Pre-tutorial instructions

1. Please download and unzip the `Tutorials/Day5.../files’ folder from the google drive

2. Launch matlab and make the `files’ folder your current directory

Tutorial:

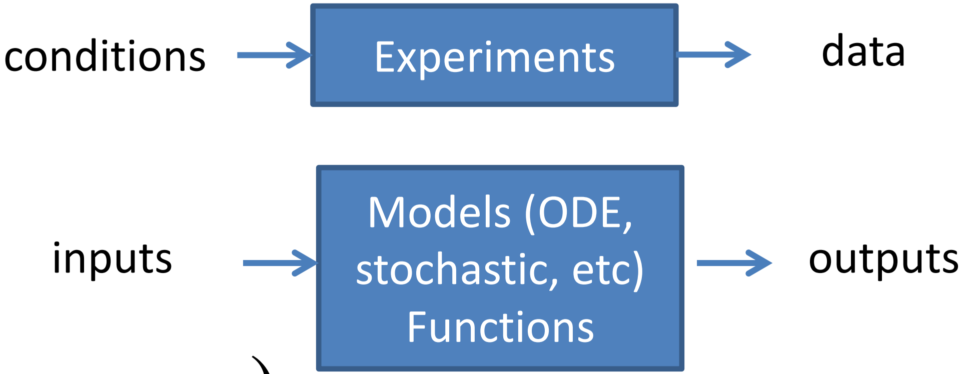
Parameter Fitting and Sensitivity Analysis

Meng Jin

Mar, 2017

Overview

1. Parameter fitting: finding values of unknown parameters that reproduce observed data



$$X = (x_1, \ldots\ldots, x_n)$$
$$\vec{p} = (p_1, \ldots, p_m)$$

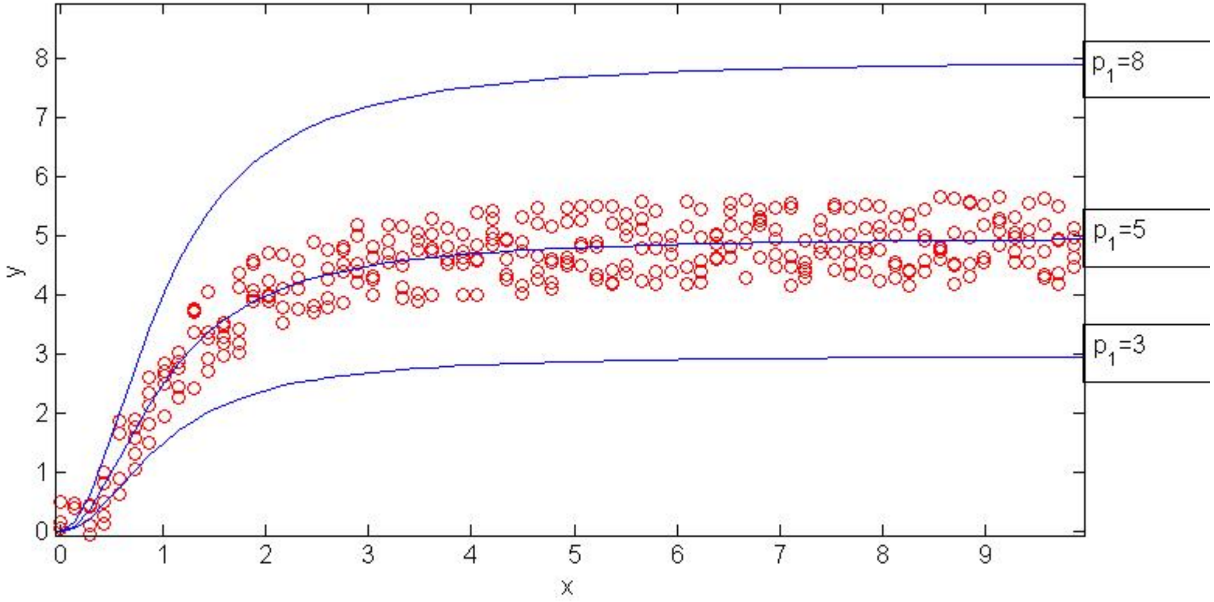
$$F(X, \vec{p})$$

$$Y = F(X, \vec{p})$$

Overview

2. Sensitivity analysis: measuring how strongly parameters affect outputs

$$y = F(x, \vec{p}) = \frac{p_1 x^{p_2}}{p_3^{p_2} + x^{p_2}}$$



Overview
Steps of parameter fitting

1. Select an appropriate mathematical model to describe observed experimental data

   what you have learned in previous tutorials

2. Define an “error” function

   which measures the agreement between experimental data and model results for given parameters

3. Adjust model parameters to get a “best fit”

   often involves minimizing the error function

4. Evaluate “goodness of fit” to experimental data

   (bootstrap, cross-validation)

   never perfect due to measurement noise

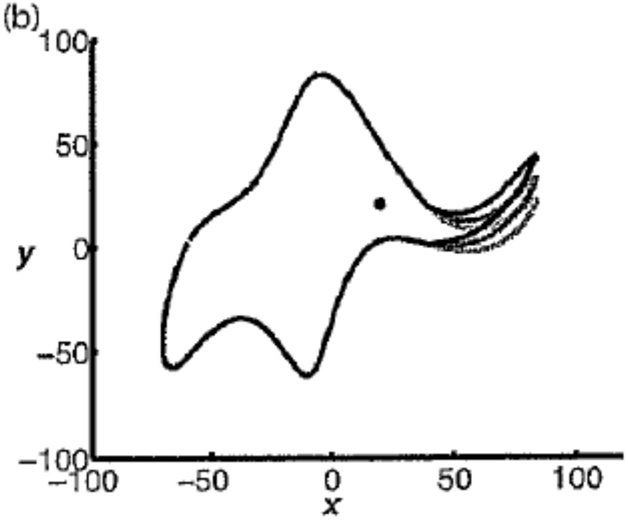
5. Estimate how parameters affect the model prediction

   (parameter sensitivity)

Parameter fitting

”With four parameters I can fit an elephant. With five I can make him wiggle his trunk.”

-John von Neumann



Parameter fitting

”With four parameters I can fit an elephant. With five I can make him wiggle his trunk.”

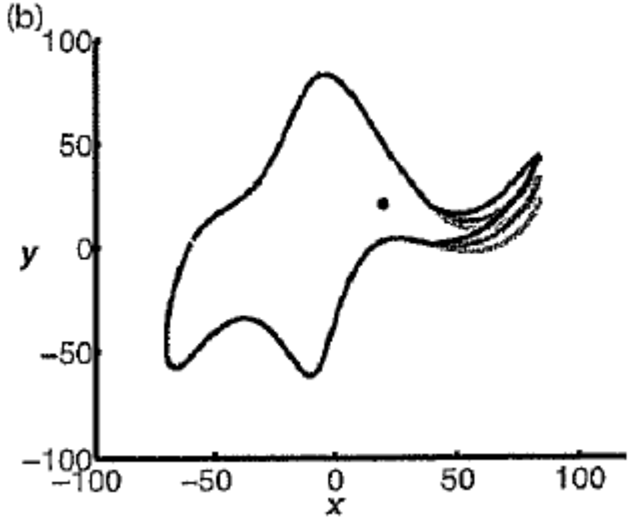
-John von Neumann

Aside:

$$x(t) = \sum_{k=0}^{\infty} (A_k^x \cos(kt) + B_k^x \sin(kt)),$$

$$y(t) = \sum_{k=0}^{\infty} (A_k^y \cos(kt) + B_k^y \sin(kt)),$$

Mayer, J., et al. Am. J. Phys. 78 (6). 2010



Parameter fitting

”With four parameters I can fit an elephant. With five I can make him wiggle his trunk.” -John von Neumann

- Given <u>enough</u> parameters, one can always fit data (overfitting)

- Goal is a <u>minimal</u> yet **predictive** model
  - Can make verifiable predictions for further experiments
  - Can be cross-validated on subsets of data

Parameter fitting

- The ”art” of hand-fitting
- Use optimization algorithms to find values of parameters that best reproduce a set of data

Fitting methods

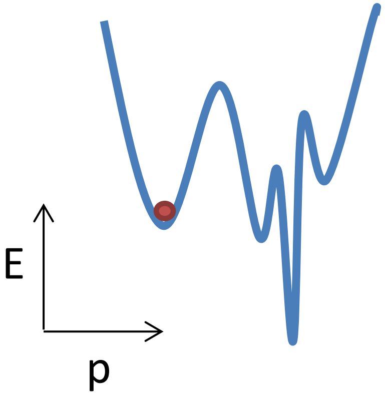
1. Define error function, E, to determine goodness of fit to data
   Ex.: $E(p)=\text{mean}((x_e(i)-x_m(i,p))^2)$

1. Then find min(E(*p*)) by taking steps in *p* space:

a) Deterministic fitting: take defined steps in *p*

– Fast, but can get stuck in local minimum

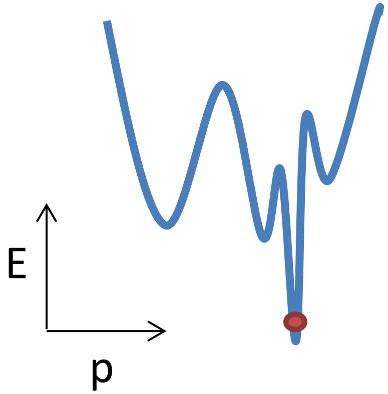
Ex.: parameter scan, steepest descent,
Newton methods, Levenberg –Marquardt, etc.



b) Stochastic fitting: take noisy steps in *p*

– Will find global minimum, given enough time

Ex.: random search, evolution algorithms,
Metropolis, simulated annealing, particle swarm, etc.



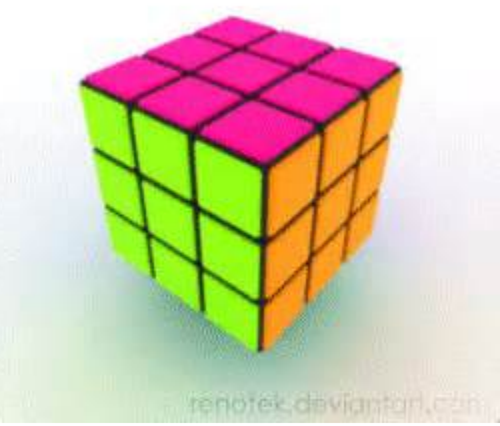
Deterministic fitting methods

- Parameter scan:

equal size steps through sections of parameter space

  - Fine with few parameters and coarse sampling

```
for p1=1..
  for p2=1..
    for p3=1..
      E(p1,p2,p3)= ...
    end
  end
end
```

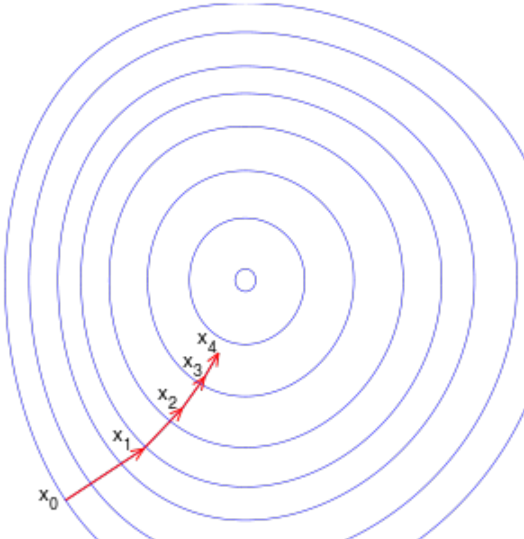


- Steepest (gradient) descent:

direction of next step is proportional to negative to gradient of E

  - Can take many steps near a minimum, and can get stuck at local minima

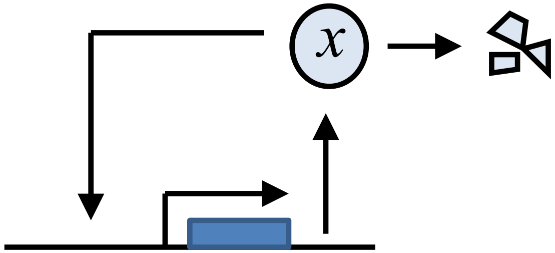
```
for i=1..
  p1=p1-dE/dp1
  ...
end
```



http://en.wikipedia.org/wiki/File:Gradient_descent.svg

Example: simple negative feedback

data

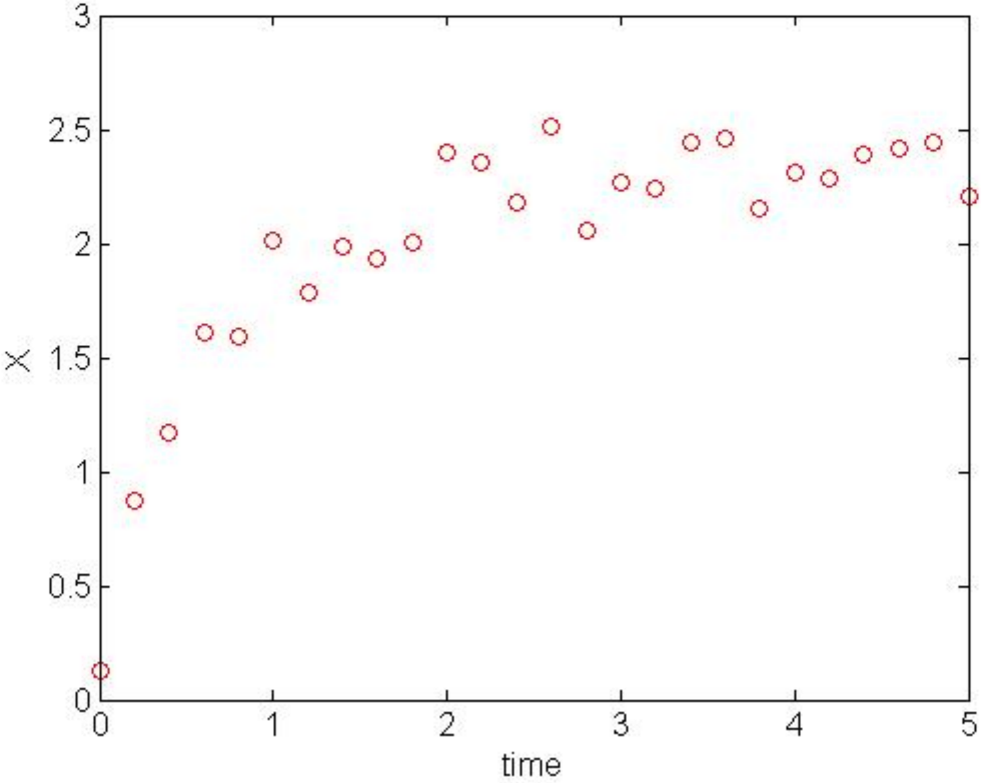


$$\dot{x} = \left(\frac{\alpha}{1 + \mathrm{K}x} \right) - \beta x$$

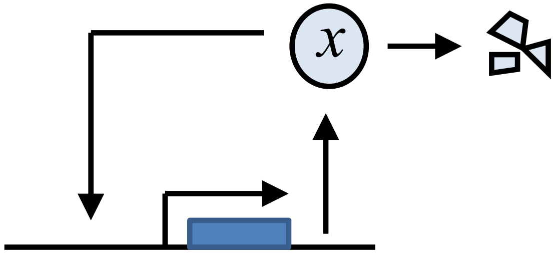
$$\beta = 1$$

$$\alpha = ?$$

$$K = ?$$



Example: simple negative feedback
guess

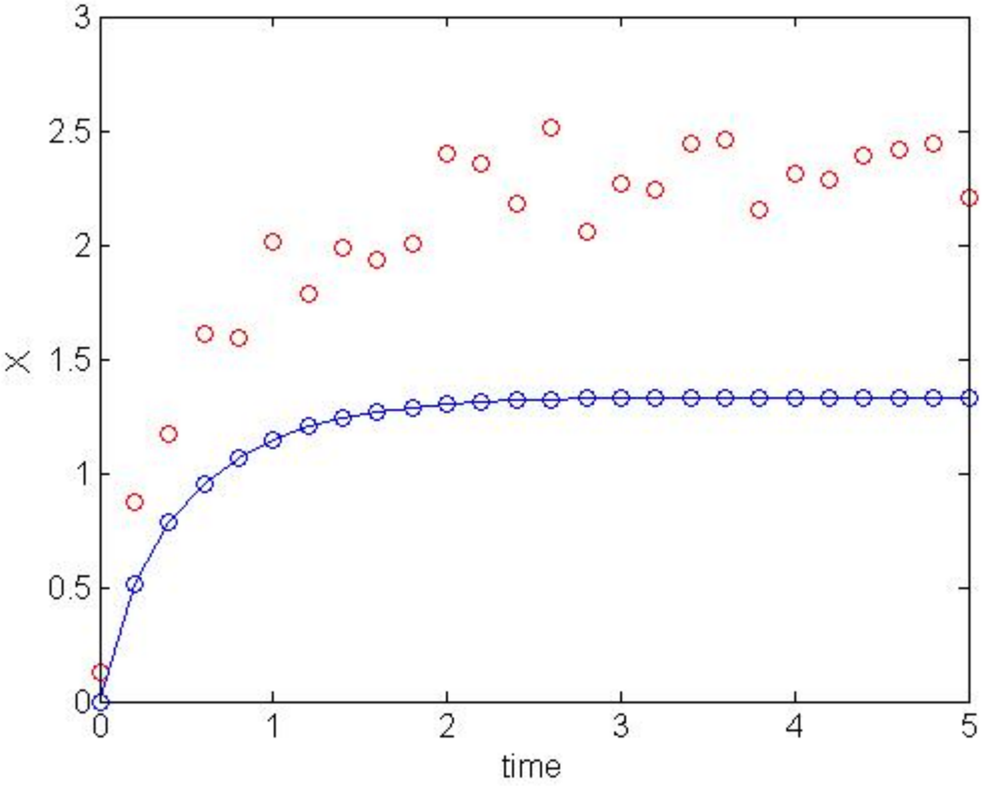


$$\dot{x} = \left(\frac{\alpha}{1+\mathrm{K}x}\right) - \beta x$$

$\beta = 1$

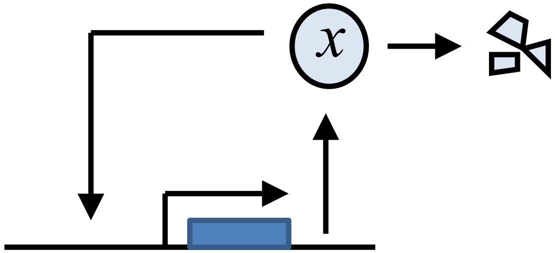
$\alpha = 4$

$K = 1.5$



NFB_simple_run.m

Example: simple negative feedback error function



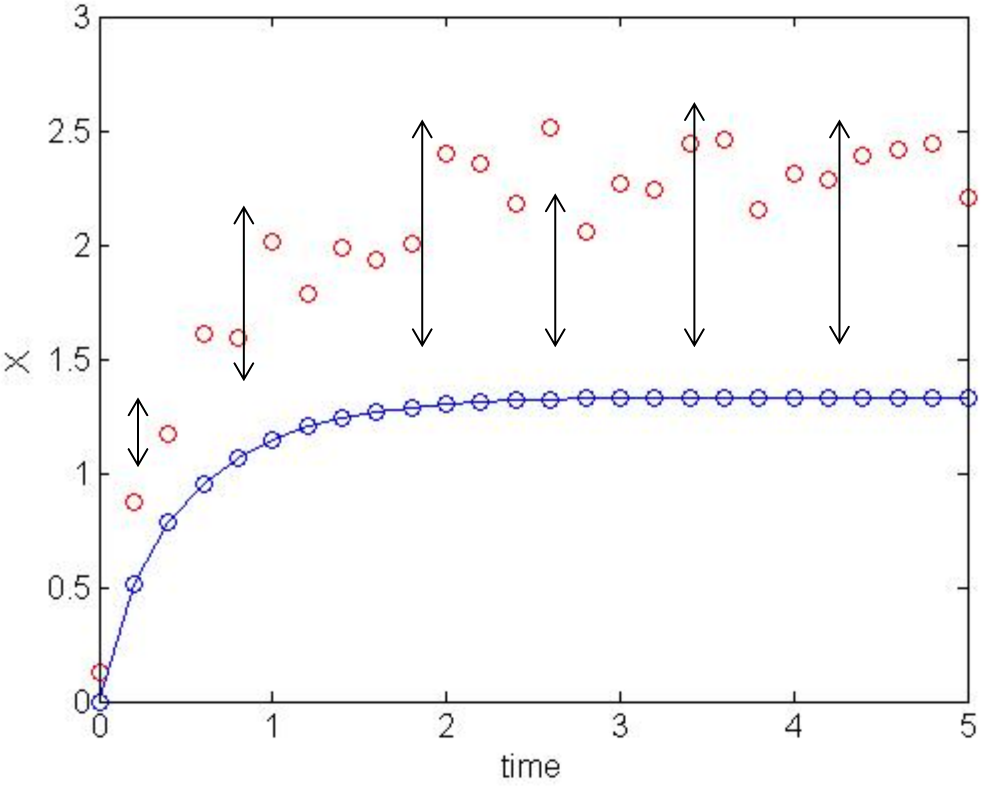
$$E(\vec{p}) = \sum_t (x_e - x_m(\vec{p}))^2$$

$$\dot{x} = \left(\frac{\alpha}{1+\mathrm{K}x}\right) - \beta x$$

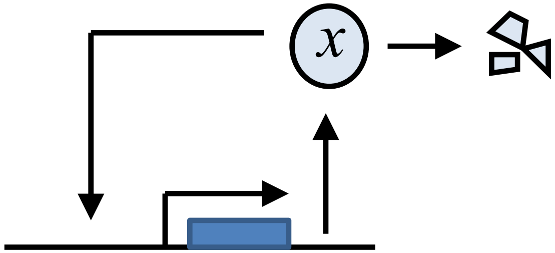
$$\beta = 1$$

$$\alpha = 4$$

$$K = 1.5 \quad \Rightarrow E = 19.6$$



Example: simple negative feedback parameter sweep



$$\dot{x} = \left(\frac{\alpha}{1 + \mathrm{K}x} \right) - \beta x$$

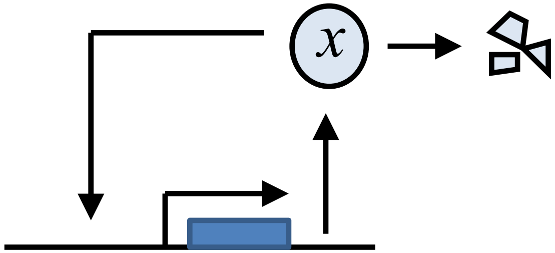
$$\beta = 1$$

$$\alpha = [0.1, 10]$$

$$K = [0.01, 1]$$

NFB_simple_paraSweep.m

Example: simple negative feedback
parameter sweep



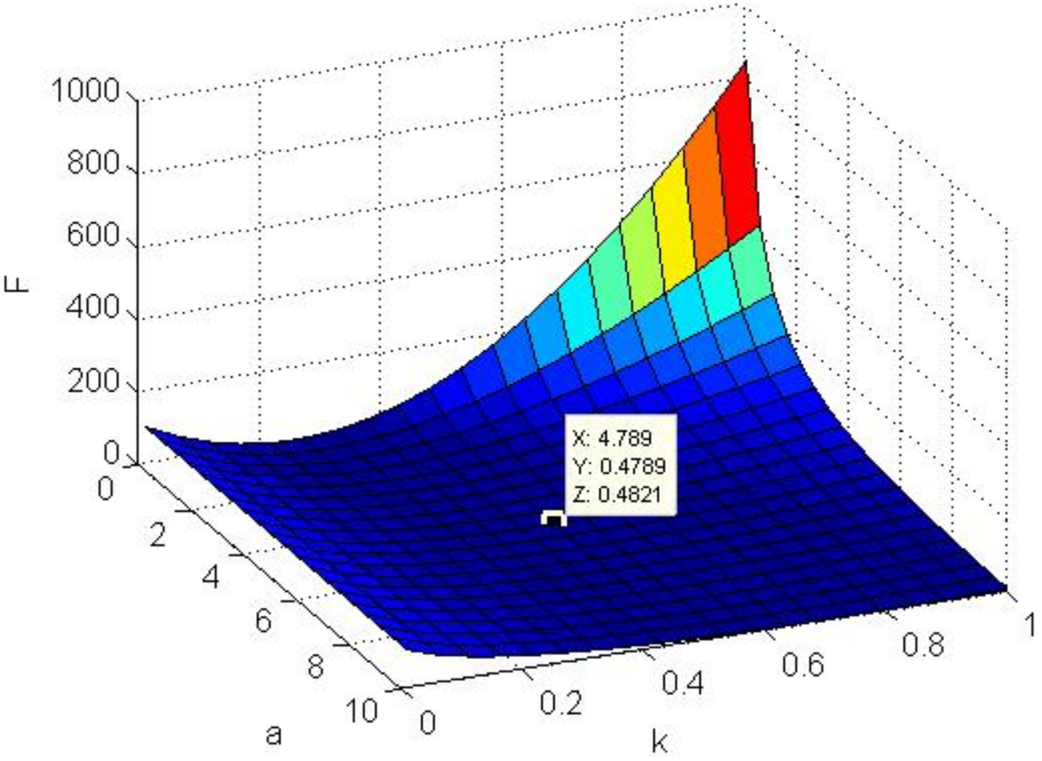
$$E(4.8,0.48) = 0.48$$

$$\dot{x} = \left(\frac{\alpha}{1+\mathrm{K}x}\right) - \beta x$$

$$\beta = 1$$

$$\alpha = [0.1,10]$$

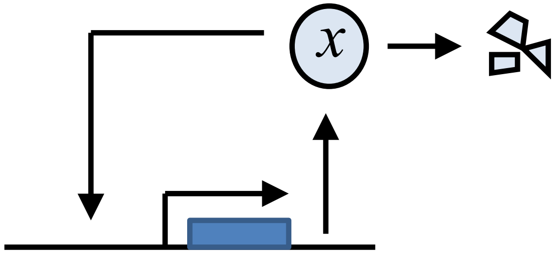
$$K = [0.01,1]$$



NFB_simple_paraSweep.m

Example: simple negative feedback

one good set of parameters found by parameter sweep

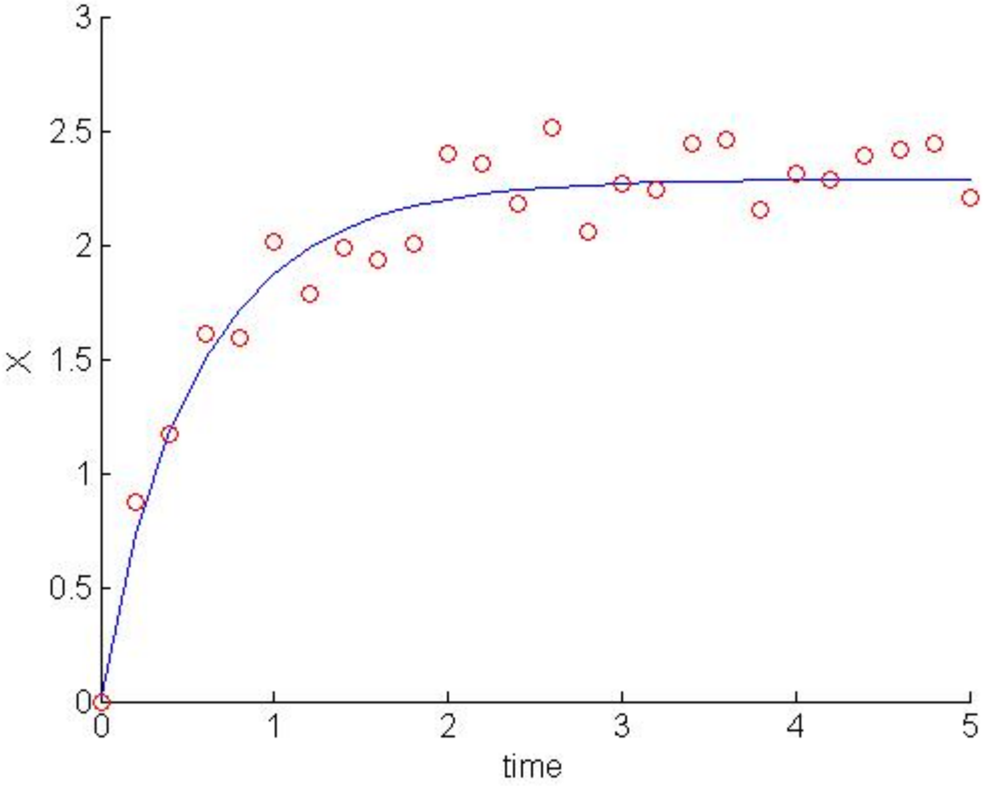


$$\dot{x} = \left(\frac{\alpha}{1+\mathrm{K}x}\right) - \beta x$$

$$\beta = 1$$

$$\alpha = 4.79$$

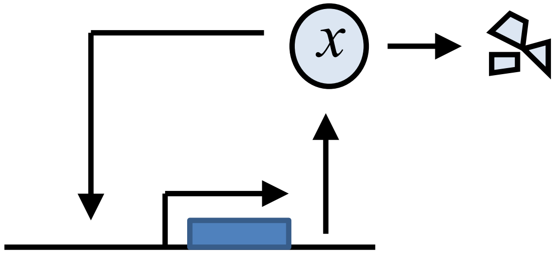
$$K = 0.479$$



NFB_simple_paraSweep.m

Example: simple negative feedback

steepest descent



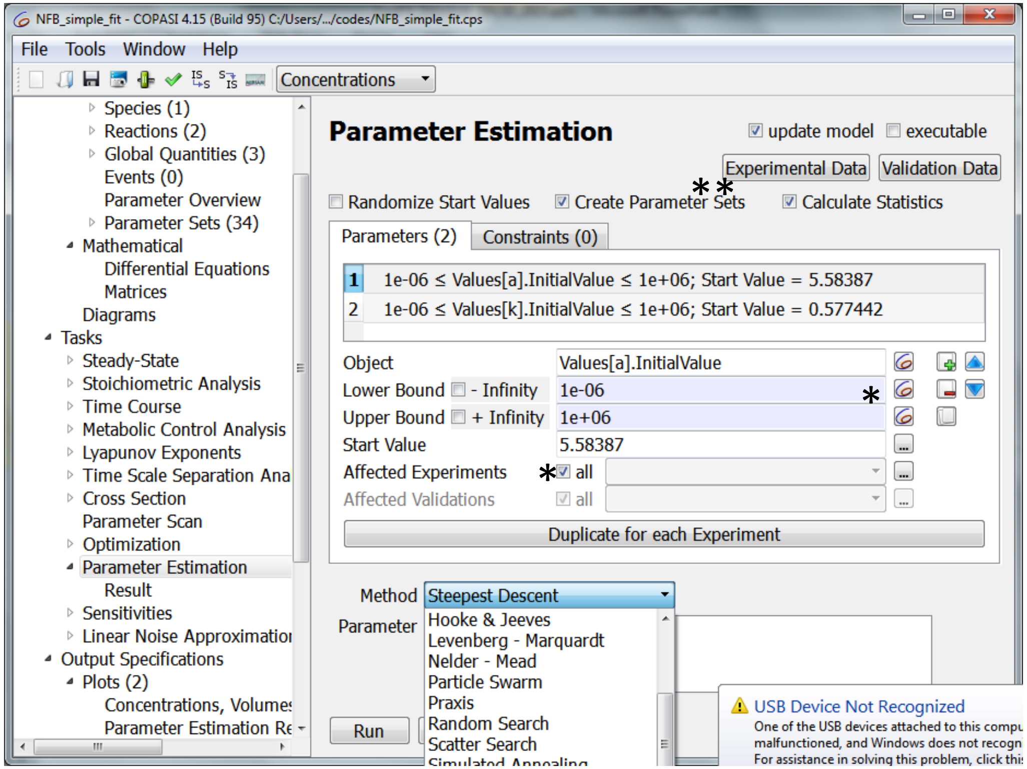
$$\dot{x} = \left(\frac{\alpha}{1+Kx}\right) - \beta x$$

$$\beta = 1$$

$$\alpha = ?$$

$$K = ?$$

- *Set parameters with initial value [a,k]=[10,1]
-**Ensure that experimental data (NFB_simple_data_cop.txt) is specified correctly



Parameter Estimation

☑ update model ☐ executable

☐ Randomize Start Values ☑ Create Parameter Sets ☑ Calculate Statistics

1 1e-06 ≤ Values[a].InitialValue ≤ 1e+06; Start Value = 5.58387

2 1e-06 ≤ Values[k].InitialValue ≤ 1e+06; Start Value = 0.577442

Object: Values[a].InitialValue

Lower Bound ☐ - Infinity: 1e-06

Upper Bound ☐ + Infinity: 1e+06

Start Value: 5.58387

Affected Experiments ☑ all

Affected Validations ☑ all

Method: Steepest Descent

Parameter

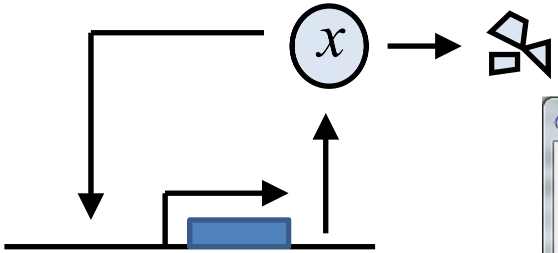
USB Device Not Recognized

One of the USB devices attached to this compu
malfunctioned, and Windows does not recogn
For assistance in solving this problem, click thi

NFB_simple_fit.cps

Example: simple negative feedback

steepest descent



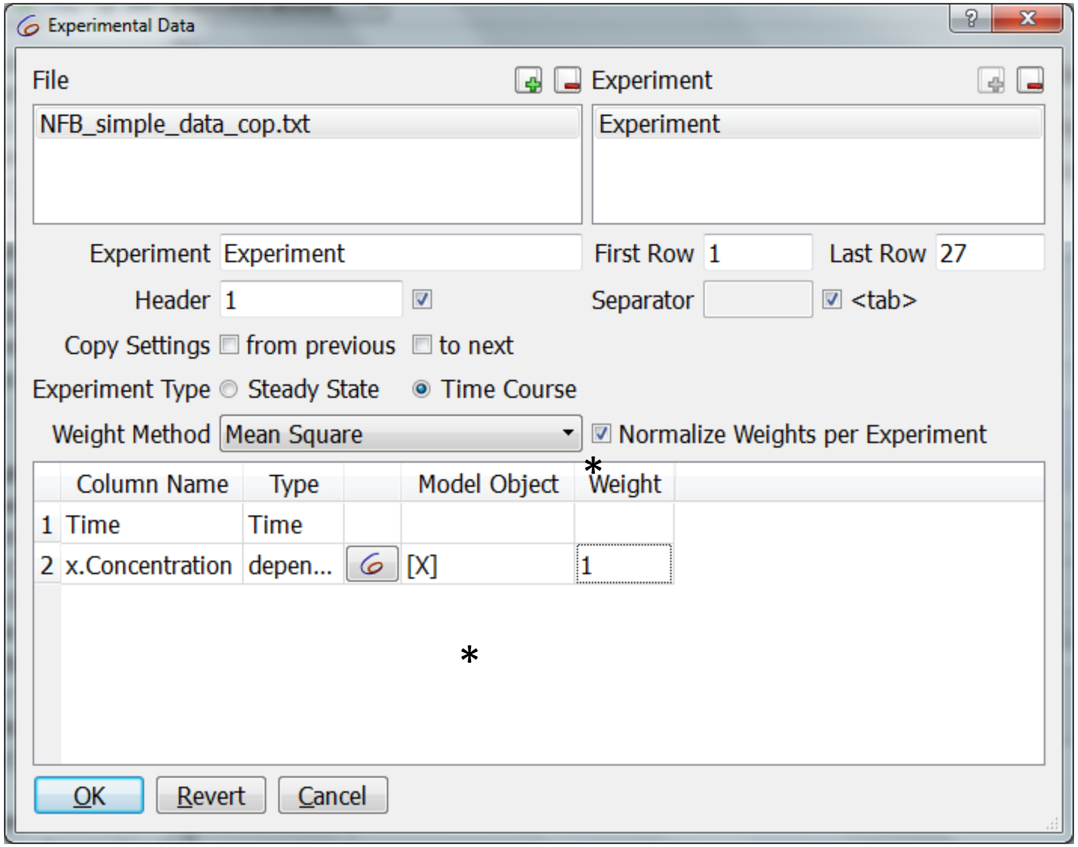
-Ensure that experimental data (NFB_simple_data_cop.txt) is specified correctly

$$\dot{x} = \left(\frac{\alpha}{1+\mathrm{K}x}\right) - \beta x$$

$$\beta = 1$$

$$\alpha = ?$$

$$K = ?$$



File: NFB_simple_data_cop.txt

Experiment: Experiment

Experiment: Experiment — First Row: 1 — Last Row: 27

Header: 1 ☑ — Separator: ☑ <tab>

Copy Settings: □ from previous □ to next

Experiment Type: ○ Steady State ● Time Course

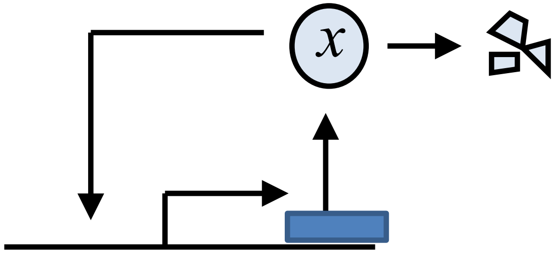
Weight Method: Mean Square — ☑ Normalize Weights per Experiment

	Column Name	Type		Model Object	* Weight
1	Time	Time			
2	x.Concentration	depen...		[X]	1

*

NFB_simple_fit.cps

Example: simple negative feedback
steepest descent



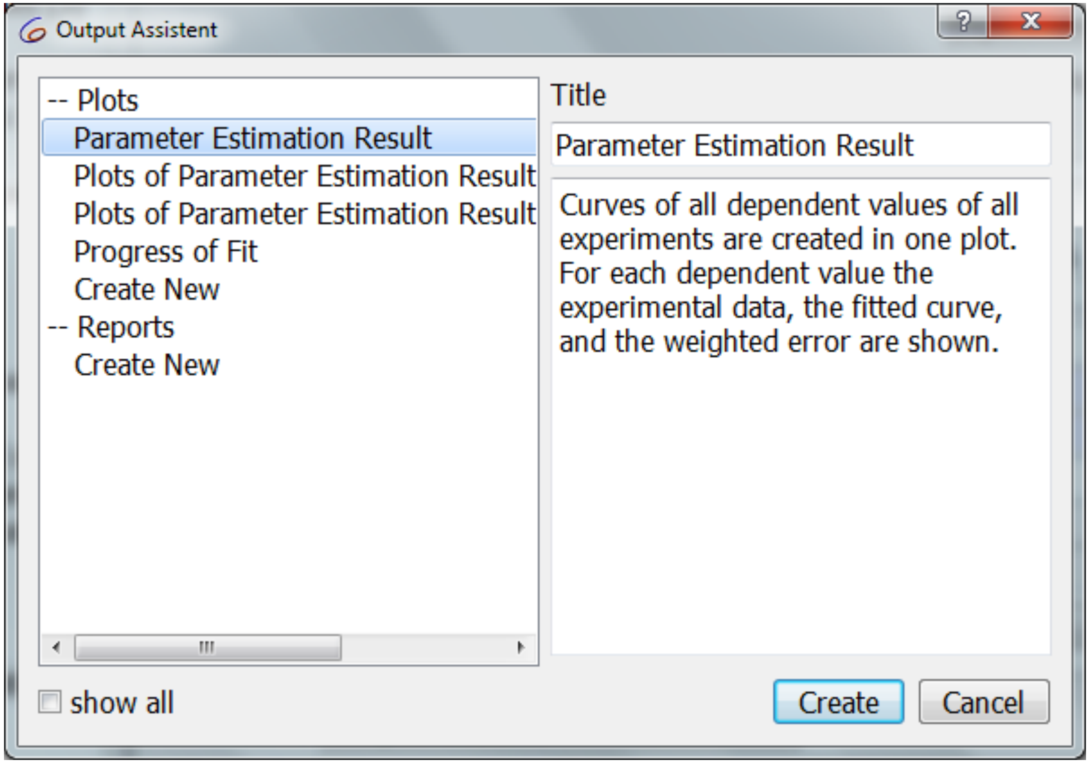
$$\dot{x} = \left(\frac{\alpha}{1+\mathrm{K}x}\right) - \beta x$$

$$\beta = 1$$

$$\alpha = ?$$

$$K = ?$$

-set up plot in output assistant



Output Assistent

-- Plots
Parameter Estimation Result
Plots of Parameter Estimation Result
Plots of Parameter Estimation Result
Progress of Fit
Create New
-- Reports
Create New

Title: Parameter Estimation Result

Curves of all dependent values of all experiments are created in one plot. For each dependent value the experimental data, the fitted curve, and the weighted error are shown.

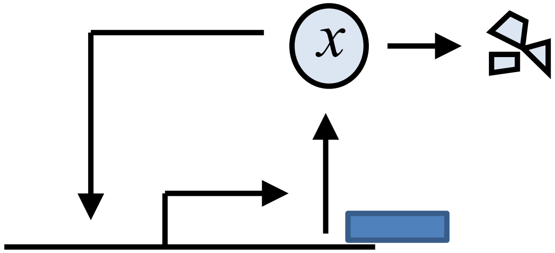
□ show all

Create Cancel

NFB_simple_fit.cps

Example: simple negative feedback

steepest descent

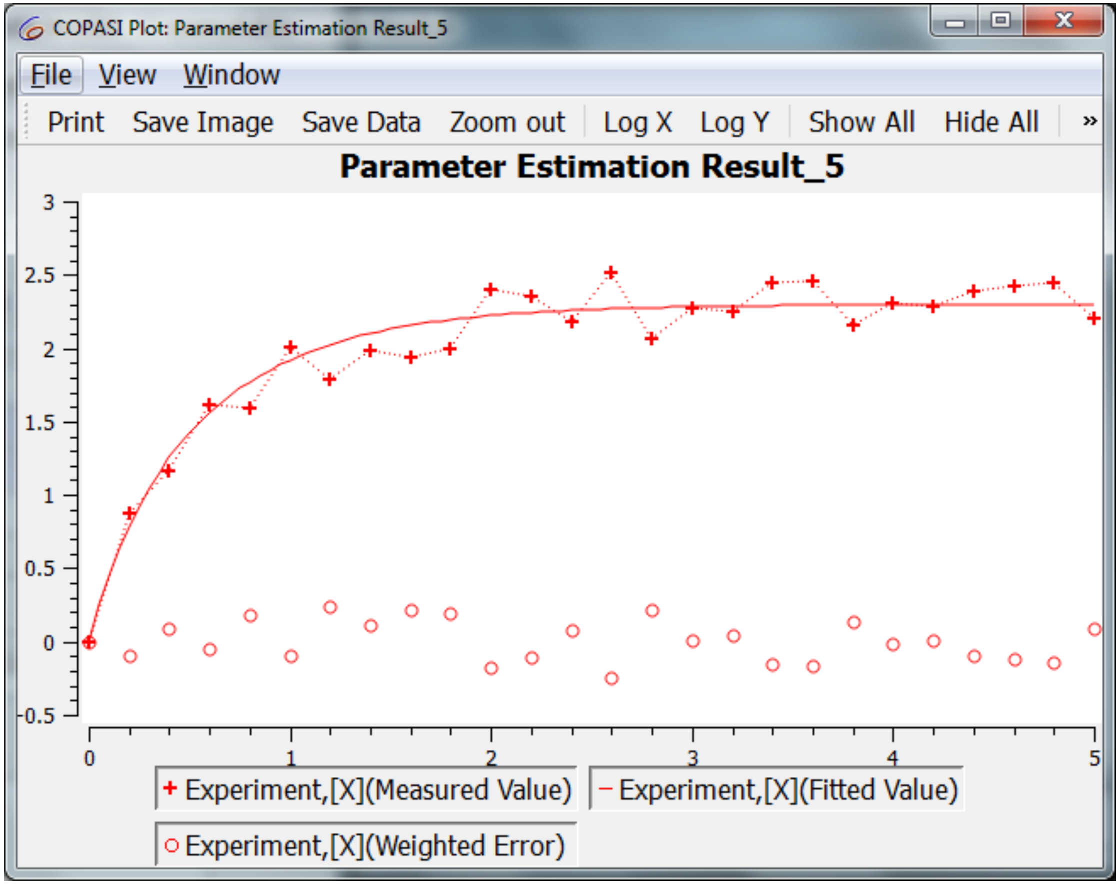


$$\dot{x} = \left(\frac{\alpha}{1 + \mathrm{K}x} \right) - \beta x$$

$\beta = 1$

$\alpha = 5.41$

$K = 0.589$

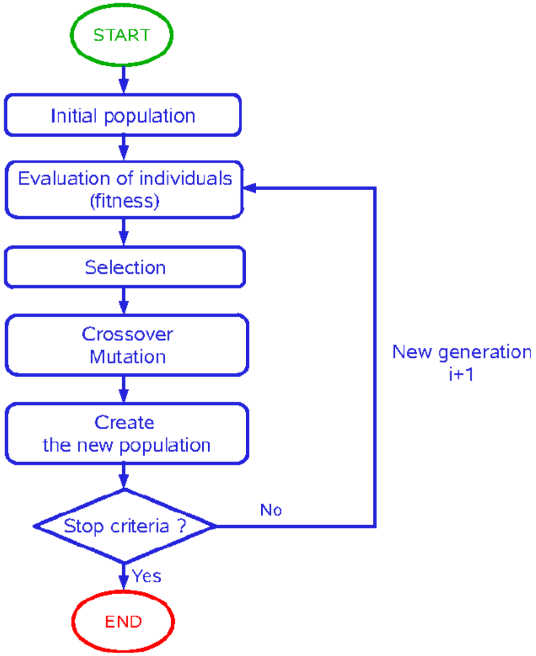


COPASI Plot: Parameter Estimation Result_5
File
View
Window
Print
Save Image
Save Data
Zoom out
Log X
Log Y
Show All
Hide All
»
Parameter Estimation Result_5
3
2.5
2
1.5
1
0.5
0
-0.5
0
1
2
3
4
5
+ Experiment,[X](Measured Value)
− Experiment,[X](Fitted Value)
○ Experiment,[X](Weighted Error)

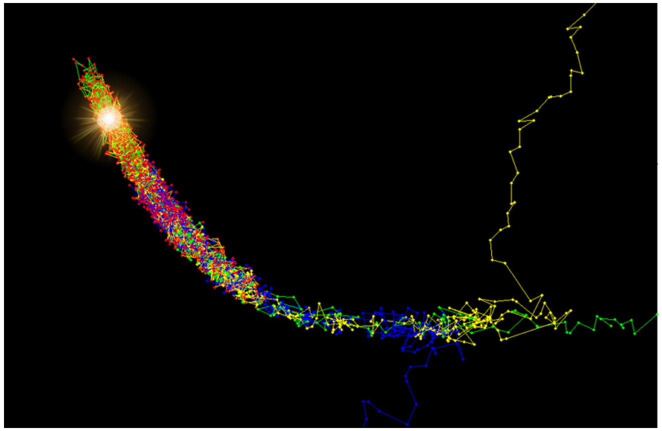
NFB_simple_fit.cps

Stochastic fitting methods

- Genetic algorithm (COPASI)
  - A population of parameter combinations (genotypes), competes for entry (selection) into the next generation
  - New members added by mutation and recombination of the fittest genotypes



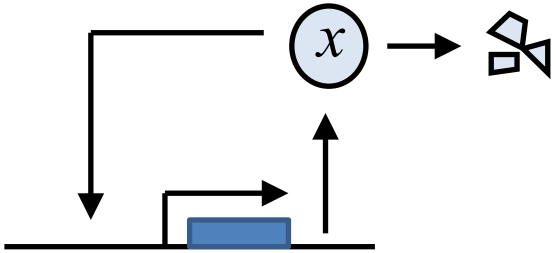
- Metropolis algorithm (Matlab)
  - At each generation, i, take a random step in p. Keep it if $E_i < E_{i+1}$, or $\exp((E_i - E_{i+1})/k_T) < \mathrm{rand}[0,1]$



http://en.wikipedia.org/wiki/File:3dRosenbrock.png

Example: simple negative feedback

Metropolis algorithm



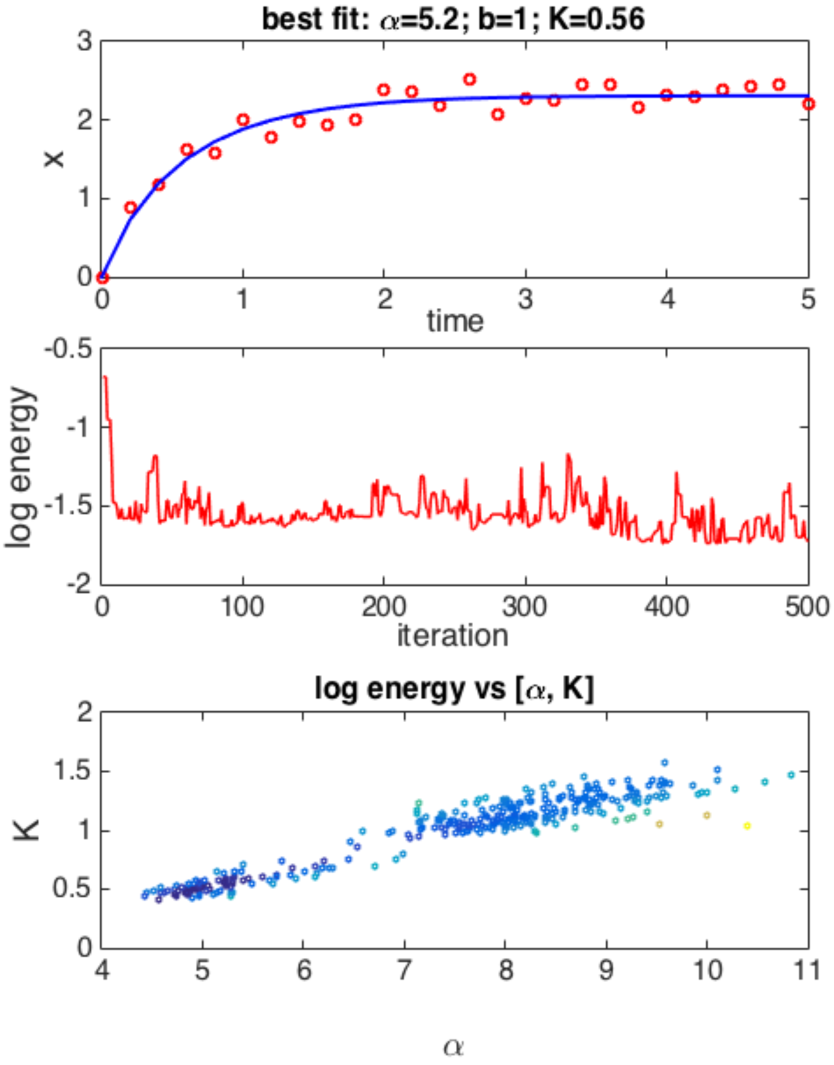
$$\dot{x} = \left(\frac{\alpha}{1 + \mathrm{K}x}\right) - \beta x$$

$$\beta = 1$$

$$\alpha \approx 5.2$$

$$K \approx 0.56$$

NFB_simple_Metropolis.m



- so I get several different parameter sets from fitting algorithms, how accurate are the estimated parameters for the data?

- What are the confidence intervals of parameters?

  (How frequently the observed interval contains the true parameters)

- Does the model overfit the data? Is the model adequate? How to compare two possible models?

Bootstrap method

- How accurate are the estimated parameters for the data?

- A statistical technique uses random sampling with replacement to measure the accuracy of sample estimates.

- It allows us to quantify the uncertainty when you don’t know enough about the estimation errors (i.e. not sure whether they are normally distributed).

Bootstrap method
key idea

If we have dataset D_0 with N data points

1. Randomly select N data points from D_0 with replacement as $D^S_{j,}$ where D^S_j differs from D_0 with a small fraction of the original points replaced by duplicated original points. (resampling with replacement)
2. Fitting the D^S_j data to get model parameter sets $\vec{p}_j^S$.
3. With big enough sampling times (j>100), we get the distribution of parameters.

“to pull oneself up by one’s bootstraps”

Bootstrap method
a simple example:

D_0

Obs	X	Y
1	1.2	2.4
2	2.1	0.5
3	5	3.3

$D^S{}_1$

Obs	X	Y
3	5	3.3
1	1.2	2.4
2	2.1	0.5

$D^S{}_2$

Obs	X	Y
2	2.1	0.5
1	1.2	2.4
1	1.2	2.4

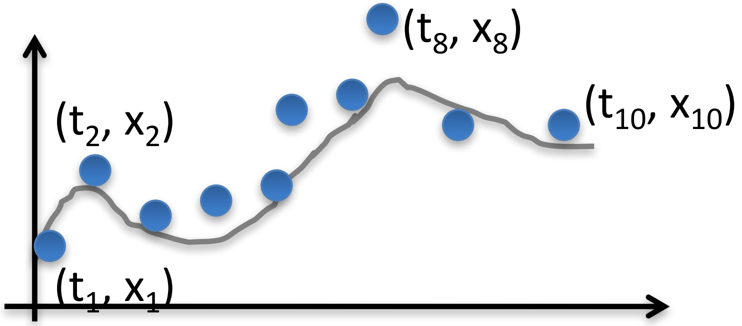
$D^S{}_3$

Obs	X	Y
2	2.1	0.5
2	2.1	0.5
3	5	3.3

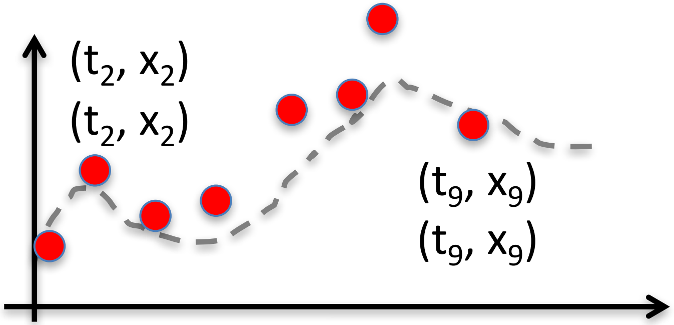
Estimate $\vec{p}_1^{\,S}$ $\vec{p}_2^{\,S}$ $\vec{p}_3^{\,S}$

Bootstrap method for time series:

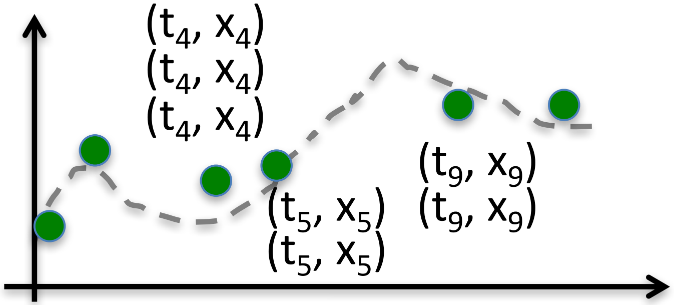
D_0 (t_i, x_i), i=1,2,…,10



$D^S{}_1$ [1, 2, 2, 3, 4, 6, 7, 8, 9, 9]

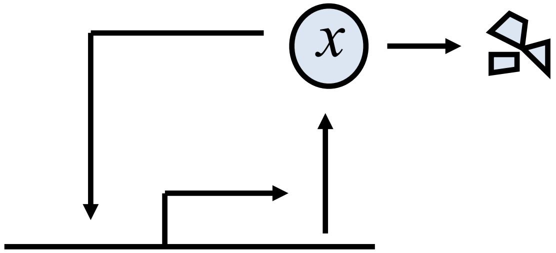


$D^S{}_2$ [1, 2, 4, 4, 4, 5, 5, 9, 9, 10]



Estimate $\vec{p}_1^{\,S}$ $\vec{p}_2^{\,S}$

Example: simple negative feedback
Bootstrap + Metropolis algorithm



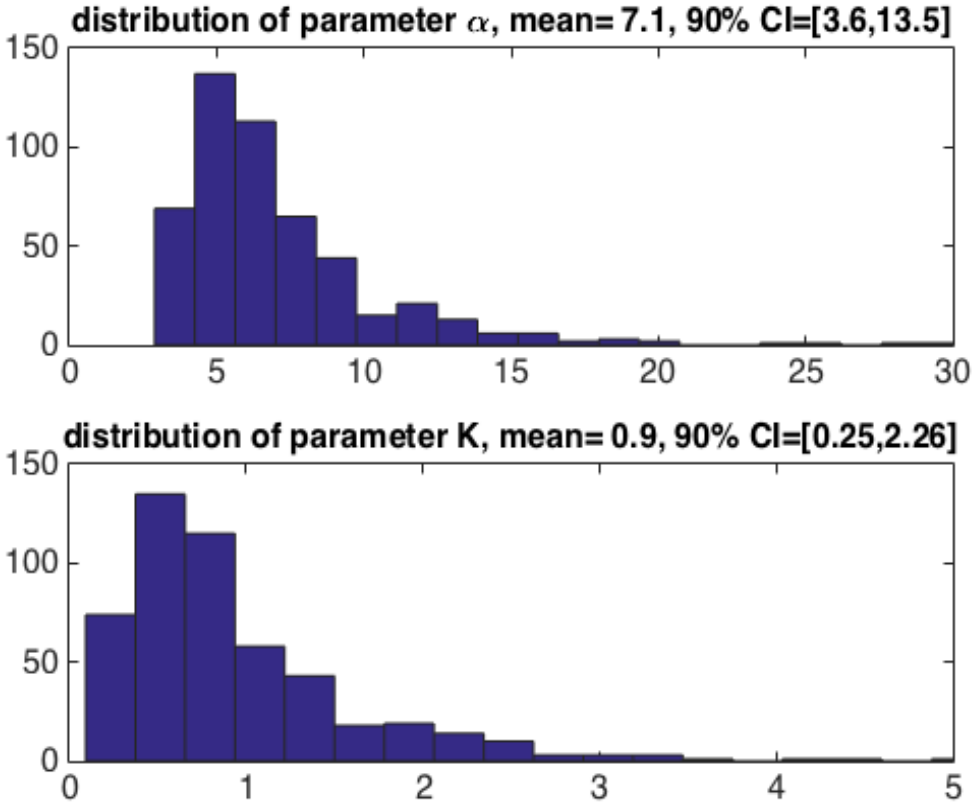
$$\dot{x} = \left(\frac{\alpha}{1+\mathrm{K}x}\right) - \beta x$$

$$\beta = 1$$

$$\alpha \approx 5.2$$

$$K \approx 0.56$$

This is just one fit

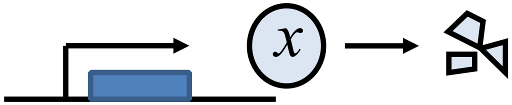

distribution of parameter α, mean= 7.1, 90% CI=[3.6,13.5]

distribution of parameter K, mean= 0.9, 90% CI=[0.25,2.26]

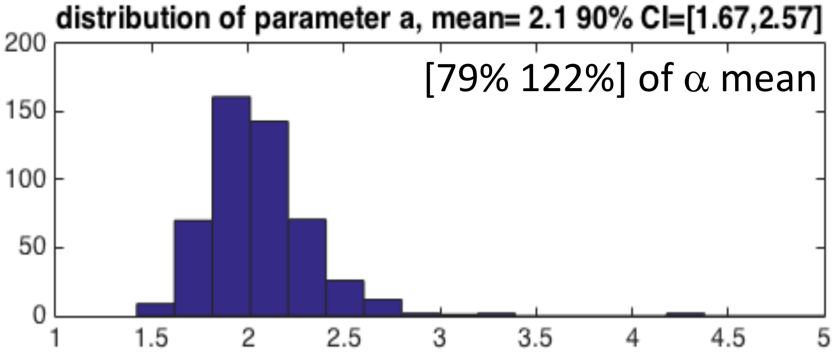
NFB_simple_metropolis_bootstrap.m

Example: synthesis-degradation

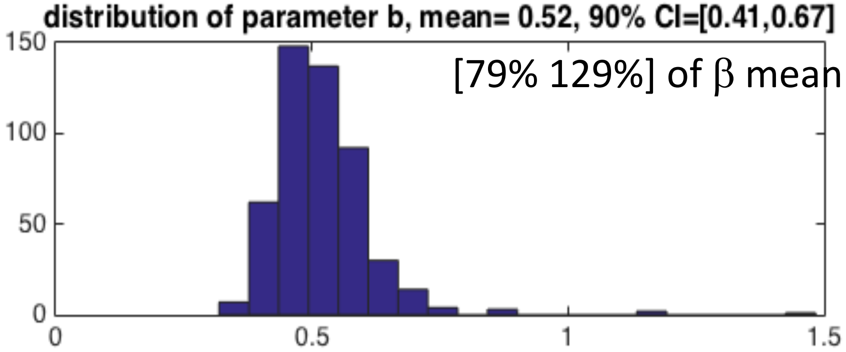
Bootstrap + Metropolis algorithm



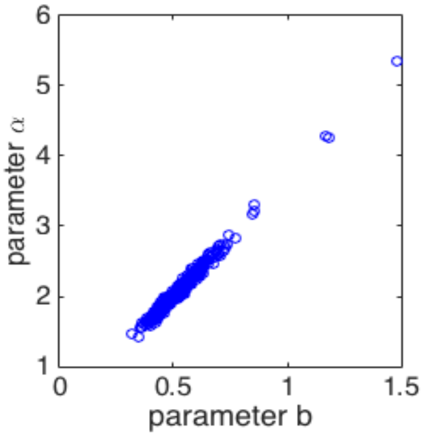
$$\dot{x} = \alpha - \beta x \quad \alpha = ? \quad \beta = ?$$

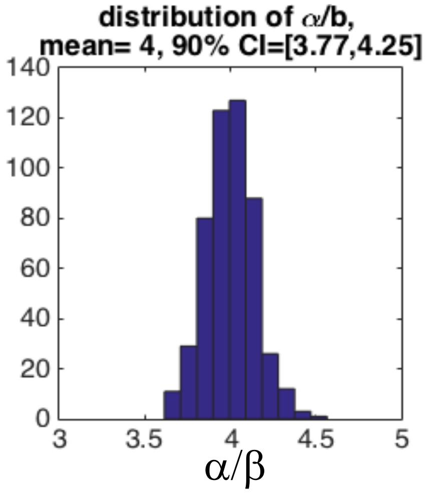

distribution of parameter a, mean= 2.1 90% CI=[1.67,2.57]

[79% 122%] of α mean


distribution of parameter b, mean= 0.52, 90% CI=[0.41,0.67]

[79% 129%] of β mean




distribution of α/b,
mean= 4, 90% CI=[3.77,4.25]

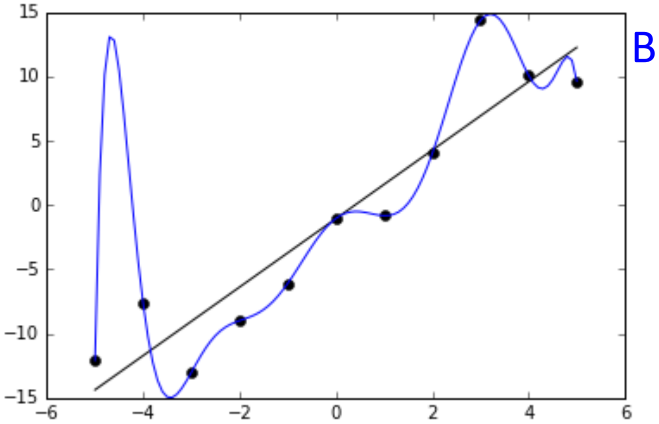
[94% 106%] of α/β mean

syndeg_simple_metropolis_bootstrap.m

Cross-validation

Concerns:

- Overfitting

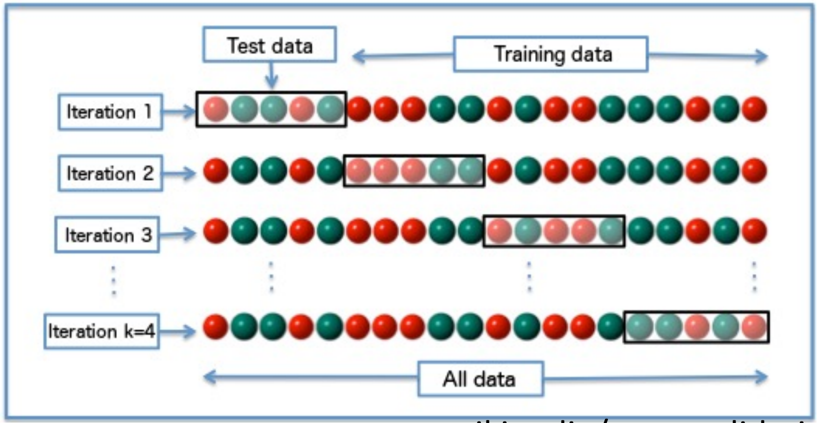


Blue: overfitting

- How do I know which model is better?

Cross-validation

- Split the data into 2 non-overlapping sets, one is training data set, the other is test data set
- use the training data set to train a model, get fitted parameters
- Use fitted parameters to predict the test data set
- calculate errors between "prediction" and test data



wikipedia/cross-validation

- The error tells how well the model predicts "new"data

(big error: likely to be overfitting and worse prediction)

Cross-validation Steps

- divide the data into K roughly equal parts

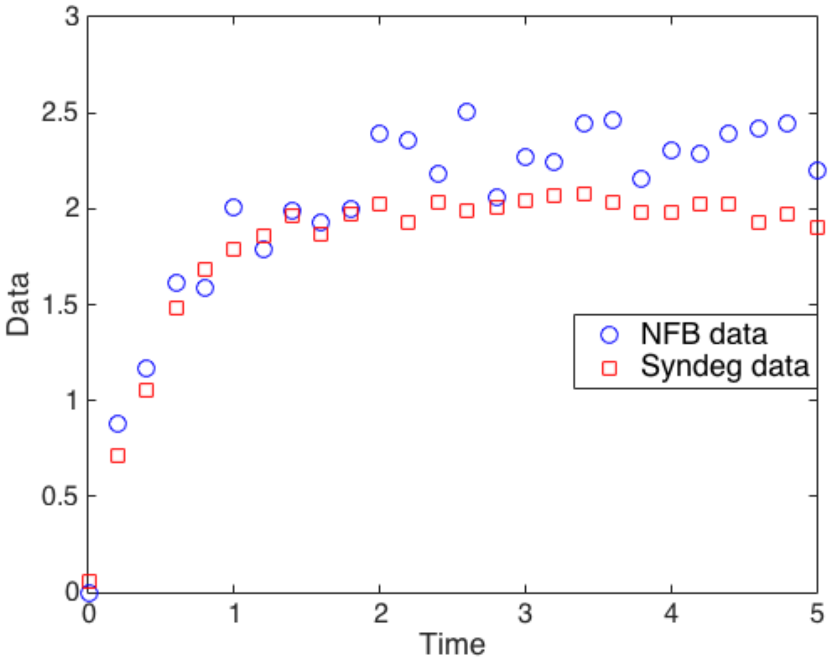
	k=1	k=2	k=3	k=4	k=5
1	Test	Train	Train	Train	Train
2	Train	Test	Train	Train	Train
…					
5	Train	Train	Train	Train	Test

- for each *k*=1, 2, … K, fit the model to the other K-1 parts to get $\vec{p}_k$ and compute its error in prediction the *k*th part: $E(\vec{p}_k) = \sum_{i \in kth \quad part} (y_i - \widehat{y}_i(\vec{p}_k))^2$
- For all k parts, this gives the cross-validation error

$$CV(\vec{p}) = \frac{1}{K} \sum_{k=1}^{K} E(\vec{p}_k)$$

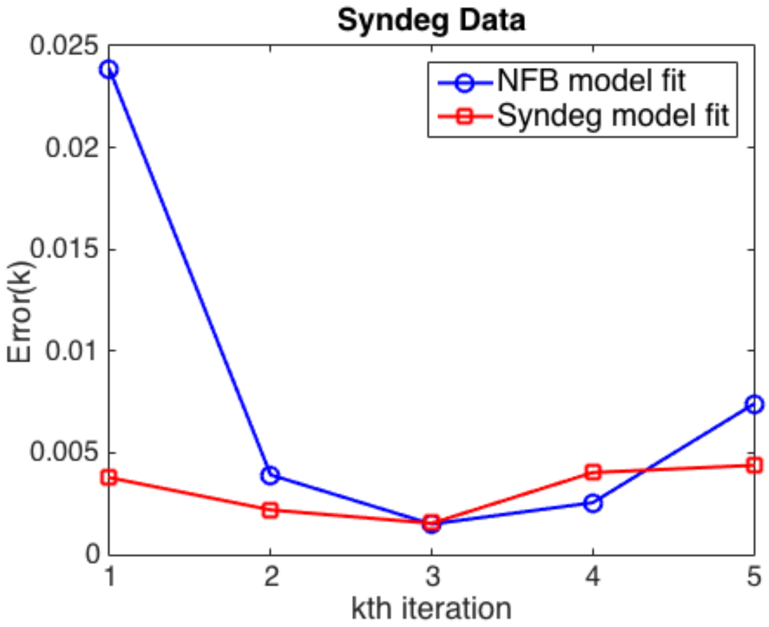
Cross-validation Example:

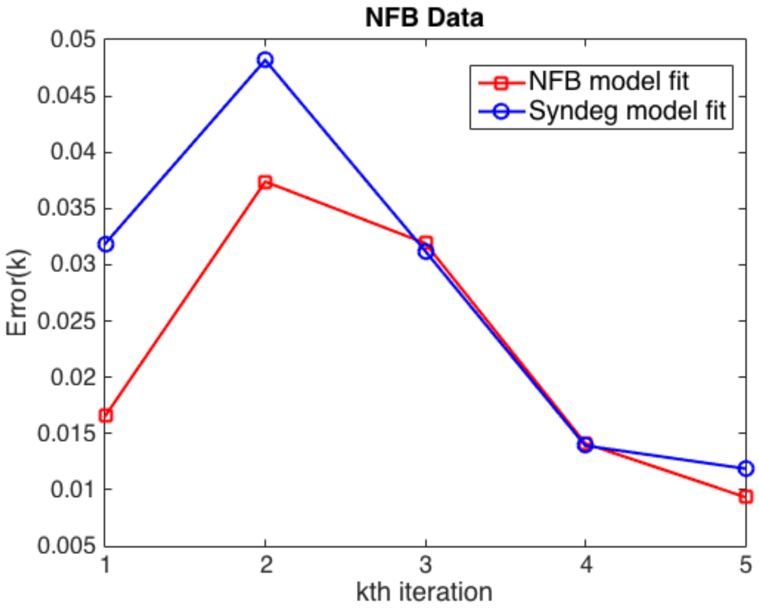
synthesis-degradation or simple negative feedback?



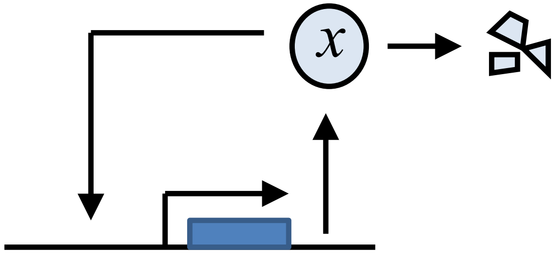
Cross-validation Example:

synthesis-degradation or simple negative feedback?





Example: delayed autorepression



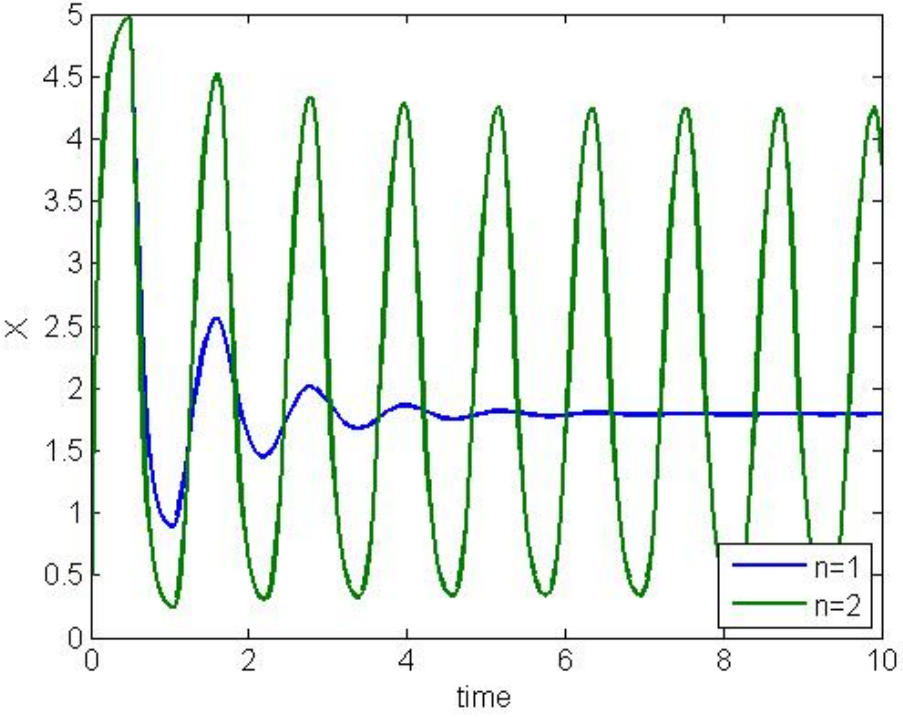
$$\dot{x} = \left(\frac{\alpha}{1 + x(\tau)^n} \right) - \beta x$$

$$\beta = 10$$

$$\alpha = 50$$

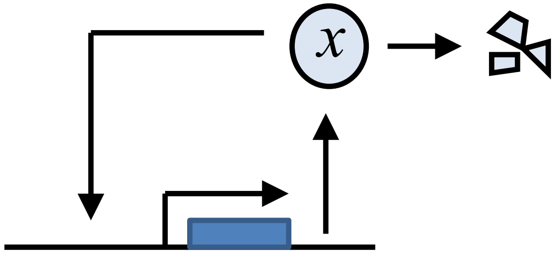
$$\tau = 0.5$$

$$n = ?$$



NFB_del_run.m

Example: delayed autorepression, Metropolis algorithm



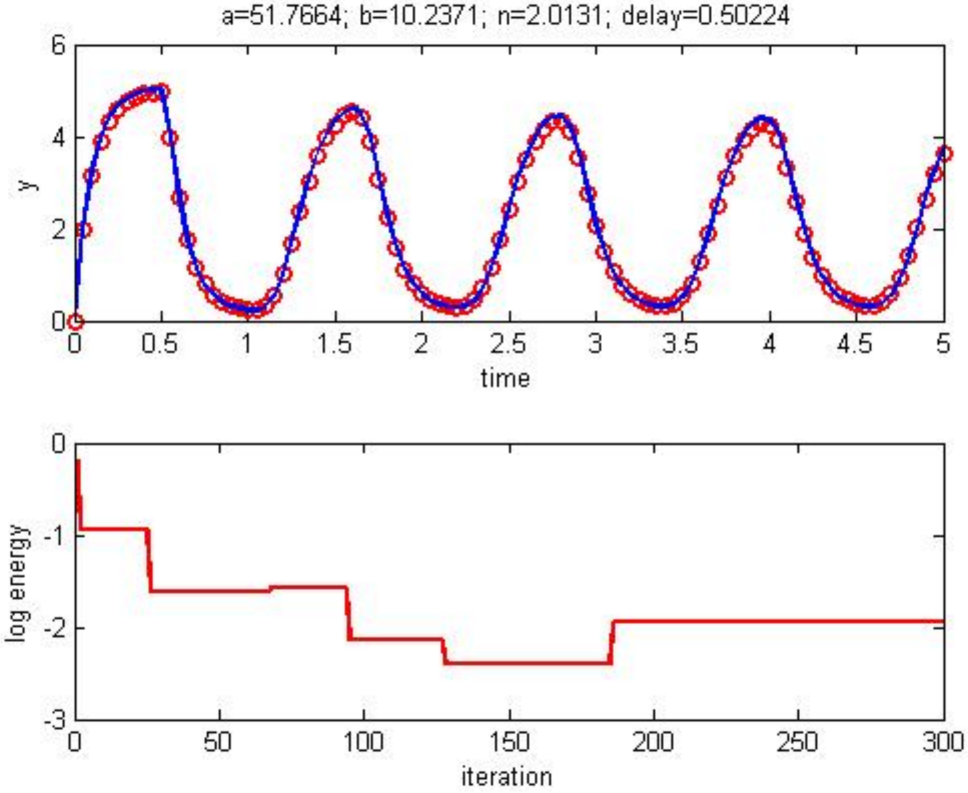
$$\dot{x} = \left(\frac{\alpha}{1+x(\tau)^n}\right) - \beta x$$

$$\beta = ?$$

$$\alpha = ?$$

$$\tau = ?$$

$$n = ?$$



NFB_del_metropolis.m

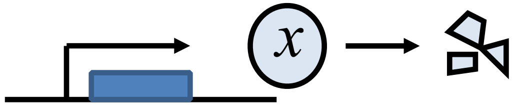
Fitting stochastic models

- Single output trajectory is noisy, may not be sufficient to fit the data
- Need good observed and simulated process statistics
- Consider other measures for the error function:
  - mean and standard deviation of variables
  - if periodic, mean of period or amplitude
  - if periodic, variance of peak heights or peak times
  - etc.
- Ex.:

  $E = \text{mean}((\text{mean}X_o - \text{mean}X)^2 + (\text{std}X_o - \text{std}X)^2))$

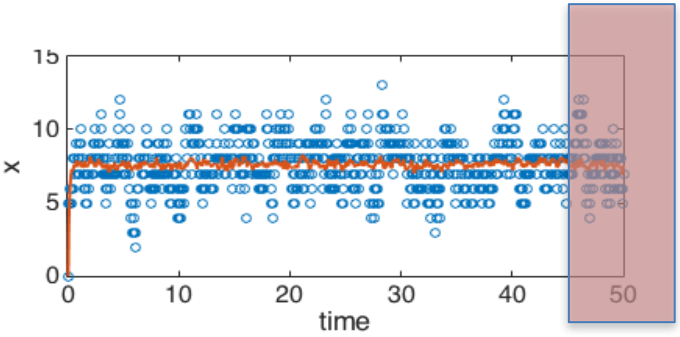
  $E = \text{mean}((\text{mean}T_o - \text{mean}T)^2 + (\text{std}T_o - \text{std}T)^2))$

Example: stochastic synthesis degradation



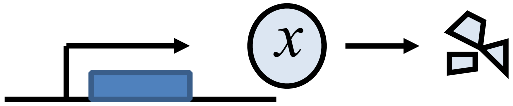
$$\dot{x} = \frac{\alpha}{1+\left(\frac{x}{C_0}\right)^n} - gx$$

$$C_0 = 5 \quad n = 3 \quad \alpha = ? \quad g = ?$$



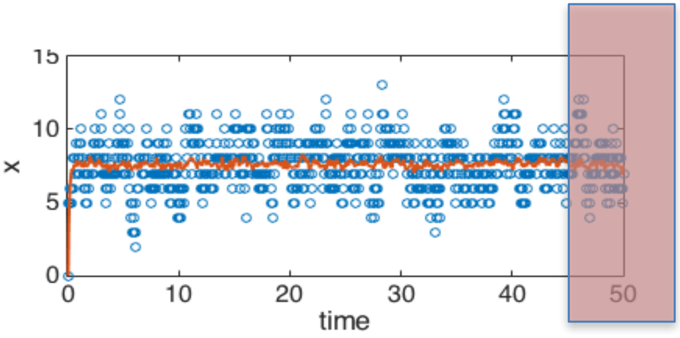
If we know the mean and std …

Example: stochastic synthesis degradation

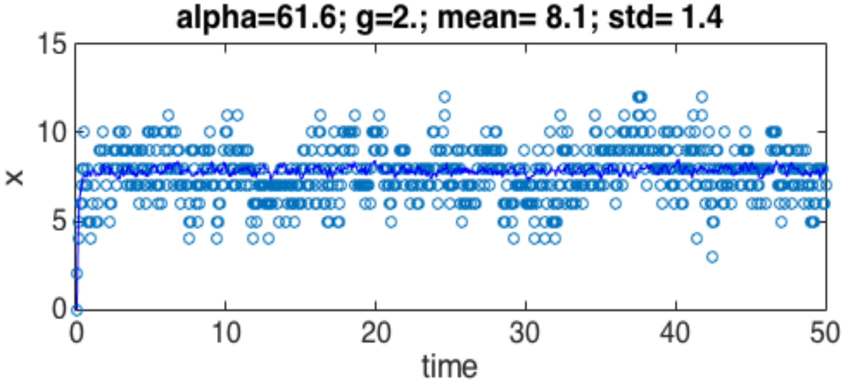


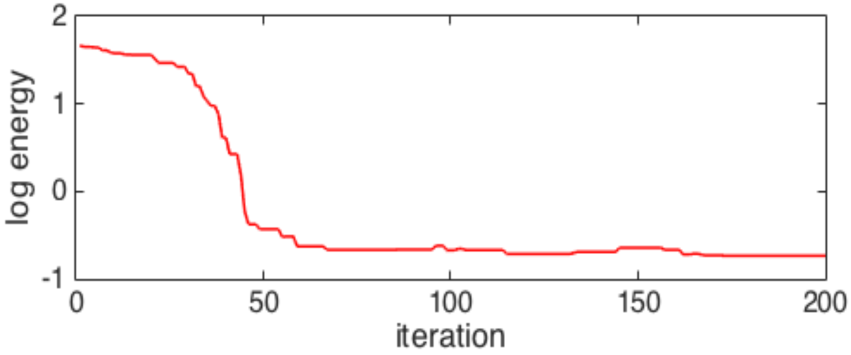
$$\dot{x} = \frac{\alpha}{1+\left(\frac{x}{C_0}\right)^n} - gx$$

$$C_0 = 5 \quad n = 3 \quad \alpha = ? \quad g = ?$$



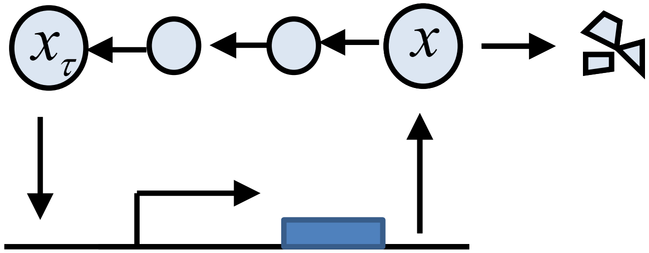
If we know the mean and std …





NFB_syndeg_gil_metropolis.m

Example: stochastic delayed autorepression

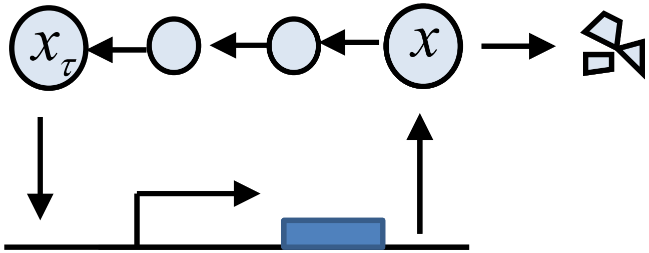


$$\dot{x} \approx \left(\frac{\alpha}{1 + x(\tau)^n} \right) - \beta x$$

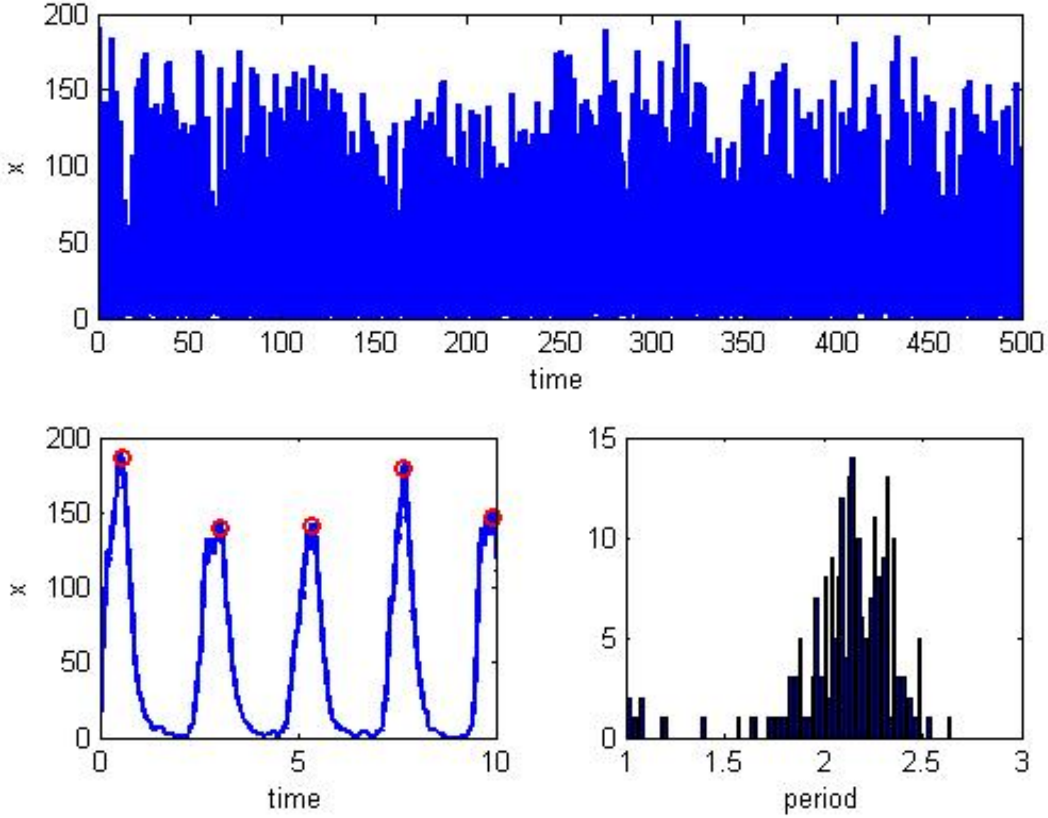
Mather, et al. PRL, 102(6) 2009

NFB_gil_run.m

Example: stochastic delayed autorepression



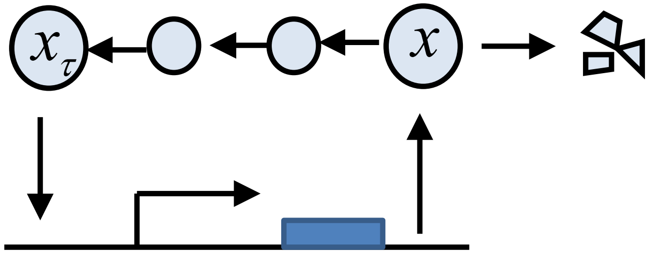
$$\dot{x} \approx \left(\frac{\alpha}{1+x(\tau)^n}\right) - \beta x$$



Mather, et al. PRL, 102(6) 2009

NFB_gil_run.m

Example: stochastic delayed autorepression

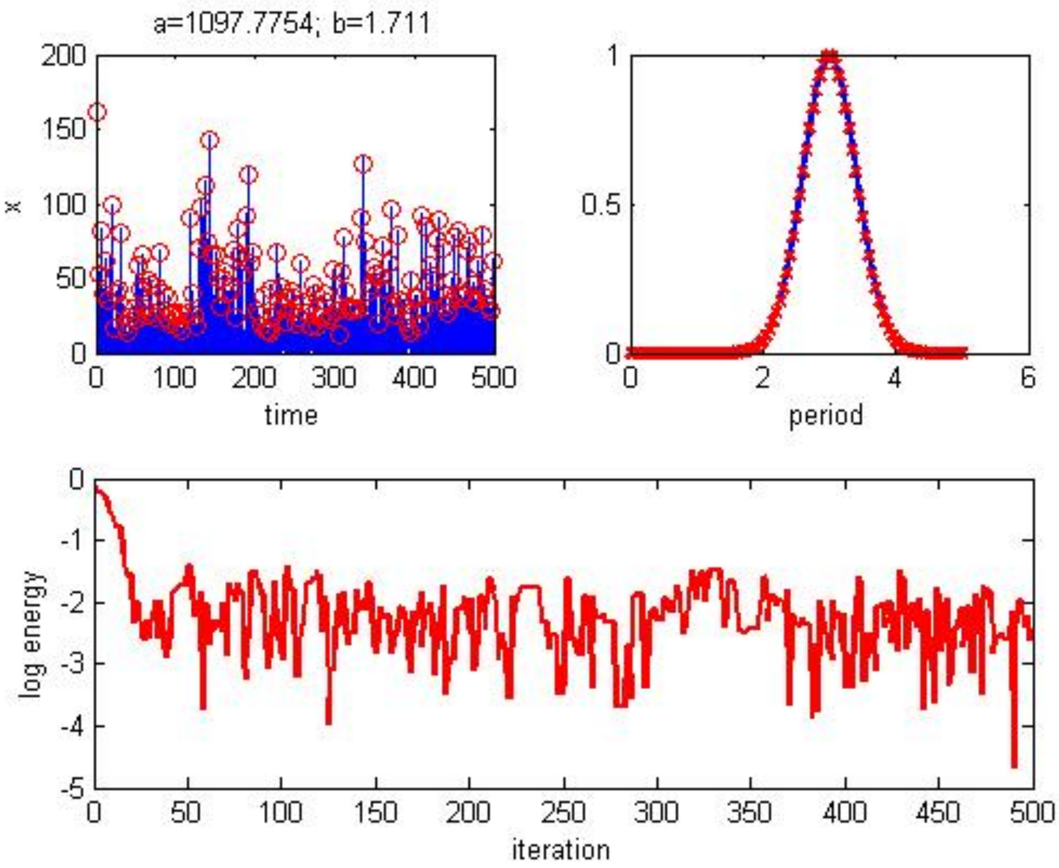


$$\dot{x} \approx \left(\frac{\alpha}{1+x(\tau)^n}\right) - \beta x$$

$\alpha = ?$

$\beta = ?$

…



Mather, et al. PRL, 102(6) 2009

NFB_gil_metropolis.m

Sensitivity Analysis

- Quantifying how much variation of inputs affects variation of output ”observables’’ in the model
  - How model responses to parameters pertubation?

- Useful for:
  - Finding redundant, or inconsequential inputs
    - these may help simplify the model
  - Identifying inputs that cause high fluctuation for a given output
    - these inputs can be prioritized for further study

- Disclaimer:

  it can give incorrect results due to assumptions of the model

Parameter sensitivity

- Quantifying how variation of parameters affect variation of output ”observables” (ex.: direct output, steady state, period, amplitude)
- Local methods
  - Parameters are varied one at a time by a small amount, around steady state
  - the effect of individual perturbations on the outputs are calculated

$$S = \frac{(Output_a - Output_{ss}) / Output_{ss}}{(P_a - P_o) / P_o}$$

Parameter sensitivity

For a general ODE model of the form: $\frac{d\vec{x}}{dt} = f(\vec{x}, \vec{p} \ , t)$

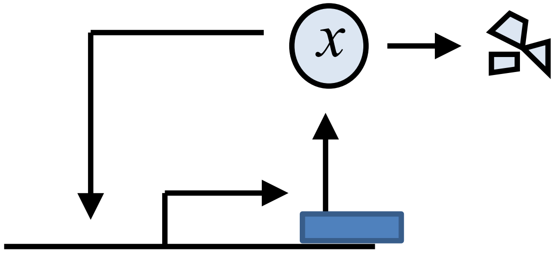
where $\vec{x}$ is the vector of variables, $\vec{p}$ is the m-vector of model parameters. Then, for given parameter set

$$x_i(t, \vec{p} + \Delta\vec{p}) = x_i(t, \vec{p}) + \sum_{j=1}^{m} \frac{\partial x_i}{\partial p_j} \Delta p_j + O(x^2)$$

First-order local sensitivity coefficients: $S(x_i, p_j) = \frac{\partial x_i}{\partial p_j}$

In reality, sensitivity coefficients are normalized to be dimensionless (no unit): $S(x_i, p_j) = \frac{\partial x_i / x_{i\_ss}}{\partial p_j / p_j}$

Example: delayed negative feedback sensitivity



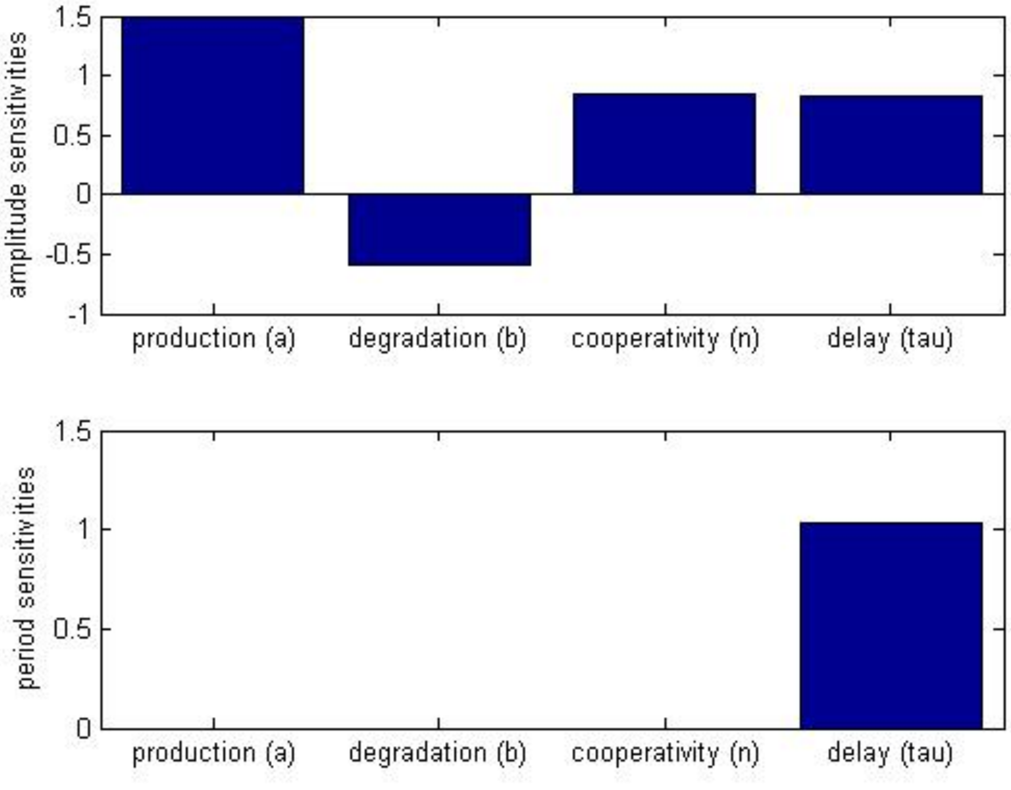
$$\dot{x} = \left(\frac{\alpha}{1+x(\tau)^n}\right) - \beta x$$

$\beta = 10;$

$\alpha = 50$

$n = 2$

$\tau = 0.5$



NFB_delay_sens.m

The end

- Thank you for your participation

- Exercise time!

Exercises

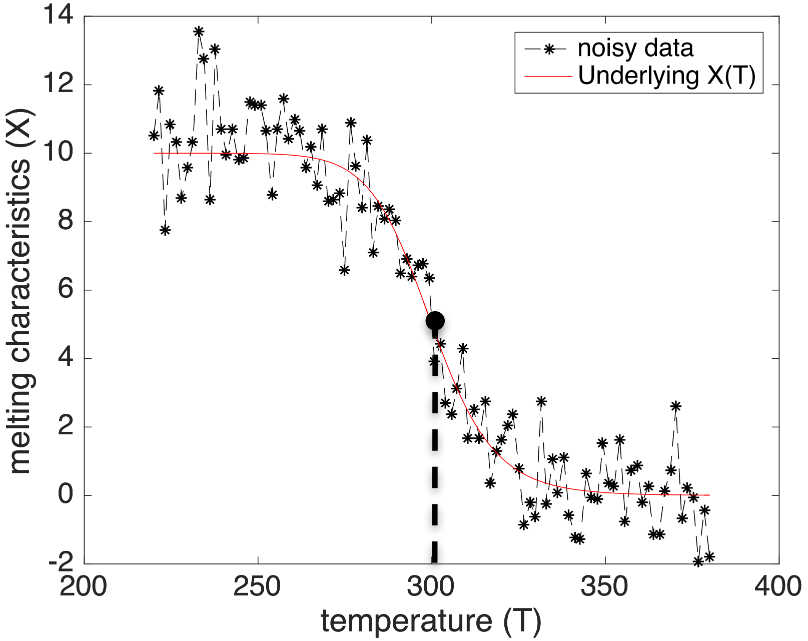
1. Load the exercise_data.mat, try to fit the data to a positive feedback model with metropolis algorithm

$$\dot{x} = a + \frac{x^n}{1 + Kx^n} - bx,\ \ (a, b, K, n)$$

2. Using bootstrap method to get the confidence intervals of parameters
3. Test whether a simple synthesis-degradation method would be a better model.

4. Tm problem (see next slide)

Melting temperature problem



Melting temperature $T_m = 300K$

Fit with:

$$X(T) = \frac{X_U + X_N K(T)}{1 + K(T)}$$

$$K(T) = \exp\left(\left(\frac{1}{T} - \frac{1}{\boxed{T_m}}\right) \cdot (\Delta E / k)\right)$$

Four parameters: $\boxed{T_m}, X_U, X_N, (\Delta E / k)$

Question: given the noisy data, how confident can we be about the fitted Tm?