼ぶ.

➡ 局所最適解に陥りやすい（欠点）

最急降下法

1. $\boldsymbol{\theta}_0$ を初期値として与える．k = 0とする

2. 次の三つの条件を同時に満たした場合に停止する

$$|f(\boldsymbol{\theta}_k) - f(\boldsymbol{\theta}_{k-1})| < 10^{-16}(1 + |f(\boldsymbol{\theta}_k)|)$$
$$||\nabla f(\boldsymbol{\theta}_k)|| < 10^{-8}(1 + |f(\boldsymbol{\theta}_k)|)$$
$$||\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}|| < 10^{-8}(1 + ||\boldsymbol{\theta}_k||)$$

3. 探索方向を $\boldsymbol{d}_k = -\nabla f(\boldsymbol{\theta}_k)$ と定める

4. $\phi(\alpha) = f(\boldsymbol{\theta}_k + \alpha \boldsymbol{d}_k)$ に対する直線探索により α_k を決定

5. $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha_k \boldsymbol{d}_k$ と更新

6. k = k+1としてステップ2に戻る

金森敬文, “機械学習のための連続最適化” (講談社, 2016)

αの選択方法

方針1　まじめに1次元の最適化問題を解く

→　$\boldsymbol{\theta}$ を固定すると　$f(\boldsymbol{\theta}_{k-1} - \alpha \nabla f)$　はαに関する1次元の最適化問題になるのでこれを真面目に解く

- 二分法
- 黄金分割探索

方針2　大体でいいや

→減少すればいいので，大枠で評価してしまう

- Wolf の基準
- etc.

参考文献：

金森敬文，鈴木大慈，竹内一郎，佐藤一誠著，
機械学習のための連続最適化，講談社

直線探索

α に関する1変数関数

$$\phi(\alpha) = f(\boldsymbol{\theta}_k + \alpha \boldsymbol{d}_k) \quad (\alpha > 0)$$

を考える．一変数関数の最適化問題を考える

$$\phi(0) = f(\boldsymbol{\theta}_k)$$

よりも目的関数の値が小さい

$$\phi(0) > \phi(\alpha)$$

となる α　を上手く選ぶ

二分法のアルゴリズム

1. $\alpha_0 = 0$ とし，　$\phi'(\alpha_1) > 0$ となる α_1 を定める．$k = 2$ とする．

2. $\alpha_k = \dfrac{\alpha_{k-2} + \alpha_{k-1}}{2}$　とする

3. 停止条件が満たされるならば，α_k を数値解として出力して停止

4. $\phi'(\alpha_k) < 0 \rightarrow \alpha_k = \alpha_{k-2}$

   $\phi'(\alpha_k) > 0 \rightarrow \alpha_k = \alpha_{k-1}$

   $\phi'(\alpha_k) = 0 \rightarrow \alpha_k$　を数値解として出力して停止

5. $k = k + 1$ として，ステップ2に戻る

サンプルプログラムのダウンロード

パラメータ推定のサンプルプログラムは以下からダウンロードしてください

http://mmc01.es.hokudai.ac.jp/els/link

数理モデルに対するパラメータ推定の解説はこちらをご覧下さい.
サンプルプログラムはこちらからダウンロードしてください.

サンプルプログラム

```
#ロジスティック方程式　最急降下法python
import numpy as np
import matplotlib.pyplot as plt
import sys
import random
#微分方程式の右辺
def f(x,par):
        return par[1]*(par[2]-x)*x

#Runge-Kutta法
def runge(x,par):
        k1 = dt*f(x,par)
        k2 = dt*f(x+0.5*k1,par)
        k3 = dt*f(x+0.5*k2,par)
        k4 = dt*f(x+k3,par)
        x += (k1+2*k2+2*k3+k4)/6
        return x
#コスト関数
def costfunc(par,y,st):
        xc=[]
        x=par[0]
        for i in range(N):
                x=runge(x,par)
                #時刻情報stと同じiのループの時にシミュレーションデータとして保存
                if i in st:
                        xc.append(x)
        s=0.0
        for i in range(Nz):
                s += (xc[i]-y[i])**2
        return(s+r*sum([x**2 for x in par]))
```

サンプルプログラム

```
#コスト関数の勾配を計算
def grad(par,y,st):
        par_grad=[]
        for m in range(len(par)):
                par_plus=[par[m]+0.01*par[m] if i==m else par[i] for i in range(len(par))]
                par_minus=[par[m]-0.01*par[m] if i==m else par[i] for i in range(len(par))]
                par_grad.append((costfunc(par_plus,y,st)-costfunc(par_minus,y,st))/(2*par[m]/100.0))
        return par_grad
```

サンプルプログラム

```
#二分法(直線探索)
def bisection(par,par_grad,y,st):
##初期設定
        s=0.0      #二分法左端初期値
        l=0.01      #二分法右端初期値
        count=0
        g_plus=[par[i]-(l+0.01*l)*par_grad[i] for i in range(len(par))]
        g_minus=[par[i]-(l-0.01*l)*par_grad[i] for i in range(len(par))]
        phiplus=costfunc(g_plus,y,st)
        phiminus=costfunc(g_minus,y,st)
        #計算可能かつ現在のコスト関数より小さいlになるまでlを小さくする
        while np.isinf(phiplus) or costfunc(par,y,st)<phiminus :
                l=l/2.0
                g_plus=[par[i]-(l+0.01*l)*par_grad[i] for i in range(len(par))]
                g_minus=[par[i]-(l-0.01*l)*par_grad[i] for i in range(len(par))]
                phiplus=costfunc(g_plus,y,st)
                phiminus=costfunc(g_minus,y,st)
                count += 1
                if count>100:
                        print("ERROR_bisection")
                        sys.exit()
        #右側の微分が負になるまでlを右にずらしていく
        while (phiplus-phiminus)<0.0:
                old_l=l
                l=1.1*l
                g_plus=[par[i]-(l+0.01*l)*par_grad[i] for i in range(len(par))]
                g_minus=[par[i]-(l-0.01*l)*par_grad[i] for i in range(len(par))]
                phiplus=costfunc(g_plus,y,st)
                phiminus=costfunc(g_minus,y,st)
```

サンプルプログラム

```
                #計算が発散したら一つ前のlを出力
                if np.isinf(phiplus) or np.isinf(phiminus):
                        print("ERROR_l=%.15e" %l)
                        return old_l
while True:
                p=(s+l)/2.0      #二分法の右端と左端の真ん中の点をpとする
                g_plus=[par[i]-(p+0.01*p)*par_grad[i] for i in range(len(par))]
                g_minus=[par[i]-(p-0.01*p)*par_grad[i] for i in range(len(par))]
                phiplus=costfunc(g_plus,y,st)
                phiminus=costfunc(g_minus,y,st)
                if (phiplus-phiminus)<0.0:      #pでの傾きが負だったらpを左端にする
                        s=p
                elif (phiplus-phiminus)>0.0:      #pでの傾きが正だったらpを右端にする
                        l=p
                elif (phiplus-phiminus)==0.0:
                        l = s = p
                else:
                        print("ERROR!!")
                        sys.exit()
                if(abs(l-s)<1e-8):          #区間が十分小さくなったら終了
                        p=(s+l)/2.0
                        return p

print("ERROR! bisection¥n")
sys.exit()
```

サンプルメイン関数

```
##########################################
###メイン文開始###
##########################################
#変更しないパラメータ
dt=0.01  #刻み幅
N=500    #ルンゲクッタ試行回数
r=0.0000001    #正則化パラメータ

#自分で実験データを作る場合 st,yを使用
#####################################
st=[25,100,250,470]  #データ点の時刻情報
Nz=len(st)                  #データ点の個数取得
par_ans=[0.3,1.0,2.0] #正解パラメータ，左からx0,a,b
#正解の解軌道
t=0.0
tpoints=np.arange(0.0,(N+1)*dt,dt)
xpoints=[]
x=par_ans[0]
y=[]
xa=[] #ノイズ用
for i in range(N+1):
                xpoints.append(x)
                x=runge(x,par_ans)
                xa.append(random.uniform(0.95*x,1.05*x))    #ノイズ用
                if i in st:
                                y.append(x)   ←   ここでy.append(xa[i] ) とすると
                                                  10%の誤差が混入できる
```

```
#実験データyの表示
map_y=map(str,y)
y_show=', '.join(map_y)
print("y="+ y_show)
########################################
###最急降下法###
#パラメータ初期値
par=[1.0,3.0,5.0]
#カウント数
n_count=0
while True:
        if n_count>10000:
                print("looperror")
                sys.exit()
        z=costfunc(par,y,st) #現在のコスト関数の値
        print("n=%d costfunc=%.15e" %(n_count,z))
        par_grad= grad(par,y,st)  #グラディエント計算
        alpha = bisection(par,par_grad,y,st)  #直線探索ステップ幅計算
        par_old=par  #一つ前のパラメータを保存
        par=[par[i]-alpha*par_grad[i] for i in range(len(par))]  #パラメータ更新
        s=0.0
        for i in range(len(par)):
                s += (par[i]-par_old[i])**2

        if s<1e-8:  #パラメータの更新が終了判定を満たしたら終了
                #計算にかかった更新の回数 コスト関数　導出されたパラメータ
                print("n=%d" %n_count)
                print("costfunc=%f" %costfunc(par,y,st))
                print(par)
                break
        n_count += 1
####主な計算終了
```

```
#導出されたパラメータの解軌道
xpoints_a=[]
x=par[0]
for i in range(N+1):
        xpoints_a.append(x)
        x=runge(x,par)

#グラフ描写
plt.plot(tpoints,xpoints)
plt.plot(tpoints,xpoints_a)
for i in st:
        plt.plot(i*dt,xpoints[i],marker='o',color='r',ms=10)
        # plt.plot(i*dt,xa[i],marker='o',color='r',ms=10) #ノイズ用

plt.show()
```

最適化問題の結果

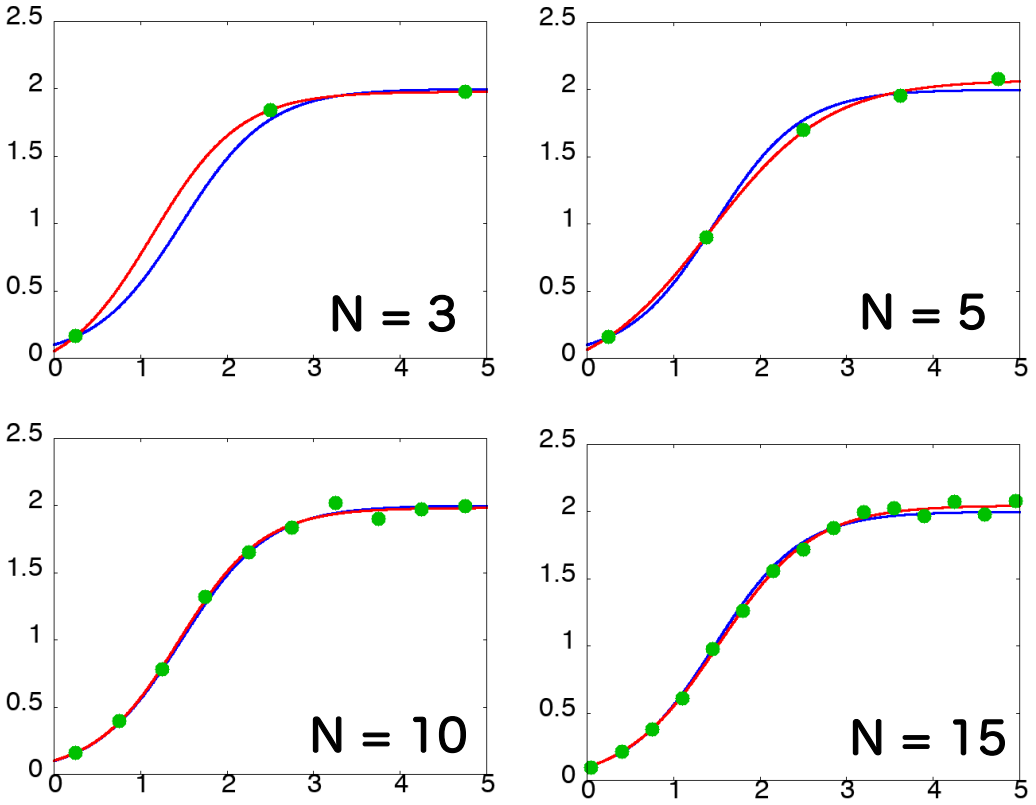
例：ロジスティック方程式

$$\begin{cases} \dfrac{dx}{dt} = a(k-x)x \\ x(0) = x_0 \end{cases}$$

パラメータは

$$\boldsymbol{\theta} = (x_0, a, k)$$

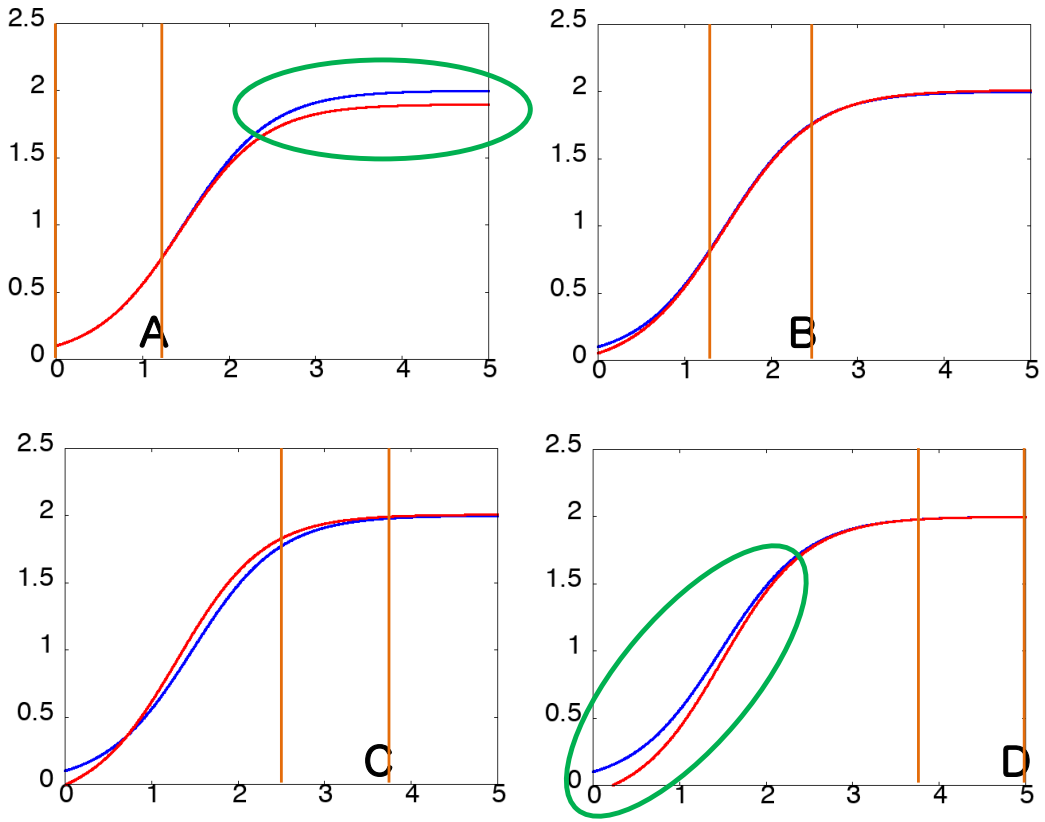
取得定量データの個数による誤差変化



取得データ数を増やす事で，パラメータ推定の精度は上がることが分かった．
また，データにノイズを加えた場合，取得データの個数が少ないときは解軌道はノイズを与えたデータ上をなぞるように動くが，取得データを増やすことでノイズを与えたデータの影響が小さくなり，与えたモデルの解軌道に近づくことが分かった．

データ点の位置による影響

取得定量データの時間位置による誤差変化



データを同じ個数取ってきても,
データの取得位置によって
解軌道に部分的な差が生じる.

データを取得してくる時間位置毎に,
取得データがどのパラメータの
情報を多く有しているかが異なる.

サンプルプログラムを動かそう！

ロジスティック方程式に対するパラメータ推定

$$\begin{cases} \dfrac{dx}{dt} = a(k-x)x \\ x(0) = x_0 \end{cases}$$

プログラム名: sample01.py

ターミナルを立ち上げて，コマンドラインから

```
% python3 sample01.py
```

とすれば計算でき，結果がグラフとして出力されます．

パッケージ不足のエラーが出たら

以下のコマンドからパッケージをインストールする

```
% pip3 install パッケージ名
```

それでも動かなかったら…，TAに聞きましょう！

演習問題

2種競争系モデル

$$\begin{cases} \dfrac{du}{dt} = u(1-u) - uv, \\ \dfrac{dv}{dt} = v(1-bv) - cuv, \end{cases}$$

初期値

$$(u, v)(0) = (u_0, v_0).$$

数理モデルの設定から
u0, v0, b, c は非負の実数

データから決定したいパラメータ

$$\boldsymbol{\theta} = (u_0, v_0, b, c)$$

sample01.pyを参考にしてプログラムを書いてみましょう！

答え

```
% python3 sample02.py
```

1. $c < 1 < b$ のとき，単安定，例えば，b=2.0, c=0.5

2. $b < 1 < c$ のとき双安定．b=0.5, c= 2.0

ベイズ推定を用いたパラメータ推定

データ点の取り方によって，モデル方程式から生成したデータにも関わらず
最適化問題では収束してもパラメータを確定できない場合がある.
（ある意味データの性質が悪いとも言える）

→パラメータ同定不能問題
→谷口さん（神戸大学）の研究：グレブナー基底を使った解析から
パラメータ同定不能条件を示している

パラメータ同定が不可能な場合を込めてベイズ推定を使って
パラメータを確率分布として求めてみる.

ベイズ推定：機械学習やデータ解析等でよく用いられる方法

ベイズの定理

$\boldsymbol{Y}$ を定数,$\boldsymbol{\theta}$ を確率変数とする．ベイズの定理より以下の関係式が成り立つ．

$$p(\boldsymbol{\theta}|\boldsymbol{Y}) = \frac{L(\boldsymbol{Y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int L(\boldsymbol{Y}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} \quad \Longrightarrow \quad p(\boldsymbol{\theta}|\boldsymbol{Y}) \propto L(\boldsymbol{Y}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

事後分布　尤度関数　事前分布

ベイズ推定とは事前分布(結果)と尤度関数からベイズの定理を用いて
事後分布(原因)を推定することである．事後確率を確率分布として
求めることでパラメータ推定を行う．

ベイズ推定の適用

データの個数をN，数理モデルのパラメータを$\boldsymbol{\theta}$,

実験データを$\boldsymbol{Y} = (y_1, \cdots, y_N) \in \mathbb{R}^N$,

パラメータ $\boldsymbol{\theta}$ 下でのシミュレーションデータを

$\boldsymbol{\Phi}(\boldsymbol{\theta}) = (x_1, \cdots, x_N) \in \mathbb{R}^N$ とする.

$\boldsymbol{\Phi}(\boldsymbol{\theta})$ に対する $\boldsymbol{\theta}$との相対誤差は正規分布：$\dfrac{1}{\sqrt{2\pi}\sigma} \exp\left(-\dfrac{x^2}{2\sigma^2}\right)$ に従うとし，

以下の式が成り立つと仮定する

$$y_j = (1 + \eta_j) x_j, \quad (1 \leq j \leq N)$$

ここで，η_j は分散 σ_j，平均0の正規分布に従う確率変数.

このとき，尤度関数は以下のように定義できる：

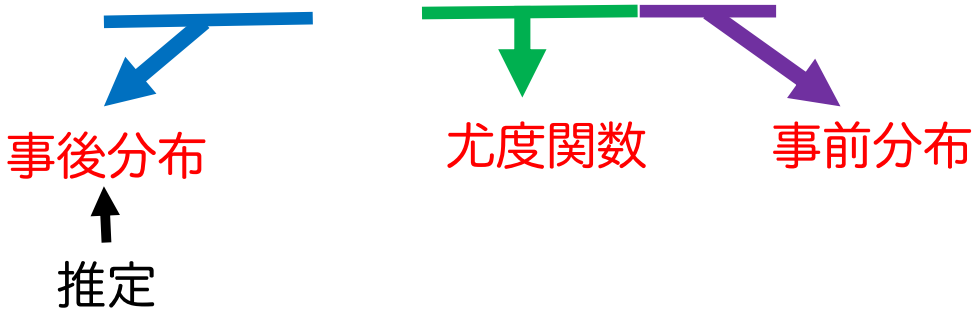
$$L(\boldsymbol{Y}|\boldsymbol{\theta}) = \sqrt{\frac{\alpha}{\pi}} \exp\left(-\alpha \sum_{j=1}^{N} \left(\frac{y_j - x_j}{x_j}\right)^2\right) \qquad (\alpha > 0)$$

実験データとシミュレーションデータの距離が近いほど良いという指標(尤度が大きい)

ベイズの定理再出

ベイズの定理

$$p(\boldsymbol{\theta}|\boldsymbol{Y}) \propto L(\boldsymbol{Y}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$



事前分布にはパラメータに関してのアプリオリな情報を取り込む．
事前に得ている情報が少ない時は，広い区間での一様分布や
分散がとても大きいガウス分布を用いる．(無情報事前分布)

事後分布 → 「与えられた実験データに対して数理モデルが尤もらしい解を生成するパラメータの確率」

事後分布の推定方法

マルコフ連鎖モンテカルロ法(MCMC)

MCMC → 多変量の確率分布からの乱数発生アルゴリズム

メトロポリス
ヘイスティング法
（M-H法） → MCMCの代表的な手法の一つ

M-H法を用いて事後分布からのサンプリングを行う

M-H法アルゴリズム（ベイズ推定）

求めたい分布 $p(\boldsymbol{\theta}|\boldsymbol{y})$ からのサンプリング

1. 初期値 $\boldsymbol{\theta}^{(0)}$ を決める.
2. $\tau = 0, 1, \ldots$ に対して次を繰り返す.

   $\boldsymbol{\theta}^*$ を提案分布 $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(\tau)})$ から発生させる.(提案分布は $\boldsymbol{\theta}^{(\tau)}$ を中心とした一様分布)
   u を区間 $(0, 1)$ の一様分布から発生させ

$$\boldsymbol{\theta}^{(\tau+1)} = \begin{cases} \boldsymbol{\theta}^* & u \leqq \min\left\{1, \dfrac{p(\boldsymbol{\theta}^*|\boldsymbol{Y})q(\boldsymbol{\theta}^{(\tau)}|\boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}^{(\tau)}|\boldsymbol{Y})q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(\tau)})}\right\} \text{ の場合} \\ \boldsymbol{\theta}^{(\tau)} & \text{そのほかの場合} \end{cases}$$

   遷移確率

   とする.

遷移確率の計算

$$\min\left\{1, \frac{p(\boldsymbol{\theta}^*|\boldsymbol{Y})q(\boldsymbol{\theta}^{(\tau)}|\boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}^{(\tau)}|\boldsymbol{Y})q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(\tau)})}\right\} = \min\left\{1, \frac{p(\boldsymbol{\theta}^*|\boldsymbol{Y})}{p(\boldsymbol{\theta}^{(\tau)}|\boldsymbol{Y})}\right\} \quad (\because \text{提案分布の対称性})$$

$$= \min\left(1, \frac{L(\boldsymbol{Y}|\boldsymbol{\theta}^*)p(\boldsymbol{\theta}^*)}{L(\boldsymbol{Y}|\boldsymbol{\theta}^{(\tau)})p(\boldsymbol{\theta}^{(\tau)})}\right) \quad (\because \text{ベイズの定理})$$

M-H法アルゴリズム（ベイズ推定）改訂

求めたい分布 $p(\boldsymbol{\theta}|\boldsymbol{y})$ のサンプリングを行う

1. 初期値 $\boldsymbol{\theta}^{(0)}$ を決める.

2. $\tau = 0, 1, \ldots$ に対して次を繰り返す

$\boldsymbol{\theta}^*$ を提案分布 $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(\tau)})$ から発生させる.

(提案分布は $\boldsymbol{\theta}^{(\tau)}$ を中心とした一様分布)

$$L(\boldsymbol{Y}|\boldsymbol{\theta}^*)p(\boldsymbol{\theta}^*) > L(\boldsymbol{Y}|\boldsymbol{\theta}^{(\tau)})p(\boldsymbol{\theta}^{\tau}) \rightarrow \boldsymbol{\theta}^{(\tau+1)} = \boldsymbol{\theta}^*$$

$$L(\boldsymbol{Y}|\boldsymbol{\theta}^*)p(\boldsymbol{\theta}^*) \leq L(\boldsymbol{Y}|\boldsymbol{\theta}^{(\tau)})p(\boldsymbol{\theta}^{\tau}) \rightarrow$$

$\boldsymbol{u}$ を区間 $(0, 1)$ の一様分布から発生させる.

$$\boldsymbol{\theta}^{(\tau+1)} = \begin{cases} \boldsymbol{\theta}^*, & u \leq \dfrac{p(\boldsymbol{\theta}^*|\boldsymbol{Y})q(\boldsymbol{\theta}^{(\tau)}|\boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}^{(\tau)}|\boldsymbol{Y})q(\boldsymbol{\theta}^{(\tau)}|\boldsymbol{\theta}^{(\tau)})} \\ \boldsymbol{\theta}^{(\tau)}, & else \end{cases}$$

とする.

遷移確率

遷移確率の計算 $\min\left\{1, \dfrac{p(\boldsymbol{\theta}^*|\boldsymbol{Y})q(\boldsymbol{\theta}^{(\tau)}|\boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}^{(\tau)}|\boldsymbol{Y})q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(\tau)})}\right\} = \min\left\{1, \dfrac{p(\boldsymbol{\theta}^*|\boldsymbol{Y})}{p(\boldsymbol{\theta}^{(\tau)}|\boldsymbol{Y})}\right\}$ (∵提案分布の対称性)

$$= \min\left(1, \frac{L(\boldsymbol{Y}|\boldsymbol{\theta}^*)p(\boldsymbol{\theta}^*)}{L(\boldsymbol{Y}|\boldsymbol{\theta}^{(\tau)})p(\boldsymbol{\theta}^{(\tau)})}\right)$$ (∵ベイズの定理)

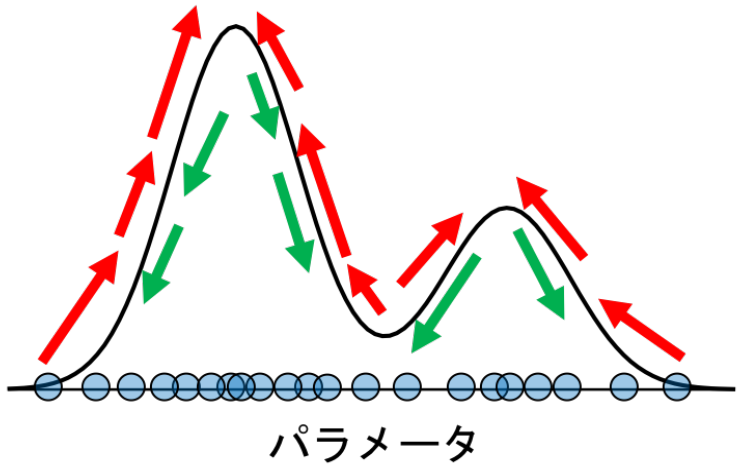
M-H法の利点

最急降下法　→　次の候補点をその点の勾配によって決定

局所最適解に陥りやすい

M-H法　→　次の候補点をランダムウォークによって決定

局所最適解に陥りにくい



M-H法によるサンプリングのイメージ

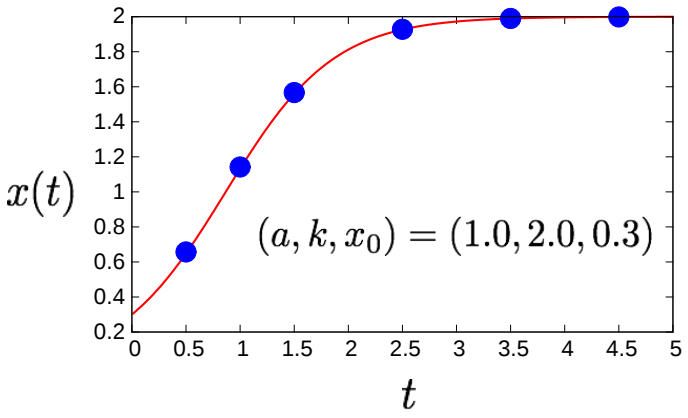
ベイズ推定を用いた数理モデルのパラメータ推定の適用例

ロジスティック方程式

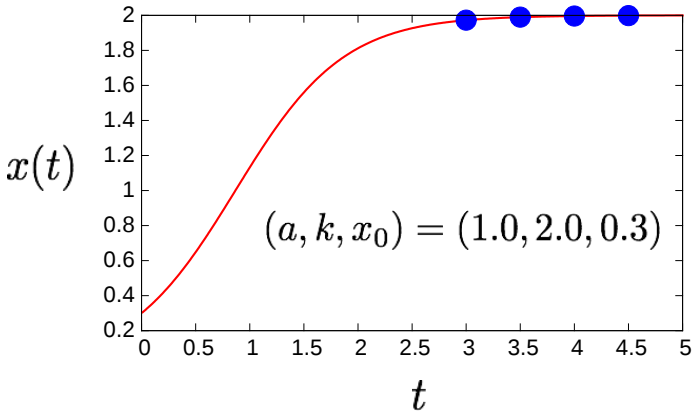
$$\begin{cases} \dfrac{dx}{dt} = a(k-x)x \\ x(0) = x_0 \end{cases}$$

パラメータを $\boldsymbol{\theta} = (a, k, x_0)$ とする.
下図の青い点を実験データとし,
これらの点を用いて実験1,2の場合における
$\boldsymbol{\theta} = (a, k, x_0)$ を推定する.

赤線：解軌道　青点：実験データ



実験1：初期状態から平衡状態まで
ほぼ均等な位置の実験データ6点

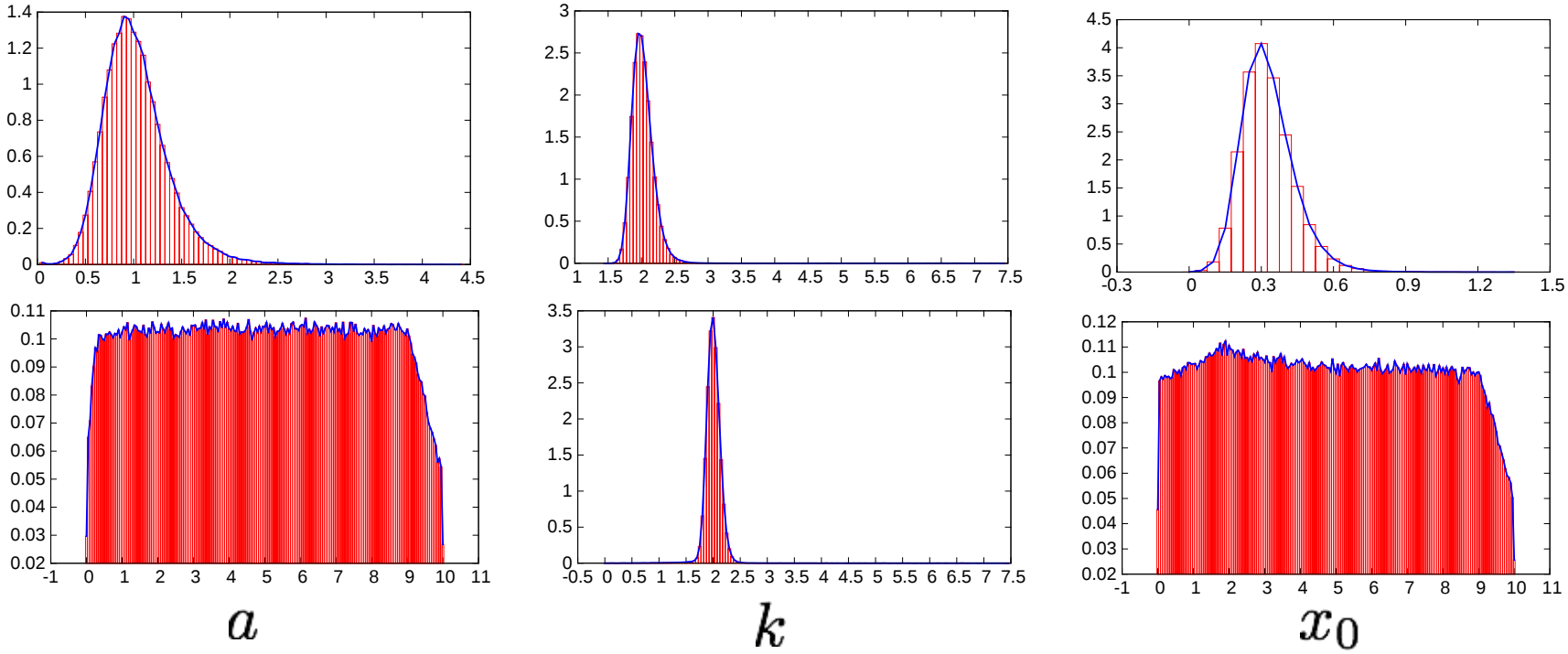


実験２：平衡解付近のみの実験データ4点

ベイズ推定を用いた数理モデルのパラメータ推定の適用例

実験1

実験2

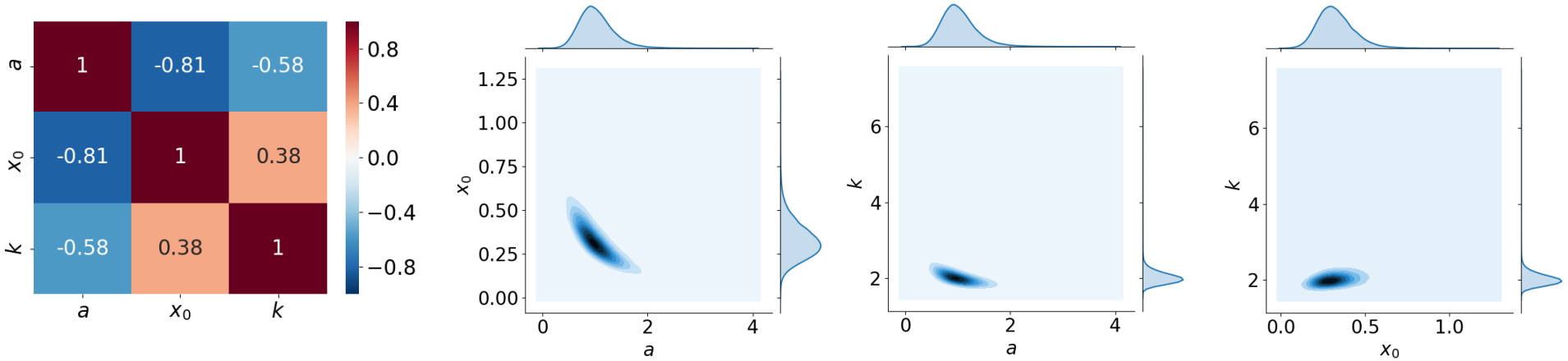


(各パラメータのサンプリングをヒストグラムにしたもの)

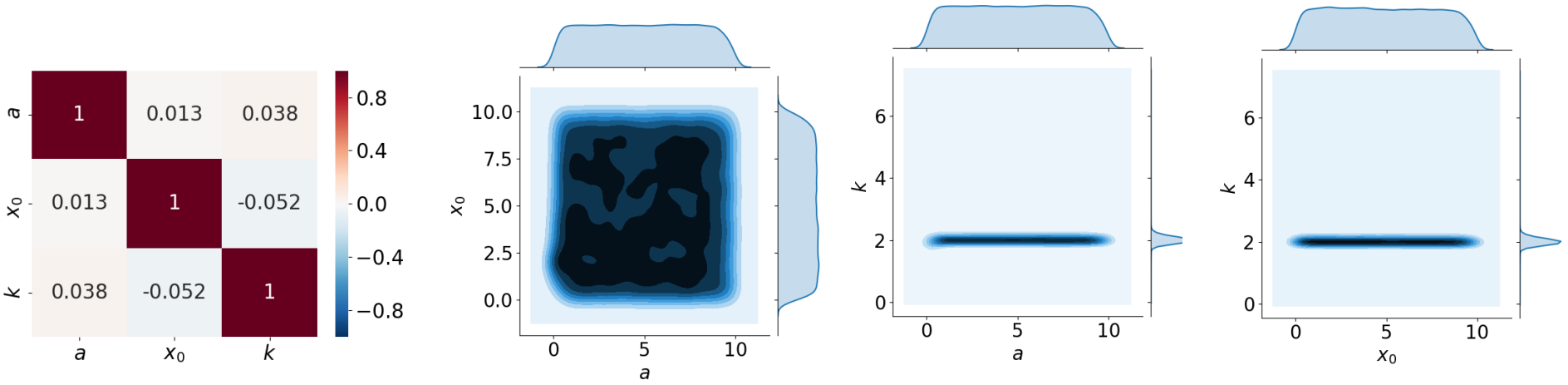
パラメータの真値 $(a, k, x_0) = (1.0, 2.0, 0.3)$

サンプリングは全て1.0×10^7 個，事前分布は$[0, 10]$の一様分布を仮定し，
メタパラメータα は40 とした.

パラメータ間の相関図



実験1におけるサンプリングされたパラメータ間の相関係数のヒートマップと散布図



実験2におけるサンプリングされたパラメータ間の相関係数のヒートマップと散布図

相関係数

データ列$\{x_i\}$と$\{y_i\}$の標本相関係数 $(i = 1, \cdots, n)$

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\left(\sum_{i=1}^{n}(x_i - \bar{x})^2\right)\left(\sum_{i=1}^{n}(y_i - \bar{y})^2\right)}}$$

ただし

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

サンプルプログラム

MHMCを用いたベイス推定の計算（ロジスティック方程式）

```
% python3 sample02.py
```

パラメータの分布を可視化

```
% pythos3 makehistogram.py
```

相関係数とデータの分布図を計算

```
% cat xxxxx.dat | python3 heatmap.py
```

演習問題

2種競争系モデル

$$\begin{cases} \dfrac{du}{dt} = u(1-u) - uv, \\ \dfrac{dv}{dt} = v(1-bv) - cuv, \end{cases}$$

初期値

$$(u, v)(0) = (u_0, v_0).$$

数理モデルの設定から
u0, v0, b, c は非負の実数

データから決定したいパラメータ

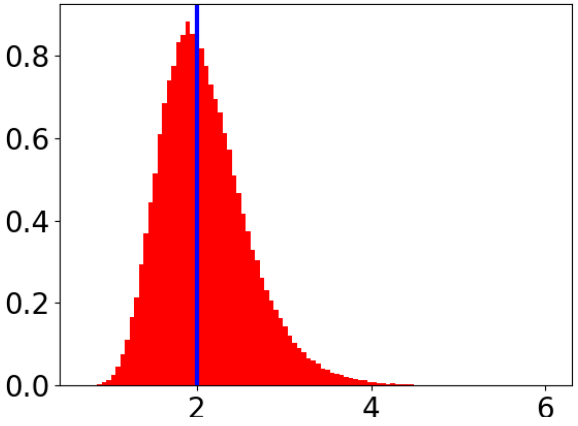
$$\boldsymbol{\theta} = (u_0, v_0, b, c)$$

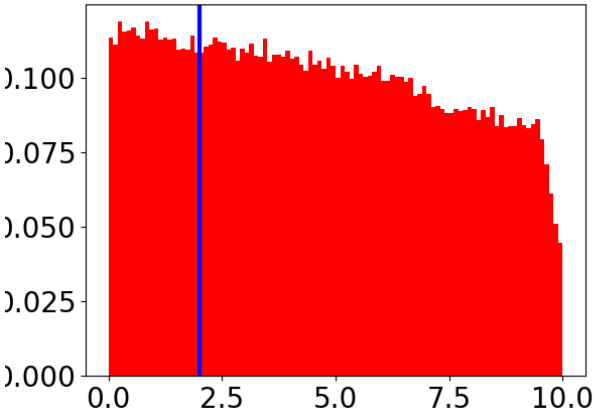
sample02.pyを参考にしてプログラムを書いてみましょう！

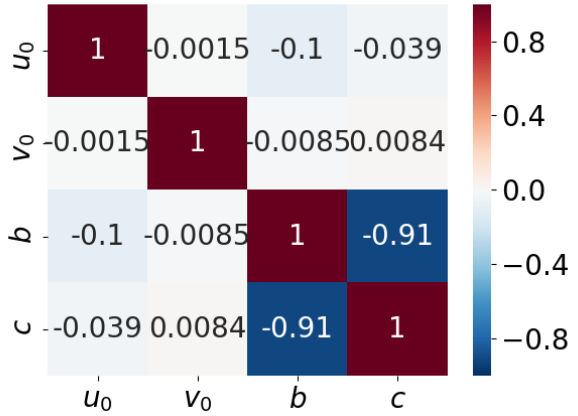
答え

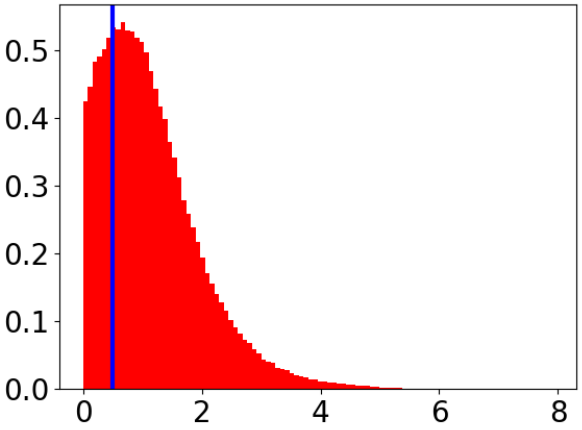
```
% python3 mcmc_Competition.py
```

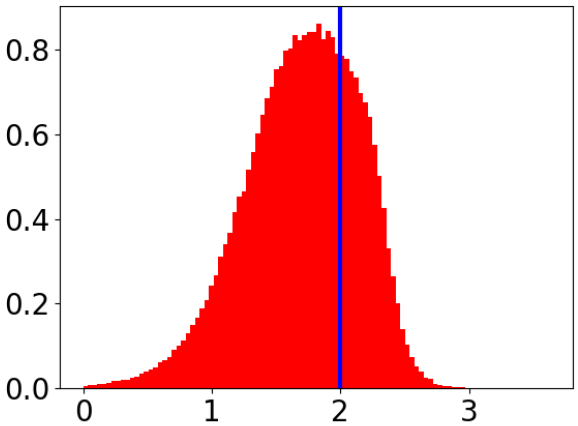
1. $c < 1 < b$ のとき，単安定，例えば，b=2.0, c=0.5
2. $b < 1 < c$ のとき双安定．b=0.5, c= 2.0











参考文献

金森敬文，鈴木大慈，竹内一郎，佐藤一誠著，
機械学習のための連続最適化，講談社

C.M.ビショップ著，パターン認識と機械学習下，丸善出版

数理モデルの数値計算法について

数値計算に必要な知識は以下から！

http://mmc01.es.hokudai.ac.jp/els/link

「私にとっての反応拡散系数値シミュレーション入門，基礎編」の
PDFファイルのダウンロード.