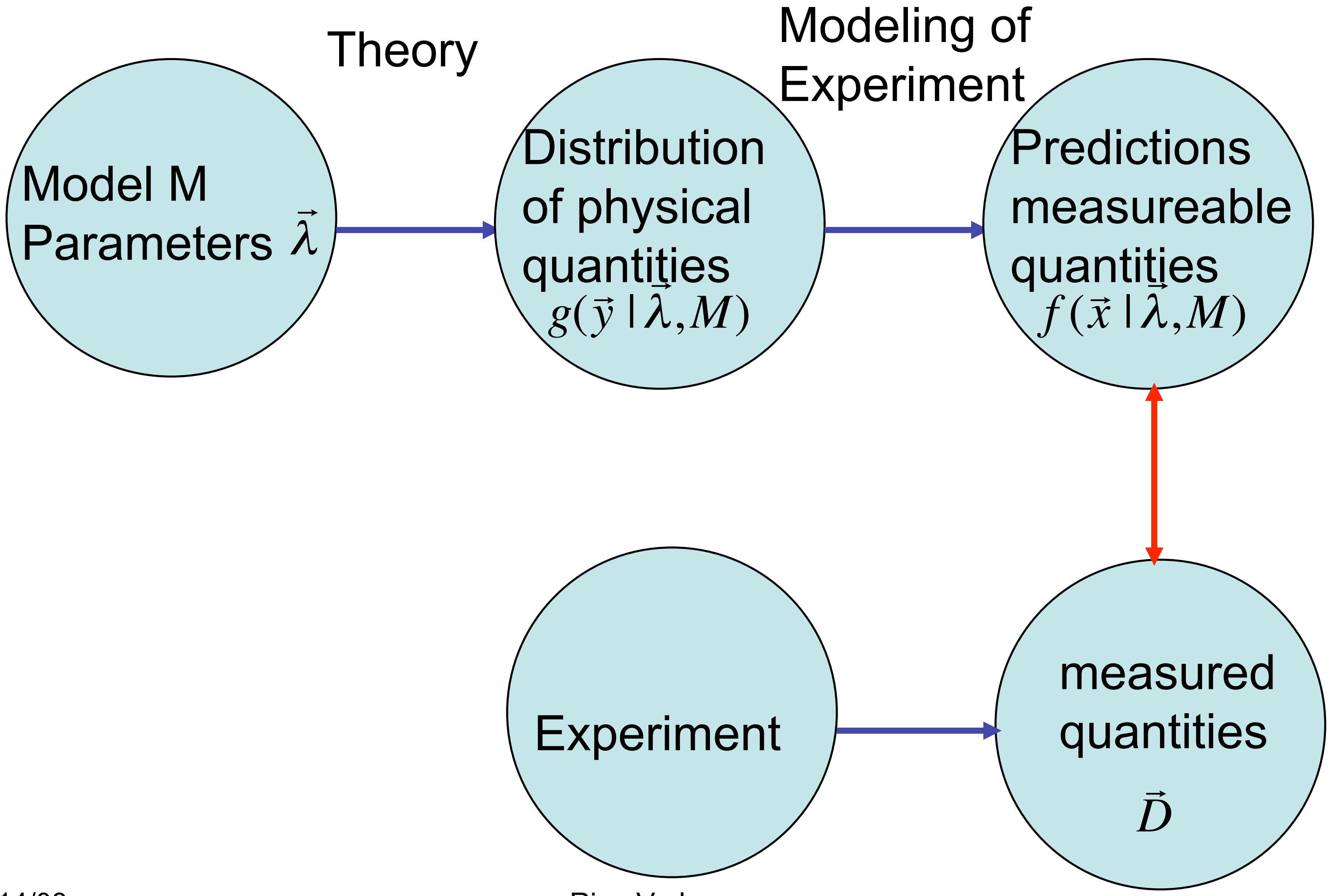
# Introduction to Bayesian Learning & Markov Chains

- Learning process
- Mathematical Formulation
  - Bayesian Learning rules
  - Discovery or not ?
  - Probability intervals & bounds
- Realization via Markov Chains

# How we learn



Theory

Modeling of Experiment

Model M Parameters $\vec{\lambda}$

Distribution of physical quantities $g(\vec{y}\,|\,\vec{\lambda},M)$

Predictions measureable quantities $f(\vec{x}\,|\,\vec{\lambda},M)$

Experiment

measured quantities $\vec{D}$

# How we Learn

We learn by comparing real data with probability distributions for predicted results assuming a theory, parameters, and a modeling of the experimental process. In this case, the probabilities can be frequencies, since we are talking about the output of a model.

What we typically want to know:

- Is the theory reasonable ? I.e., is the observed data a reasonably likely result from this theory (+ experiment)

- If we have more than one potential explanation, then we want to be able to quantify which theory is more likely to be correct given the observations

- Assuming we have a reasonable theory, we want to estimate the most probable values of the parameters, and their uncertainties.

# Formulation

Model provides ‘Direct Probabilities’; i.e., relative frequencies of possible outcomes if perform the experiment many times. Possible because the model is a mathematical construction. The function $f(\vec{x}|\vec{\lambda},M)$ with

$$f(\vec{x}|\vec{\lambda},M) \geq 0 \qquad \int f(\vec{x}|\vec{\lambda},M)d\vec{x} = 1$$

is the prediction from the model for the probability (density) for the result, with

$\vec{x}$ a possible realization of the data

$\vec{\lambda}$ the model parameters

M model including assumptions

# Formulation

The modeling of the experiment will typically add other parameters (e.g., factor representing how much energy measurement can be varied).

There could be additional information which is not built into the model, but which could limit the values of the parameters. E.g., one parameter could be the mass of a new particle. We then have m≥0.

The normalization of $f(\vec{x}|\vec{\lambda}, M)$ is usually not needed.

# Formulation

What we want to know is the probability of our model or particular set of parameters. For the model, we have

$$0 \leq P(M) \leq 1$$

For the parameters:

$$P(\vec{\lambda}|M) \geq 0$$

$$\int P(\vec{\lambda}|M) d\vec{\lambda} = 1$$

In the Bayesian approach, these quantities are treated in the same way as the frequency distributions from the model, but they are more accurately described as ‘Degrees-of-Belief’

# Formulation

There is no way to talk about the probability of a model being right as a frequency – there is only a 'Degree-of-Belief'. The role of experimental science is to modify our beliefs. If we have complete faith in model M being correct, then

$$P(M) = 1$$

The process of learning from experiment is:

$$P_{i+1}(\vec{\lambda}, M|\vec{D}) \propto f(\vec{x} = \vec{D}|\vec{\lambda}, M)P_i(\vec{\lambda}, M)$$

where the index represents a 'state-of-knowledge'

# Formulation

We have to satisfy our normalization condition, so

$$P_{i+1}(\vec{\lambda}, M|\vec{D}) = \frac{P(\vec{x} = \vec{D}|\vec{\lambda}, M)P_i(\vec{\lambda}, M)}{\sum_M \int P(\vec{x} = \vec{D}|\vec{\lambda}, M)P_i(\vec{\lambda}, M)d\vec{\lambda}}$$

We usually write $P_i = P_0$

and call $P_0$ the ‘prior’. It contains all information on the model and parameter values which we want to use before adding the new data. Setting the denominator to $P(\vec{D})$

$$P(\vec{\lambda}, M|\vec{D}) = \frac{P(\vec{D}|\vec{\lambda}, M)P(\vec{\lambda}, M)}{P(\vec{D})}$$ Bayes Equation

# Parameter Estimation

Keep the model fixed $P(\vec{\lambda}, M) = P(\vec{\lambda}|M)P(M)$

$$P(\vec{\lambda}|\vec{D}, M) = \frac{P(\vec{D}|\vec{\lambda}, M)P_0(\vec{\lambda}|M)}{\int P(\vec{D}|\vec{\lambda}, M)P_0(\vec{\lambda}|M)d\vec{\lambda}}$$

$P_0(\vec{\lambda}|M)$ Prior. If constant, max likelihood fit

$P(\vec{D}|\vec{\lambda}, M)$ The likelihood, if Gaussians, then $\chi^2$fit

Formulation includes max likelihood and chi-squared as approximations

# Parameter Estimation

The posterior pdf gives the full probability distribution for all parameters, including all correlations – no approximations. If interested in subset of parameters, then marginalize. E.g., for one parameter:

$$P(\lambda_i|\vec{D},M) = \int P(\vec{\lambda}|\vec{D},M)d\vec{\lambda}_{\mathrm{J}\neq i}$$

Quantities of interest which can be determined:

$$\text{Mode} \quad \overset{\lambda_i}{\max}\{P(\lambda_i|D,M)\}$$

$$\text{Mean of } \lambda_i \quad <\lambda_i> = \int P(\lambda_i|\vec{D},M)\lambda_i d\lambda_i$$

$$\text{Median} \quad \int_{\lambda_{min}}^{\lambda_{med}} P(\lambda_i|\vec{D},M)d\lambda_i = 0.5$$

$$\text{Central Interval} \quad \alpha = \int_{\lambda_{min}}^{\lambda_{lower}} P(\lambda_i|\vec{D},M)d\lambda_i = \int_{\lambda_{upper}}^{\lambda_{max}} P(\lambda_i|\vec{D},M)d\lambda_i$$

$$\text{rms} \quad rms_i = \sqrt{\left[\int P(\lambda_i|\vec{D},M)\lambda_i^2 d\lambda_i - \left(\int P(\lambda_i|\vec{D},M)\lambda_i d\lambda_i\right)^2\right]}$$

# Model Testing

We can ask – how likely are the results assuming the model.

First, a couple definitions:

$$\begin{aligned} f^*(\vec{x}) &= P(\vec{x}|\vec{\lambda}^*, M) \\ f^D &= P(\vec{D}|\vec{\lambda}^*, M) \end{aligned}$$

where $\vec{\lambda}^*$ is the set of parameters values for the mode of the full posterior pdf. The quantity we propose for model testing is:

$$p = \frac{\int_{f^*(\vec{x})<f^D} f^*(\vec{x})d\vec{x}}{\int f^*(\vec{x})d\vec{x}}$$

In words: ‘tail-area’ probability to have found a result with smaller probability than that observed (as with chi-squared prob test). If model is correct, then p has flat distribution between 0,1

# Model testing

A model which represents the data should have a p-value not too small.

When comparing models, just compare p-values.  The bigger the better.

If several models have similar p-values, choose the simplest model (Occam’s razor).

Discovery can be defined if background (standard physics) has small p-value, whereas new physics gives good p-value.

# Setting Limits

Setting limits is conceptually easy – just integrate the posterior pdf. E.g.,

$$0.9 = \int_{\lambda_{min}}^{\lambda_{upper}} P(\lambda_i|\vec{D}, M)d\lambda_i$$

# Markov Chains

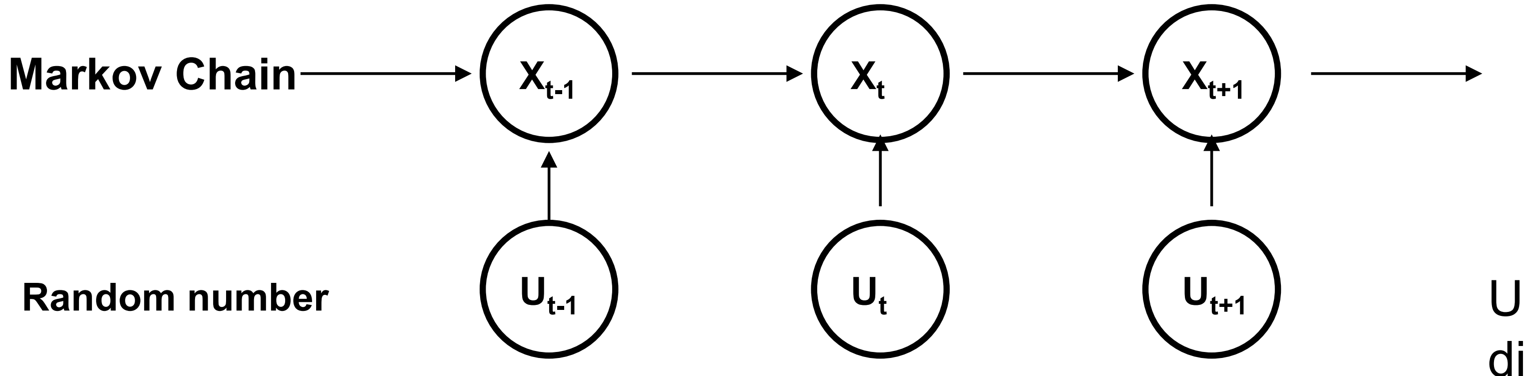
Basic Property of a Markov Process:

$$\Pr\{a < X_t \le b \mid X_{t_1} = x_1, \cdots, X_{t_n} = x_n\} = \Pr\{a < X_t \le b \mid X_{t_n} = x_n\}$$

$$t_1 < t_2 < \cdots < t_n < t$$

I.e., the probability distribution for the variable *X* depends only the current state, not on any previous behavior. For a finite or denumerable state space (which is always the case on a computer), have a Markov Chain. E.g., Poisson process is a continuous time Markov Chain.

# Markov Chain Monte Carlo



$X_t \sim \pi(x)$

$U_t$ iid from uniform dist between (0,1)

Goal of MCMC is to find a chain with $(\pi_i)_{i=0}^{\infty}$=pdf of interest. Sampling according to the Markov Chain will then correspond to sampling from the desired pdf.

Define Markov Chain Monte Carlo as any method producing an ergodic Markov chain $X_t$ whose stationary distribution in the distribution of interest.

The original algorithm is due to Metropolis. Later generalized by Hastings.

# Markov Chain Monte Carlo

Basic Limit Theorem (for aperiodic, irreducible and recurrent Markov Chains)

$$\boxed{\lim_{n\to\infty} P_{ii}^{n} = \frac{1}{\sum_{n=0}^{\infty} n f_{ii}^{n}} = \pi_i \qquad \lim_{n\to\infty} P_{ji}^{n} = P_{ii}^{n} = \pi_i}$$

$\pi$ is the stationary distribution. Ergodic - does not depend on the starting point. Strongly ergodic class, all $\pi_i>0$.

Note that: $\lim_{n\to\infty} P_{jj}^{n} = \boxed{\pi_j = \sum_{i=0}^{\infty} \pi_i P_{ij}}$ $\sum_{i=0}^{\infty} \pi_i = 1$ Eigenvalue equation

Detailed balance: $\pi_i P_{ij} = \pi_j P_{ji}$ Sufficient condition for $\pi_i$ to be stationary distribution of $P_{ij}$

# Markov Chain Monte Carlo

Uses:

1. Simulation of physical system which follows a known probability rule

$$x \sim \pi(x) \quad \text{where } x \text{ is a configuration}$$

2. Calculation of expectation values in a large number of dimensions

$$E[g(x)] = \int g(x)\pi(x)dx$$

3. Optimization with an annealing scheme

$$x^{*} = \arg\max \pi(x)$$

4. Learning (probability calculations)

# Metropolis Algorithm

1. Suppose we have $X_t=x$. Generate a proposed new value, Y, according to a symmetric function $g(y,x)$. Symmetric means $g(y,x)=g(x,y)$.
2. Calculate $r=f(y)/f(x)$, where $f(x)$ is the desired density distribution. Generate a random number $U$ from a uniform distribution between 0,1. Then,

    set $X_{t+1}=y$ if $U<r$;

    else, $X_{t+1}=x$

Note that all steps with $f(y)>f(x)$ are accepted. If $f(y)<f(x)$, take new position with probability r, else stay in current state.

original Metropolis et al. paper:

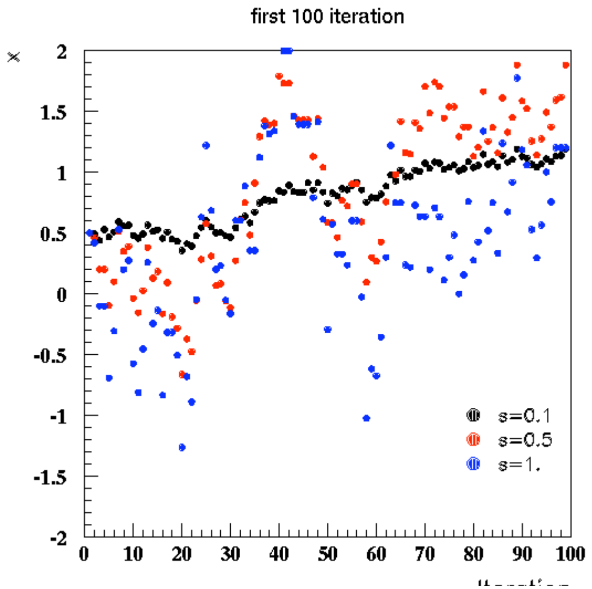
N. Metropolis et al., J. Chem. Phys. 21 (1953) 1087.

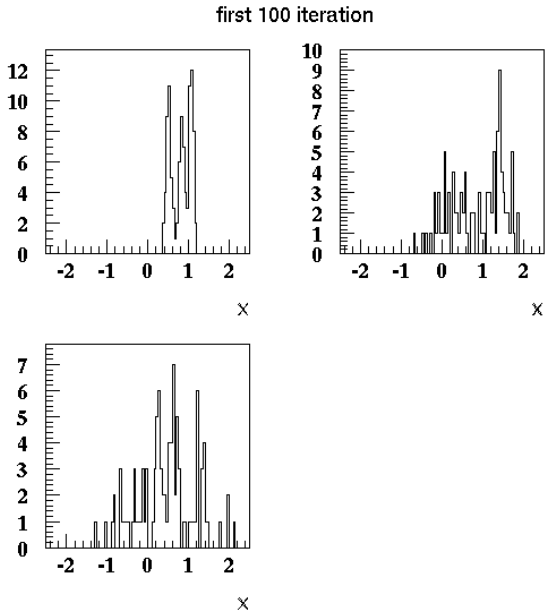
# Markov Chain Monte Carlo

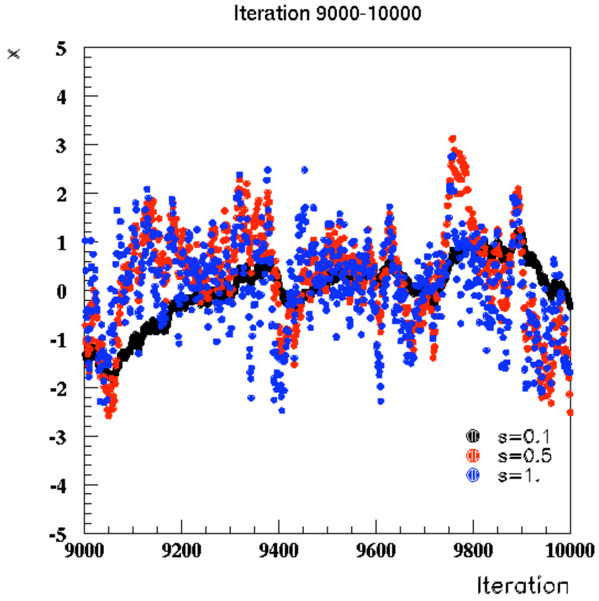
For example, generate a Gaussian distribution with zero mean and $\sigma$=1 from a random walk Markov Chain with a step derived from a flat distribution as follows:

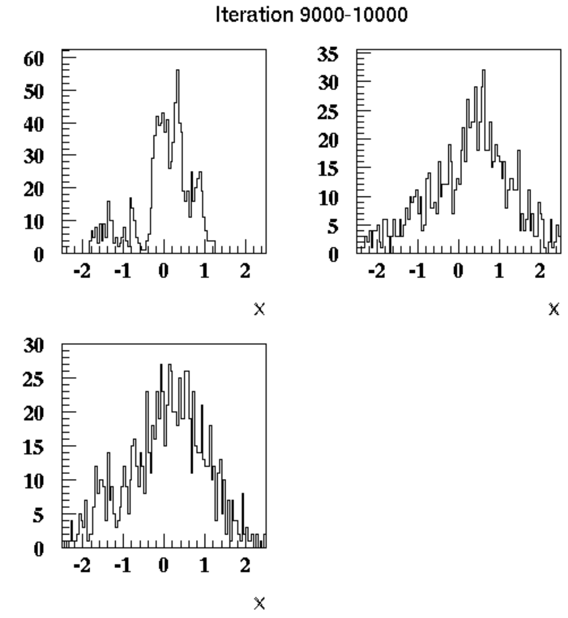
1. Generate a number from a flat distribution between *[-s,s]*; call it $\varepsilon$. Now set $y = x_t + \varepsilon$
2. Calculate $\rho = \min\left\{ \frac{e^{-y^2/2}}{e^{-x^2/2}}, 1 \right\}$ (note that *q(y|x)=q(x|y)*)
3. Set $x_{t+1}$=y if $U<\rho$, where *U* is a r.v. from a uniform distribution between (0,1)

# Example


first 100 iteration
x
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
10
20
30
40
50
60
70
80
90
100
Iteration
● s=0.1
● s=0.5
● s=1.


first 100 iteration
12
10
8
6
4
2
0
-2
-1
0
1
2
x
10
9
8
7
6
5
4
3
2
1
0
-2
-1
0
1
2
x
7
6
5
4
3
2
1
0
-2
-1
0
1
2
x


Iteration 9000-10000
x
5
4
3
2
1
0
-1
-2
-3
-4
-5
9000
9200
9400
9600
9800
10000
Iteration
● s=0.1
● s=0.5
● s=1.


Iteration 9000-10000
60
50
40
30
20
10
0
-2
-1
0
1
2
x
35
30
25
20
15
10
5
0
-2
-1
0
1
2
x
30
25
20
15
10
5
0
-2
-1
0
1
2
x

# Markov Chain Monte Carlo

If you define your likelihood function and priors, then you have your target distribution, because

$$P(\vec{\lambda}|\vec{D}, M) \propto P(\vec{D}|\vec{\lambda}, M) P_0(\vec{\lambda}|M)$$

The MCMC, once it has converged, will output sets of parameter values  which are distributed according to the posterior pdf.  You can then use this, e.g., in your root program, to calculate anything you want.

Technical realization – BAT program