

# Computer aspects of statistical simulation

Peter Sestoft (sestoft@dina.kvl.dk)  
 Department of Mathematics and Physics  
 Royal Veterinary and Agricultural University  
 Frederiksberg, Denmark

## Simulation of homogenous Markov chains $(X_n) = (X_0, X_1, X_2, \dots)$

- The index set  $T$  is discrete.
- The transition mechanism is the same from  $X_n = x$  to  $X_{n+1}$  as from  $X_0 = x$  to  $X_1$ .

This permits a simulation algorithm of the form:

```
x[0] = x0; // initial state
for (int n=0; n<rmax; n+=1)
    x[n+1] = q(x[n]);
```

where  $q$  is the transition mechanism: a (non-deterministic) 'function' of the previous state  $x[n]$  only.

This simulates a finite prefix  $(X_0, X_1, \dots, X_{\text{rmax}})$  of the chain.

Because of the Markov property, we need not store all elements of the chain:

```
x = x0; // initial state
for (int n=0; n<rmax; n+=1)
    x = q(x);
```

**Main question:** How to program the transition 'function'  $q$  ?

## Case A1: A finite state space $E = \{1, \dots, K\}$

Assume we have the transition matrix  $P = p_{ij}$ :

$$P = \begin{pmatrix} p_{11} & p_{12} & p_{13} & \dots & p_{1K} \\ p_{21} & p_{22} & p_{23} & \dots & p_{2K} \\ p_{31} & p_{32} & p_{33} & \dots & p_{3K} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{K1} & p_{K2} & p_{K3} & \dots & p_{KK} \end{pmatrix}$$

Given  $X_n = i$ , how choose  $X_{n+1} = q(i)$ ?

Example:  $p_i = (0, 0.05, 0, 20, 0, 0.0, 0, 55, 0, 20)$ .

Make the size of a state proportional to its probability, pick a point at random and go to the indicated state:

1	2	4	5
---	---	---	---

In general: Draw a random  $u$  from  $U[0, 1]$ , the uniform distribution on  $[0, 1]$ , then take  $q(i)$  to be least  $j$  such that

$$p_{i1} + \dots + p_{ij} \geq u$$

## Straightforward implementation: linear search

```
double u = Math.random();
int j = 0;
double psum = 0;
while (psum < u) {
    j += 1;
    psum += P(i, j);
}
```

When the loop terminates,  $psum = p_{i1} + \dots + p_{ij} \geq u$ , and  $j$  is the least  $j$  for which this holds.

The Java 'function' `Math.random()` draws a random  $u$  from  $U[0, 1]$ .

On average, the loop requires  $K/2$  iterations to determine  $j$ .

### Doing it efficiently: binary search

Precompute a table of the cumulative transition probabilities

$$p_{c_j} = p_{i1} + \dots + p_{ij}$$

then use binary search in the table  $p_{c_j}$  to find the least  $j$  for which  $p_{c_j} \geq u$ :

```
double u = Math.random();
int j, left = 1, right = K;
while (left <= right) {
    j = (left+right) / 2;
    if (u < p_c[j]) right = j-1;
    else if (p_c[j] < u) left = j+1;
    else if (left == j) right = left - 1;
    else
        right = j;
}
j = left;
```

Halves the distance from left to right in each iteration, hence requires at most  $\log_2 K$  iterations to find  $j$ .

When  $K = 1000$  this means 10 iterations (instead of 500).

Monday 16 August 1999

Dina PhD Summerschool

Page 5

### Doing it efficiently: binary search

Precompute a table of the cumulative transition probabilities

$$p_{c_j} = p_{i1} + \dots + p_{ij}$$

then use binary search in the table  $p_{c_j}$  to find the least  $j$  for which  $p_{c_j} \geq u$ :

```
double u = Math.random();
int j, left = 1, right = K;
while (left <= right) {
    j = (left+right) / 2;
    if (u < p_c[j]) right = j-1;
    else if (p_c[j] < u) left = j+1;
    else if (left == j) right = left - 1;
    else
        right = j;
}
j = left;
```

Halves the distance from left to right in each iteration, hence requires at most  $\log_2 K$  iterations to find  $j$ .

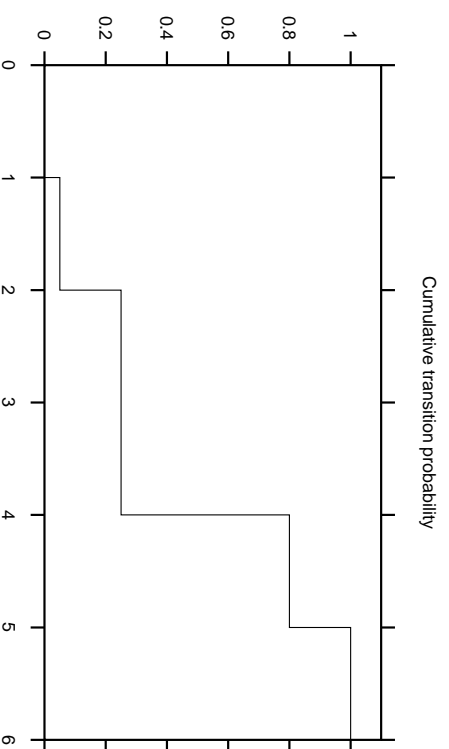
When  $K = 1000$  this means 10 iterations (instead of 500).

Monday 16 August 1999

Dina PhD Summerschool

Page 5

Example: if  $p_x = (0.05, 0.20, 0.00, 0.55, 0.20)$  then  $p_{c_x} = (0.05, 0.25, 0.25, 0.80, 1.00)$  then



Note that the transition  $i \rightarrow 3$  will never be taken.

Monday 16 August 1999

Dina PhD Summerschool

Page 6

### Case A2: A discrete (countable) state space $E = \{0, 1, 2, \dots\}$

Given  $X_n = i$ , how choose  $X_{n+1}$ ?

Assume we have a probability density function  $f_i$  on  $E$  given  $X_n = i$ .

We draw a random  $u$  from  $U[0, 1]$ , then take  $q(i)$  to be least  $j$  such that

$$f_i(1) + \dots + f_i(j) \geq u$$

In general we cannot precompute and tabulate the cumulative transition probabilities.

Monday 16 August 1999

Dina PhD Summerschool

Page 7

### Case B: Continuous state space $E = [0, +\infty[$

Given  $X_n = x$ , how choose  $X_{n+1} = q(x)$ ?

Assume we have the cumulative distribution function  $F_x : E \rightarrow [0, 1]$  on  $E$  given  $X_n = x$ .

We draw a random  $u$  from  $U[0, 1]$ , then take  $q(x)$  to be the least  $y \in E$  for which  $F_x(y) \geq u$ .

This is just what we did in the discrete cases A1 and A2.

Since  $F_x$  is monotone we can always define its 'lower inverse'  $F_x^{-1}$  and take  $q(x) = F_x^{-1}(u)$ .

Sometimes  $F_x^{-1}(u)$  can be computed efficiently.

Monday 16 August 1999

Dina PhD Summerschool

Page 8

### A generalization: Metropolis-Hastings chains

```
int x = x0;
for (int n=0; n<nmax; n+=1) {
    y = q(x);
    if (Math.random() < alpha(x, y)) // Proposal state y
        x = y;                       // Accept y as next state?
}
```

If  $p \in [0, 1]$  then `Math.random() < p` with probability  $p$ .

In the special case where  $\alpha(x, y) = 1$  we get a Markov chain.

The Metropolis-Hastings algorithm is used in Markov Chain Monte Carlo (MCMC) simulations (next week).

### Online Metropolis-Hastings tool

A collection of Java applets at <http://www.dina.kvl.dk/~sestoft/stat/mcmc.html>

When the state space is multi-dimensional (e.g.  $E = R^2$ ) then the state is a vector  $\vec{x} = (x_1, x_2)$ .

In *single-component Metropolis-Hastings* (SCMH) the state vector is updated one component at a time.

The simulation setup is the same regardless of state space  $E$  and its dimension.

In general, we only need to specify the initial state  $\vec{x}_0$  and the  $q$  and  $\alpha$  functions.

### Case A1: Finite state space Markov chain (class Transition)

Must specify the cumulative transition probabilities  $p_{ij} = \sum_{k=1}^j p_{ik}$ :

```
double[][] pc = new double[K][K]; // K-by-K matrix
pc = ...
new SCMH(new Transition(pc), x0);
```

When `alpha` is not specified, it defaults to 1 (thus giving a Markov chain).

### Case A2: Discrete (countable) state space Markov chain

Must specify  $\pi(i, j) = f_i(j)$ , the transition probability from state  $i$  to state  $j$ :

Example:

```
class Poisson extends MHParam {
    double pi(int i, int j)
    { ... }
}
...
new SCMH(new Poisson(), x0);
```

Above `pi(i, j)` computes  $\pi(i, j)$ .

### Case B: Continuous state space

Example:  $E = R^2$

Must specify  $q$  and  $\alpha$ :

```
class Twon extends SCMHParam {
    double qobs(int i, double[] xt)
    { ... }

    double alpha(double[] xt, int i, double yi)
    { ... }
}
...
new SCMH(new Twon(), x0);
```

Above, `qobs(i, x)` proposes a new value  $y_i$  for the  $i$ 'th component of state  $x$ .

and `alpha(x, i, y)` is the probability with which  $y_i$  is accepted, given previous state  $x$ .

### Computing derived quantities from Markov chains

#### The average proportions of 0's and 1's

```
int zs = 0, os = 0;
int x = x0;
for (int n=0; n<nmax; n+=1) {
    x = q(x);
    if (x == 0) zs += 1;
    if (x == 1) os += 1;
}
```

Now the proportion of 0's is  $zs/nmax$ , and that of 1's is  $os/nmax$ .

More concise and efficient:

```
int[] freq = new int[K]; // State frequencies
int x = x0;
for (int n=0; n<nmax; n+=1) {
    x = q(x);
    freq[x] += 1;
}
```

Now the proportion of  $i$ 's is  $freq[i]/nmax$ .

Monday 16 August 1999

Dina PhD Summerschool

Page 13

### Waiting time till absorption

A state  $a$  is absorptive if  $p_{aa} = 1$ .

The set of  $A$  absorptive states is  $A = \{a \in E \mid p_{aa} = 1\}$

This program fragment may be used to estimate the waiting time till absorption:

```
int wsum = 0;
for (int s=1; s<=smax; s+=1) {
    int n = 0;
    int x = x0;
    while (x not in A) {
        x = q(x);
        n += 1;
    }
    wsum += n;
}
```

Note: when the expected waiting time is infinite, this program may loop forever.

The conditions for this can be detected by analysing the transition matrix (which is used when defining  $q$ ).

Monday 16 August 1999

Dina PhD Summerschool

Page 15

### Distribution of first time in state 1

We must simulate many chains:

```
int wsum = 0;
for (int s=1; s<=smax; s+=1) { // Perform smax simulations
    int n = 0;
    int x = x0;
    while (x != 1) {
        x = q(x);
        n += 1;
    }
    wsum += n;
}
```

The computed average waiting time is  $wsum/smax$ .

Monday 16 August 1999

Dina PhD Summerschool

Page 14