Parameter estimation, Monte Carlo methods, parametric Uncertainty Quantification (UQ)

Heikki Haario
LUT, Lappeenranta University of Technology,
FMI, Finnish Meteorological Institute

Outline

1. Model types and their uncertainties
2. **Linear, nonlinear models**, classical error analysis, design of experiments
3. **Monte Carlo:** MCMC, the Metropolis algorithm
4. Demos

Mathematical Models

Mathematical modeling is a central tool in most fields of science and engineering. Mathematical models can be either

- **Mechanistic:** based on principles of natural sciences. Also known as 'physiochemical' or 'physics-based' or 'hard' models.
- **Empirical:** inferring relationships between variables directly from the available data. Also known as 'data-driven' or 'soft' models.

Empirical models try to learn the relationship between input variables $\mathbf{X}$ and output variables $\mathbf{Y}$ using the empirical data alone.

$$\mathbf{X} = \begin{array}{c} \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_p \end{pmatrix} \\ \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix} \end{array} \quad \mathbf{Y} = \begin{array}{c} \begin{pmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \dots & \mathbf{y}_q \end{pmatrix} \\ \begin{pmatrix} y_{11} & y_{12} & \dots & y_{1q} \\ y_{21} & y_{22} & \dots & y_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \dots & y_{nq} \end{pmatrix} \end{array}.$$

Empirical Models

+ Often result in easier computation.
- Possibly problems in interpreting the modeling results.
- Usually poorly extendable to new situations from which measurements are not available (extrapolation).
+ Often the preferred choice, if a 'local' description (interpolation) is enough or if the mechanism is not known or too complicated to be modeled in detail.

The methodology used in analyzing empirical model is often called *regression analysis* (today: neural nets)

Mechanistic Models

Built on first principles of natural sciences, and are often formulated as differential equations. For instance, the elementary chemical reaction $A \to B \to C$ can be modeled as an ODE system

$$\begin{aligned}
\frac{dA}{dt} &= -k_1 A \\
\frac{dB}{dt} &= k_1 A - k_2 B \\
\frac{dC}{dt} &= k_2 B.
\end{aligned}$$

- \- Model building is often more demanding as with empirical models.
- \- Require knowledges of numerical methods.
- \+ Often extends well to new situations.
- \+ Chosen if the phenomenon is understood well enough and if the model is needed outside the experimental region.

Combining Models and Data

- Mathematical models must be verified against measured data.
- The models usually contain some unknown *parameters* that need to be calibrated from the measurements.
- Standard notation:

$$\mathbf{y} = f(\mathbf{x}, \theta) + \varepsilon,$$

  where $\mathbf{y}$ denotes the measurements, $f(\mathbf{x}, \theta)$ is the mathematical model, $\mathbf{x}$ are the control (input, experimental) variables, θ contains the unknown parameters and ε gives measurement error.
- Parametric uncertainty (UQ) at 'model fitting' stage: uncertainty in the data $\mathbf{y}$ implies uncertainty in the parameter values θ.

UQ for linear and nonlinear models

- Mathematical models are either linear or nonlinear.
- Here, we mean linearity with respect to the parameters θ.
- For instance $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2$ is linear, while $y = \theta_1(1 + \exp(-\theta_2 x))$ is nonlinear.
- Dilemma: most of the classical statistical theory is for linear models, while most of physical phenomena are nonlinear.
- Solution by Monte Carlo methods: let the computer create the statistics in 'histogram sense', by repeatedly evaluating the model at different parameter values.

Parameter Estimation

- Traditionally, point estimates for the parameters are obtained by solving a least squares (LSQ) optimization problem.
- In LSQ, we search for parameter values $\hat{\theta}$ that minimize the sum of squared differences between the observations and the model:

$$SS(\theta) = \sum_{i=1}^{n} [y_i - f(x_i, \theta)]^2.$$

- The LSQ method does not say anything about the uncertainty of the estimator, which this is the purpose of statistical analysis in modeling.
- The theory for statistical analysis of linear models is well established, and often used as an approximation for nonlinear models as well.
- Efficient numerical Monte Carlo techniques have been introduced lately to allow full statistical analysis for nonlinear models.

Prequisities: statistical distributions

- The PDF of the univariate normal (or Gaussian) distribution with mean x_0 and variance σ^2 is

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{x - x_0}{\sigma}\right)^2\right).$$

  For a random variable x that follows the normal distribution, we write $x \sim \mathrm{N}(x_0, \sigma^2)$.

- The sum of n univariate Gaussian random variables, $s = \sum_{i=1}^{n} x_i^2$, follows the *chi-square* distribution with n degrees of freedom, which we denote by $s \sim \chi_n^2$.

- The PDF of the d-dimensional Gaussian distribution with mean vector $\mathbf{x}_0$ and covariance matrix Σ is

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \Sigma^{-1} (\mathbf{x} - \mathbf{x}_0)\right),$$

  where $|\Sigma|$ denotes the determinant of Σ.

Visualizing Gaussian Densities as Ellipses

- The contours of multivariate Gaussian densities are ellipsoids, where the principal axes are given by the eigenvectors of the covariance matrix, and the eigenvalues correspond to the lengths of the axes.

- Therefore, two-dimensional covariance matrices can be visualized as ellipses.

- In practice, this can be done with the `ellipse` function given in the code package.

- Check out the demo program `ellipse_demo.m`.

Linear Models

- Let us consider a linear model
  $f(\mathbf{x}, \theta) = \theta_0 + \theta_1 \mathbf{x}_1 + \theta_2 \mathbf{x}_2 + \ldots + \theta_p \mathbf{x}_p$.
- Assume that we have noisy measurements $\mathbf{y} = (y_1, y_2, ..., y_n)$ obtained at points $\mathbf{x}_i = (x_{1i}, x_{2i}, ..., x_{ni})$ where $i = 1, ..., p$.
- Now, we can write the model in matrix notation:

$$\mathbf{y} = \mathbf{X}\theta + \varepsilon,$$

where $\mathbf{X}$ is the design matrix that contains the measured values for the control variables, augmented with a column of ones to account for the intercept term θ_0:

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1p} \\ 1 & x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{np} \end{pmatrix}.$$

Linear Models

- For linear models, we can derive a direct formula for the LSQ estimator.
- The LSQ estimate, that minimizes $SS(\theta) = ||\mathbf{y} - \mathbf{X}\theta||_2^2$, is obtained as the solution to the *normal equations* $\mathbf{X}^T\mathbf{X}\theta = \mathbf{X}^T\mathbf{y}$:

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

- To obtain the statistics, we can compute the covariance matrix $\mathrm{Cov}(\hat{\theta})$.
- Assume i.i.d. measurement error: $\mathrm{Cov}(\mathbf{y}) = \sigma^2\mathbf{I}$.
- Then, we can show (exercise) that

$$\mathrm{Cov}(\hat{\theta}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}.$$

Linear Models

- Let us further assume that the measurement errors are Gaussian.
- Then, we can also conclude that the distribution of $\hat{\theta}$ is Gaussian, since $\hat{\theta}$ is simply a linear transformation of a Gaussian random variable $\mathbf{y}$.
- That is, the unknown parameter follows the normal distribution with mean and covariance matrix given by the above formulae: $\theta \sim \mathrm{N}(\hat{\theta}, \sigma^2(\mathbf{X}^T\mathbf{X})^{-1})$.
- The probability density of the unknown parameter can be written as

$$p(\theta) = K \exp\left(-\frac{1}{2}(\theta - \hat{\theta})^T \mathbf{C}^{-1}(\theta - \hat{\theta})\right),$$

where $K = ((2\pi)^{d/2}|\Sigma|^{1/2})^{-1}$ is the normalization constant and $\mathbf{C} = \mathrm{Cov}(\hat{\theta})$.

Design of Experiments

- So far, we have assumed that the measurements for the input and output variables $\mathbf{X}$ and $\mathbf{y}$ are given.
- The task of Design of Experiments (DOE) is to figure out how to choose $\mathbf{X}$ so that maximal information about θ is obtained with minimal experimental effort.
- In linear models, the covariance matrix of the parameters is $\mathrm{Cov}(\hat{\theta}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$.
- That is, the uncertainty in $\hat{\theta}$ depends on the noise level σ^2 and on the design matrix $\mathbf{X}$ (not on $\hat{\theta}$).
- This suggests that, for linear models, one can derive general, 'case-independent' theory about how to choose $\mathbf{X}$.

Nonlinear Models

- For nonlinear models, no direct formulas for statistics are available, one has to resort to numerical methods and different approximations.
- Let us consider a nonlinear model $\mathbf{y} = f(\mathbf{x}, \theta) + \varepsilon$
- To compute the LSQ estimate, one has to numerically minimize the sum of squares

$$l(\theta) = \sum_{i=1}^{n} [y_i - f(x_i, \theta)]^2.$$

- For simple models, one can use standard optimization routines in computational software packages. For this course, the MATLAB gradient-free nonlinear simplex optimizer `fminsearch` is enough.
- Let us recall how approximative error analysis can be performed for the parameters of a nonlinear model.

Nonlinear Models: classical error analysis

- The first three terms of the Taylor series expansion for $l(\theta)$ at a point $\hat{\theta}$ can be written as

$$l(\theta) \approx l(\hat{\theta}) + \nabla l(\hat{\theta})^T(\theta - \hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})^T \mathbf{H}(\theta - \hat{\theta}),$$

  where ∇ denotes the gradient and $\mathbf{H}$ is the Hessian matrix.

- The 2nd derivatives, or elements $[\mathbf{H}]_{pq}$ of the Hessian matrix, are

$$\frac{\partial^2 \ell(\hat{\theta})}{\partial \theta_p \partial \theta_q} = 2\sum_{i=1}^{n} \frac{\partial f(x_i, \hat{\theta})}{\partial \theta_p} \frac{\partial f(x_i, \hat{\theta})}{\partial \theta_q} + 2\sum_{i=1}^{n} (f(x_i, \hat{\theta}) - y_i) \frac{\partial^2 f(x_i, \hat{\theta})}{\partial \theta_p \partial \theta_q}.$$

- Assuming that the residuals $f(x_i, \hat{\theta}) - y_i$ are small, the Hessian matrix can be approximated using only first derivatives by dropping the residual terms:

$$[\mathbf{H}]_{pq} = \frac{\partial^2 \ell(\hat{\theta})}{\partial \theta_p \partial \theta_q} \approx 2\sum_{i=1}^{n} \frac{\partial f(x_i, \hat{\theta})}{\partial \theta_p} \frac{\partial f(x_i, \hat{\theta})}{\partial \theta_q}.$$

Nonlinear Models: classical error analysis

- The first derivatives can be collected into a Jacobian matrix $\mathbf{J}$, which has elements

$$[\mathbf{J}]_{ip} = \frac{\partial f(x_i;\theta)}{\partial \theta_p}|_{\theta=\widehat{\theta}}.$$

- Now, the Hessian approximation can be written in a matrix form:

$$\mathbf{H} \approx 2\mathbf{J}^T\mathbf{J}.$$

- Inserting this into the Taylor expansion, and noting that $\nabla l(\hat{\theta}) = \mathbf{0}$, we obtain

$$l(\theta) \approx l(\hat{\theta}) + (\theta - \hat{\theta})^T\mathbf{J}^T\mathbf{J}(\theta - \hat{\theta}).$$

Nonlinear Models: classical error analysis

- For linear models, the least squares expression is

$$l(\theta) = ||\mathbf{y}-\mathbf{X}\theta||^2 = (\mathbf{y}-\mathbf{X}\theta)^T(\mathbf{y}-\mathbf{X}\theta) = \mathbf{y}^T\mathbf{y}-2\mathbf{y}^T\mathbf{X}\theta+\theta^T\mathbf{X}^T\mathbf{X}\theta.$$

- Differentiating the function twice gives the Hessian matrix $\mathbf{H} = \mathbf{X}^T\mathbf{X}$, and the Taylor expansion is

$$l(\theta) = l(\hat{\theta}) + (\theta - \hat{\theta})^T\mathbf{X}^T\mathbf{X}(\theta - \hat{\theta}).$$

- Now, compare the nonlinear and linear expressions.
- We observe that **the Jacobian matrix J assumes the role of the design matrix X in the linear case**.
- That is, the covariance matrix of $\hat{\theta}$ can be approximated with

$$\mathrm{Cov}(\hat{\theta}) = \sigma^2(\mathbf{J}^T\mathbf{J})^{-1}.$$

Estimating σ^2

- The measurement error σ^2 can be estimated using repeated measurements.
- Often, however, replicated measurements are not available.
- In this case, the measurement noise can be estimated using the residuals of the fit, using the assumption that *residuals* $\approx$ *measurement error*.
- An estimate for the measurement error can be obtained using the mean square error (MSE):

$$\sigma^2 \approx MSE = RSS/(n-p),$$

where RSS (residual sum of squares) is the minimum of the least squares function, n is the number of measurements and p is the number of parameters.

Code Example: Nonlinear LSQ Fitting and Approximative Error Analysis

- Let us consider estimating the parameters in a model $\mathbf{y} = \theta_1(1 - \exp(-\theta_2 \mathbf{x}))$ using the data $\mathbf{x} = (1, 3, 5, 7, 9)$ and $\mathbf{y} = (0.076, 0.258, 0.369, 0.492, 0.559)$.

- We create two files, the main program `bod_fit.m` and the function `bod_ss.m` that computes the sum of squares objective function that is minimized.

- We use the `fminsearch` optimizer, and compute the Jacobian matrix analytically.

- Check the demo program `bod_fit.m`.

Code Example: Nonlinear LSQ Fitting and Approximative Error Analysis

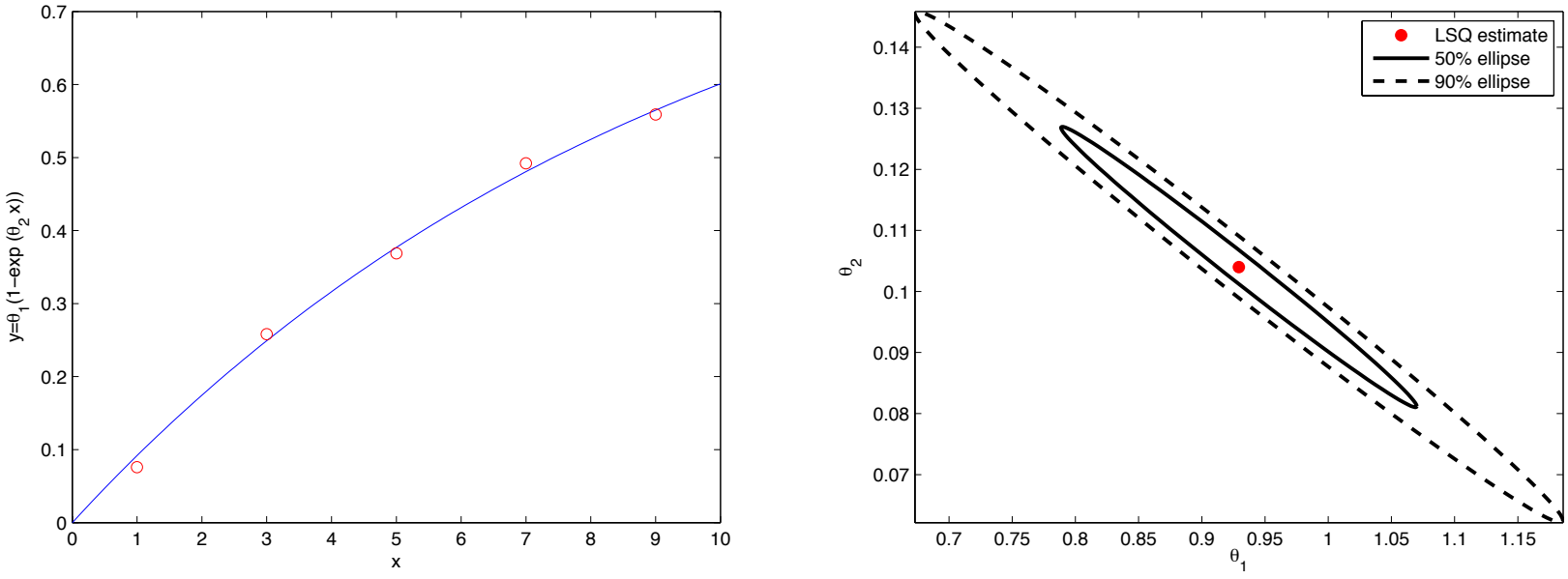


Figure: Left: the model (blue line) fitted to the data (red circles). Right: the LSQ estimates and two confidence ellipses.

Monte Carlo Methods for Parameter Estimation

- The approximative error analysis for nonlinear models described above can give misleading results.
- An alternative way to obtain statistics for parameter estimates is to use various Monte Carlo (MC) random sampling methods.
- Before proceeding to Bayesian estimation and MCMC topics, we briefly present several 'classical' Monte Carlo methods that one can use to evaluate the uncertainty in $\hat{\theta}$.

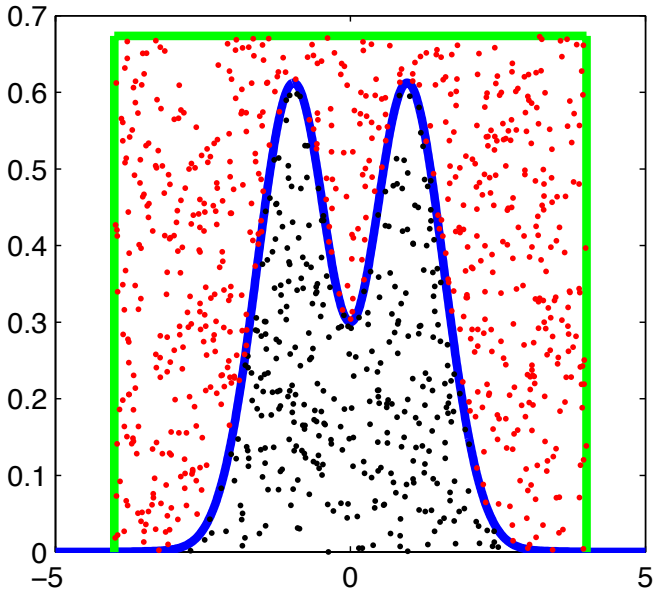
MC methods: Accept-Reject

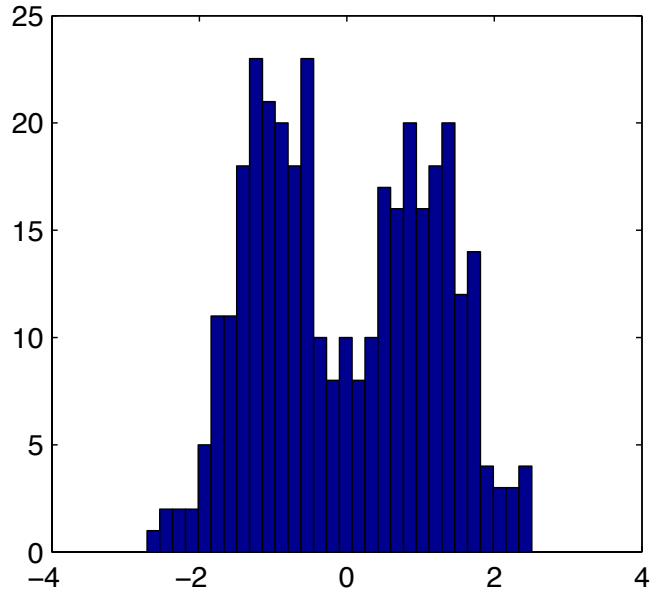
- Suppose that f is a positive (but non-normalized) function on the interval $[a, b]$, bounded by M.
- Consider uniform random points (x_i, u_i) in the 'box' $[a, b] \times [0, M]$
- The points that satisfy $u_i < f(x_i)$ form a uniform sample under the graph of f.
- The area of any slice $\{(x, y)|x_l < x < x_u, y \leq f(x)\}$ is proportional to the number of sampled points in it. So the histogram of the points x_i gives an approximation of the PDF given by f.

MC methods: Accept-Reject

A (very) straightforward algorithm:

1. Sample $x \sim U([a,b])$, $u \sim U([0,M])$.
2. Accept points x for which $u < f(x)$





Works, in principle, in R^n. Problem: how to find values for a, b, M ?

MC methods: Adding Noise to Data

- Uncertainty in the model parameters θ in a model

$$\mathbf{y} = f(\mathbf{x}, \theta) + \epsilon$$

  is caused by the noise ϵ.
- The LSQ fit with given data leads to a single estimated value $\hat{\theta}$.
- So, to obtain a distribution of θ, a natural idea is to generate new data by adding random noise to the existing data and repeatedly fit different values $\hat{\theta}$.

\+ Simple to implement, and can work well, if the noise is correctly generated so that it agrees with the true measurement noise.

\- Often the structure of the noise is not properly known.

\- An iterative optimization needs to be performed after every time new data is generated, which can be time consuming.

\- The results are dependent on the optimizer settings.

Bootstrap

- A popular (non-Bayesian) statistical analysis method, in spirit similar to the above 'adding noise to data' approach.
- No new data is generated, but new random combinations of the existing data are used.
- The basic idea can be written as a pseudo-algorithm as follows:
  1. From the existing data $\mathbf{x} = (x_1, ..., x_n)$, $\mathbf{y} = (y_1, ..., y_n)$, sample new data $\tilde{\mathbf{x}}$, $\tilde{\mathbf{y}}$ with replacement. In practice, select n indices randomly from $1, ..., n$ and choose the data points corresponding to the chosen indices.
  2. Compute the fit using the resampled data $\tilde{\mathbf{x}}$, $\tilde{\mathbf{y}}$.
  3. Go to step 1, until a desired number of θ samples are obtained.
- Bootstrap by residuals: create LSQ fit, get residuals. New fits by swapping residuals between the measurements. Advisable for small data sets especially .

Bootstrap: RTO

- The statistics of the bootstrapped result (parameter posterior) is not known.
- Partial solution by RTO: 'randomize then optimize', (Bardsley et al, see http://helios.fmi.fi/ lainema/): modify the optimization step so that the results coincides with the posterior of Bayesian MCMC (more below)
- Sampling in high dimensional parameter spaces is challenging. So RTO can be a good alternative especially in cases where a numerical solution for an inverse problem already exists, and can be repeatedly used in the RTO optimization.

Bayesian Estimation and MCMC

- In Bayesian parameter estimation, θ is interpreted as a random variable and the goal is to find the *posterior distribution* $\pi(\theta|\mathbf{y})$ of the parameters.
- The posterior distribution gives the probability density for values of θ, given measurements $\mathbf{y}$.
- Using the Bayes' formula, the posterior density is

$$\pi(\theta|\mathbf{y}) = \frac{l(\mathbf{y}|\theta)p(\theta)}{\int l(\mathbf{y}|\theta)p(\theta)d\theta},$$

  where $l(\mathbf{y}|\theta)$ is the *likelihood* and $p(\theta)$ is the *prior* distribution.
- The likelihood gives the probability density of observing $\mathbf{y}$ given the parameter value θ, and the prior contains all existing information about the parameters, such as bound constraints.
- The integral in the denominator is the normalization constant.

Likelihood

- The likelihood function contains the measurement error model.
- Let us consider the model $\mathbf{y} = f(\mathbf{x}, \theta) + \epsilon$ and employ a Gaussian i.i.d. error model, $\epsilon \sim N(0, \sigma^2 \mathbf{I})$
- Noting that $\epsilon = \mathbf{y} - f(\mathbf{x}, \theta)$ gives the likelihood

$$l(\mathbf{y}|\theta) \propto \prod_{i=1}^{n} l(y_i|\theta) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}[y_i - f(x_i, \theta)]^2\right).$$

- That is, the likelihood contains the familiar sum of squares term $SS(\theta) = \sum_{i=1}^{n}[y_i - f(x_i, \theta)]^2$.

Point Estimates

- Different point estimates can be derived from the posterior distribution:
  - *Maximum a Posteriori* (MAP) estimator maximizes $\pi(\theta|\mathbf{y})$.
  - *Maximum Likelihood* (ML) estimator maximizes $l(\mathbf{y}|\theta)$.

- If the prior distribution is uniform within some bounds, ML and MAP coincide.

- With the Gaussian i.i.d. error assumption, ML coincides also with the classical Least Squares (LSQ) estimate.

Bayesian Computation: MCMC

- In principle, the posterior distribution gives the solution to the parameter estimation problem in a fully probabilistic sense.
- We can find the peak of the probability density, and determine, for instance, the 95% credibility regions for the parameters.
- However, working with the posterior density directly is challenging, since we need to compute the normalization constant $\int l(\mathbf{y}|\theta)p(\theta)d\theta$.
- This often cannot be computed analytically, and classical numerical integration methods also become infeasible, if the number of parameters is larger than a few.
- With the *Markov chain Monte Carlo* (MCMC) methods, statistical inference for the model parameters can be done without explicitly computing this difficult integral.

Bayesian Computation: MCMC

- MCMC methods aim at generating a sequence of random samples $(\theta_1, \theta_2, ..., \theta_N)$, whose distribution asymptotically approaches the posterior distribution as N increases.
- That is, the posterior density is not used directly, but samples from the posterior distribution are produced instead.
- The *Monte Carlo* term is used to describe methods that are based on random number generation.
- The samples are generated so that each new point θ_{i+1} only depends on the previous point θ_i, and the samples therefore form a *Markov Chain*.
- Markov Chain theory can be used to show that the samples approach the correct target (posterior).

The Metropolis Algorithm

- One of the most widely used MCMC algorithms.
- Works by generating candidate parameter values from a *proposal distribution* and then either accepting or rejecting the proposed value according to a simple rule.
- The Metropolis algorithm can be written as follows:
  - 1 Initialize by choosing a starting point θ_1
  - 2 Choose a new candidate $\hat{\theta}$ from a suitable **proposal distribution** $q(.|\theta_n)$, that may depend on the previous point of the chain.
  - 3 **Accept** the candidate with probability

$$\alpha(\theta_n, \hat{\theta}) = \min\left(1, \frac{\pi(\hat{\theta})}{\pi(\theta_n)}\right).$$

    If rejected, repeat the previous point in the chain. Go back to step 2.

The Metropolis Algorithm

- One can see that the candidate points that give a higher posterior density value than the previous point (points where $\pi(\hat{\theta}) > \pi(\theta_n)$) are always accepted.
- However, moves 'downward' may also be accepted, with probability given by the ratio of the posterior density values.
- That is, the algorithm randomly jumps around the peak of the target, without converging to a single point.
- In code level, the accept-reject step can be implemented by generating a uniform random number $u \sim U(0, 1)$ and accepting if $u \leq \pi(\hat{\theta})/\pi(\theta_i)$.
- Note that we only need to compute ratios of posterior densities, and the normalization constant (nasty integral) cancels out!

The Metropolis Algorithm

- The problem remaining in the implementation of the Metropolis algorithm is defining the proposal distribution q.
- The proposal should be chosen so that it is easy to sample from and 'close' to the underlying target distribution.
- An unsuitable proposal can lead to inefficient implementation:
  - if the proposal is too large, the new candidates mostly miss the essential support π and are only rarely accepted.
  - if the proposal is too small, the new candidates are mostly accepted, but from a small neighborhood of the previous point, and the chain moves slowly.

The Metropolis Algorithm

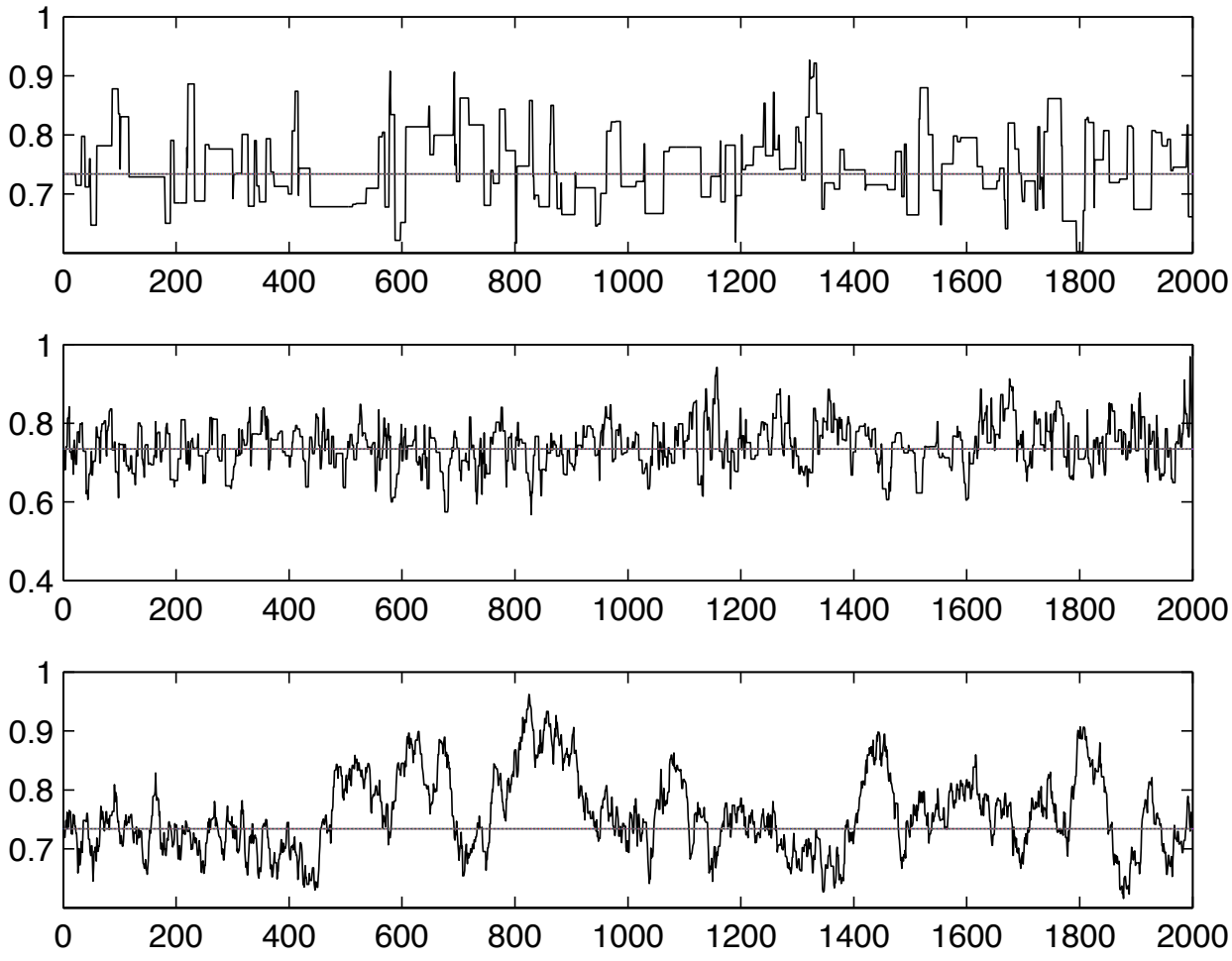


Figure: Top: too wide, bottom: too narrow, middle: good 'mixing'.

The Metropolis Algorithm

- Note that although we assume here a flat prior, it is easy to add possible prior information about the parameters.
- In this course, we will mostly use bound constraints as prior information.
- Implementing simple bound constraints is easy: if the proposed parameter is out of bounds, it is simply rejected.
- The progress of the Metropolis algorithm is animated in the `mcmcmovie` demo program.

The Metropolis Algorithm

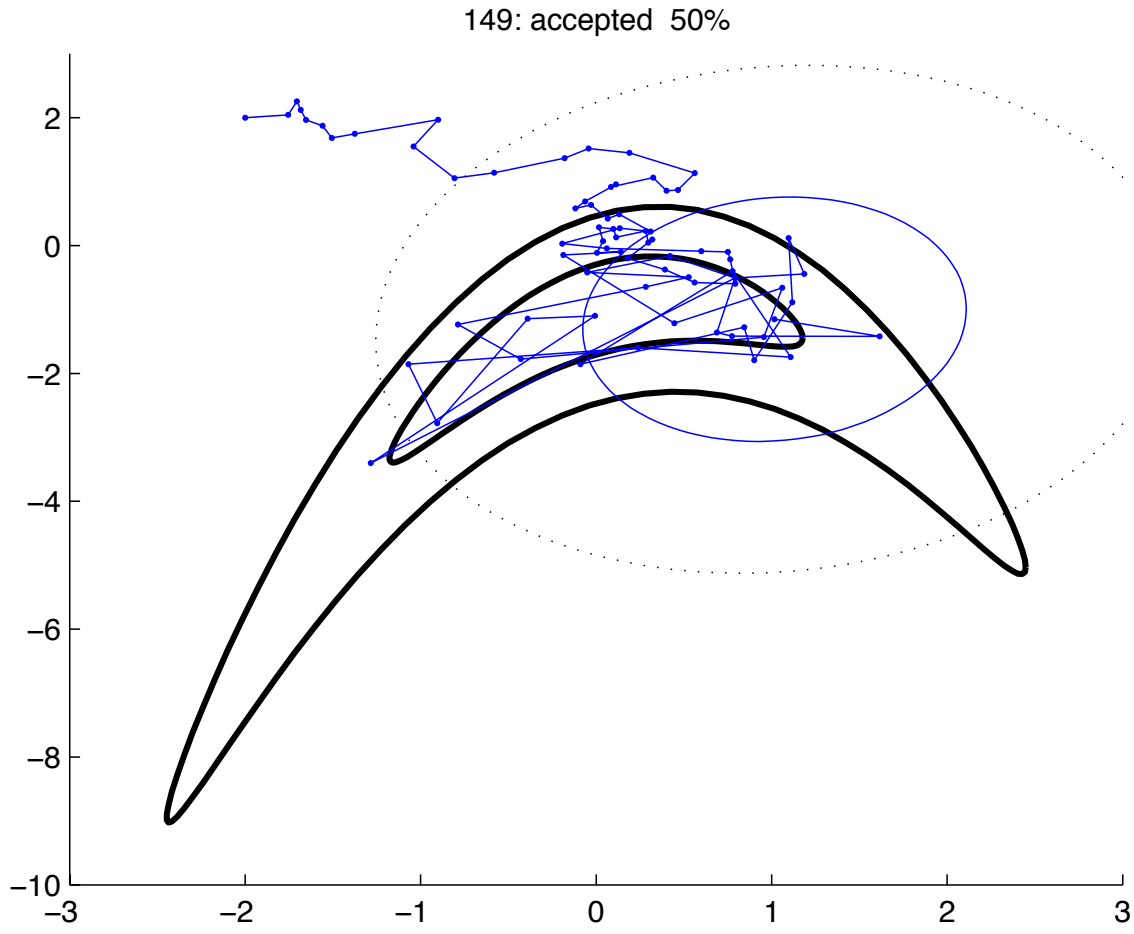


Figure: The path of the Metropolis sampler (blue line), when sampling a non-Gaussian target with a Gaussian proposal distribution (ellipses).

Code example: implementing the Metropolis algorithm

- Let us consider again model $\mathbf{y} = \theta_1(1 - \exp(-\theta_2 \mathbf{x}))$ with data $\mathbf{x} = (1, 3, 5, 7, 9)$, $\mathbf{y} = (0.076, 0.258, 0.369, 0.492, 0.559)$.
- We first compute the LSQ estimate by minimizing the sum of squares, and use that as the starting point for MCMC.
- A spherical proposal covariance $\mathbf{C} = \alpha \mathbf{I}$ is used, where α controls the size of the proposal distribution.
- The measurement error variance is estimated from the residuals using the MSE formula.
- The code is given in two files, the main program `bod_mcmc.m` and the sum of squares objective function `bod_ss.m`.

Code example: implementing the Metropolis algorithm

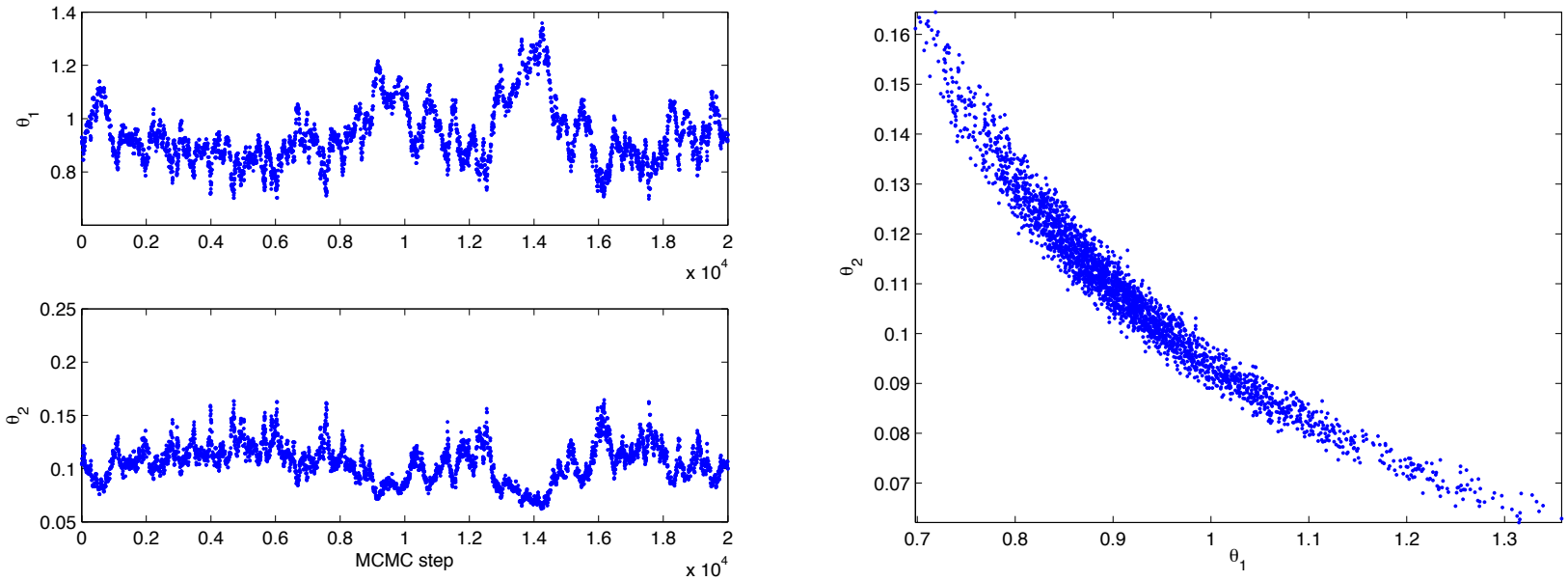
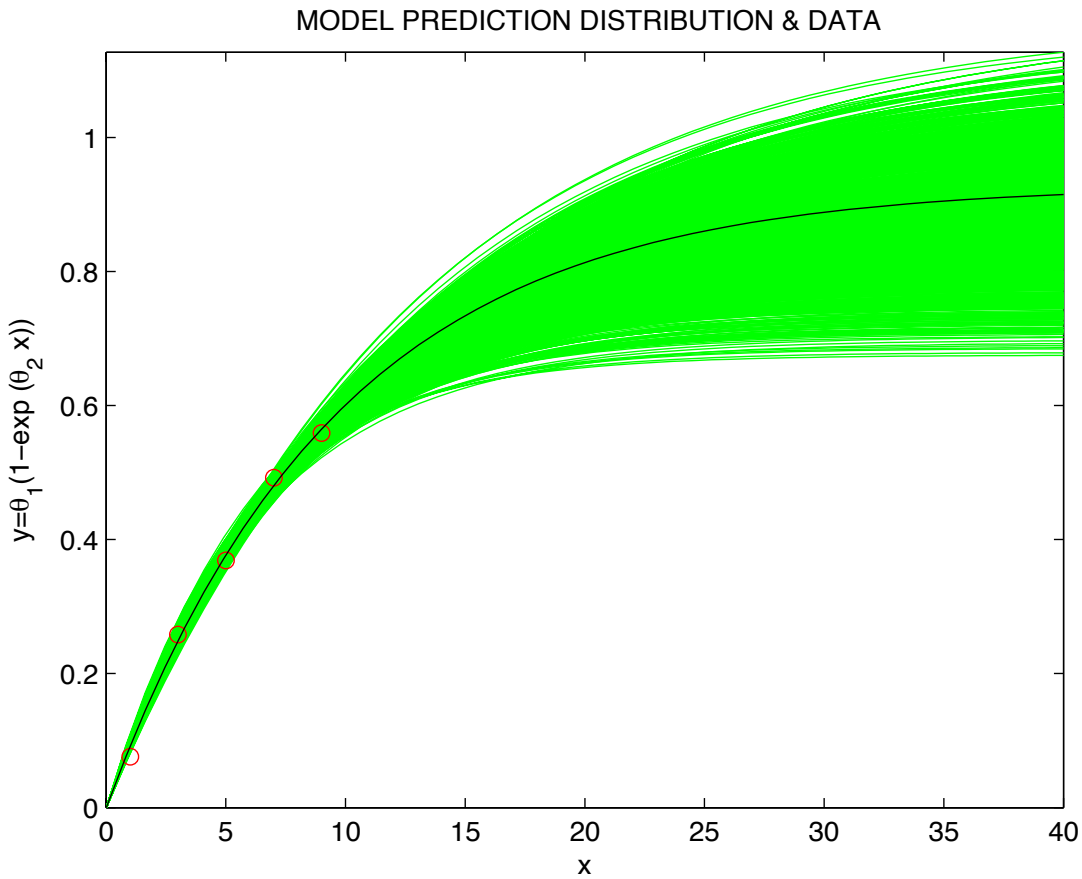


Figure: Left: the path of the MCMC sampler for both parameters. Right: the posterior distribution of the parameters.

Predictive Distributions

In addition to finding the distribution of the model parameters, the MCMC results can be used to simulate the distribution the model predictions, the *predictive distribution*.



Selecting the Proposal Covariance Matrix

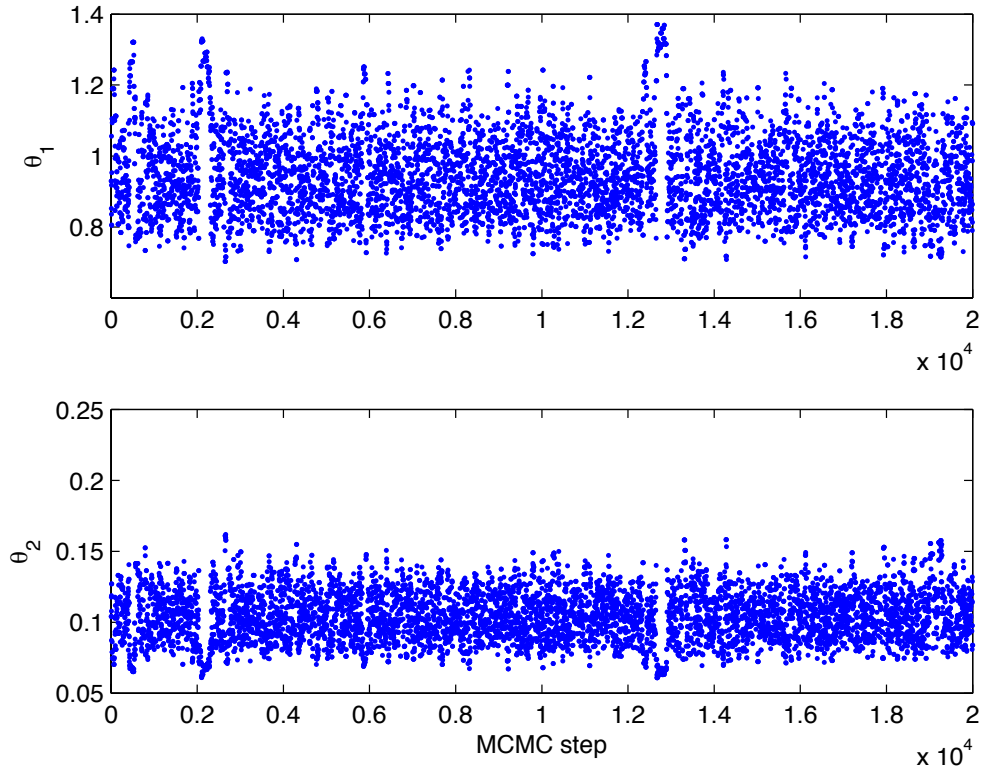


Figure: The path of the MCMC sampler for the two parameters with proposal covariance matrix $\mathbf{C} = s_d \sigma^2 (\mathbf{J}^T \mathbf{J})^{-1}$.

Adaptive Methods: computer 'learns' to optimize $\mathbf{C}$.

Adaptive MCMC

- The bottleneck in MCMC computations is usually selecting a proposal distribution.
- The covariance matrix using the linearization of the model is a good starting point, but does not always lead to efficient sampling.
- The purpose of adaptive MCMC methods is to tune the proposal 'on the run' as the sampling proceeds, using the information of the previously sampled points.
- In the Adaptive Metropolis (AM) algorithm, the proposal covariance matrix is taken to be the empirical covariance matrix computed from the history.
- Note that now the sampled points depend on the earlier history of the chain, and the chain is therefore no longer Markovian.
- However, it can be shown that the AM algorithm gives correct (ergodic) results. See again http://helios.fmi.fi/ lainema/dram/

AM Algorithm

- The choice of the length of the *burn-in period* n_0 is free.
- The adaptation might not be efficient if done at each time step, and one should adapt only at given time intervals.
- Finally, the AM algorithm as a pseudocode:
  - Choose the lenght of the chain N and initialize θ_1 and $\mathbf{C}_1$.
  - For $k = 1, 2, ..., N$
    * Perform the Metropolis step, using proposal $N(\theta_k, \mathbf{C}_k)$.
    * Update $\mathbf{C}_{k+1} = \mathrm{Cov}(\theta_1, ..., \theta_k)$.
- One may compute the covariance by the whole chain $(\theta_1, ..., \theta_k)$ or by an increasing part of it, for instance $(\theta_{k/2}, ..., \theta_k)$.

MCMC in practice: the `mcmcrun` tool

- In this course we use a MATLAB code package that makes it easy to run MCMC analyses.
- The code is written by Marko Laine, and it can be downloaded from `http://helios.fmi.fi/ lainema/mcmc/`.
- The toolbox provides a unified interface for specifying models, and implements the AM and DRAM methods.
- Let us demonstrate the toolbox by considering again the simple BOD model $\mathbf{y} = \theta_1(1 - \exp(-\theta_2 \mathbf{x}))$ with data $\mathbf{x} = (1, 3, 5, 7, 9)$, $\mathbf{y} = (0.076, 0.258, 0.369, 0.492, 0.559)$.
- The main program for running the MCMC analysis is `bod_mcmcrun.m`. The code uses the same sum of squares function `bod_ss.m` as before.
- The MCMC is run with the `mcmcrun` function, and the results are visualized with `mcmcplot`, see lecture notes for details.

Visualizing MCMC Output

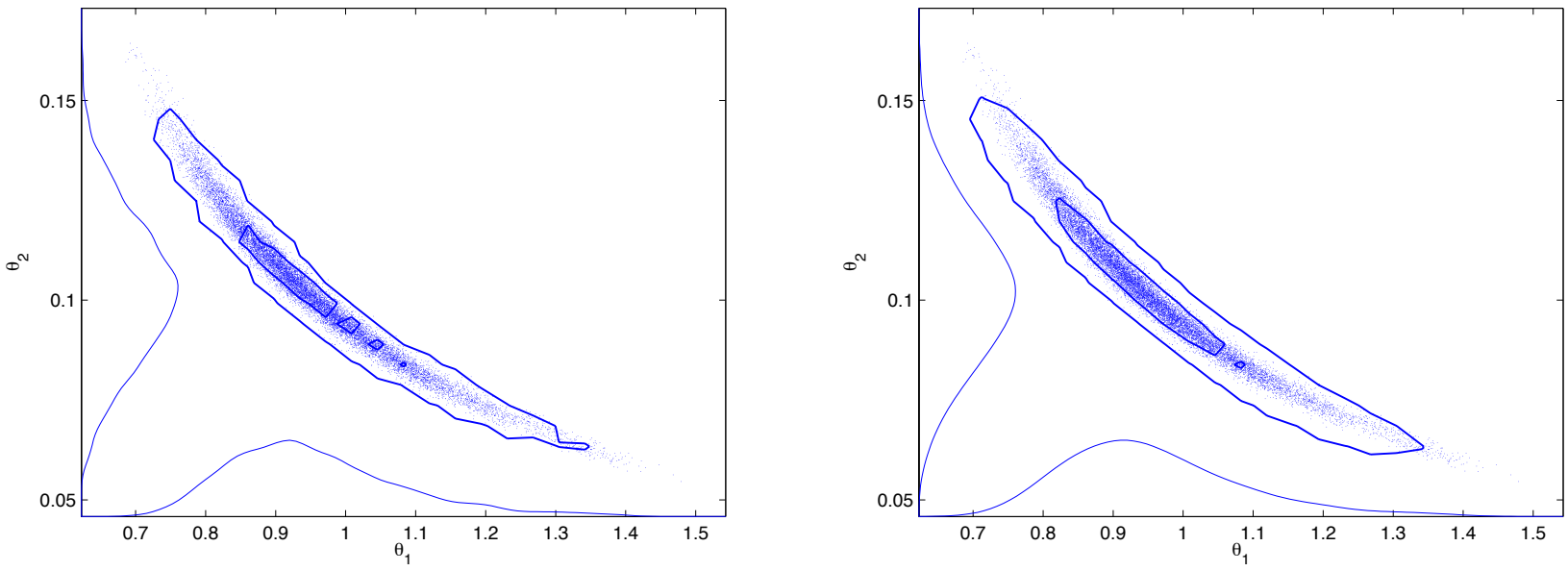


Figure: Pairs plots with two different width parameters for kernel density estimation.

Visualizing MCMC Output

- In addition to visualizing the parameter posterior, we are often interested in the predictive distribution.
- Predictive distributions can be visualized simply by simulating the prediction model with different parameter values and drawing the prediction curves.
- Another way is to compute, for instance, 50% and 95% envelopes for the predictions.
- The MCMC package contains functions for visualizing predictive distributions:
  - `mcmcpred` simulates the model responses and computes different confidence envelopes.
  - `mcmcpredplot` draws the envelopes
- Check the demo program `bod_mcmcrun_pred.m`.

Visualizing MCMC Output

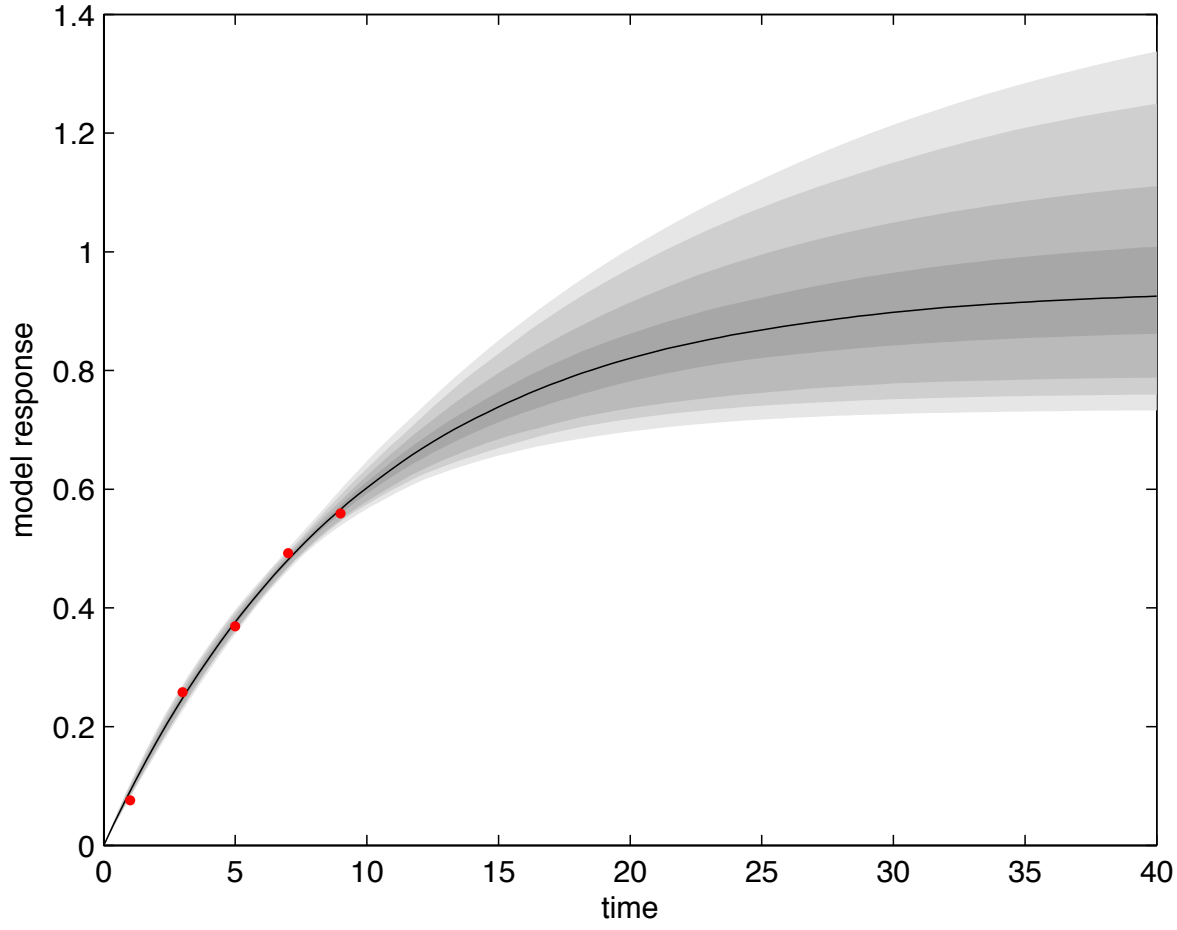


Figure: The data (red dots), the median of the predictions (black line) and the 50%, 95% and 99% confidence envelopes for the predictions (grey areas).

Dynamical State Estimation

- Besides estimating the static parameters θ, one is often interested in estimating the dynamically changing *state* of the system.
- As time proceeds, new measurements become available, that can be used to update the state estimates.
- The state estimation problem: at time k, estimate the system state $\mathbf{x}_k$ using previous observations $\mathbf{y}_{1:k} = (\mathbf{y}_1, ..., \mathbf{y}_k)$, when the model and observation equations are given as

$$\begin{aligned} \mathbf{x}_k &= \mathcal{M}(\mathbf{x}_{k-1}) + \varepsilon_k^p \\ \mathbf{y}_k &= \mathcal{K}(\mathbf{x}_k) + \varepsilon_k^o, \end{aligned}$$

where $\mathcal{M}$ is the *evolution model* and $\mathcal{K}$ is the *observation model*. The terms ε_k^p and ε_k^o represent the model error and the observation error.

Dynamical State Estimation

- In dynamical state estimation problems, measurements are obtained in real-time and the state estimate needs to be updated after the measurements are obtained.
- This can be achieved by applying the Bayes' formula sequentially:
  1. The prior is given by evolving the state from the previous time step using the model $\mathcal{M}$ (prediction step)...
  2. ... the prior is updated with the likelihood of the measurement (update step) to get the posterior ...
  3. ... which is evolved with the model and used as the prior in the next time step.
- Repeating this procedure allows 'on-line' estimation of model states.

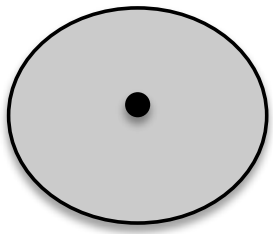
Dynamical State Estimation

Dynamical state estimation techniques are needed in many applications:

- Target tracking: estimate the position and velocity of an object. For instance, the Global Positioning System (GPS) uses state estimation techniques (extended Kalman filtering).
- Combining accelerometer and gyroscope data to compute the orientation, position and velocity of a gaming device.
- Process tomography: estimate the dynamically changing concentrations of different compounds in a pipe by combining fluid dynamics and chemistry models with tomographic measurements.
- Numerical Weather Prediction (NWP): estimate the state of the weather by correcting the previous prediction with different kinds of observations (ground based, satellite etc.) to allow real-time weather predictions.
- ...

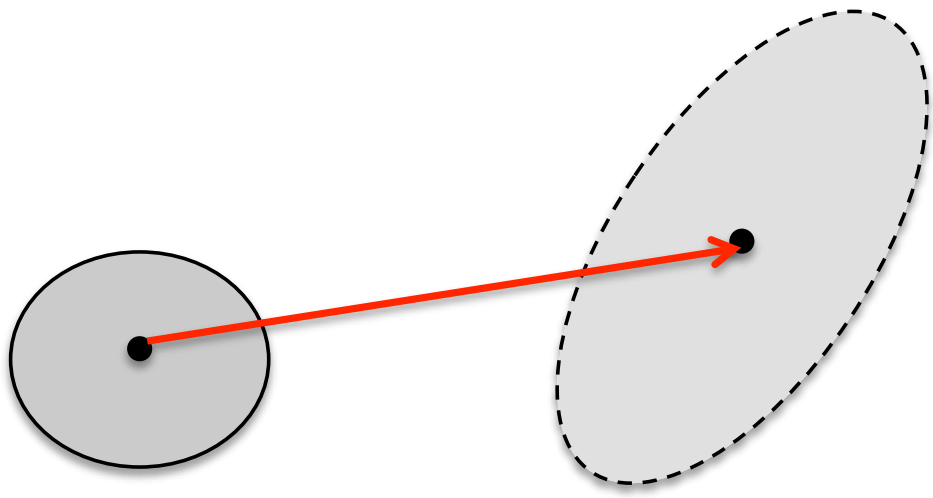
Dynamical State Estimation: Idea of Kalman Filter

1) Current state distribution



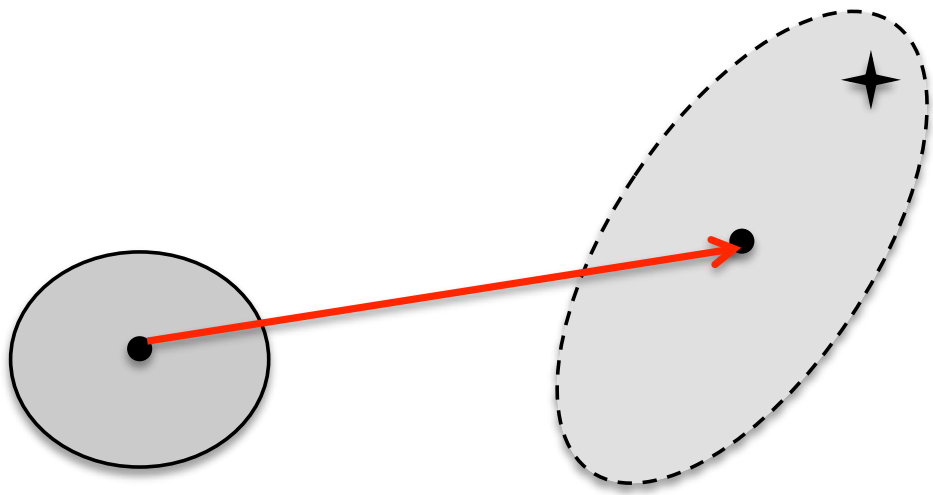
Dynamical State Estimation: Idea of Kalman Filter

2) Predict: move the state distribution with the model



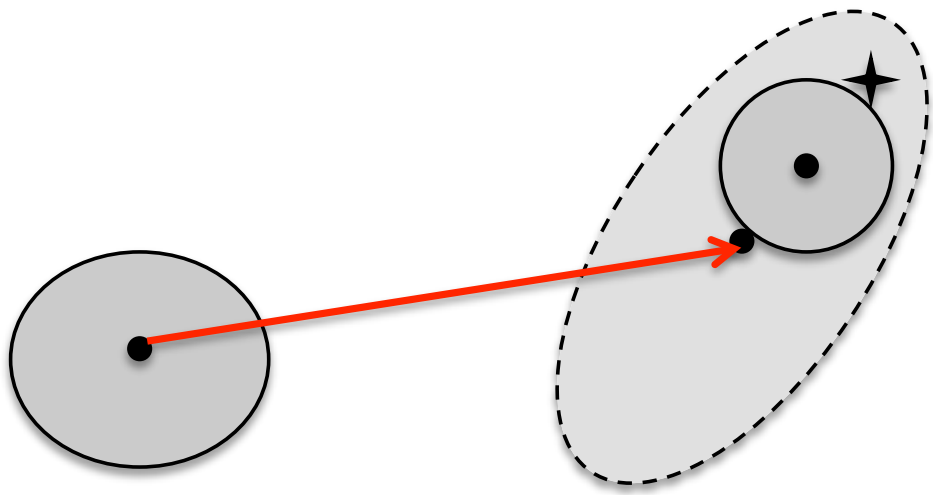
Dynamical State Estimation: Idea of Kalman Filter

3) New observations become available



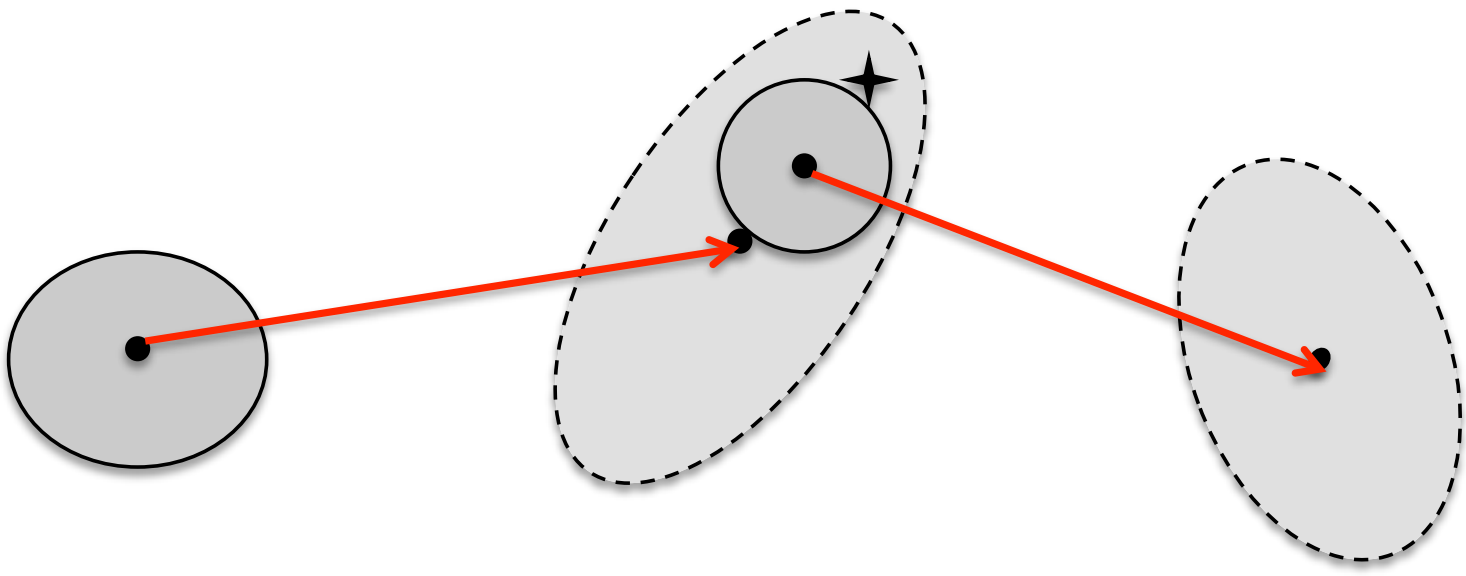
Dynamical State Estimation: Idea of Kalman Filter

4) Update the state estimate with the new observations



Dynamical State Estimation: Idea of Kalman Filter

5) Predict forward



Kalman Filter and Extended Kalman Filter

- The Kalman filter (KF) is meant for situations, where the model is linear and the model and observation errors are assumed to be zero mean Gaussians.

- The extended Kalman filter (EKF) is its extension to nonlinear situations, where the model is linearized and the KF formulas are applied.