

# 生化学反応系の数理モデル -パラメータフィッティング-

大野 聡

東京大学 黒田研

# お知らせ

今日使うファイル類は

<http://kurodalab.bi.s.u-tokyo.ac.jp/class/Summer/2015/Day3/>

に置いてあります。(テキストエンコーディングはSJIS)

慣れてきたら自力で全部書く、あるいは、  
これまで作ったプログラムを応用して作るようにして下さい。

課題が終わった人は、  
積極的に発展課題に取り組んで下さい。

# 前回の復習

生化学反応を常微分方程式 (ODE) で記述



数値計算で解く (数値積分)



解 (反応系の振る舞い) の解析

# そもそも、なぜモデルを作るの？

現象を定量的に理解したい

- 観測データの再現には、どのパラメータが重要？

予測したい

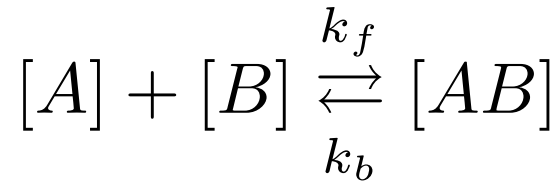
- どんな実験条件なら、面白い現象が見られる？

実験で確かめるのが困難なことを知りたい

- コスト・技術・倫理などの問題でそもそも実験できないことを知りたい

# モデルを使ったシミュレーション

反応式



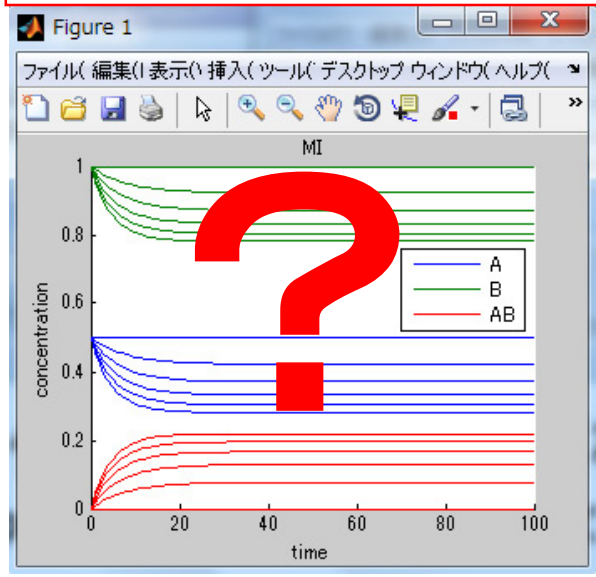
モデル

$$\frac{d[AB]}{dt} = k_f[A][B] - k_b[AB]$$



数値シミュレーション

モデル = 構造 + パラメータ  
(定性的) (定量的)



モデルさえ決まってしまうえば  
様々な条件下での振る舞いが  
シミュレーション可能！

⇒ 解析、実験デザインに有用

# 構造が決まった。パラメータはどうやって決める？

○文献を参照 ⇒ 実験条件によってまちまち

○反応の速度定数を実験的に測る！

⇒ 1)パラメータが多いとしんどい...

2)誤差を含む(正確にわかるわけでない)

多くの場合、速度定数そのものが見たいわけではない！

自分がほしいパラメータは自分で求めないといけない！

(興味のあるもの:**システム全体の振る舞いを再現するモデル**)

誤解を恐れず極論すれば

たとえ**物理的に正確な速度定数**が神様から告げられても

モデルに対して**現象を再現するパラメータ**とは限らない！

※そもそも、全ての要素を考慮したモデルは作れない。

今回の例では、空間的な効果は考えない。

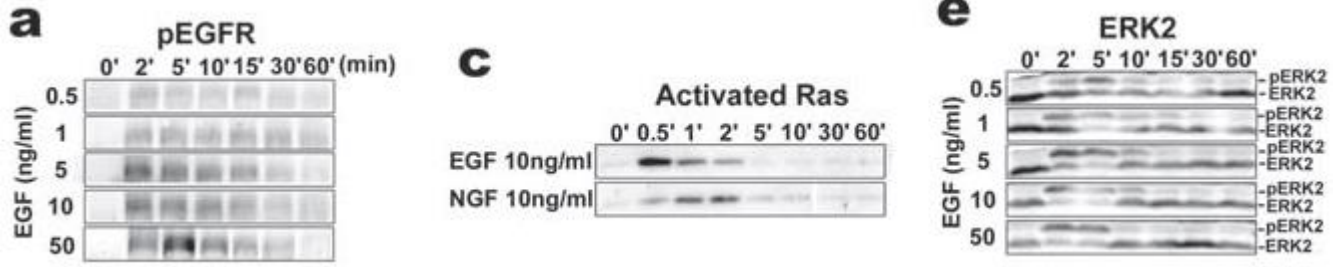
# 欲しいパラメータは目的(モデル)依存

モデルの構築は目的依存(自分がみたいものを見る)

➡ **パラメータはモデルに依存して決まる。**

例: 細胞を刺激することで測定された  
**各シグナル分子の時間波形を再現**するモデルが欲しい!

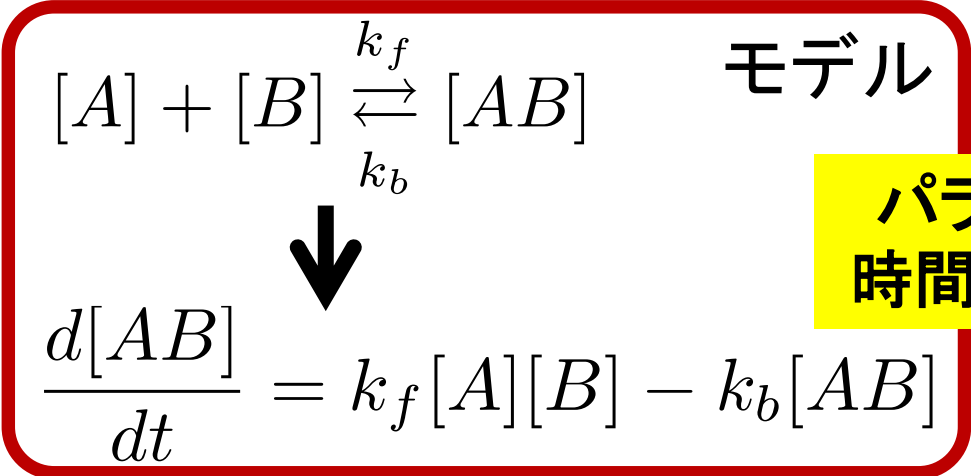
実験データ: ウェスタンブロッティング(WB)やイメージングなどにより取得



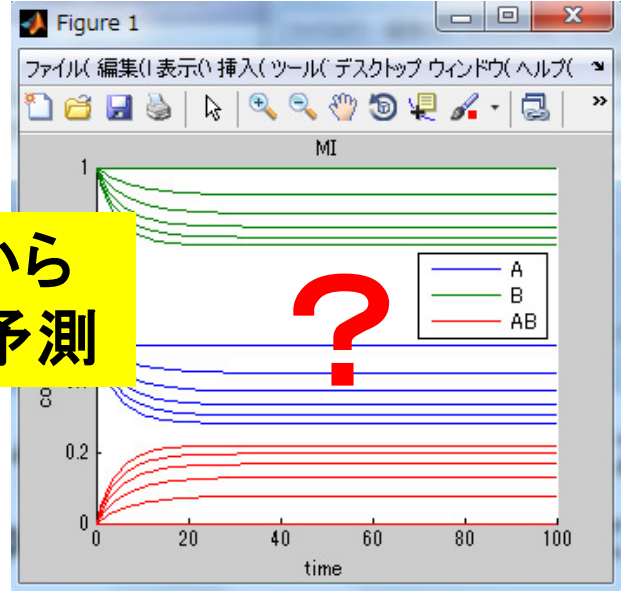
モデルの構造を決定

➡ **このモデルで実験データを再現するパラメータを推定**

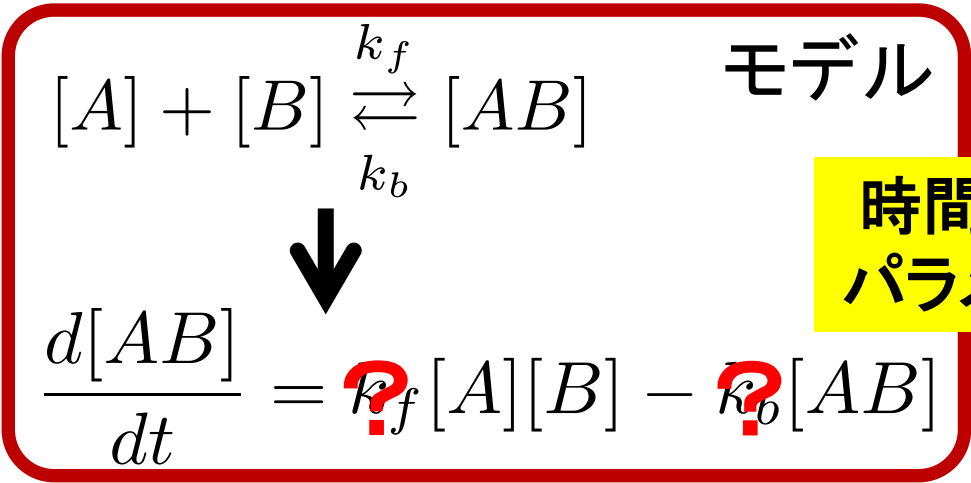
# 順問題(前回)



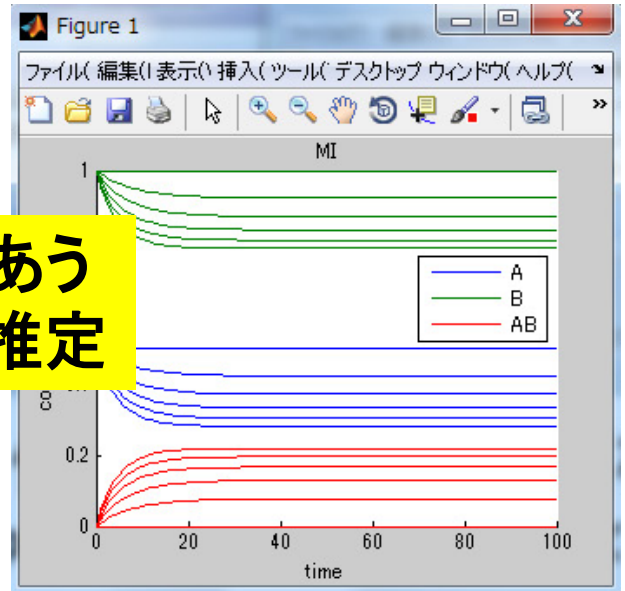
パラメータから  
時間波形を予測



# 逆問題(今回すること)



時間波形にあう  
パラメータを推定





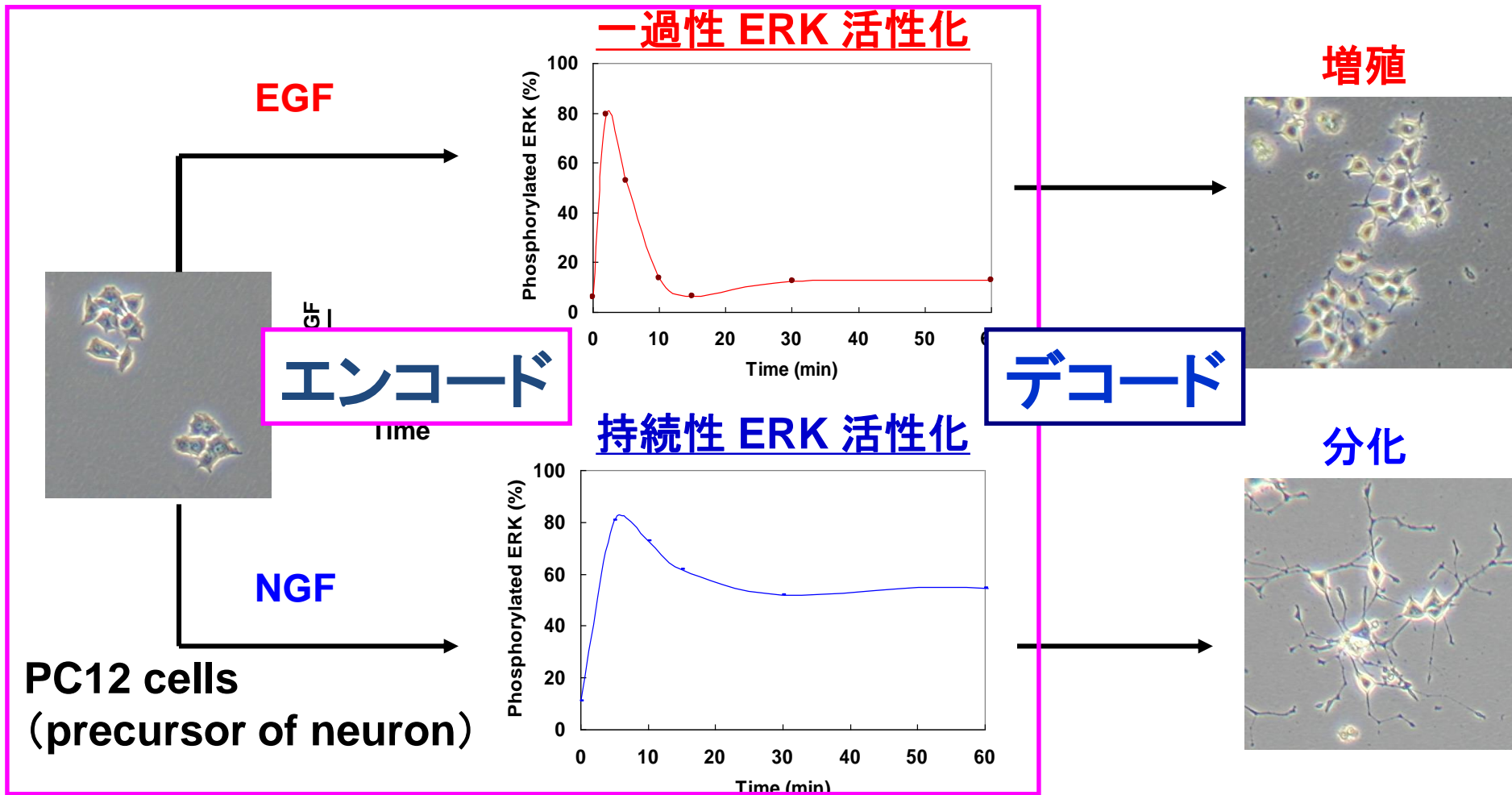
# 今日の流れ

1. Ras, Rap1のODEモデルを作成
2. パラメータ推定と最小2乗法について
3. ODEモデルの1つの未知パラメータを推定
4. ODEモデルの4つの未知パラメータを推定

# 細胞運命決定機構：増殖と分化のスイッチ

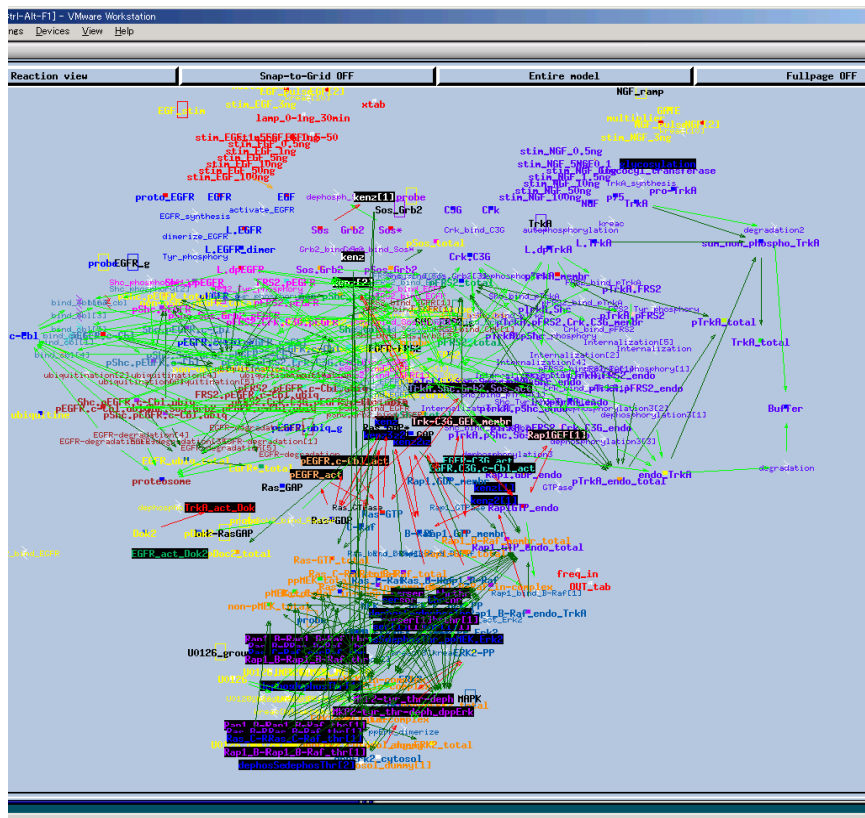
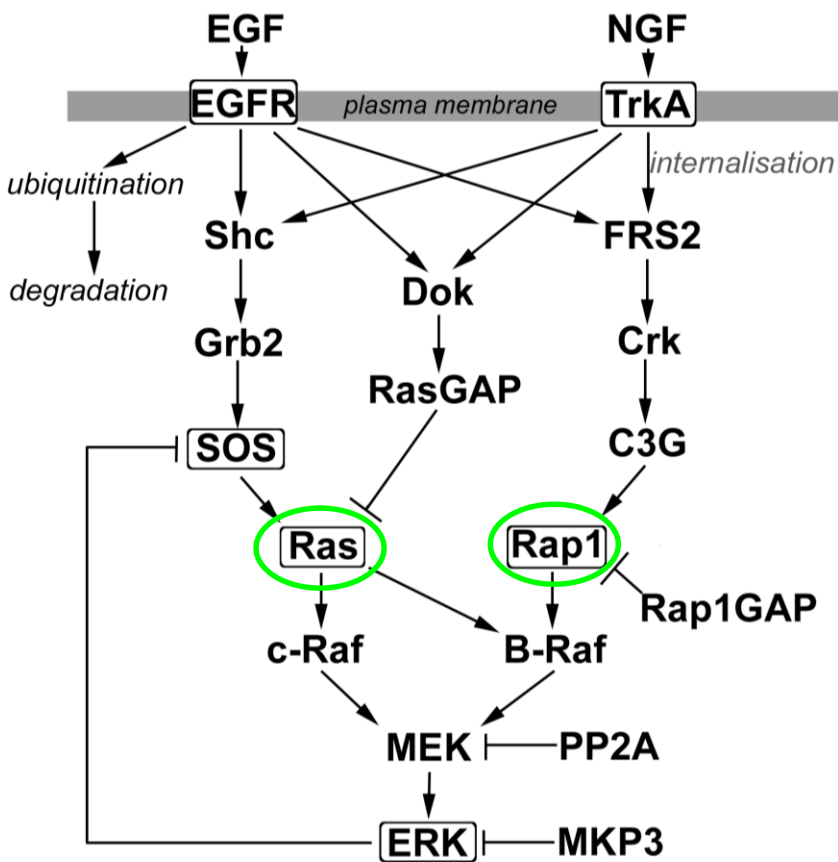
現象を分子に帰着せざるできない！

## ERKの時間波形が異なるだけで異なる現象を制御



# Ras経路とRap1経路の生化学反応モデル

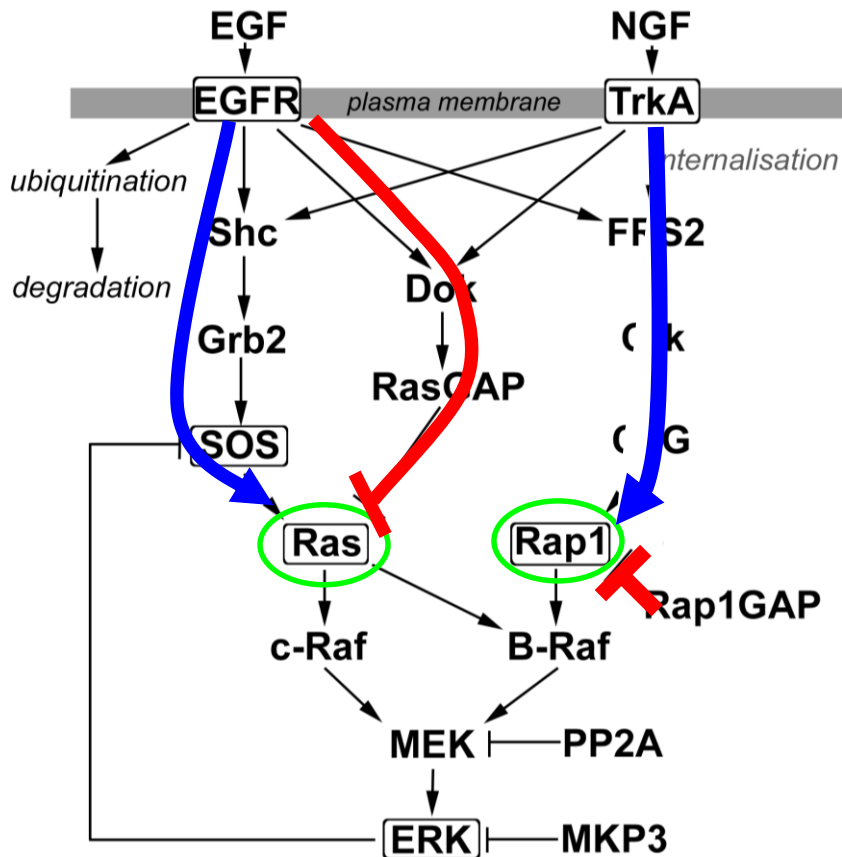
## Erk経路のブロック線図



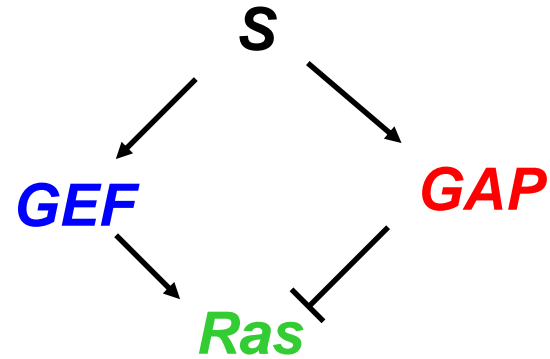
変数・パラメータが多すぎる！  
どの作用が大事なの？

# モデルの簡単化

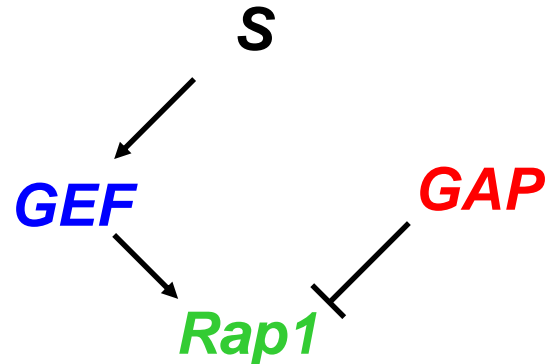
## Erk経路のブロック線図



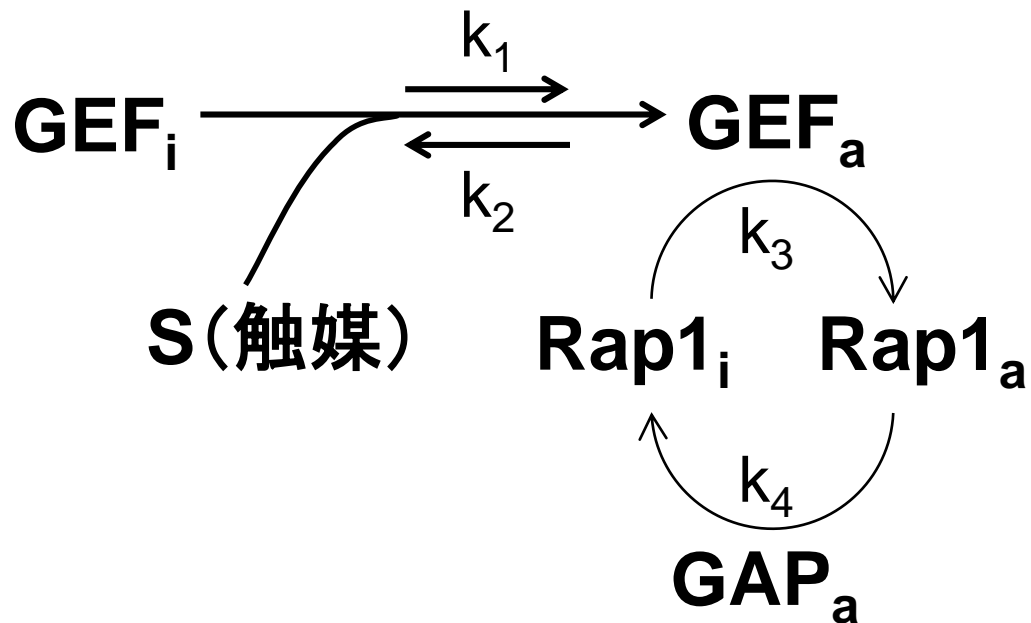
## シンプルRasモデル



## シンプルRap1モデル



# シンプルRap1モデル



変数(分子)	初期濃度
S	1 (定数)
GEF <sub>a</sub>	0
GAP <sub>a</sub>	0.005 (定数)
Rap1 <sub>a</sub>	0

パラメータ	値
k <sub>1</sub>	0.5
k <sub>2</sub>	0.5
k <sub>3</sub>	0.2
k <sub>4</sub>	10

不活性型と活性型の総和は保存

$$\text{GEF}_i + \text{GEF}_a = 1$$

$$\text{GAP}_i + \text{GAP}_a = 1$$

$$\text{Rap}_i + \text{Rap}_a = 1$$

\* ひな形をダウンロードして, Rap1\_model.mとして保存

<http://kurodalab.bi.s.u-tokyo.ac.jp/class/Summer/2015/>

# “Rap1\_model.m”をダウンロードして空欄を埋める

## その1/2

```
function Rap1_model()  
    % Rap1モデル  
    time = 0:100;  
    param = [  ]; % [k1, k2, k3, k4]の順に  
    y0 = [  ]; % [S, GEFa, GAPa, Rap1]の順に  
  
    [t_sim, timeCourse_sim] = ode15s(@(t, y) ODE(t, y, param), time, y0);  
  
    figure;  
    plot(t_sim,timeCourse_sim(:,4));  
    xlabel('time');  
    ylabel('concentration');  
    title('Rap1 model');  
  
end
```

計算の途中を見たいときは？  
デバッグ機能と変数表示

Rap1\_model()は次ページにも続く

# “Rap1\_model.m”をダウンロードして空欄を埋める

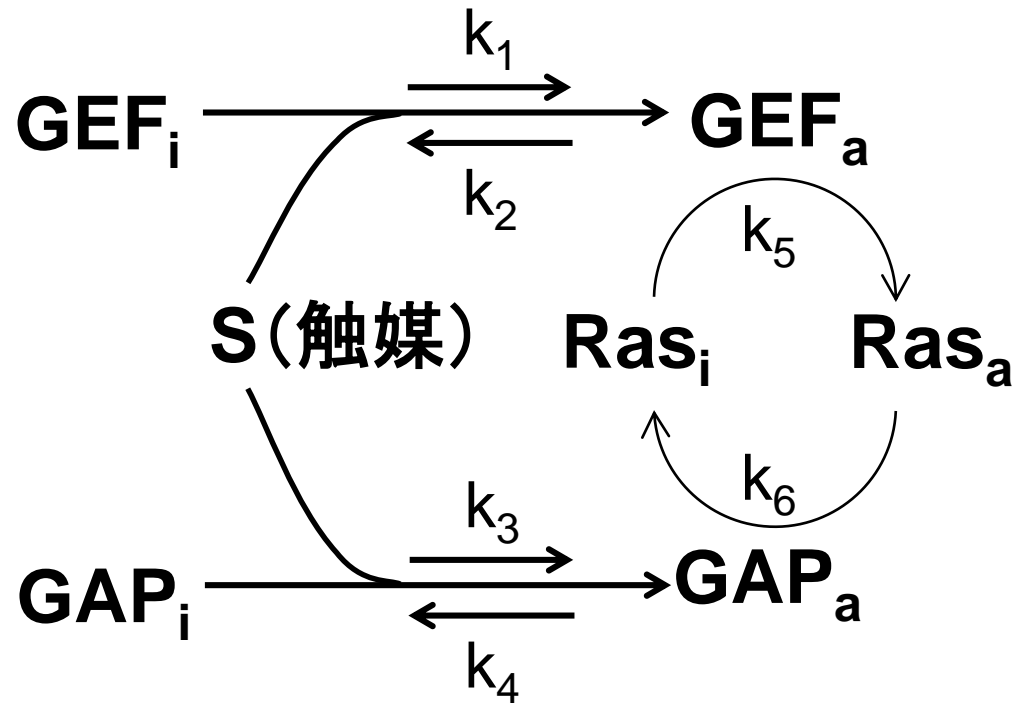
## その2/2

```
function dydt = ODE(t, y, param)
    S = y(1);
    GEFa = y(2); % GEFi = 1 - GEFa
    GAPa = y(3); % GAPi = 1 - GAPa
    Rap1a = y(4); % Rap1i = 1 - Rap1a

    k1 = param(1);
    k2 = param(2);
    k3 = param(3);
    k4 = param(4);

    dydt(1, :) =  % S
    dydt(2, :) =  % GEFa
    dydt(3, :) =  % GAPa
    dydt(4, :) =  % Rap1a
end
```

# シンプルRasモデル



変数(分子)	初期濃度
S	1 (定数)
$GEF_a$	0
$GAP_a$	0
$Ras_a$	0

パラメータ	値
$k_1$	0.5
$k_2$	5
$k_3$	0.0005
$k_4$	0.005
$k_5$	0.05
$k_6$	100

不活性型と活性型の総和は保存

$$GEF_i + GEF_a = 1$$

$$GAP_i + GAP_a = 1$$

$$Ras_i + Ras_a = 1$$

\* ひな形をダウンロードして, Ras\_model.mとして保存

<http://kurodalab.bi.s.u-tokyo.ac.jp/class/Summer/2015/>



# “Ras\_model.m”をダウンロードして空欄を埋める

## その1/2

```
function Ras_model()  
    % Rasモデル  
    time = 0:100;  
    param = [  ]; % [k1, k2, k3, k4, k5, k6]の順に  
    y0 = [  ]; % [S, GEFa, GAPa, Rasa]の順に  
  
    [t_sim, timeCourse_sim] = ode15s(@(t, y) ODE(t, y, param), time, y0);  
  
    figure;  
    plot(t_sim,timeCourse_sim(:,4));  
    xlabel('time');  
    ylabel('concentration');  
    title('Rap1 model');  
  
end
```

# “Ras\_model.m”をダウンロードして空欄を埋める

## その2/2

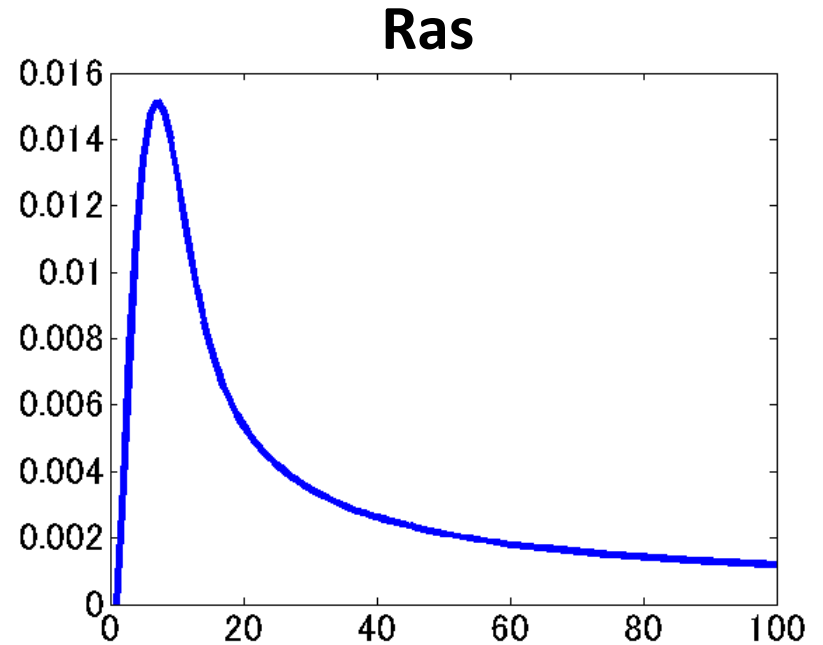
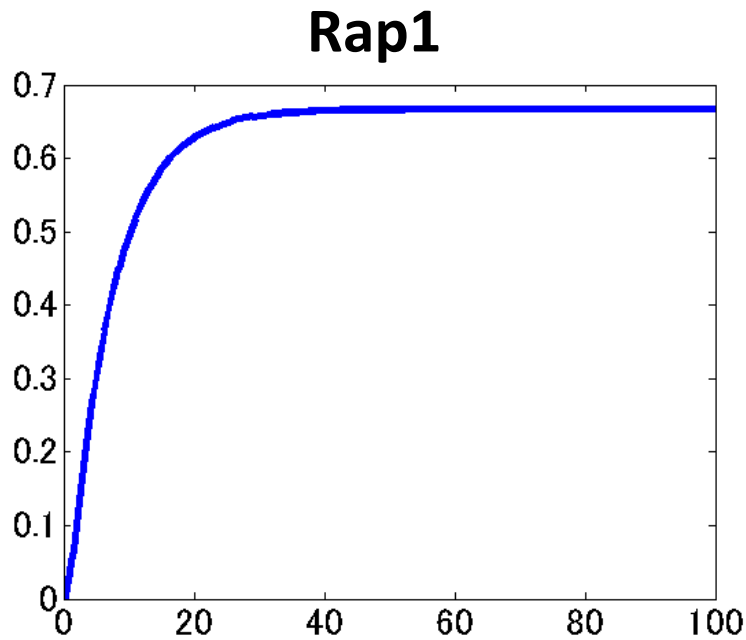
```
function dydt = ODE(t, y, param)
    S = y(1);
    GEFa = y(2); % GEFi = 1 - GEFa
    GAPa = y(3); % GAPi = 1 - GAPa
    Rasa = y(4); % Rasi = 1 - Rasa

    k1 = param(1);
    k2 = param(2);
    k3 = param(3);
    k4 = param(4);
    k5 = param(5);
    k6 = param(6);

    dydt(1, :) =  % S
    dydt(2, :) =  % GEFa
    dydt(3, :) =  % GAPa
    dydt(4, :) =  % Rasa
end
```

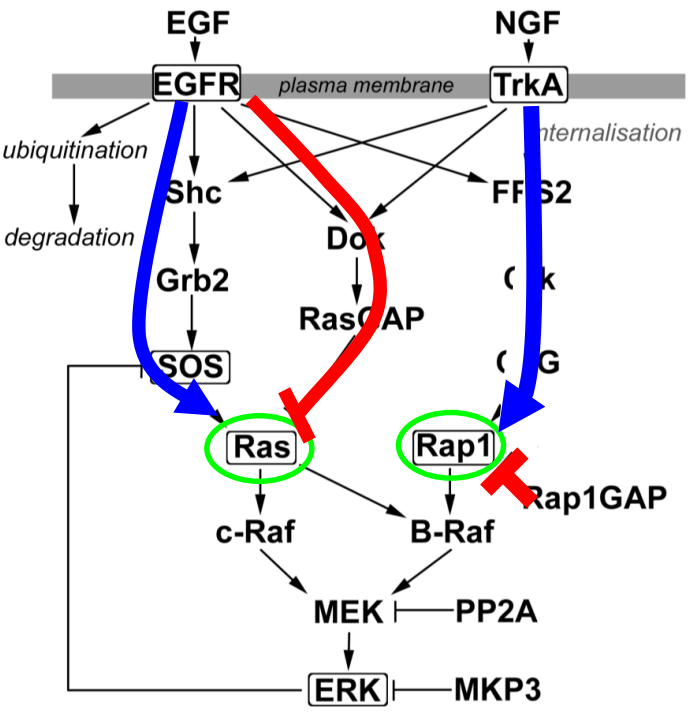
# モデルの確認

波形をプロットして、モデルが正しく作れているか確認する。



# RasとRap1からERKの時間波形が決まるようだ

## Erk経路のブロック線図



EGF

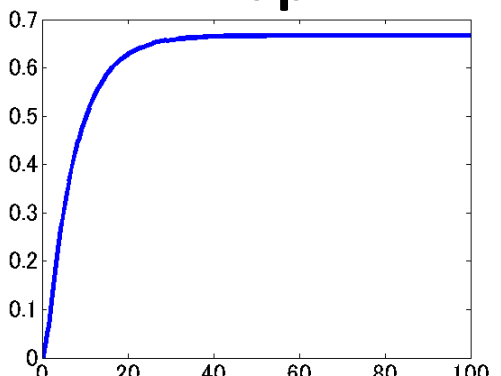
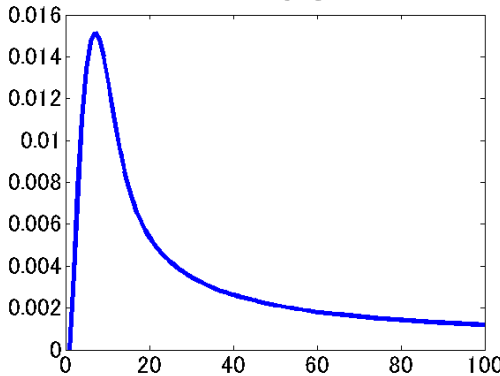


Ras

NGF

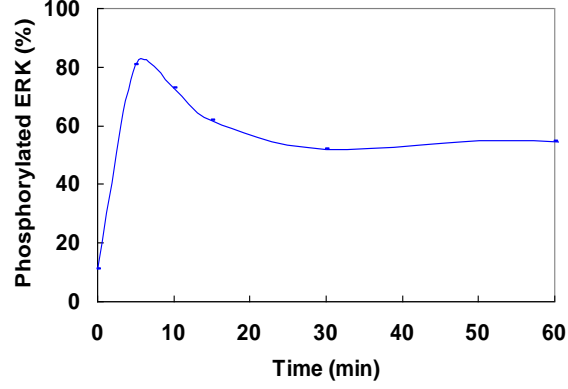
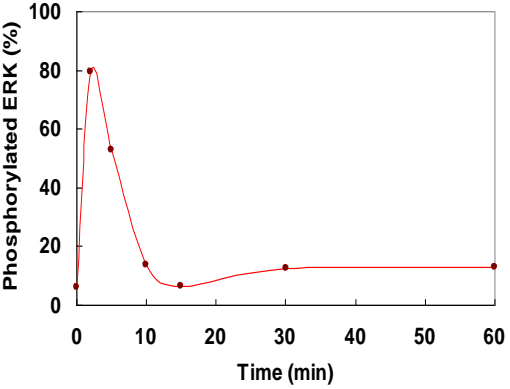


Rap1



一過性 ERK 活性化

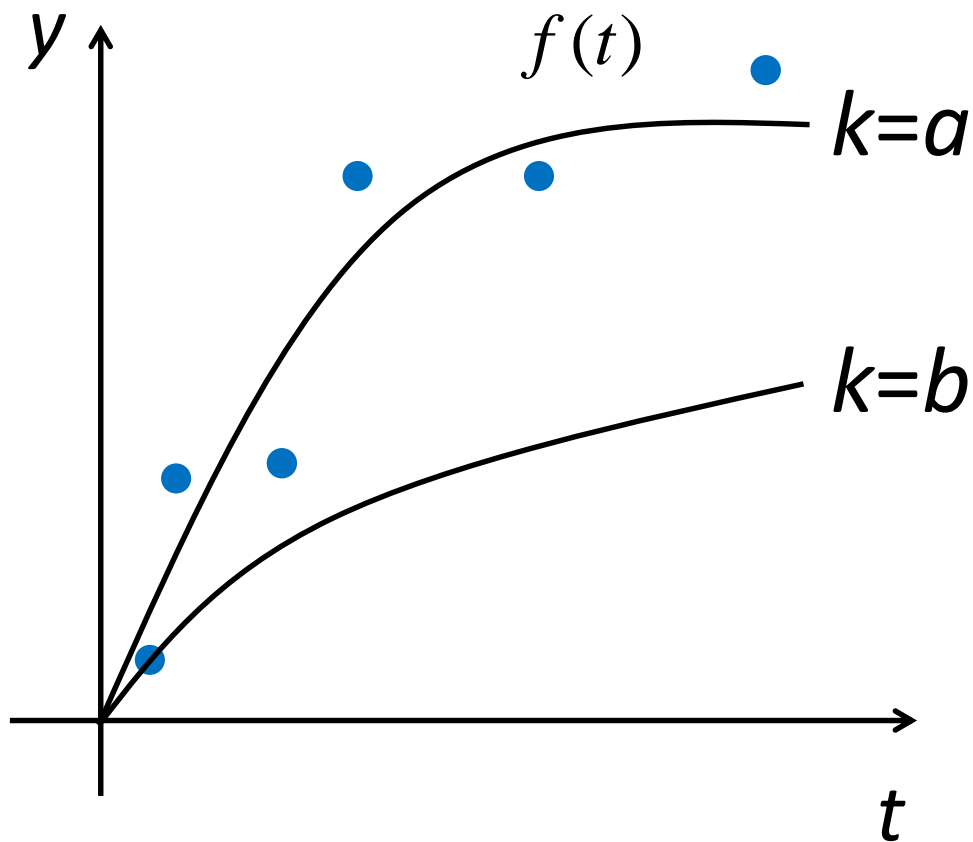
持続性 ERK 活性化



# 今日の流れ

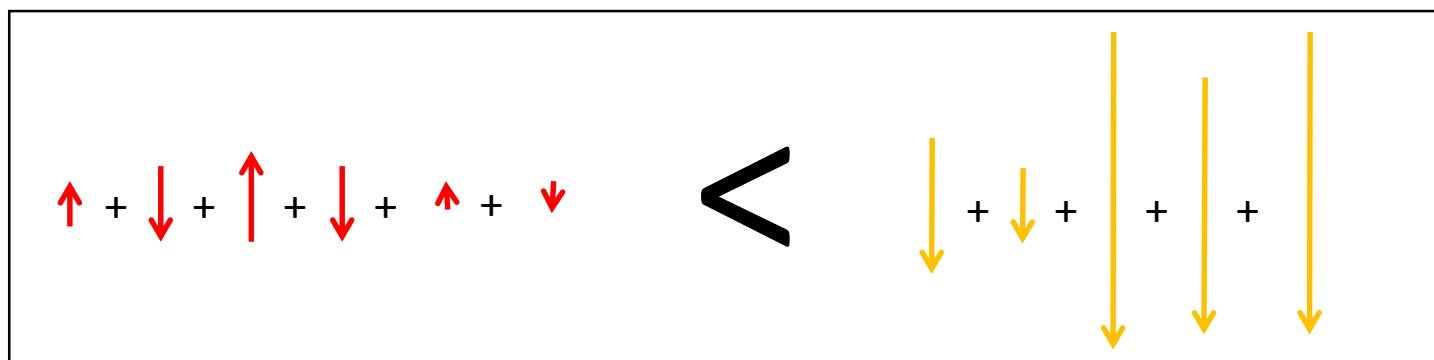
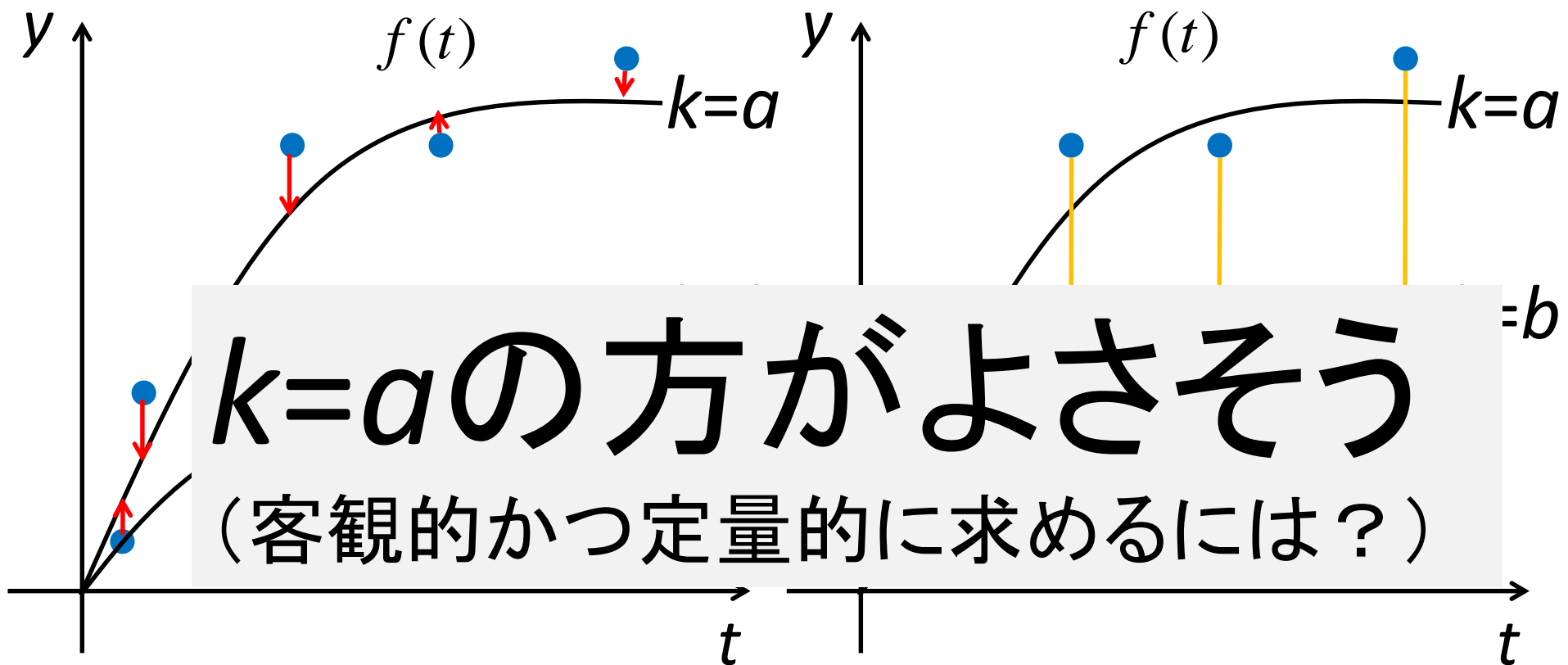
1. Ras, Rap1のODEモデルを作成
2. パラメータ推定と最小2乗法について
3. ODEモデルの1つの未知パラメータを推定
4. ODEモデルの4つの未知パラメータを推定

# パラメータの決め方(1)



$k=a$ と $k=b$ , どちらがよいパラメータか？

# パラメータの決め方(2)

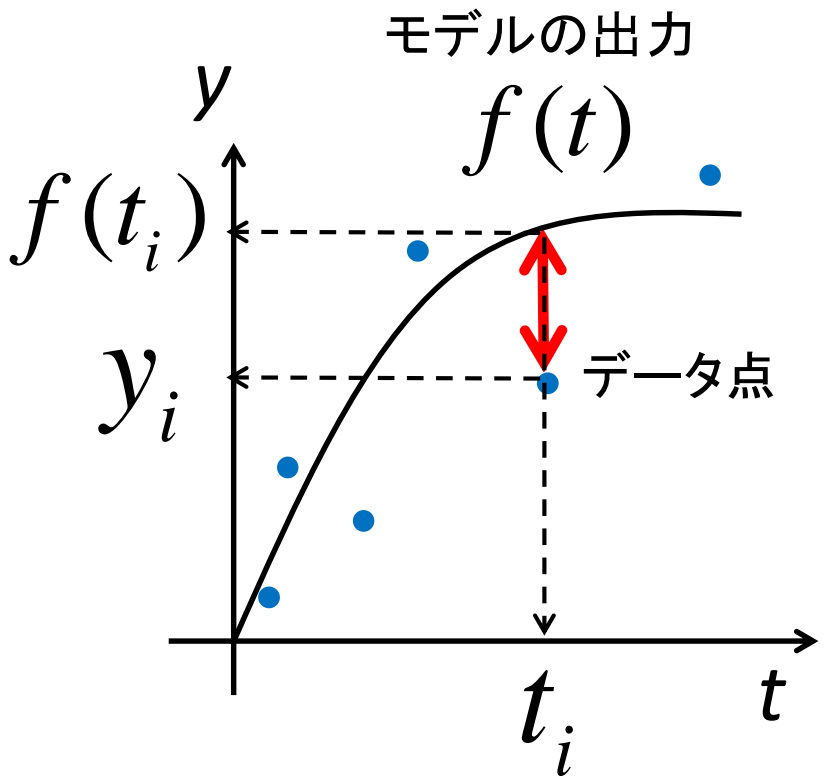


# 客観的かつ定量的な指標 → 評価関数

実験値  $y_i$       推定値  $f(t_i)$

○残差  $y_i - f(t_i)$       モデルの出力(シミュレーションで得られる分子の時間波形)と、データ点の差

○残差2乗和  $\varepsilon^2 = \sum_{i=1}^N (y_i - f(t_i))^2$       ユークリッド距離に相当



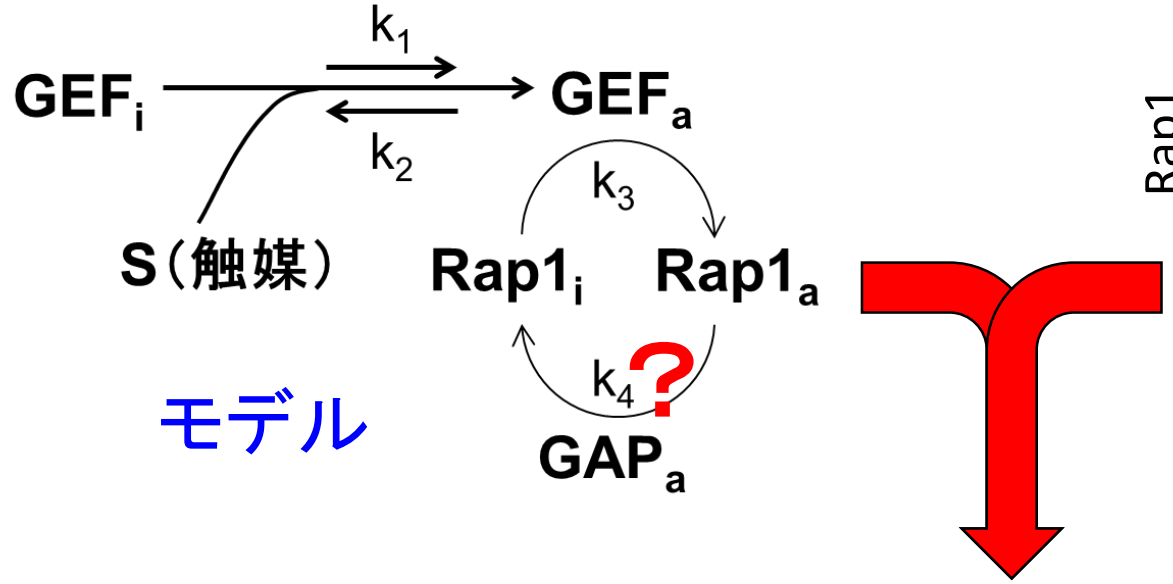
残差2乗和を評価関数として、  
最小にするようにパラメータを  
決める方法を最小2乗法という。



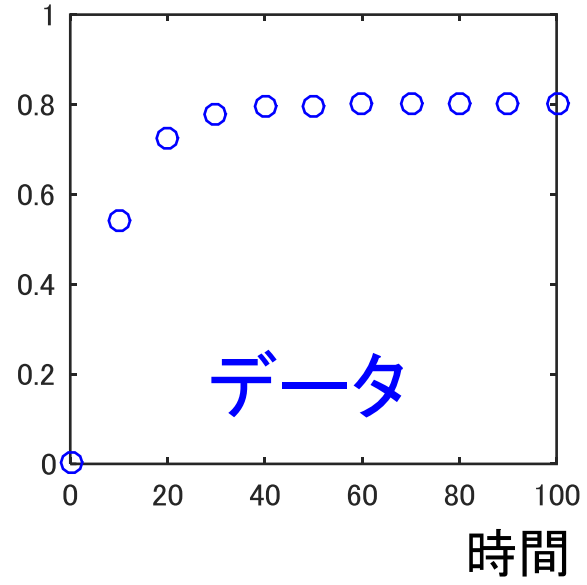
# 今日の流れ

1. Ras, Rap1のODEモデルを作成
2. パラメータ推定と最小2乗法について
3. ODEモデルの1つの未知パラメータを推定
4. ODEモデルの4つの未知パラメータを推定

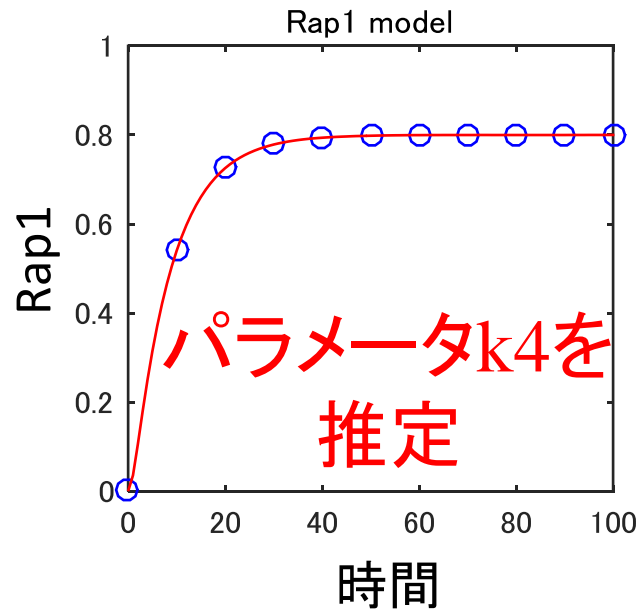
# 手でRap1のパラメータk4を求めてみる



モデル



k4を手で色々変えて、データにもっともよく合う値を探してみる。



k4以外のパラメータは既知とする

- k1 = 0.5
- k2 = 0.5
- k3 = 0.2

# 手でRap1のパラメータk4を求めてみる

データ(ノイズなし)"data\_Rap1\_1para\_noiseless.mat"をダウンロード  
<http://kurodalab.bi.s.u-tokyo.ac.jp/class/Summer/2015/>

"Rap1\_model.m"をコピーして, "hand\_Rap1\_1param.m"などとして改造  
外から, 引数でk4を与えるようにする.

↓

```
function hand_Rap1_1param(k4)
    param = [0.5, 0.5, 0.2,  ]; % [k1, k2, k3, k4]の順に
    y0 = [1, 0, 0.005, 0]; % [S, GEFa, GAPa, Rap1]の順に

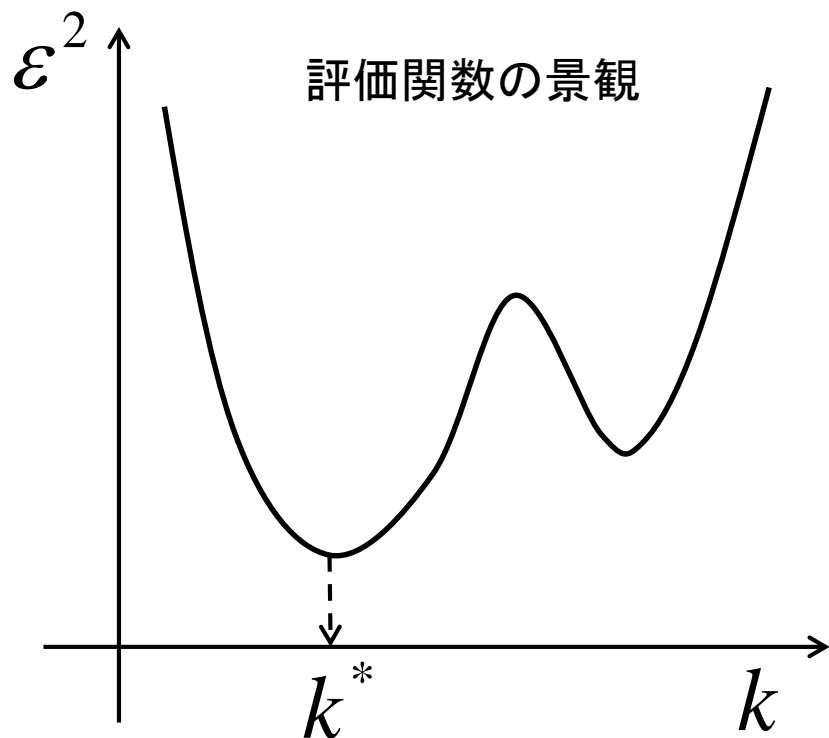
    load('data_Rap1_1para_noiseless.mat');
    [t_sim, timeCourse_sim] = ode15s(@(t, y) ODE(t, y, param), time, y0);

    figure;
    plot(t, x, 'bo'); % データ点の表示 ←表示部分を追加
    hold on;
    plot(t_sim, timeCourse_sim(:, 4), 'r-'); %シミュレーション結果の表示
    (略)
end
```

実行するときはコマンドウィンドウ上から「`hand_Rap1_1param(数値)`」

# Rap1モデルのパラメータ $k$ を推定する (ノイズなし)

1. 評価関数の景観を書く.
2. 残差2乗和が最小となるときのパラメータ $k$ を求める.



パラメータ $k$ の値を少しずつ変えながら、残差2乗和を求める。  
横軸にパラメータ $k$ の値、縦軸に残差2乗和 $\varepsilon^2$ をとってプロットする。

# “fit\_Rap1\_1param.m”をダウンロードして空欄を埋める

## その1/3

```
function [k4_est] = fit_Rap1_1param()
```

```
param = [0.5, 0.5, 0.2, NaN]; % [k1, k2, k3, k4]の順に(k4は後で変えるので、仮にNaNを代入)  
y0 = [1, 0, 0.005, 0]; % [S, GEFa, GAPa, Rap1]
```

```
load('data_Rap1_1para_noiseless.mat');
```

```
%% k4を振り、残差2乗和(RSS)が最小となるときのk4を求める
```

```
k4 = linspace(0.1, 10, 201);
```

```
[x_Rap1_RSS] = landscape_Rap1(k4, t, x, param, y0); % t及びxはデータ点の時間とRap1aの波形
```

```
[RSS_min, idx] =  % 最小値を探す関数minを使って、最小となるRSSとそのときの番号を求める  
k4_est =  % RSSが最小となるときのk4を代入。k4は配列であることに注意。
```

```
%% 上で求めた残差2乗和(RSS)が最小となるときのk4を代入し、シミュレーション
```

```
temp_param = param;
```

```
temp_param(4) = ; % k4は、RSS最小となるときの値を代入
```

```
[t_sim, timeCourse_sim] = ode15s(@(t, y) ODE(t, y, temp_param), t, y0);
```

fit\_Rap1\_1param()は次ページにも続く

k4を0.1から10まで201分割刻みで動かし  
残差2乗和(RSS)をサンプリング

# “fit\_Rap1\_1param.m”をダウンロードして空欄を埋める

## その2/3

%% 描画

```
figure(1);
```

%% 時間波形

```
subplot(1, 2, 1)
```

```
plot(t, x, 'bo'); % データ点
```

```
hold on
```

```
plot(t_sim, timeCourse_sim(:, 4), 'r-'); % シミュレーションの結果
```

%% RSSとk4の関係

```
subplot(1, 2, 2);
```

```
plot(k4, x_Rap1_RSS, 'b-');
```

```
hold on
```

```
plot(k4_est, RSS_min, 'ro');
```

```
end
```

# “fit\_Rap1\_1param.m”をダウンロードして空欄を埋める

## その3/3

```
function [x_Rap1_RSS] = landscape_Rap1(k4, SamplingTime, x_Rap1, param, y0)
```

```
    vary_x_Rap1 = zeros(length(x_Rap1), length(k4)); % 各列毎に各k4でのRap1の時間波形が入るような配  
    列を準備
```

```
    for i = 1:length(k4)
```

```
        param(4) = ; % 変えているパラメータは？配列で与えられていることに注意
```

```
        [~, tc] = ode15s(@(t, y) ODE(t, y, param), SamplingTime, y0);
```

```
        vary_x_Rap1(:, i) = tc(:, 4);
```

```
    end
```

```
    x_Rap1_Residual = vary_x_Rap1 - x_Rap1 * ones(1, size(vary_x_Rap1, 2));
```

```
    x_Rap1_RSS = ; % 残差はx_Rap1_Residual。残差2乗和はどう表すか？
```

```
end
```

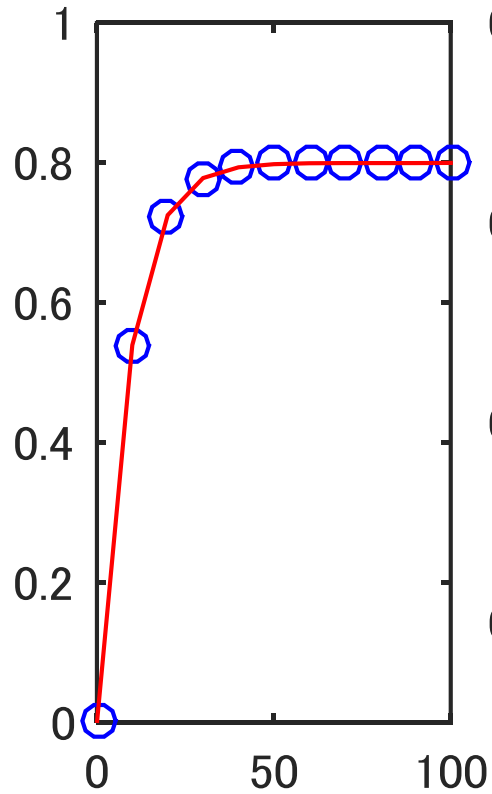
```
function dydt = ODE(t, y, param)
```

```
    % ここにモデルをコピー
```

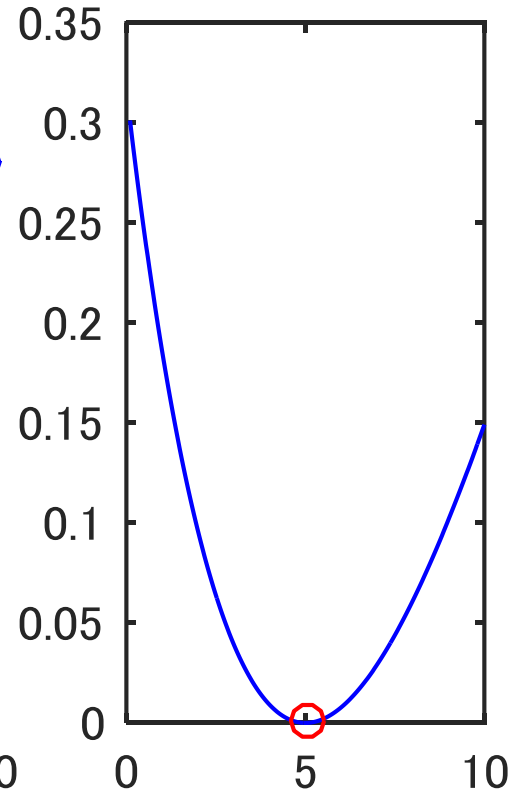
```
end
```

# 正解

データとODEの出力



評価関数の景観



$K4 = 5.005$

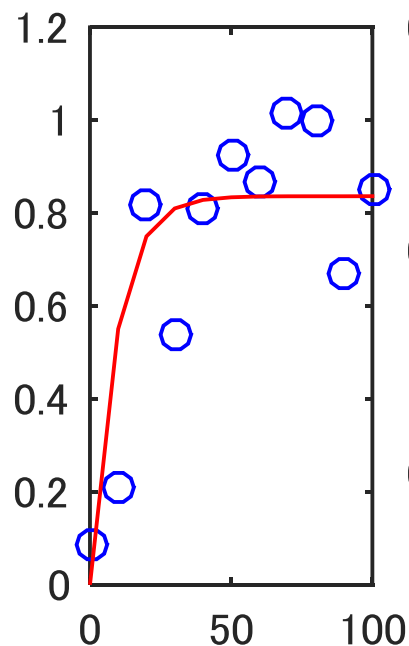
(真のモデルは,  $k4 = 5.0$ )



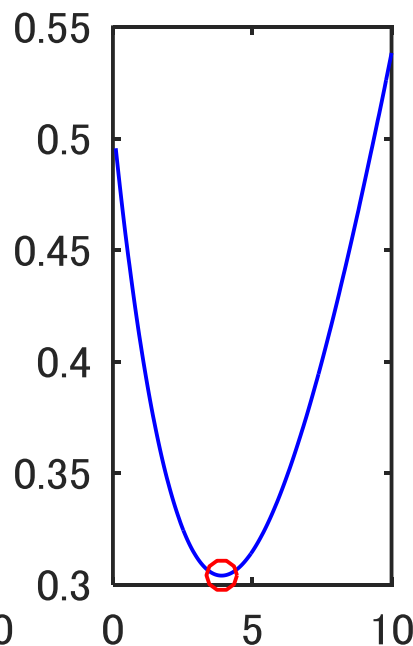
# Rap1モデルのパラメータ $k_4$ を推定する (ノイズあり)

1. データ(ノイズあり) "data\_Rap1\_1para\_noise.mat"をダウンロード  
<http://kurodalab.bi.s.u-tokyo.ac.jp/class/Summer/2015/>
2. 評価関数の景観を書く.
3. 残差2乗和が最小となるときのパラメータ $k$ を求める.

データとODEの出力



評価関数の景観



$$K4 = 3.9115?$$

# 読み込むデータの変更

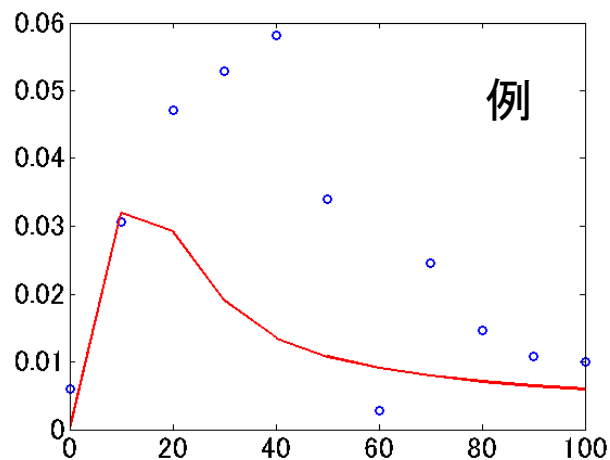
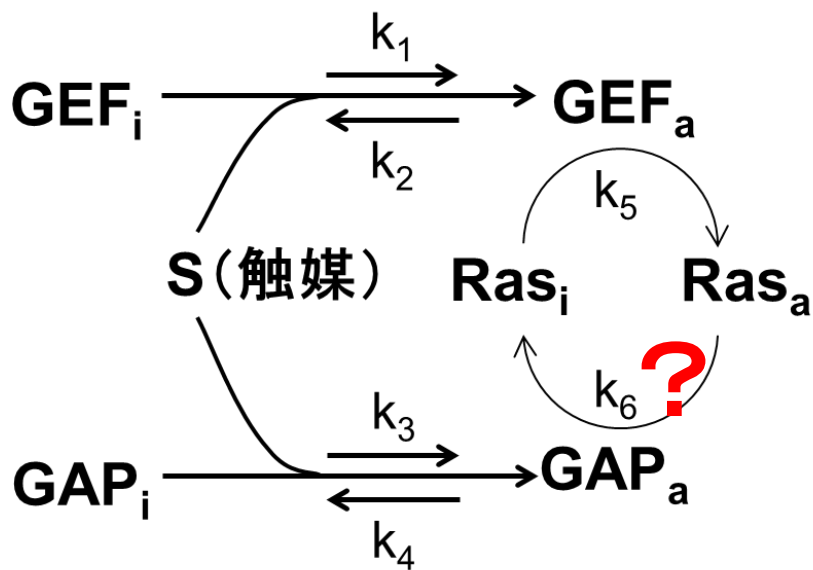
```
function [k4_est] = fit_Rap1_1param()  
  
param = [0.5, 0.5, 0.2, NaN]; % [k1, k2, k3, k4]の順に(k4は後で変えるので、仮にNaNを代入)  
y0 = [1, 0, 0.005, 0]; % [S, GEFa, GAPa, Rap1]  
  
% load('data_Rap1_1para_noiseless.mat');  
load('data_Rap1_1para_noise.mat');  
  
%% k4を振って、残差2乗和(RSS)が最小となるときのk4を求める  
k4 = linspace(0.1, 10, 201);  
[x_Rap1_RSS] = landscape_Rap1(k4, t, x, param, y0); % t, xはデータ点の時間とRap1aの波形
```

# 手でRasのパラメータk6を求めてみる

データ(ノイズなし)"data\_Ras\_1para\_noise"をダウンロード  
<http://kurodalab.bi.s.u-tokyo.ac.jp/class/Summer/2015/>

"Ras\_model.m"をコピーして, "hand\_Ras\_1param.m"などとして改造  
外から, 引数でk6を与えるようにする.

Rap1モデルの時と同様に  
k6を手で色々変えて, データにもっともよく合う値を探してみる.

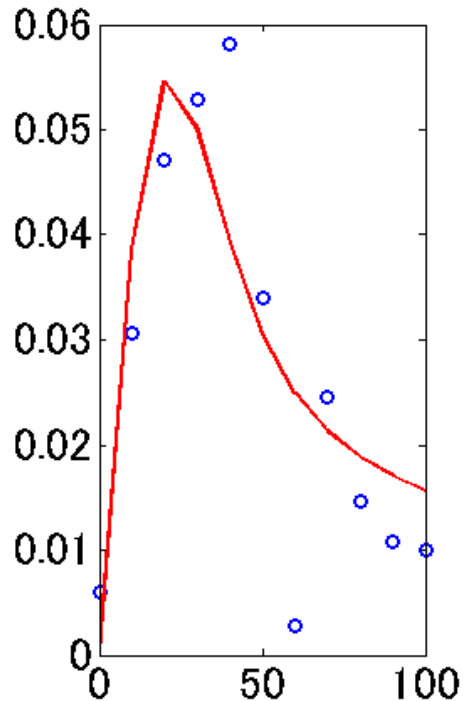


実行するときはコマンドウィンドウ上から「`hand_Ras_1param('数値')`」

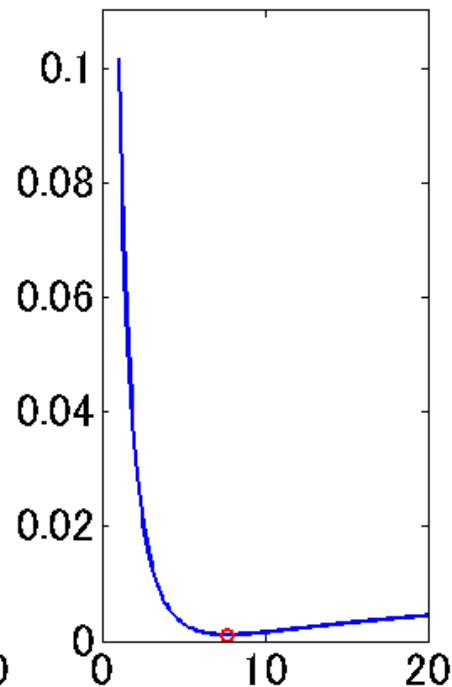
# Rasモデルのパラメータ $k_6$ を推定する(ノイズあり)

1. 評価関数の景観を書く.
2. 残差2乗和が最小となるときのパラメータ $k$ を求める.

データとODEの出力



評価関数の景観



手で求めた値と  
比べてどうか？

# “fit\_Ras\_1param.m”をダウンロードして空欄を埋める

## その1/3

```
function [k6_est] = fit_Ras_1param()
```

```
param = [0.5, 5, 0.0005, 0.005, 0.05, NaN]; % [k1, k2, k3, k4, k5, k6]の順に(k6は後で変えるので、仮にNaNを代入)
```

```
y0 = [1, 0, 0, 0]; % [S, GEFa, GAPa, Ras]
```

```
load('data_Ras_1para_noise.mat');
```

k6を1から10まで201分割した上で動かし  
残差2乗和(RSS)を計算

```
%% k6を振って、残差2乗和(RSS)が最小となるときのk6を求める
```

```
k6 = linspace(1, 20, 201);
```

```
[x_Ras_RSS] = landscape_Ras(k6, t, x, param, y0); % t及びxはデータ点の時間とRasの波形
```

```
[RSS_min, idx] =  % 最小値を探す関数minを使って、最小となるRSSとそのときの番号を求める
```

```
k6_est =  % RSSが最小となるときのk6を代入。k6は配列であることに注意。
```

```
%% 上で求めた残差2乗和(RSS)が最小となるときのk6を代入し、シミュレーション
```

```
temp_param = param;
```

```
temp_param() = ; % k6は、RSS最小となるときの値を代入
```

```
[t_sim,timeCourse_sim] = ode15s(@(t, y) ODE(t, y, temp_param), t, y0);
```

fit\_Ras\_1param()は次ページにも続く

# “fit\_Ras\_1param.m”をダウンロードして空欄を埋める

## その2/3

%% 描画

```
figure(1);
```

%% 時間波形

```
subplot(1, 2, 1)
```

```
plot(t, x, 'bo'); % データ点
```

```
hold on
```

```
plot(t_sim, timeCourse_sim(:, 4), 'r-'); % シミュレーションの結果
```

%% RSSとk6の関係

```
subplot(1, 2, 2);
```

```
plot(k6, x_Ras_RSS, 'b-');
```

```
hold on
```

```
plot(k6_est, RSS_min, 'ro');
```

```
end
```

# “fit\_Ras\_1param.m”をダウンロードして空欄を埋める

## その3/3

```
function [x_Ras_RSS] = landscape_Ras(k6, SamplingTime, x_Ras, param, y0)
```

```
    vary_x_Ras = zeros(length(x_Ras), length(k6)); % 各列毎に各k6でのRasの時間波形が入るような配列を準備
```

```
    for i = 1:length(k6)
```

```
        param( ) = ; % 変えているパラメータは？配列で与えられていることに注意
```

```
        [~, tc] = ode15s(@(t, y) ODE(t, y, param), SamplingTime, y0);
```

```
        vary_x_Ras(:, i) = tc(:, 4);
```

```
    end
```

```
    x_Ras_Residual = vary_x_Ras - x_Ras * ones(1, size(vary_x_Ras, 2));
```

```
    x_Ras_RSS = ; % 残差はx_Rap1_Residual。残差2乗和はどう表すか？
```

```
end
```

```
function dydt = ODE(t, y, param)
```

```
    % ここにモデルをコピー
```

```
end
```

# 議論1

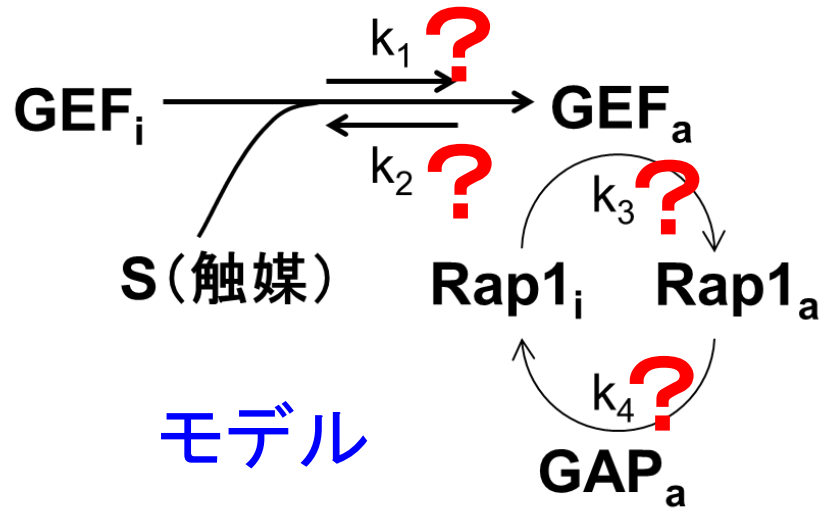
- 手と数値探索の結果は、どれぐらい一致しましたか？
- ノイズがデータにのると、パラメータの値はどうなりましたか？



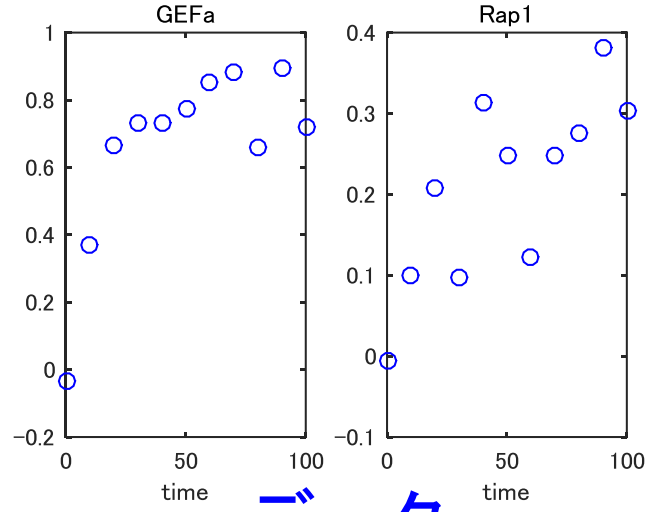
# 今日の流れ

1. Ras, Rap1のODEモデルを作成
2. パラメータ推定と最小2乗法について
3. ODEモデルの1つの未知パラメータを推定
4. ODEモデルの4つの未知パラメータを推定

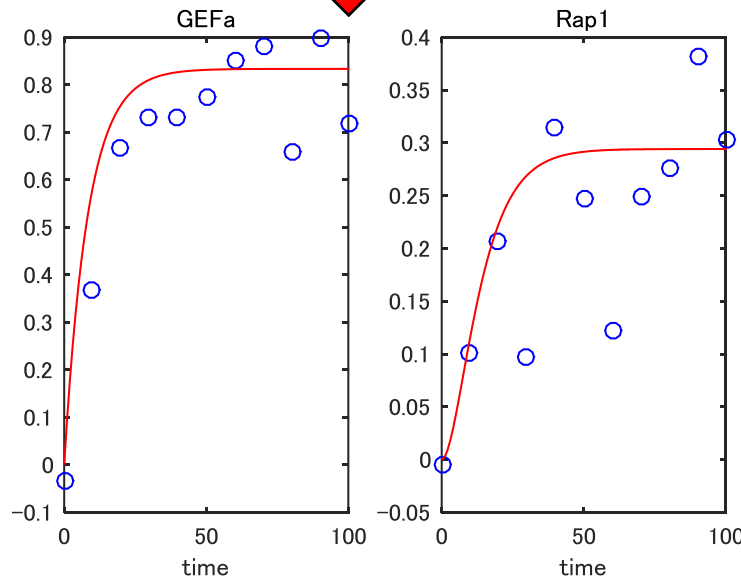
# 手でRap1の4パラメータを合わせる？



モデル



データ



k1~k4を手で色々変えて、データにもっともよく合う値を探してみる。

k1~k4を推定

※ 手で合わせるのが大変なことを確認する程度でよい。真面目に合わせる必要はない。

# 手でRap1の4パラメータを合わせる？

データ(ノイズあり) "data\_Rap1\_4para.mat" をダウンロード

"hand\_Rap1\_1param.m" をコピーして, "hand\_Rap1\_4param.m" などとして改造する.

```
function hand_Rap1_4param(k)
```

```
    param = ; % [k1, k2, k3, k4]の順に
```

```
    y0 = [1, 0, 0.005, 0]; % [S, GEFa, GAPa, Rap1]の順に
```

```
    load('data_Rap1_4para.mat');
```

```
    [t_sim, timeCourse_sim] = ode15s(@(t, y) ODE(t, y, param), time, y0);
```

```
    figure;
```

```
    for i = 1:2
```

```
        subplot(1, 2, i);
```

```
        plot(t, x(:, i*2), 'bo'); % データ点の表示
```

```
        hold on;
```

```
        plot(t_sim, timeCourse_sim(:, i * 2), 'r-'); % シミュレーション結果の表示
```

```
    end
```

```
    xlabel('time');    ↑表示部分を追加
```

```
    (略)
```

```
end
```

実行時: hand\_Rap1\_4param([数値, 数値, 数値, 数値])

※ 手で合わせるのが大変なことを確認する程度でよい。真面目に合わせる必要はない。

# パラメータの数が多いと推定が大変...

パラメータの数が少ない → for loopでしらみつぶし

パラメータ数が多い → しらみつぶしに探すのは難しい

単純に、パラメータの数だけ for loopを回すと... ?

→ (パラメータ候補の数)<sup>(パラメータ数)</sup>: 膨大な回数

**どうやってデータの再現性の高い**

**(評価関数の値が小さい)**

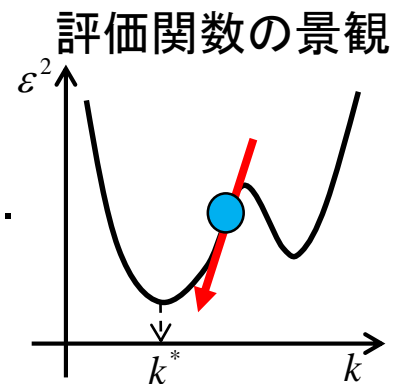
**パラメータの候補を探すか？**

# 評価関数の最適化方法はいろいろあります

## ・勾配 (微分) を用いる方法

ニュートン法, 共役勾配法など.

拘束条件の有無によってアプローチが異なってくる.  
凸最適化問題 (局所的な最小と大域的な最小が一致する問題) に向いている.



## ・確率を用いる方法

- シミュレーテッドアニーリング (SA)

熱力学的な焼きなましの過程をまねた方法

- 遺伝的アルゴリズム (GA) 交叉 + 突然変異

- 進化的プログラミング (EP) 主に突然変異

様々な最適化問題に対して上手く適用できることが  
経験的に知られている方法

# Evolutionary programming (EP)

Evolutionary programming (EP, 進化的計算)

- 最適化アルゴリズムの一種
- 生物学における自然淘汰を模して作られている
- メタヒューリスティック((ある程度)汎用性のある経験的な)

ここでは、残差2乗和が最小となるようなODEのパラメータを得るためにEPを使う。

## ○大雑把な説明

- 1) パラメータを個体とみなして、最初に $N$ 個の個体がある
- 2) 各個体が子をつくる。各子には確率的な変異が入る  $\Rightarrow 2N$ 個体
- 3) 残差2乗和を小さいパラメータ  $\rightarrow$  優れている個体  $\rightarrow$  生き残る  
残差2乗和が大きいパラメータ  $\rightarrow$  劣っている個体  $\rightarrow$  淘汰  
 $\rightarrow$  個体数を $N$ 個にする。
- 4) 2)と3)を十分長く繰り返す
- 5) 最終的に最も優れた個体を、パラメータとして採用

# もう少し詳細なEPのアルゴリズム

1. 乱数を初期値として,  $N$  個の個体(解の候補)を生成する.
2. 個体のそれぞれのコピーをつくる.
3. 2.で作った各コピーに正規乱数を加える.(変異)
4. 3.の操作で作られた新しい  $N$  個の個体と, 元の個体を混ぜた  $2N$  個の各個体に対してスコアを求める. スコアは以下のよう<sup>に</sup>決める.
  - a. スコアを決める対象の個体を除いて,  $2N$ 個体の集団から無作為に  $q$  個体を選ぶ.
  - b. 選んだ  $q$  個体のうち, スコアを決める対象の個体より評価関数の値(RSSなど)が悪い個体(最小2乗法の場合は, 対象の個体よりRSSの値が大きい個体)の数をその個体のスコアとする.
5. スコアの下位  $N$  個の個体を削除する.(淘汰)
6. 終了条件を満たすまで2. ~7.(1世代に相当)の操作を繰り返す.
7. 残った中で最も評価関数の値が良い個体を“解”とする.

※EPには, バリエーションがいくつか存在する.

# “fitEP\_Rap1\_4param.m”をダウンロードして空欄を埋める

## その1/5

```
function [res_bestIndv, res_bestScore] = fitEP_Rap1_4para(n, numGeneration)
% Rap1モデルのパラメータをEPで推定するモデル。
% 引数のnは系列数。numGenerationは世代数。

res_bestIndv = zeros(n, 4); % 結果(パラメータ)を格納する変数
res_bestScore = zeros(n, 1); % 結果(RSS)を格納する変数

for i = 1:n
    % 各系列で最小のRSSとなるパラメータを探し出し、変数に格納
    [res_bestIndv(i, :), res_bestScore(i, :)] = work_EP_Rap1(numGeneration);
end
end
```

**copasiep.m**もダウンロード

<http://kurodalab.bi.s.u-tokyo.ac.jp/class/Summer/2015/>



# “fitEP\_Rap1\_4param.m”をダウンロードして空欄を埋める

## その2/5

```
function [bestIndv, bestScore, report] = work_EP_Rap1(numGeneration)
    y0 = [1, 0, 0.005, 0]; % [S, GEFa, GAPa, Rap1]
    param = zeros(1, 4);
    load('data_Rap1_4para.mat');

    %%% EPパートここから %%%
    currentTime = clock; % 現在の時刻を取得
    second = currentTime(6); % 現在の時刻から'秒'を取得

    strSeed = '2015';
    seed = str2num(strSeed)*second*1000; % 乱数の種に'秒'をかけることで、よりランダムに
    try RandStream.setGlobalStream(RandStream('mt19937ar', 'Seed', seed)); % 乱数の初期設定
    catch; RandStream.setDefaultStream(RandStream('mt19937ar', 'Seed', seed)); % ver. 2012以前
    end

    lb = 1e-3 * ones(size(param)); % パラメータ探索範囲の下限
    ub = 1e+2 * ones(size(param)); % パラメータ探索範囲の上限
    numParents = 20; % 子を生成する親の数
    [bestIndv, bestScore, report] = copasiep(@(param) cal_RSS(x, t, param, y0), ...
        numParents, numGeneration, lb, ub); % copasiのEPでパラメータサーチ
    %%% EPパートここまで %%%
```

# “fitEP\_Rap1\_4param.m”をダウンロードして空欄を埋める

## その3/5

```
%% 各系列で最小のRSSのときのパラメータを用いてシミュレーション
```

```
time = [0, 100];
```

```
[t_est, x_est] = ode15s(@(t, y) ODE(t, y, bestIndv), time, y0);
```

```
% データ点とシミュレーション結果の重ね描き
```

```
figure(1);
```

```
for i = 1:2
```

```
    subplot(1, 2, i);
```

```
    hold on
```

```
    plot(t, x(:, i*2), 'o'); % データ点
```

```
    hold on
```

```
    plot(, 'r'); % シミュレーション結果
```

```
    if i == 1
```

```
        title('GEF_a');
```

```
    else
```

```
        title('Rap1');
```

```
    end
```

```
end
```

# “fitEP\_Rap1\_4param.m”をダウンロードして空欄を埋める

## その4/5

% EPでRSSが減少して行く様子を描画

```
figure(2);
```

```
subplot(1, 2, 1);
```

```
hold on
```

```
plot(report(:, 1), 'b-');
```

```
title('最も良い個体');
```

```
xlabel('世代数');
```

```
ylabel('残差2乗和');
```

```
subplot(1, 2, 2);
```

```
hold on
```

```
plot(report(:, 2), 'b:');
```

```
title('全個体の平均');
```

```
xlabel('世代数');
```

```
ylabel('残差2乗和');
```

```
end
```

# “fitEP\_Rap1\_4param.m”をダウンロードして空欄を埋める

## その5/5

```
function [RSS] = cal_RSS(x_data, SamplingTime, param, y0)
    % 残差2乗和を計算
    [~, tc] = ode15s(@(t, y) ODE(t, y, param), SamplingTime, y0);
    residual = tc - x_data; % 残差
    RSS = ; % 残差2乗和
end
```

```
function dydt = ODE(t, y, param)
    % ここにモデルをコピー
```

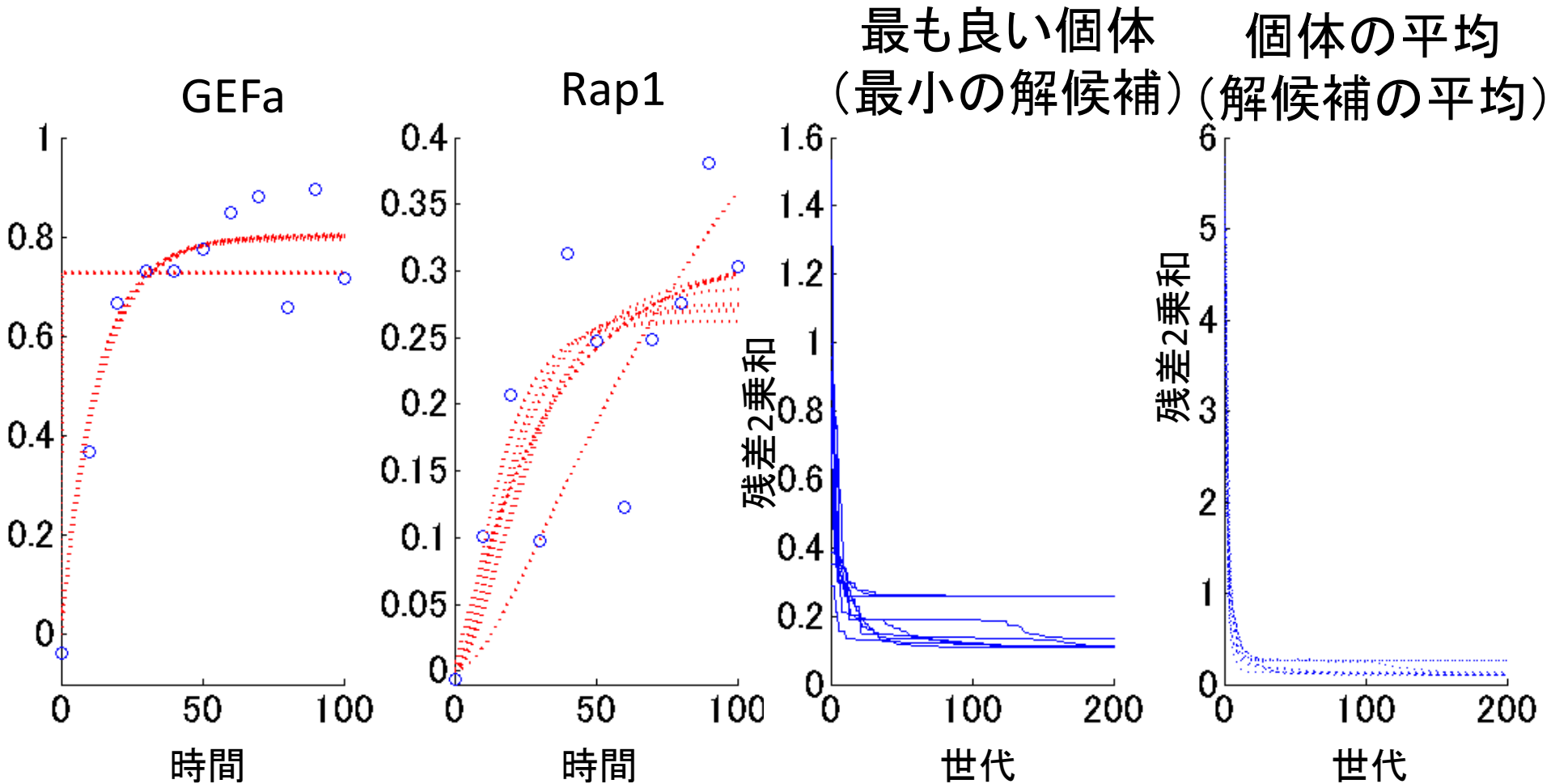
```
end
```

# 注意点・補足事項

- 1次元空間の探索は、しらみつぶしにやるより効率的な方法がある。⇒ 黄金比探索法など。
- 手元のデータを最もよく再現できるパラメータが、最もよいパラメータとは限らない。(なぜか?)
- 今回は、どのデータにどのモデルを使うかは予めわかっていた。しかし、現実問題としては、そのようなことは、文献などからその系について事前に信頼性の高い情報がある場合に限られる。データからモデルを決めるには、統計的モデル選択規準がある。

# 実行結果の例

回数:  $n=10$ , 世代数:  $\text{numGeneration}=200$ などとして, 実行.



ワークスペースの変数をクリックして, "res\_bestIndv", "res\_bestScore"をみて, 全体で最もよい解を確認する. (※実行時に出力を格納するようにする.)

## 議論2

- パラメータはどれくらい合っていたか？
- 実行ごとに、結果が異なるのはなぜか？
- EPで、残差2乗和が変化しなくなったら、計算をやめてもよいか？
- パラメータの数の増加に対して、解を探索すべき対象となる領域は、どのように増えるか？
- そもそも、なぜEPを使うのか？

# 発展課題



# 発展課題1

- RasモデルでRap1データをfitしてみる
  - Rap1モデルとRasモデルの関係は、どうなっているか？
  - Rap1モデルでRap1データをfitした結果と、RasモデルでRap1データをfitした結果を比べて、どのような違いがあるか？
  - ノイズのある有限データから、RasモデルとRap1モデルを判別するにはどうすればよいか？

## 発展問題2

- Rap1モデル, RasモデルのODEを手で解いて, 解析解を求める.

# 発展問題3:オイラー法

(特に時間が余った人向け, 本編とは直接関係ない問題です.)

☆ODEを数値的に解く非常に簡単な方法:

$$\frac{dx}{dt} = f(x, t) \approx \frac{x_{n+1} - x_n}{h}$$

と近似して,

$$x_{n+1} \approx x_n + hf(x_n, t_n)$$

を計算する解法があり, オイラー法と呼ばれている.

オイラー法を実装して, 1次反応モデルなどの簡単なODEを解いてみる.

※数値解法には様々な手法があり, 問題に応じて適切な数値解法を用いる必要がある. オイラー法はODEの数値解法の初歩を学ぶためには有用だが, 実用性は高くない.