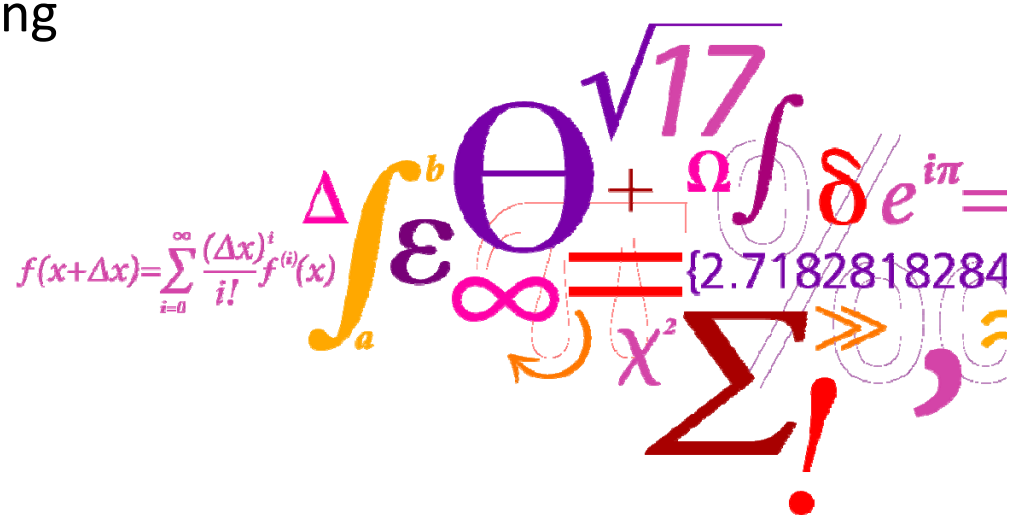
L2_1 Bayesian framework for uncertainty analysis

Gürkan Sin, Associate Professor
PROSYS- DTU Chemical Engineering



DTU Kemiteknik
Institut for Kemiteknik

Agenda

9:00-12:00	L2.1 Bayesian framework for parameter estimation
	Exercises
Lunch break	
1:00-4:00	L2.2 Monte Carlo procedure
	Exercises

Objective of this lecture

- At the end of the lecture, you should be able to:
  - Understand and apply in a generic sense bayesian inference to the parameter estimation and uncertainty problem for nonlinear models
  - Understand basic concept of numerical methods for Bayesian inference : in particular mcmc (markov-chain monte carlo) sampling methods such as metropolis algorithm, metropolis-hasting, adaptive metropolis, etc …
  - Calculate & interpret joint posterior probability density of model parameters as well as make inferences about the uncertainty of model outputs from mcmc simulations (including mean, variance,…)

Outline

- Bayesian theorem
- Bayesian analysis for nonlinear regression
- Numerical methods for solving Bayesian PE problems
- Example 1: Bivariate distribution function
- Example 2: Monod model
- Exercise: OUR model
- Recap
- Copyright notice related to Matlab scripts

Bayesian inference

Unlike other statistical inference methods such as Frequentists, Bayesian statistics treats parameters, θ, and, model variables, y_{model}, as variables described by *probability* statements

The probability statements are conditioned on the measurements, y :

$$p(\theta \mid y) \quad \& \quad p(y_{\mathbf{model}} \mid y)$$

Probability notation:

$p(\cdot \mid \cdot)$ conditional probability density

$p(\cdot)$ marginal distribution

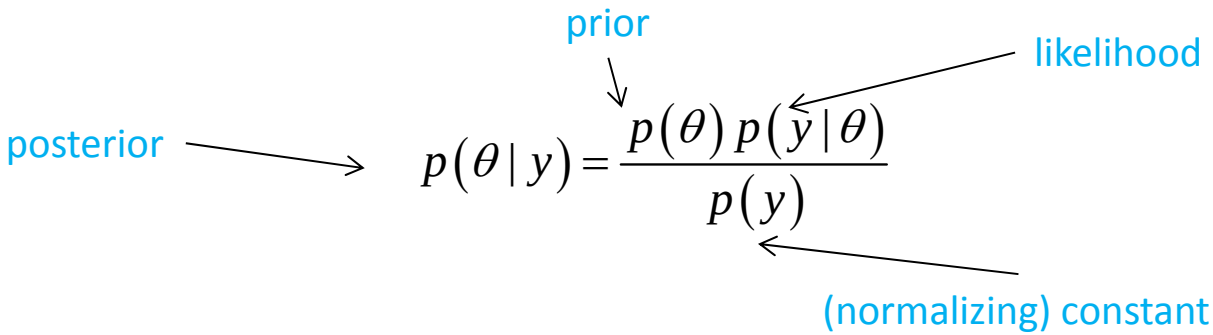
The terms 'distribution' and 'density' are used interchangeably.

Standard distributions are identified with their names. E.g. if θ has a normal distribution with mean, μ and variance, σ^2, one writes this as follows:

$$\theta \sim N(\mu, \sigma^2) \quad or \quad p(\theta) = N(\theta \mid \mu, \sigma^2) \quad or \quad p(\theta \mid \mu, \sigma^2) = N(\theta \mid \mu, \sigma^2)$$

Bayesian theorem

Bayes' rule provides a mathematical framework to update your beliefs (prior information) about a variable, θ , given observations, y. Mathematically, Bayes' rule calculates the posterior probability of θ given observations, p(θ|y), as follows:



Where $p(y) = \int p(\theta)\, p(y \mid \theta)\, d\theta$ is also called a normalizing constant that make sure the sum of integral of posterior density is equal to 1. An equivalent form of this rule is the following which yields unnormalized posterior density:

$$p(\theta \mid y) \propto p(\theta)\, p(y \mid \theta)$$

This summarizes the technical core of Bayesian inference. The primary task of any application is to develop a model and make necessary computations for p(θ|y)

Bayesian inference for predictive distributions

Given the observations y and the model p(y|θ), predictive distribution of y*, p(y*|y), can be calculated once the posterior probability , p(θ|y), is obtained as follows:

$$p(y^*|y) = \int p(y^*|\theta)\, p(\theta|y)\, d\theta$$

In simple words, future prediction are calculated using the updated probability of p(θ|y) similar to making a prediction for y* using single value of θ in traditional sense.

This integral is caculated using monte carlo simulation strategies such as markov-chain monte-carlo algorithms.

Numerical methods for bayesian inference

The functions involved in bayesian theorem (likelihood , the integral of normalizing constant) are higher dimensional functions for which analytical solutions rarely exist. Instead simulation methods known as sampling are used .

Some simulations methods are:

- Rejection sampling
- Importance sampling
- Gibbs sampling
- Markov Chain Monte Carlo (MCMC) simulation

Markov chain simulation –in brief

Markov chain simulation draw samples from *a target distribution*, $\pi(\theta)$, which after a sequence of iteration (simulations) will converge to posterior distribution, ,$p(\theta|y)$.

By definition, Markov chain is a sequence of random variables θ^1 ,θ^2, θ^3,… for which, for any k, the distribution of θ^k depends only on the most recent one, θ^{k-1}.

In practice, several independent sequences of Markov chain simulation are created:

θ^k k=1,2,3… by starting at some point θ^0, and then for each k a sample, θ^k , is drawn from a transition distribution $T(\theta^k|\ \theta^{k-1})$ which depends on previous value.

Many sampling methods have been developed for sampling from transition distributions and are still being developed (active research area). We will look at metropolis and metropolis-hasting algorithms developed early on giving birth to this field.

Metropolis algorithm

The metropolis algorithm is an adaptation of a random walk that uses an acceptance/rejection rule to converge to the specified target distribution. The algorithm proceeds as follows:

Step 1: select initial parameter vector, θ_0

Step 2: iterate as follows for k=1,2,3…

1. Create a new trial position, $\theta^* = \theta^{k-1} + \Delta\theta$
   where $\Delta\theta$ is randomly sampled from jumping distribution $q(\Delta\theta)$
2. Calculate metropolis ratio $$r = \frac{\pi\left(\theta^* \mid y\right)}{\pi\left(\theta^{k-1} \mid y\right)}$$
3. Accept new sample if:

$$\theta^k = \begin{cases} \theta^* & \text{with probability } \min(r,1) \\ \theta^{k-1} & \text{otherwise} \end{cases}$$

Note this requires jumping distribution to be symmetric: $q(\theta^*, \theta^{k-1}) = q(\theta^{k-1}, \theta^*)$

Metropolis-hasting algorithm

The metropolis-hasting algorithm is a generalisation of metropolis algorithm in which jumping distribution need not be symmetric and the acceptance/rejection rule is updated as follows:

$$r = \frac{\pi\left(\theta^* \mid y\right) q\left(\theta^{k-1} \mid \theta^*\right)}{\pi\left(\theta^{k-1} \mid y\right) q\left(\theta^* \mid \theta^{k-1}\right)}$$

The rest remains the same.

Simple example: bivariate unit normal distribution

To illustrate the metropolis algorithm (mcmc in general for that matter), we will use bivariate unit normal density function.

the task is to use mcmc sampling to converge to the target density function.

Target density is bivariate unit normal: $\pi(\theta|y)=N(\theta|0,C)$ where C is 2x2 identity matrix

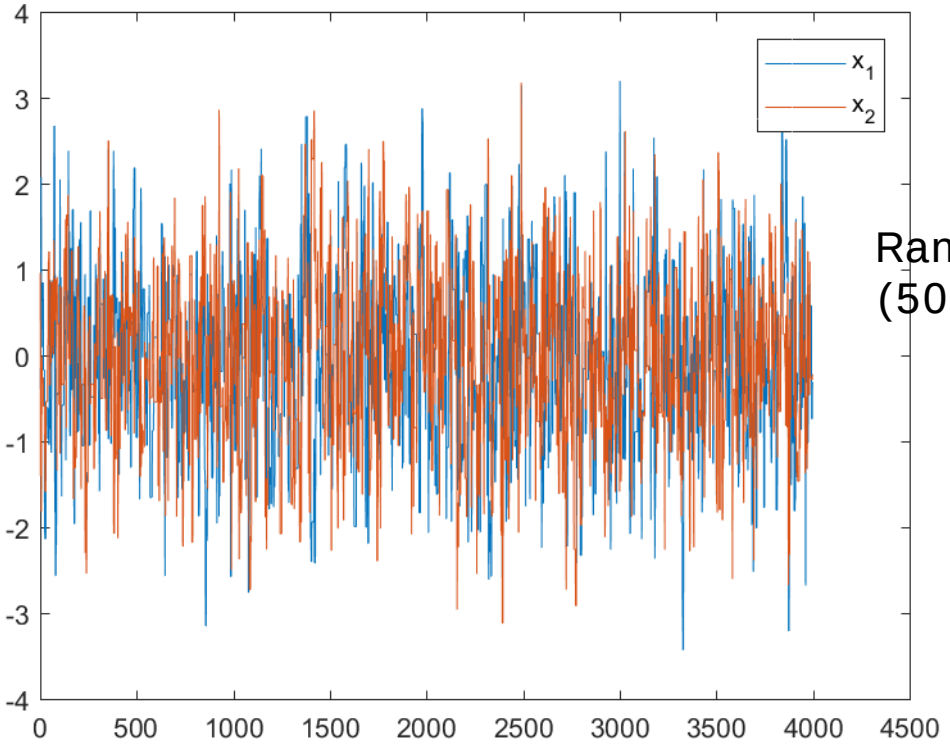
Jumping distribution also bivariate normal centered at current iteration and scaled to 1/5 the size: $q(\theta^*|\ \theta^{k-1})= N(\theta^*|\theta^{k-1},0.2^2C)$

Matlab implementation of metropolis algorithm also known as random walk algorithm

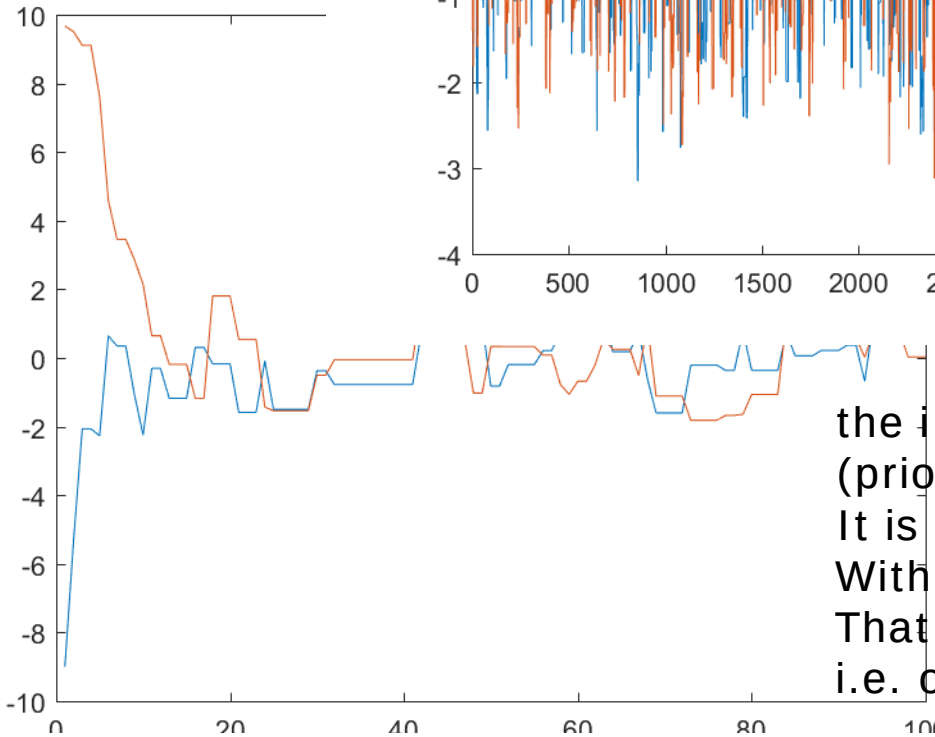
Metropolis algorithm in Matlab

```
%%step 1 draw a starting point for each chain.
mu = [0 0]; % C = eye(2); % C is identity matrix R = chol(C);
  npar=length(mu); covscale=2.4 / sqrt(npar); % important.
x0= mvnrnd(mu,C,chainnumber) ; %bivariate normal
 %step 2: perform sampling
for j=1:chainnumber
    x=x0(j,:);
    for i=1:samplingnumber
        %2.1 sample a proposal from jumping distribution
        u = randn(1,npar);  % draw a random number
        dx=u*R*covscale; % incremental walk
        xp=x+u*R*covscale; % new trial position
        %2.2 calculate the metropolis ratio:
        r= mvnpdf(xp,mu,C) ./ mvnpdf(x,mu,C); ratio(i,:)=r';
        %2.3 reject/accept new sample
        if min(r,1) > random('unif',0,1) % probability min(r,1)
            x=xp ; else x=x;
   end
end
```

Simple exa[illegible]

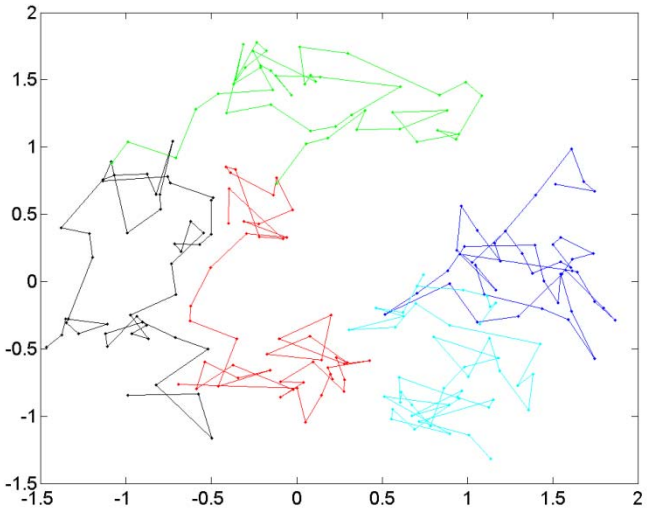


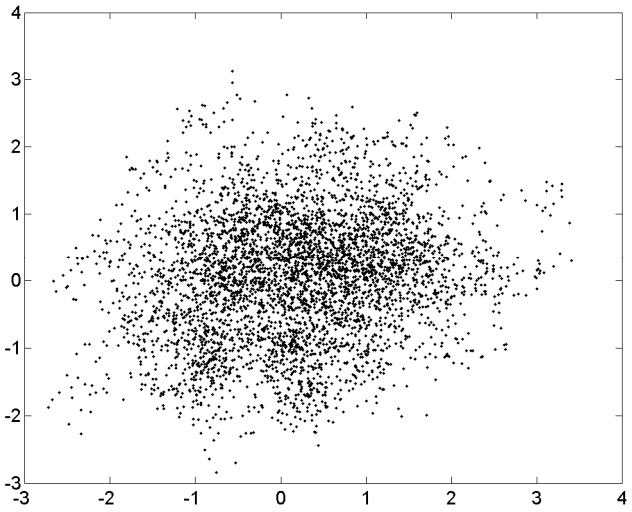
Random walk
(5000 mcmc steps)



the impact of initial point
(prior knowledge).
It is quickly updated (bayesian)
With new information.
That is the reason why we use burn in ratio.
i.e. omit first 20 or 50% of data in postproce

Simple example: bivariate normal density



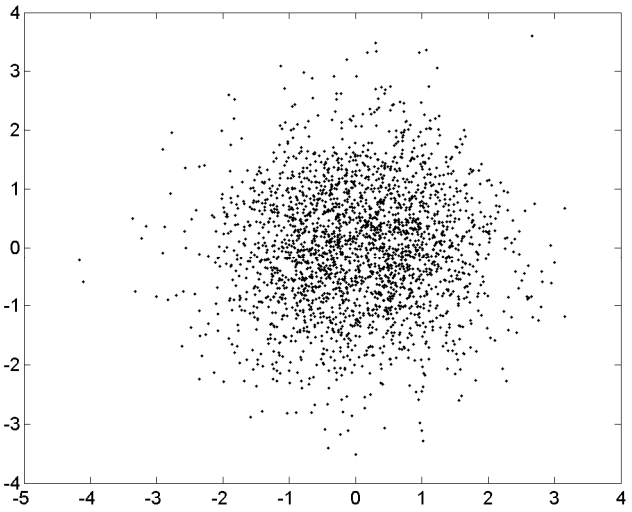


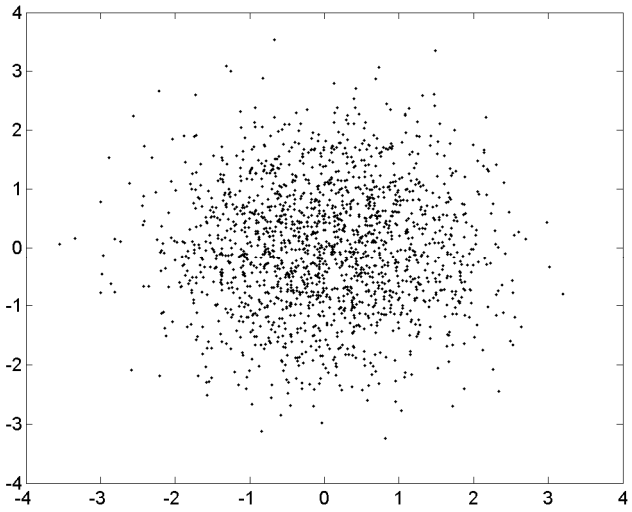
5 chains and 50 simulations: random walk starting from different points

5 chains and 1000 simulations: random walk s hidden

The scaling factor used for jumping distribution is an important property. It was purposely chosen as 0.2 to make the jumping inefficient to see the random walk aspect in the figure-left. For a better performance, suggested efficiency scaling ~ 2.4/sqrt(d) where d is the dimension of the target distribution function (Gelman et al. 2004).

Simple example: bivariate normal density scaling factor versus acceptance ratio





5 chains and 1000 simulations:

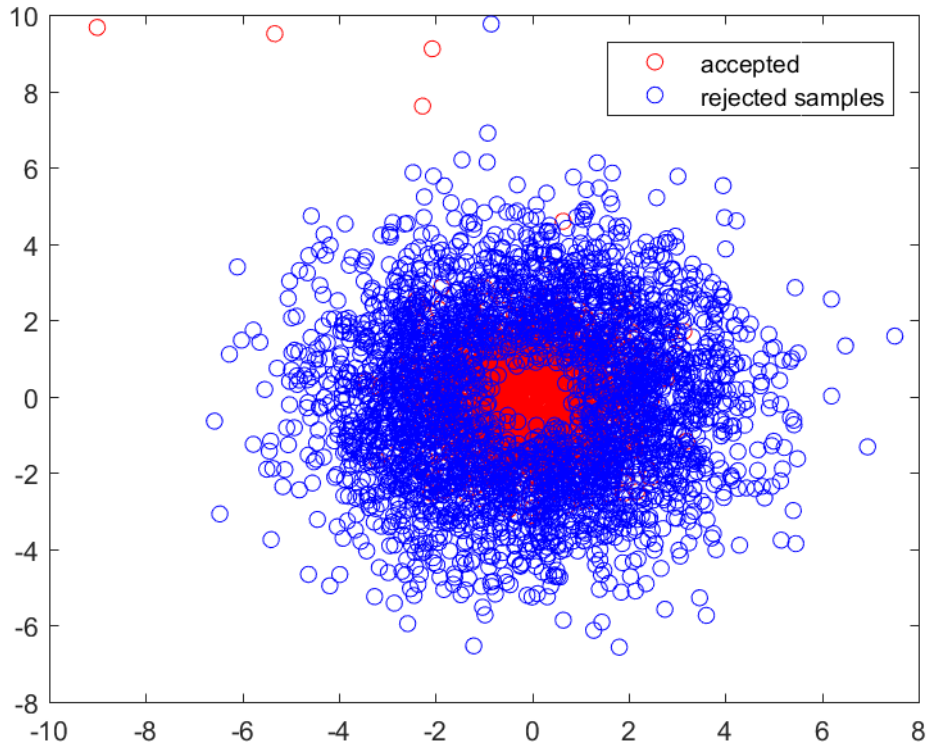
Scale = 1.2 vs AR=0.48

5000 simulations:

scale 2.4/sqrt(d) v s AR=0.35

Optimal acceptance ratio (AR) is 0.44 for d=1 and decreases to 0.23 for d>5 (Gelman et al 2004)

Compare the accepted versus rejected samples



Random walk metropolis algorithm is simple and beautiful, i.e. it works.

Adaptive metropolis algorithm

we update the covariance matrix based on previous accepted samples.

```
function [x px R] =am(prior,pdf,T,d)
q =@(C,d) mvnrnd(zeros(1,d),C);  % dvariate normal proposal distri
sd=2.38/sqrt(d); % covariance scale
C=sd^2*eye(d); % scaled covariance matrix for proposal distribution
x=nan(T,d); px=nan(T,1); %preallocate memory
x(1,:)=prior(1,d); % initialize chain by sampling from prior
px(1)=pdf(x(1,:)); % compute density of initial state chain
R=[]; % initialize rejected samples
for t=2:T
    if mod(t,10)==0  % update covariance matrix every 10th iteration
        C=sd^2*(cov(x(1:t-1,:))+1e-4*eye(d)); % update the covariance
    end
    xp=x(t-1,:)+q(C,d); % generate proposal
    pxp=pdf(xp); % calculate density of hte proposal
    pac=min(1,pxp/px(t-1)) ; % compute probability of acceptance
    if pac>rand    % pacc larger than U(0, 1)?
        x(t,:)=xp; px(t)=pxp; % true: accept the proposal
    else
        x(t,:)=x(t-1,:); px(t)=px(t-1);
        R=[R;xp];  % store rejected samples
    end
```

Bayesian inference for nonlinear regression

Let us start with definition of regression model, *f*, and data,*y*, with independent gaussian errors, ε, with some unknown variance, σ^2,:

$$y = f(\theta) + \varepsilon \qquad \text{where } \varepsilon \sim N(0, \sigma^2 I)$$

θ is a vector that refer to unknown model parameters, and assumed to have multivariate normal distribution as prior distribution:

$$\theta_i \sim N(\nu_i, \eta_i^2)$$

The inverse of variance of the residuals typically assumed to follow gamma distribution which approximates gaussian error models (see Gelman et al., 2004):

$$p(\sigma^{-2}) = \Gamma\left(\frac{n_0}{2}, \frac{n_0}{2} s_0^2\right)$$

n_0 is the degrees of freedom and s_0^2 is the sample variance. Note that chi-sq is a special case of gamma distribution:

$$\frac{(n_0 - 1) s_0^2}{\sigma_0^2} \sim \chi^2 (n_0 - 1)$$

Bayesian inference for nonlinear regression – Monod example

Let us start with definition of Monod model *f*, which describes substrate limited microbial growth kinetics, where μ is the growth rate (h^{-1}), μ_{max}, is the maximum growth rate (h^{-1}) and K_s affinity coefficient (mgCOD/L):

$$f(S,\theta): \quad \mu = \mu_{\max} \frac{S}{S + K_s}$$

The parameter estimation problem is the following: given a set of measured growth rates,y, under the following substrate conditions ,S, estimate the unknown μ_{max}, and K_s :

S = [28 55 83 110 138 225 375]; % (mg / L COD)

y= [0.053 0.060 0.112 0.105 0.099 0.122 0.125]; % (h^{-1})

Monod example-
MCMC implementation

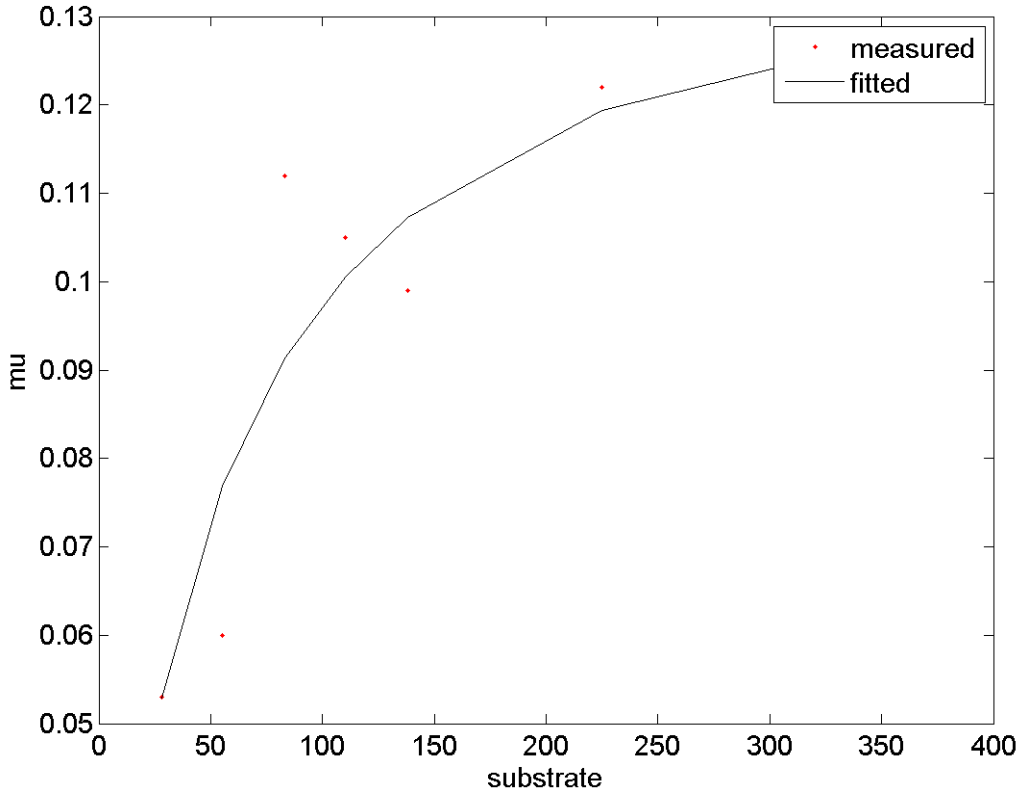
Step 0. Preparatory phase. This step to gain useful information about prior distribution function for the model parameters and to estimate the error variance

Solution: do a quick MLE estimation

```
% step 0: provide prior information about the unknowns (parameters)
and also error function (variance). here we can do an MLE estimation
Ssfun = @(theta,data) sum((data.mu-modelfun(data.s,theta)).^2);
[tmin,ssmin]=fminsearch(ssfun,theta*2,[],data);
n = length(data.mu);
p = length(theta);
s2 = ssmin/(n-p); % estimate the sample variance
J = [data.s./(tmin(2)+data.s), ...
    -tmin(1).*data.s./(tmin(2)+data.s).^2];
tcov = s2*inv(J'*J) ; %estimate of the covariance matrix for the
priors
tsigma=sqrt(diag(tcov))';
theta=tmin;
```

Monod example-
MCMC implementation

Step 0: Preliminary results for initilisation of MCMC



MLE estimation

Mean estimates	
μ_{max}	K_s
0.1454	49.0529
Covariance: Cov	
0.000244	0.25011
0.25011	320.8347
Correlation: Cor	
1	0.8926
0.8926	1
Sample variance,$s^2=\sigma^2$	
1.6335e-04	

Monod example-
MCMC implementation

Step 1. select initial parameter vector, θ_0

Multivariate normal distribution is assumed for the parameters using covariance matrix estimated from MLE, C

```
%%step 1 initialization: draw a starting point for model
parameters,define error variance and covariance scale for
the jumping distribution

C=tcov;
x0 = mvnrnd(theta,C,chainnumber) ; % sampled from joint
multivariate normal distribution using covariance C
sigma2=s2; %error variance as estimated from MLE
npar=length(theta);
covscale=2.4/sqrt(npar); %covariance scale
```

Monod example-
MCMC implementation

Step 2. do markov chain simulations to sample from target distribution

Recall that the target distribution is a product of prior and likelihood functions.

$$p(\theta \mid y) \propto p(\theta)\, p(y \mid \theta)$$

The likelihood function is defined as follows (prior were as defined previously):

$$p\left(y \mid \theta, \sigma^2\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2} SS(\theta)\right)$$

Where $$SS(\theta) = \sum_i^n \left(y - f(S, \theta)\right)^2$$

The metropolis ratio becomes then:

$$r = \frac{p\left(\theta^* \mid y, \sigma^2\right)}{p\left(\theta^{k-1} \mid y, \sigma^2\right)} = \exp\left(-\frac{1}{2\sigma^2}\left(SS\left(\theta^*\right) - SS\left(\theta^{k-1}\right)\right)\right)$$

Monod example-
MCMC implementation

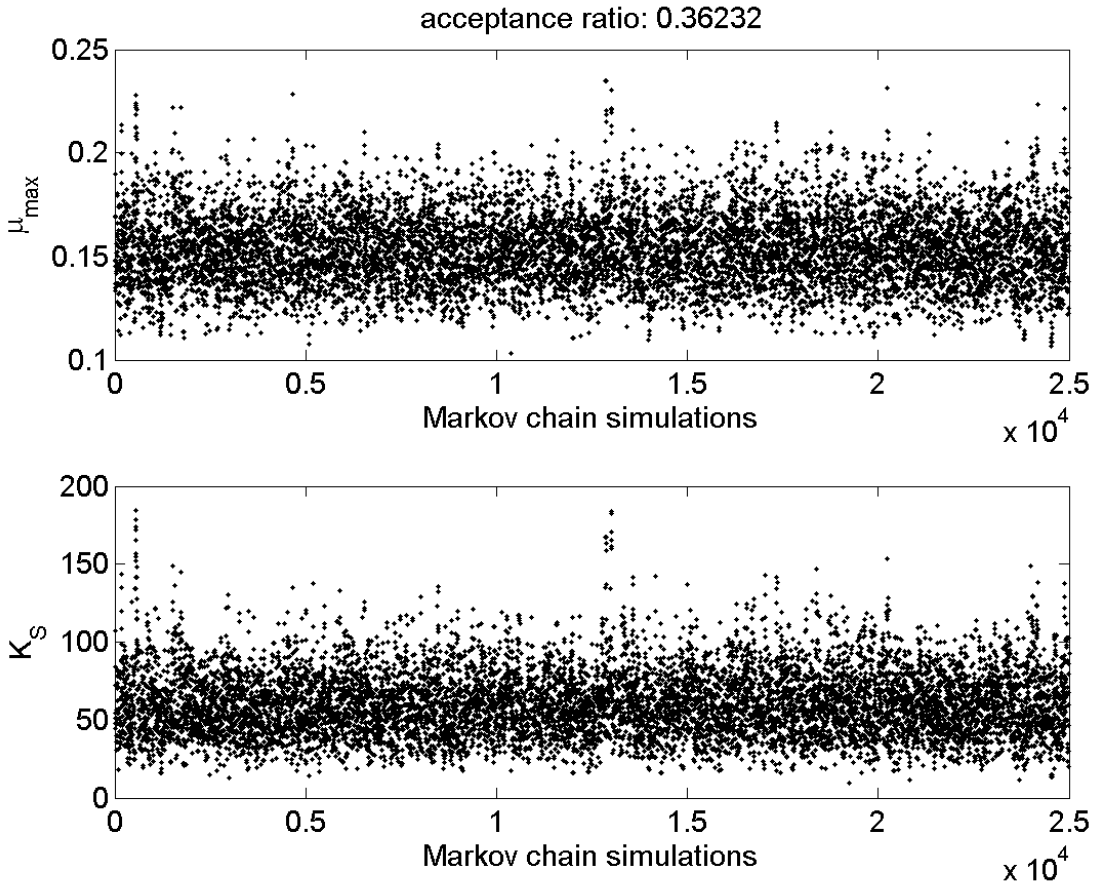
Step 2. Matlab script of this part of the code

```
%%step 2: perform mcmc sampling
for j=1:chainnumber
    x=x0(j,:);
       for i=1:samplingnumber
        %2.a sample a proposal from jumping distribution
        u = randn(1,npar); %random draw
        dx=u*R*covscale; % incremental walk
         xnew=x+dx;  %new proposal
        %2.b calculate the ratio of densities
         ssnew=ssfun(xnew,data); sscur=ssfun(x,data);
         rm= exp(-0.5*((ssnew-sscur)./sigma2) ); % metropolis ratio

         %2.c reject/accept new sample
         if min(rm,1) > random('unif',0,1) % probability min(r,1)
             x=xnew ;
          else
             x=x;
         end
end
```

Monod example-
MCMC implementation

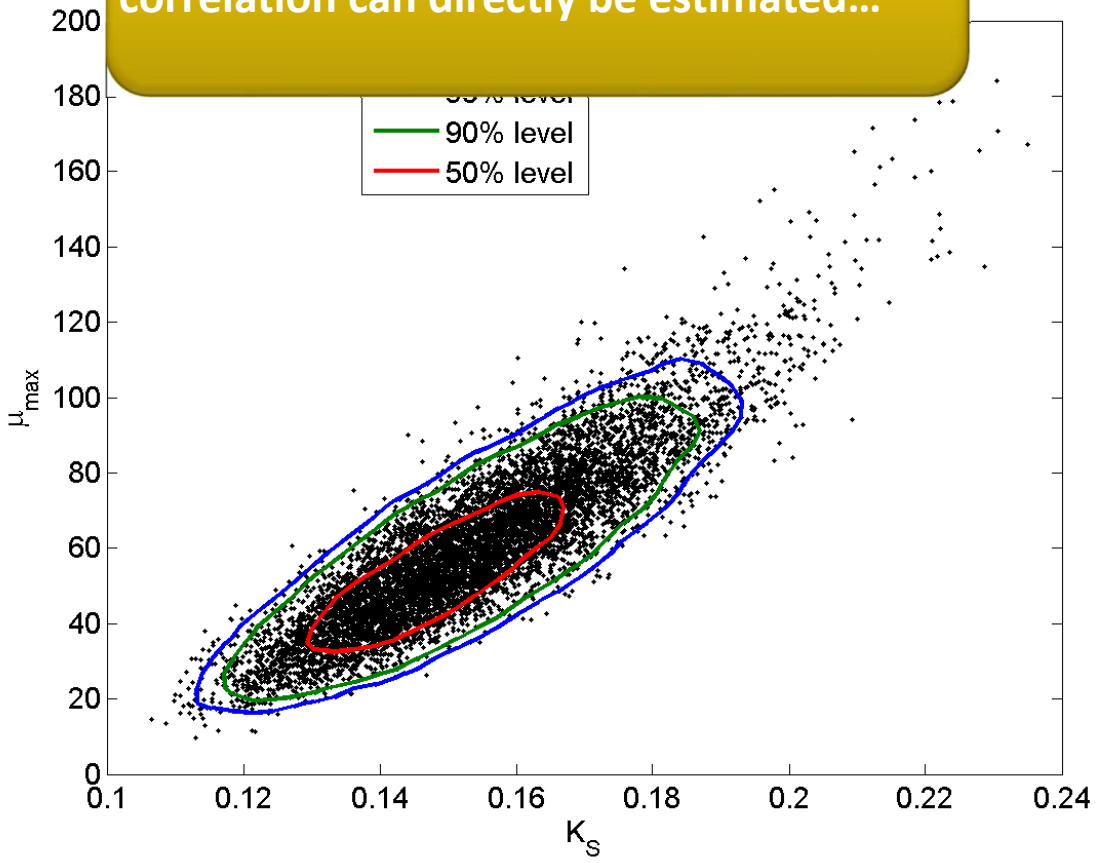
- **Step 2.** Results of mcmc sampling using metropolis algorithm what is shown is raw data from simulations using 5 markov chains with 5000 sampling



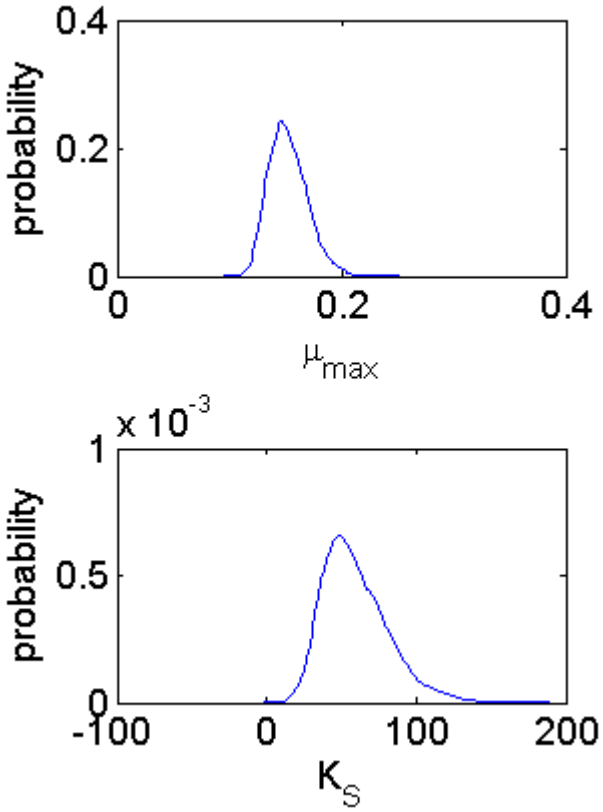
Monod example-
MCMC implementation

- of markov chain simulations provides
pl posterior distribution for μ^{max} and K_S

Uncertainty characterised entirely all relevant statistics (mean, standard dev, correlation can directly be estimated…



posterior marginal density



Monod example
posterior simulations

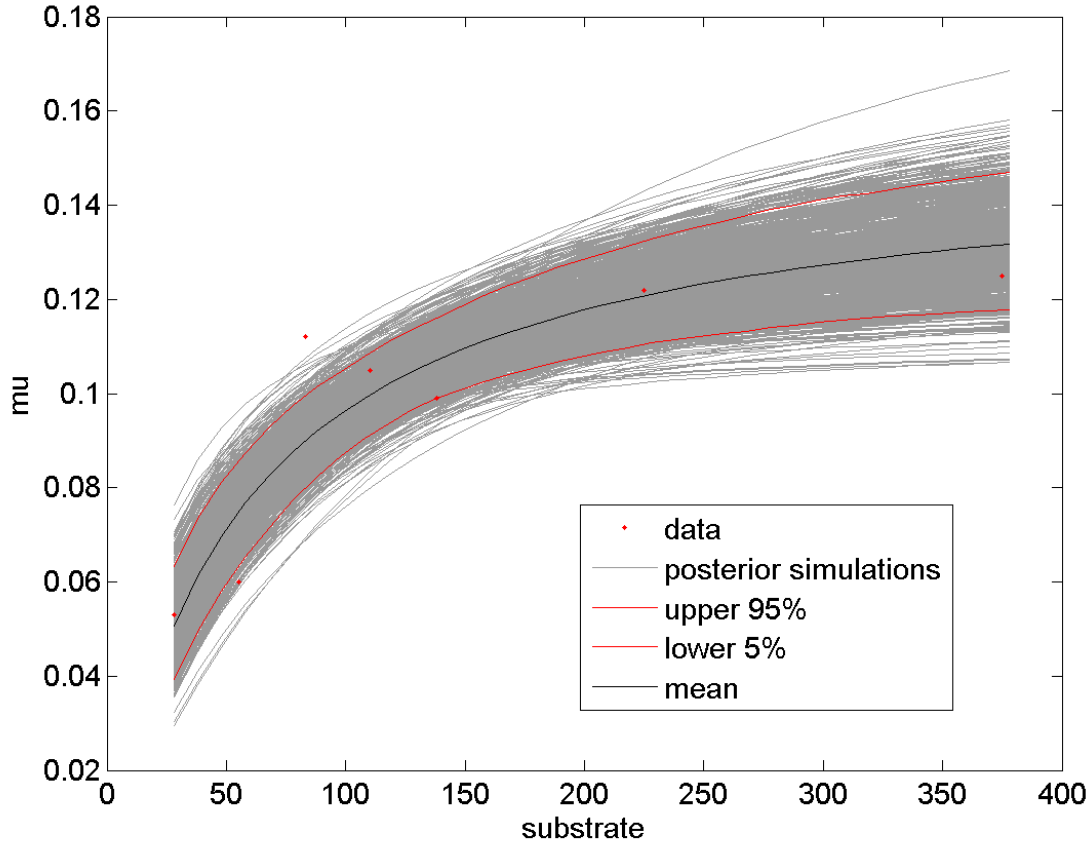
- Posterior simulations for the model prediction can be performed by drawing randomly from posterior joint distribution of the model parameters (see previous slide)

```
%% posterior simulations
% randomly sample from joint posterior distribution
chainb = [y1b y2b]; postsample = 1000;
s=data.s(1):10:data.s(end)+10; % calculate the function at
different substrate conc
posttheta = chainb(randi(length(chainb), [postsample
1]),:); % randomly draw from the chain nsample values
ypost=[];
for i=1:postsample
    ypost(i,:) = modelfun(s,posttheta(i,:));  % model
parameter uncertainty
end
for i=1:length(s)
    [fp,xp,flo,fup] = ecdf(ypost(:,i));
    x95(i,1)=xp(find(fp<0.95,1,'last')); % Pr(x<X)= 0.95
    x05(i,1)=xp(find(fp<0.05,1,'last'));  % Pr(x<X)=0.05
end
```

Monod example posterior simulations

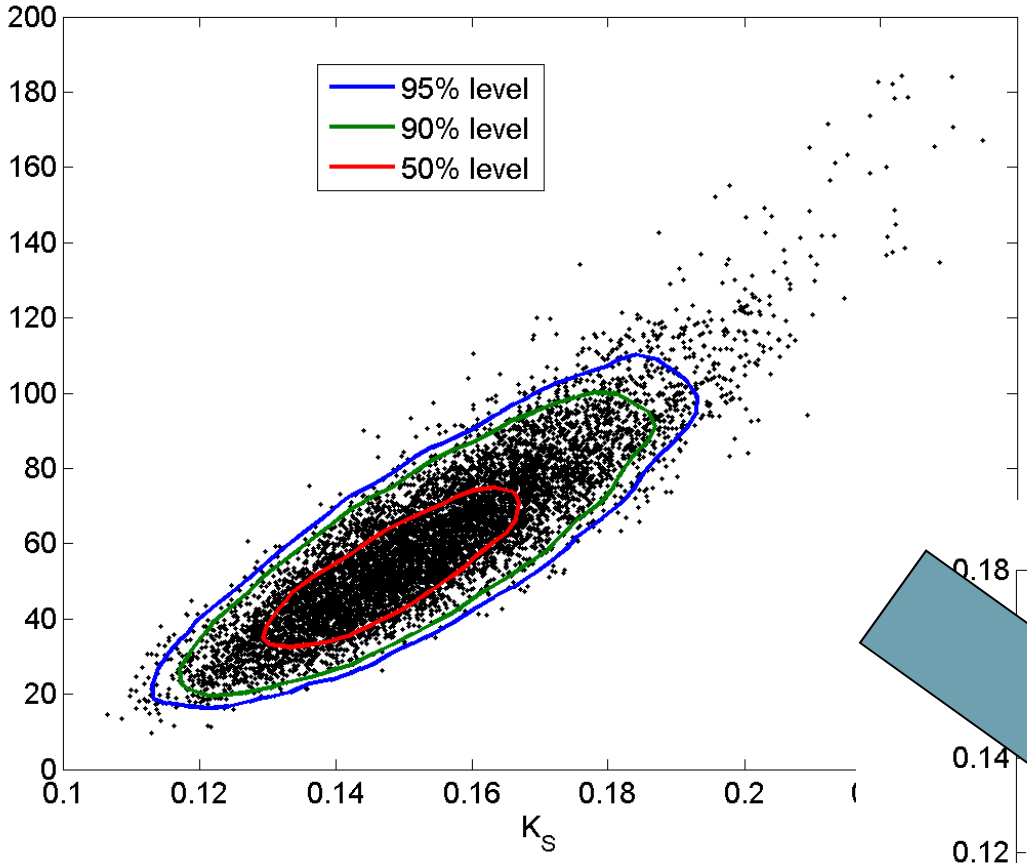
- Posterior simulations with samp ion of the model parameters

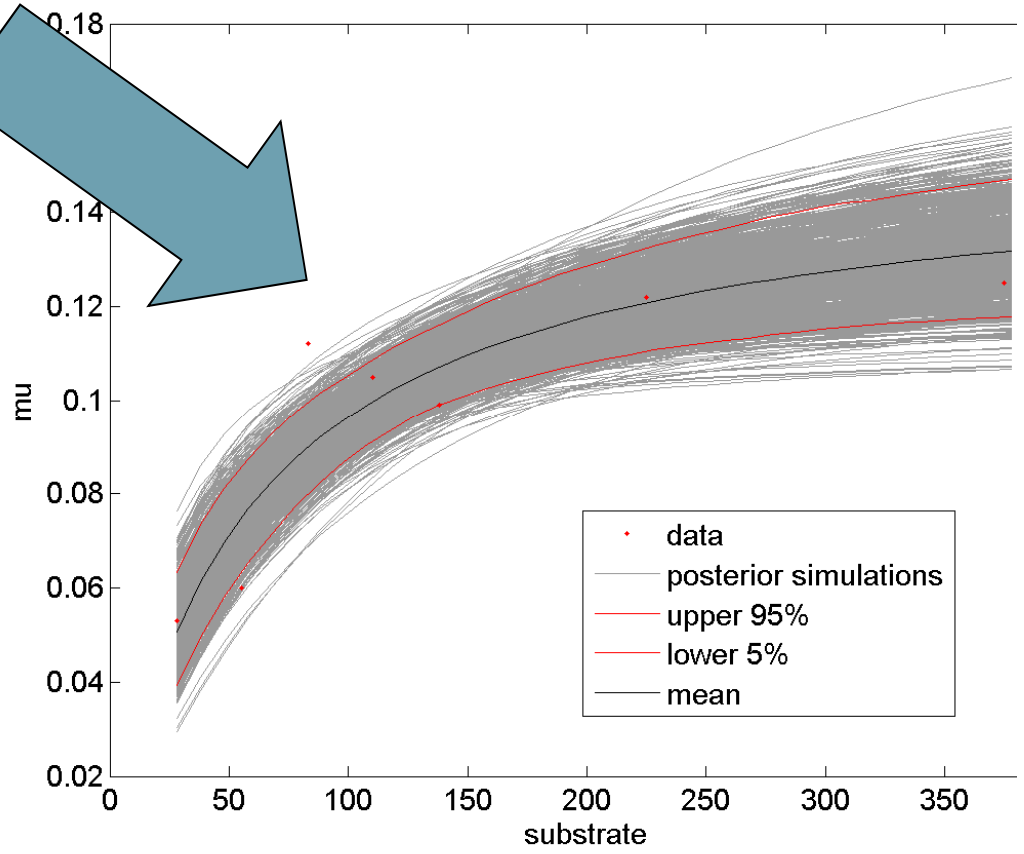
This concludes the uncertainty analysis of modelling exercise from model parameters to error structure and finally to the predictions of the model….



Bayesian estimation

Mean estimates	
μ_{max}	K_s
0.1526	59.6541
Covariance: Cov	
0.0003	0.3211
0.3211	437.7008
Correlation: Cor	
1	0.9005
0.9005	1
Sample variance,$s^2=\sigma^2$	
1.6335e-04	


95% level
90% level
50% level
200
180
160
140
120
100
80
60
40
20
0
0.1
0.12
0.14
0.16
0.18
0.2
K_S


0.18
0.14
0.12
0.1
0.08
0.06
0.04
0.02
mu
0
50
100
150
200
250
300
350
substrate
data
posterior simulations
upper 95%
lower 5%
mean

Monod example
posterior simulations – convergence statistics

- Convergence statistics via potential scale reduction factor, R

This convergence statistics indicate the potential scale reduction for the current distribution if the sampling were to continue to infinity , i.e. n →∞

  – The sampling is said to converge if R is close to 1

- Monte carlo error is defined as the error of integration which tends to 0 as n →∞. Hence error on the mean estimate of the distribution is σ/sqrt(n)

Convergence statistics		
	μ_{max}	K_s
R	1.0132	1.0136
MC error	0.0002	0.2715

convergence achieved & MC error is low!

Bayesian inference in practice – slowly but surely picking up in engineering community!

Bayesian inference is a thriving area of research thanks to increasing availability of computational power as well as increasing of size and diversity of community (from statistics to engineering in various fields) working on this area

There are some software: BUGS a leading community in this area (both scientifically and software wise).For engineering PE, see the r-project called DREAM: http://dream.r-forge.r-project.org/

Like any other problem solving, when using Bayesain inference, one needs to select the relevant probability models to carry out the analysis. This includes:

- Likelihood function (a probability model)
- Error models (a probability model)
- Prior functions (a probability function)
- Jumping function and covariance scale for the parameters (update necessary or not? Then using adaptive algorithms such as AM)
- Check convergence statistics (especially acceptance ratio and some variance estimation errors see Gelman et al 2004)

In either case, Bayesian inference provides an improved understanding of sources of uncertainty and how they can be approximated. This is quite an added value.

Uncertainty analysis via Bayesian inference

EXERCISE: AEROBIC GROWTH OF MICROORGANISMS

Exercise

- Please go step by step analysis shown for the Monod example above and apply bayesian inference for the parameter estimation problem in OUR case!
- Use the parameter estimation results from lecture 1.2 as a starting point for MCMC sampler. Follow then the steps outlined in slide 10.
- The parameter estimation problem was given the OUR example of L1.2. Please check it for the necessary information about data and model to be used.

Copyright notice of Matlab scripts

Everything contained in this directory and its subdirectories is the property of Department of Chemical and Biochemical Engineering, Technical University of Denmark.

Copyright notice:

- The code of these files is free software. Users may modify and redistribute this code. The sole restrictions are that:
- In the academic spirit of collaboration, the source code should be appropriately acknowledged in the resulting scientific disseminations. You may cite this code as follows: "Sin G, Lantz AE, Gernaey KV. Good Modeling Practice for PAT Applications: Propagation of Input Uncertainty and Sensitivity Analysis. Biotechnology progress. Biotechnol. Prog., 25: 1043–1053,2009.DOI 10.1021/bp.166."
  - (ii) the source code must accompany compiled distributions of this code, and
  - (iii) this notice must remain unchanged.
- Please report any bugs related to the code to the following email gsi@kt.dtu.dk
- Copyright Gürkan Sin DTU Chemical Engineering, Lyngby, DK.

Solution to OUR exercise
setting up Bayesian PE problem

- Foremost we choose our probability models for bayesian inference (these are commonly made choices. Other pdf can also be assumed and Bayesian PE can be repeated several times):
  - Likelihood function: multivariate normal pdf $p\left(y \mid \theta, \sigma^2\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2} SS\left(\theta\right)\right)$
  - Metropolis ratio: $r = \frac{p\left(\theta^* \mid y, \sigma^2\right)}{p\left(\theta^{k-1} \mid y, \sigma^2\right)} = \exp\left(-\frac{1}{2\sigma^2}\left(SS\left(\theta^*\right) - SS\left(\theta^{k-1}\right)\right)\right)$
  - Error model: Chisq distribution $\frac{\left(n_0 - 1\right)s_0^2}{\sigma_0^2} \sim \chi^2\left(n_0 - 1\right)$
  - Prior functions: multivariate normal pdf $p\left(\theta \mid \mu, \Sigma\right) = \left(2\pi\right)^{-d/2} \left|\Sigma\right|^{-1/2} \exp\left(-\frac{1}{2}\left(\theta - \mu\right)^T \Sigma^{-1}\left(\theta - \mu\right)\right)$
  - Jumping function : gaussian normal pdf $J\left(\theta^* \mid \theta^{k-1}, \Sigma^{k-1}\right) \sim N\left(\theta^{k-1}, \Sigma^{k-1}\right)$
  - Covariance scale: $\Sigma = 2.4^2 / d * \Sigma^{prior}$

Solution to OUR exercise

- The solution is coded in the folders our exercise3par and our exercise4par

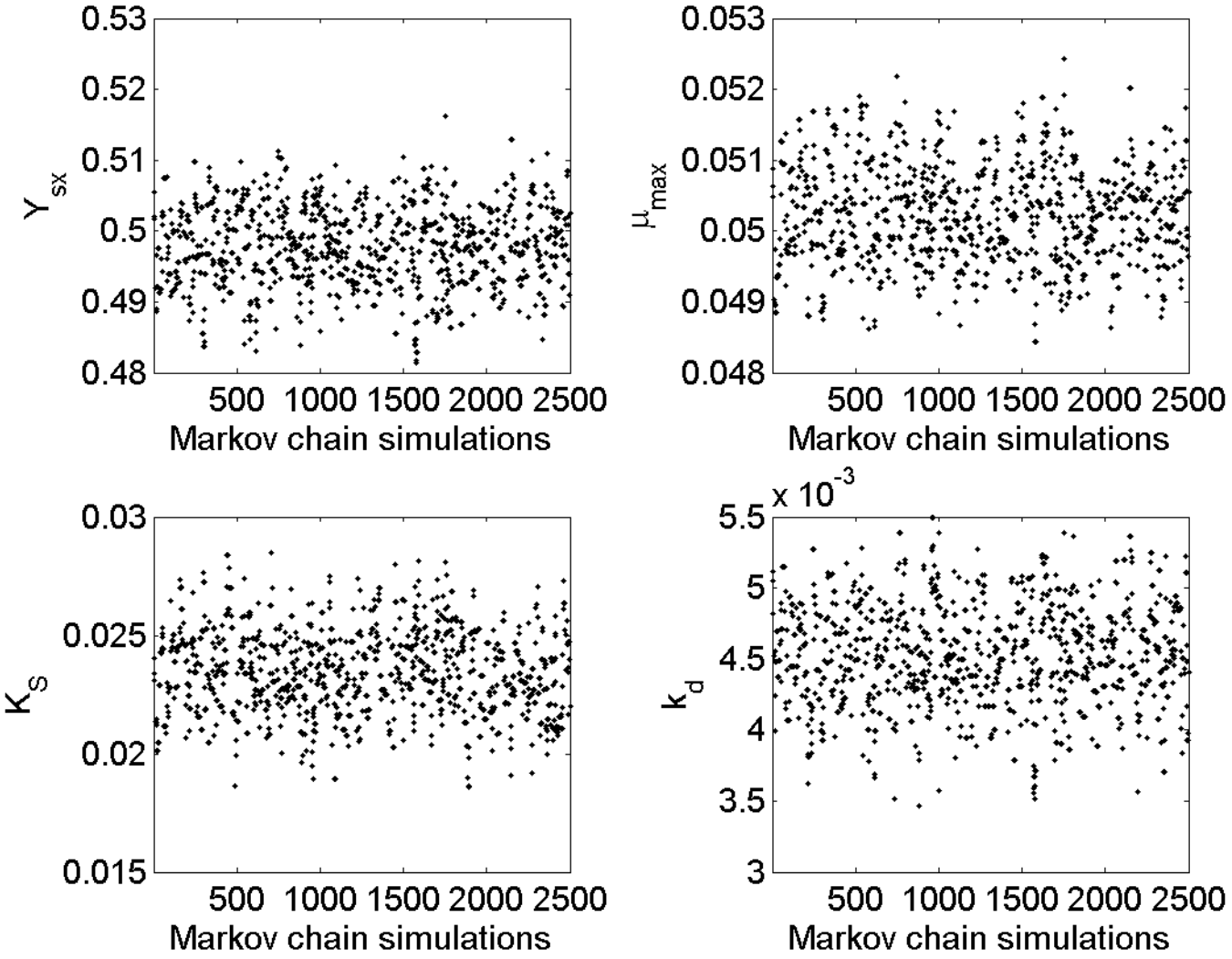
Parameters	Mean	std	Correlation matrix			
Y_{SX}	0.4978	0.0057	1	0.847	-0.2211	0.833
μ_{max}	0.0502	0.0007	0.847	1	0.2632	0.7049
K_S	0.0232	0.0018	-0.2211	0.2632	1	-0.187
kd	0.0045	0.0004	0.833	0.7049	-0.187	1

MLE

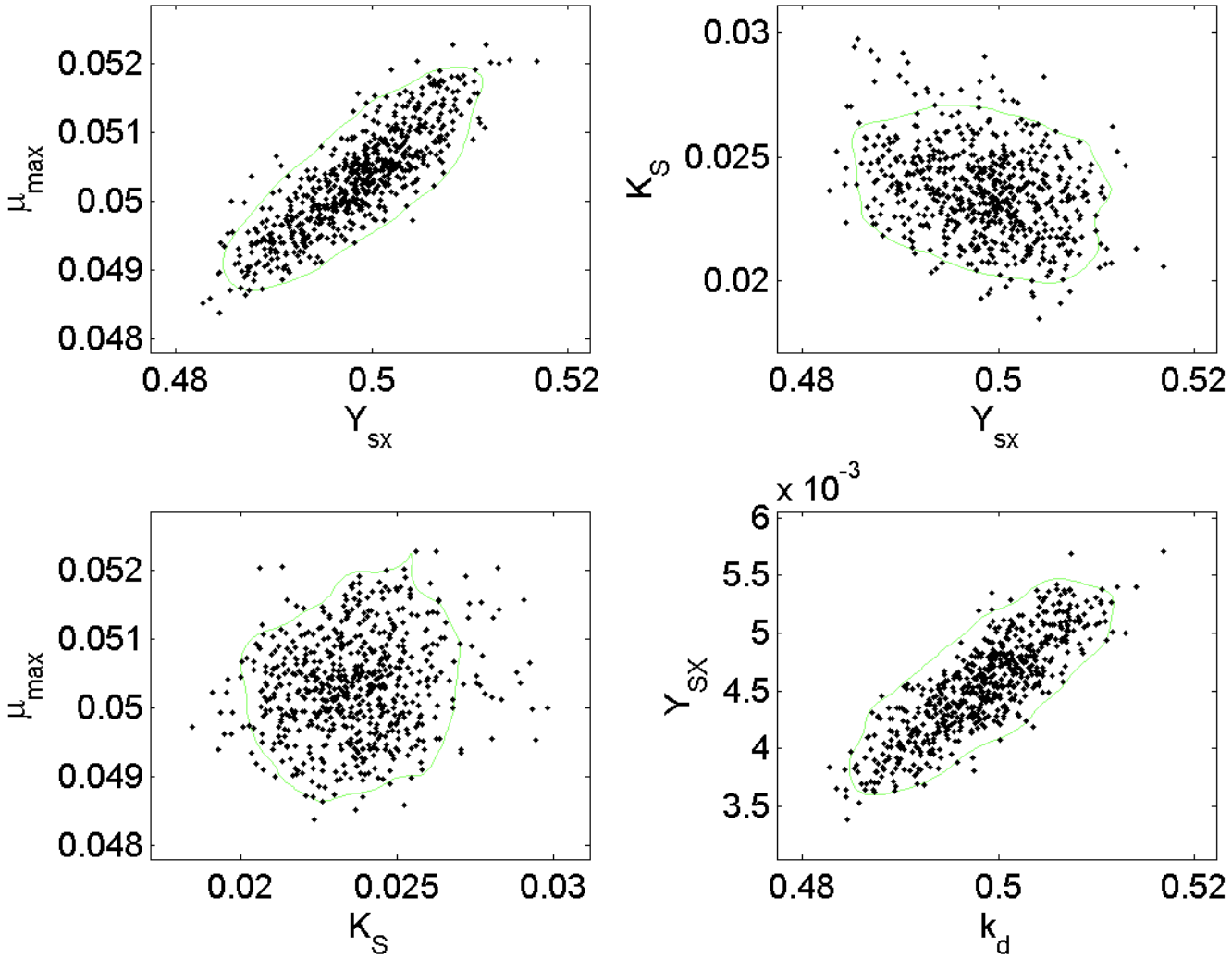
Bayesian

Parameters	Mean	std	Correlation matrix			
Y_{SX}	0.4983	0.0059	1	0.8555	-0.3146	0.8456
μ_{max}	0.0503	0.0007	0.8555	1	0.1546	0.7351
K_S	0.0236	0.0017	-0.3146	0.1546	1	-0.2772
kd	0.0045	0.0004	0.8456	0.7351	-0.2772	1

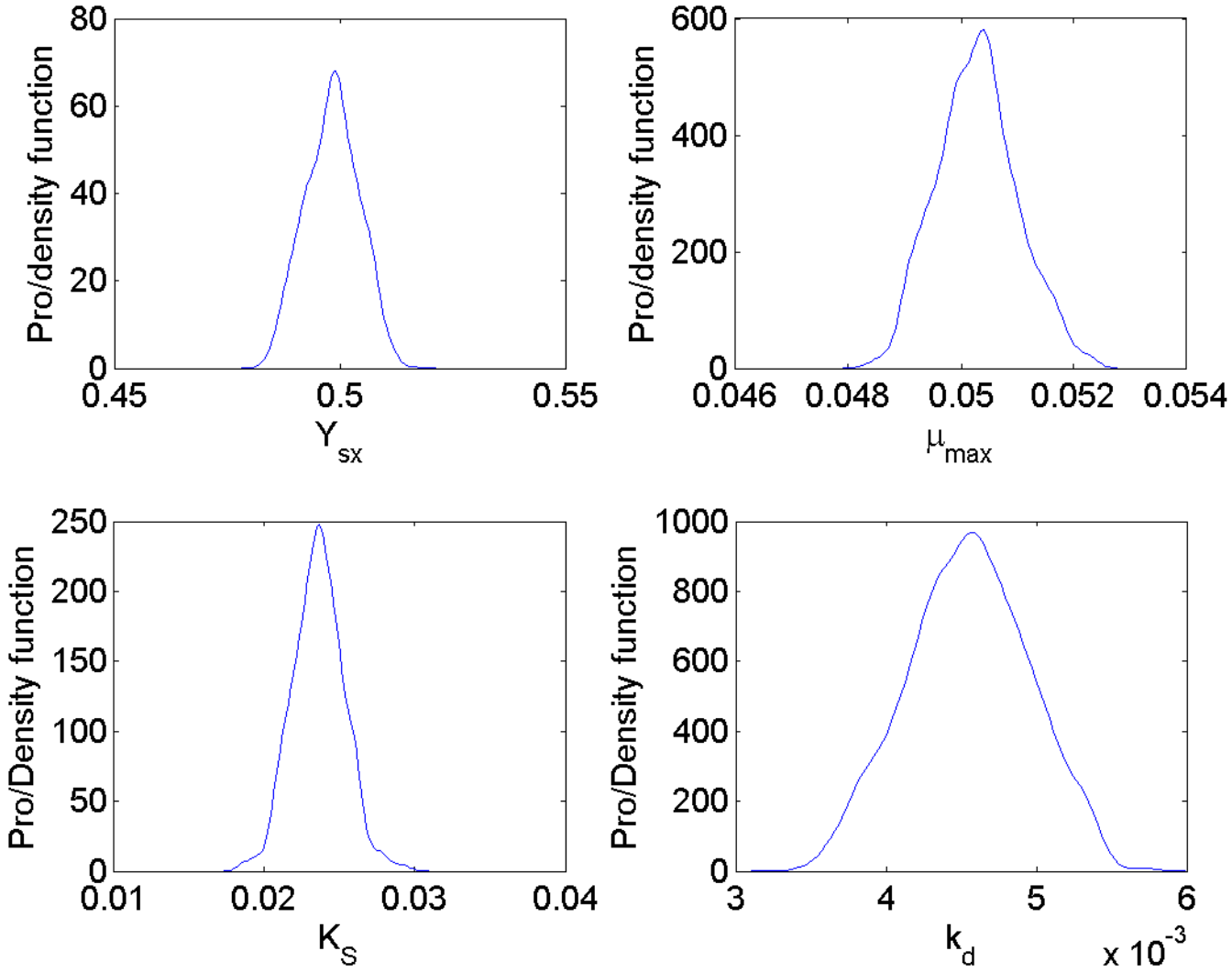
Solution to OUR exercise: MCMC sampling results



Confidence ellipsoids



Posterior densities of parameter estimates



Posterior simulations