STK3405 – Lecture 12
Part 1

A. B. Huseby

Department of Mathematics
University of Oslo, Norway



Chapter 6

Computing reliability using conditional Monte Carlo methods



Section 6.1

Monte Carlo simulation and conditioning



Monte Carlo simulation and conditioning

Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be a random vector with a known distribution, and let $\phi = \phi(\boldsymbol{X})$. The distribution of ϕ cannot be derived analytically in polynomial time with respect to n. We want to estimate:

$$h = \mathrm{E}[\phi(\boldsymbol{X})]$$

We run a Monte Carlo simulation generating N independent vectors $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$, all having the same distribution as $\boldsymbol{X}$, and estimate h by the simple *Monte Carlo* estimate:

$$\hat{h}_{MC} = \frac{1}{N} \sum_{r=1}^{N} \phi(\boldsymbol{X}_r).$$

Then $\mathrm{E}[\hat{h}_{MC}] = h$, and we have:

$$\mathrm{Var}\left(\hat{h}_{MC}\right) = \frac{1}{N^2} \sum_{r=1}^{N} \mathrm{Var}(\phi) = \frac{1}{N} \mathrm{Var}(\phi)$$



Monte Carlo simulation and conditioning (cont.)

Let $S = S(\boldsymbol{X})$ be a *discrete* variable with values in the set $\{s_1, \ldots, s_k\}$, and with a distribution that can be calculated analytically in polynomial time with respect to n.

We also introduce:

$$\theta_j = \mathrm{E}[\phi \mid S = s_j], \quad j = 1, \ldots, k.$$

We then have:

$$h = \mathrm{E}[\phi] = \sum_{j=1}^{k} \mathrm{E}[\phi \mid S = s_j] P(S = s_j) = \sum_{j=1}^{k} \theta_j P(S = s_j)$$



Monte Carlo simulation and conditioning (cont.)

Instead of generating N samples from the distribution of $\boldsymbol{X}$, we divide the sample set into k groups, one for each possible value of S.

The j-th group has size N_j, $j = 1, \ldots, k$, and $N_1 + \cdots + N_k = N$.

The samples in the j-th group, $\boldsymbol{X}_{1,j}, \ldots, \boldsymbol{X}_{N_j,j}$, are sampled from the conditional distribution of $\boldsymbol{X}$ given that $S = s_j$, $j = 1, \ldots, k$.

Then θ_j is estimated by:

$$\hat{\theta}_j = \frac{1}{N_j} \sum_{r=1}^{N_j} \phi(\boldsymbol{X}_{r,j}), \quad j = 1, \ldots, k.$$



Monte Carlo simulation and conditioning (cont.)

We then have:

$$\mathrm{E}[\hat{\theta}_j] = \frac{1}{N_j}\sum_{r=1}^{N_j}\mathrm{E}[\phi(\boldsymbol{X}_{r,j})] = \frac{1}{N_j}\sum_{r=1}^{N_j}\mathrm{E}[\phi|S=s_j]$$
$$= \frac{1}{N_j}\sum_{r=1}^{N_j}\theta_j = \theta_j, \qquad j=1,\ldots,k.$$

Moreover, the variances of these estimates are:

$$\mathrm{Var}(\hat{\theta}_j) = \frac{1}{N_j^2}\sum_{r=1}^{N_j}\mathrm{Var}(\phi(\boldsymbol{X}_{r,j})) = \frac{1}{N_j^2}\sum_{r=1}^{N_j}\mathrm{Var}(\phi|S=s_j)$$
$$= \frac{1}{N_j}\mathrm{Var}(\phi|S=s_j), \qquad j=1,\ldots,k.$$



Monte Carlo simulation and conditioning (cont.)

By combining $\hat{\theta}_1, \ldots, \hat{\theta}_k$, we get the *conditional Monte Carlo estimate*:

$$\hat{h}_{CMC} = \sum_{j=1}^{k} \hat{\theta}_j P(S = s_j).$$

Since $\hat{\theta}_1, \ldots, \hat{\theta}_k$ are independent, the variance of the conditional Monte Carlo estimate is:

$$\mathrm{Var}(\hat{h}_{CMC}) = \mathrm{Var}[\sum_{j=1}^{k} \hat{\theta}_j P(S = s_j)] = \sum_{j=1}^{k} \mathrm{Var}(\hat{\theta}_j)[P(S = s_j)]^2$$

$$= \sum_{j=1}^{k} \frac{1}{N_j} \mathrm{Var}(\phi | S = s_j)[P(S = s_j)]^2.$$



Monte Carlo simulation and conditioning (cont.)

NOTE: The variance of $\hat{h}_{CMC}$ depends on the choices of the N_j's.

In order to compare the result with the variance of $\hat{h}_{MC}$, we let:

$$N_j \approx N \cdot P(S = s_j), \quad j = 1, \ldots, k.$$

Hence:

$$\sum_{j=1}^{k} N_j \approx \sum_{j=1}^{k} N \cdot P(S = s_j) = N \cdot \sum_{j=1}^{k} P(S = s_j) = N.$$



Monte Carlo simulation and conditioning (cont.)

With this choice we get:

$$\begin{aligned}\mathrm{Var}(\hat{h}_{CMC}) &\approx \sum_{j=1}^{k} \frac{1}{N \cdot P(S = s_j)} \mathrm{Var}(\phi|S = s_j)[P(S = s_j)]^2 \\ &= \frac{1}{N} \sum_{j=1}^{k} \mathrm{Var}(\phi|S = s_j) P(S = s_j) = \frac{1}{N} \mathrm{E}[\mathrm{Var}(\phi|S)] \\ &= \frac{1}{N} \big(\mathrm{Var}(\phi) - \mathrm{Var}[\mathrm{E}(\phi|S)] \big) \leq \frac{1}{N} \mathrm{Var}(\phi) = \mathrm{Var}(\hat{h}_{MC}).\end{aligned}$$

Here we have used the well-known relation:

$$\mathrm{Var}(\phi) = \mathrm{Var}[\mathrm{E}(\phi|S)] + \mathrm{E}[\mathrm{Var}(\phi|S)]$$



Monte Carlo simulation and conditioning (cont.)

Since:

$$\begin{aligned}\mathrm{Var}(\hat{h}_{CMC}) &\approx \frac{1}{N}\big(\mathrm{Var}(\phi) - \mathrm{Var}[\mathrm{E}(\phi|S)]\big) \\ &\leq \frac{1}{N}\mathrm{Var}(\phi) = \mathrm{Var}(\hat{h}_{MC}),\end{aligned}$$

it follows that $\mathrm{Var}(\hat{h}_{CMC}) < \mathrm{Var}(\hat{h}_{MC})$ whenever $\mathrm{Var}[\mathrm{E}(\phi \mid S)] > 0$.

$\mathrm{Var}[\mathrm{E}(\phi \mid S)]$ can be interpreted as a measure of how much information S contains relative to ϕ.



Monte Carlo simulation and conditioning (cont.)

1. S and ϕ independent. Then $\mathrm{E}(\phi \mid S) = \mathrm{E}(\phi)$, and thus,

$$\mathrm{Var}[\mathrm{E}(\phi \mid S)] = 0,$$

   implying that $\mathrm{Var}(\hat{h}_{CMC}) = \mathrm{Var}(\hat{h}_{MC})$.

2. $S = \phi$. Then $\mathrm{E}(\phi \mid S) = \phi$, and thus, $\mathrm{Var}[\mathrm{E}(\phi \mid S)] = \mathrm{Var}(\phi)$, implying that $\mathrm{Var}(\hat{h}_{CMC}) = 0$.

3. $S = \boldsymbol{X}$. Then $\mathrm{E}(\phi \mid S) = \phi(\boldsymbol{X})$, and thus, $\mathrm{Var}[\mathrm{E}(\phi \mid S)] = \mathrm{Var}(\phi)$, implying that $\mathrm{Var}(\hat{h}_{CMC}) = 0$.



Monte Carlo simulation and conditioning (cont.)

In order to minimize the variance of the estimate, the variable S should contain as much information about ϕ as possible.

At the same time S must be chosen so that:

- The distribution of S can be derived analytically in polynomial time
- The number of possible values of S, i.e., k, must be bounded by a polynomial function of n
- It must be possible to sample efficiently from the distribution of $\boldsymbol{X}$ given S



Conditioning on the sum of the component state variables



Conditioning on the sum

Consider a binary monotone system (C, ϕ) where $C = \{1, \ldots, n\}$, and let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be the vector of the component state variables. We now let:

$$S = S(\boldsymbol{X}) = \sum_{i=1}^{n} X_i$$



Conditioning on the sum

Consider a binary monotone system (C, ϕ) where $C = \{1, \ldots, n\}$, and let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be the vector of the component state variables. We now let:

$$S = S(\boldsymbol{X}) = \sum_{i=1}^{n} X_i$$

Hence, the set of possible values for S is $\{0, 1, \ldots, n\}$, and we let:

$$\theta_s = E[\phi | S = s], \qquad s = 0, 1, \ldots, n.$$



Conditioning on the sum

Consider a binary monotone system (C, ϕ) where $C = \{1, \ldots, n\}$, and let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be the vector of the component state variables. We now let:

$$S = S(\boldsymbol{X}) = \sum_{i=1}^{n} X_i$$

Hence, the set of possible values for S is $\{0, 1, \ldots, n\}$, and we let:

$$\theta_s = E[\phi | S = s], \qquad s = 0, 1, \ldots, n.$$

The conditional Monte Carlo estimate can now be expressed as:

$$\hat{h}_{CMC} = \sum_{s=0}^{n} \hat{\theta}_s P(S = s)$$



Conditioning on the sum

Consider a binary monotone system (C, ϕ) where $C = \{1, \ldots, n\}$, and let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be the vector of the component state variables. We now let:

$$S = S(\boldsymbol{X}) = \sum_{i=1}^{n} X_i$$

Hence, the set of possible values for S is $\{0, 1, \ldots, n\}$, and we let:

$$\theta_s = E[\phi | S = s], \qquad s = 0, 1, \ldots, n.$$

The conditional Monte Carlo estimate can thus be expressed as:

$$\hat{h}_{CMC} = \sum_{s=0}^{n} \hat{\theta}_s P(S = s)$$

To proceed, we must find an efficient way of sampling from the conditional distribution of $\boldsymbol{X}$ given $S = s$, $s = 0, 1, \ldots, n$.



Conditioning on the sum (cont.)

This can be done as follows:

STEP 1. Sample X_1 from the conditional distribution of $X_1|S = s$, and let x_1 be the result.

STEP 2. Sample X_2 from the conditional distribution of $X_2|S = s, X_1 = x_1$, and let x_2 be the result.

...

STEP n. Sample X_n from the conditional distribution of $X_n|S = s, X_1 = x_1, \ldots X_{n-1} = x_{n-1}$, and let x_n be the result.



Conditioning on the sum (cont.)

This can be done as follows:

STEP 1. Sample X_1 from the conditional distribution of $X_1|S = s$, and let x_1 be the result.

STEP 2. Sample X_2 from the conditional distribution of $X_2|S = s, X_1 = x_1$, and let x_2 be the result.

...

STEP n. Sample X_n from the conditional distribution of $X_n|S = s, X_1 = x_1, \ldots X_{n-1} = x_{n-1}$, and let x_n be the result.



Conditioning on the sum (cont.)

This can be done as follows:

STEP 1. Sample X_1 from the conditional distribution of $X_1|S = s$, and let x_1 be the result.

STEP 2. Sample X_2 from the conditional distribution of $X_2|S = s, X_1 = x_1$, and let x_2 be the result.

...

STEP n. Sample X_n from the conditional distribution of $X_n|S = s, X_1 = x_1, \ldots X_{n-1} = x_{n-1}$, and let x_n be the result.



Conditioning on the sum (cont.)

This can be done as follows:

STEP 1. Sample X_1 from the conditional distribution of $X_1|S = s$, and let x_1 be the result.

STEP 2. Sample X_2 from the conditional distribution of $X_2|S = s, X_1 = x_1$, and let x_2 be the result.

...

STEP n. Sample X_n from the conditional distribution of $X_n|S = s, X_1 = x_1, \ldots X_{n-1} = x_{n-1}$, and let x_n be the result.

To compute the necessary conditional distributions we introduce the partial sums, $S_1, \ldots, S_n$:

$$S_m = \sum_{i=m}^{n} X_i, \quad m = 1, \ldots, n.$$



Conditioning on the sum (cont.)

We then have:

$$
\begin{aligned}
&P(X_m = x_m \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1}, S = s) \\
&\quad = \frac{P(X_m = x_m, S = s \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1})}{P(S = s \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1})} \\
&\quad = \frac{P(X_m = x_m, S_{m+1} = s - \sum_{j=1}^{m} x_j)}{P(S_m = s - \sum_{j=1}^{m-1} x_j)} \\
&\quad = \frac{p_m^{x_m}(1-p_m)^{1-x_m} P(S_{m+1} = s - \sum_{j=1}^{m} x_j)}{P(S_m = s - \sum_{j=1}^{m-1} x_j)},
\end{aligned}
$$



Conditioning on the sum (cont.)

We then have:

$$
\begin{aligned}
&P(X_m = x_m \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1}, S = s) \\
&\quad = \frac{P(X_m = x_m, S = s \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1})}{P(S = s \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1})} \\
&\quad = \frac{P(X_m = x_m, S_{m+1} = s - \sum_{j=1}^{m} x_j)}{P(S_m = s - \sum_{j=1}^{m-1} x_j)} \\
&\quad = \frac{p_m^{x_m}(1-p_m)^{1-x_m} P(S_{m+1} = s - \sum_{j=1}^{m} x_j)}{P(S_m = s - \sum_{j=1}^{m-1} x_j)},
\end{aligned}
$$



Conditioning on the sum (cont.)

We then have:

$$
\begin{aligned}
&P(X_m = x_m \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1}, S = s) \\
&\quad = \frac{P(X_m = x_m, S = s \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1})}{P(S = s \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1})} \\
&\quad = \frac{P(X_m = x_m, S_{m+1} = s - \sum_{j=1}^{m} x_j)}{P(S_m = s - \sum_{j=1}^{m-1} x_j)} \\
&\quad = \frac{p_m^{x_m}(1-p_m)^{1-x_m} P(S_{m+1} = s - \sum_{j=1}^{m} x_j)}{P(S_m = s - \sum_{j=1}^{m-1} x_j)},
\end{aligned}
$$



Conditioning on the sum (cont.)

We then have:

$$
\begin{aligned}
&P(X_m = x_m \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1}, S = s) \\
&\quad = \frac{P(X_m = x_m, S = s \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1})}{P(S = s \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1})} \\
&\quad = \frac{P(X_m = x_m, S_{m+1} = s - \sum_{j=1}^{m} x_j)}{P(S_m = s - \sum_{j=1}^{m-1} x_j)} \\
&\quad = \frac{p_m^{x_m}(1-p_m)^{1-x_m} P(S_{m+1} = s - \sum_{j=1}^{m} x_j)}{P(S_m = s - \sum_{j=1}^{m-1} x_j)},
\end{aligned}
$$



Conditioning on the sum (cont.)

We then have:

$$
\begin{aligned}
&P(X_m = x_m \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1}, S = s) \\
&\quad = \frac{P(X_m = x_m, S = s \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1})}{P(S = s \mid X_1 = x_1, \ldots, X_{m-1} = x_{m-1})} \\
&\quad = \frac{P(X_m = x_m, S_{m+1} = s - \sum_{j=1}^{m} x_j)}{P(S_m = s - \sum_{j=1}^{m-1} x_j)} \\
&\quad = \frac{p_m^{x_m}(1-p_m)^{1-x_m} P(S_{m+1} = s - \sum_{j=1}^{m} x_j)}{P(S_m = s - \sum_{j=1}^{m-1} x_j)},
\end{aligned}
$$

The distributions of the partial sums $S_1, \ldots, S_n$ can be calculated before running the simulations using the same method as we did for k-out-of-n systems.



Conditioning on the sum (cont.)

NOTE: We only calculate the conditional probabilities we actually need along the way during the simulations, not the entire set of all possible conditional probabilities corresponding to all possible combinations of values of the X_j's.

Thus, in each simulation we calculate n probabilities, one for each X_j.

Each probability is calculated using a fixed number of operations (independent of n).

Hence, sampling from the conditional distribution of $\boldsymbol{X}$ given $S = s$, can be done in $O(n)$ time, i.e., just as fast as sampling from the unconditional distribution.



Conditioning on the sum (cont.)

When choosing the sample sizes $N_0, \ldots, N_n$, we may let $N_s \approx N \cdot P(S = s)$, $s = 0, 1, \ldots, n$.

However, it is possible to improve the results slightly. By examining the system, it is often easy to determine the size of the smallest minimal path and cut sets, and we introduce:

$$d = \text{The size of the smallest minimal path set}$$

$$c = \text{The size of the smallest minimal cut set}$$

We then have that $\theta_s = 0$ for $s < d$, and $\theta_s = 1$ for $s > n - c$, implying that $\mathrm{Var}(\phi|S = s) = 0$ for $s < d$, or $s > n - c$.

Hence, there is no point in spending simulations on estimating θ_s for $s < d$, or $s > n - c$, so we let $N_s = 0$ for $s < d$, or $s > n - c$. As a result, we have more simulations to spend on the remaining quantities.



Conditioning on the sum (cont.)

When choosing the sample sizes $N_0, \ldots, N_n$, we may let $N_s \approx N \cdot P(S = s)$, $s = 0, 1, \ldots, n$.

However, it is possible to improve the results slightly. By examining the system, it is often easy to determine the size of the smallest minimal path and cut sets, and we introduce:

$$d = \text{The size of the smallest minimal path set}$$

$$c = \text{The size of the smallest minimal cut set}$$

We then have that $\theta_s = 0$ for $s < d$, and $\theta_s = 1$ for $s > n - c$, implying that $\mathrm{Var}(\phi|S = s) = 0$ for $s < d$, or $s > n - c$.

Hence, there is no point in spending simulations on estimating θ_s for $s < d$, or $s > n - c$, so we let $N_s = 0$ for $s < d$, or $s > n - c$. As a result, we have more simulations to spend on the remaining quantities.



Conditioning on the sum (cont.)

When choosing the sample sizes $N_0, \ldots, N_n$, we may let $N_s \approx N \cdot P(S = s)$, $s = 0, 1, \ldots, n$.

However, it is possible to improve the results slightly. By examining the system, it is often easy to determine the size of the smallest minimal path and cut sets, and we introduce:

$$d = \text{The size of the smallest minimal path set}$$

$$c = \text{The size of the smallest minimal cut set}$$

We then have that $\theta_s = 0$ for $s < d$, and $\theta_s = 1$ for $s > n - c$, implying that $\mathrm{Var}(\phi|S = s) = 0$ for $s < d$, or $s > n - c$.

Hence, there is no point in spending simulations on estimating θ_s for $s < d$, or $s > n - c$, so we let $N_s = 0$ for $s < d$, or $s > n - c$. As a result, we have more simulations to spend on the remaining quantities.



Conditioning on the sum (cont.)

When choosing the sample sizes $N_0, \ldots, N_n$, we may let $N_s \approx N \cdot P(S = s)$, $s = 0, 1, \ldots, n$.

However, it is possible to improve the results slightly. By examining the system, it is often easy to determine the size of the smallest minimal path and cut sets, and we introduce:

$$d = \text{The size of the smallest minimal path set}$$

$$c = \text{The size of the smallest minimal cut set}$$

We then have that $\theta_s = 0$ for $s < d$, and $\theta_s = 1$ for $s > n - c$, implying that $\mathrm{Var}(\phi|S = s) = 0$ for $s < d$, or $s > n - c$.

Hence, there is no point in spending simulations on estimating θ_s for $s < d$, or $s > n - c$, so we let $N_s = 0$ for $s < d$, or $s > n - c$. As a result, we have more simulations to spend on the remaining quantities.



Conditioning on the sum (cont.)

An extreme situation occurs when the system is a k-out-of-n-system, i.e., when $\phi(\boldsymbol{X}) = I(S \geq k)$. For such systems $d = k$, and $c = (n - k + 1)$.

This implies that $n - c = n - (n - k + 1) = k - 1$.

Hence, $\theta_s = 0$ for $s < k$ and $\theta_s = 1$ for $s \geq k$.

Thus, the CMC-estimate simplifies to:

$$\hat{h}_{CMC} = \sum_{s=0}^{n} \hat{\theta}_s P(S = s) = \sum_{s=k}^{n} P(S = s)$$

We observe that in this case $\hat{h}_{CMC}$ does not contain any uncertain parameters. Thus, this estimate is equal to the true value h, and can be calculated without doing *any simulations at all*.



Conditioning on the sum (cont.)

For all other nontrivial systems, however, it is easy to see that we always have that $d \leq n - c$.

In order to ensure that we get improved results, we assume that $P(d \leq S \leq n - c) > 0$, and let:

$$N_s \approx N \cdot P(S = s)/P(d \leq S \leq n - c), \quad s = d, \ldots, n - c \, .$$

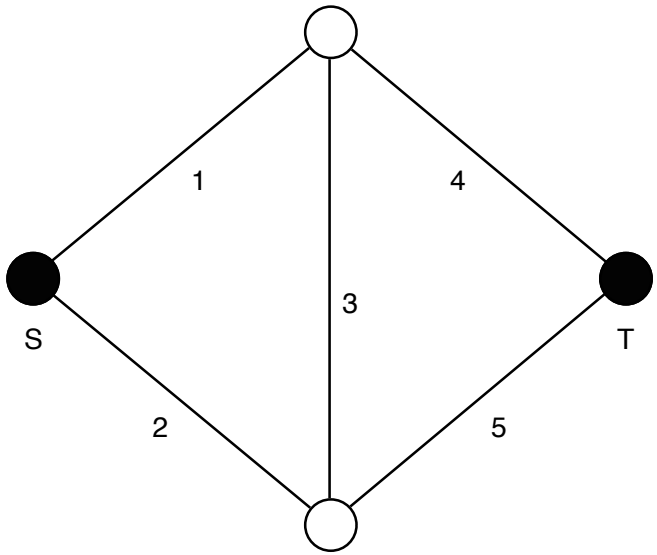
We can now show that:

$$\mathrm{Var}(\hat{h}_{CMC}) \leq P(d \leq S \leq n - c)\, \mathrm{Var}(\hat{h}_{MC})$$

Hence, if d and $n - c$ are close, the variance is reduced considerably.



Example – Bridge structure



Let (C, ϕ) be the bridge structure shown in the figure, where $C = \{1, \ldots, 5\}$. The vector of component state variables is $\boldsymbol{X} = (X_1, \ldots, X_5)$, and the structure function $\phi = \phi(\boldsymbol{X})$ is given by:

$$\phi(\boldsymbol{X}) = X_3(X_1 \amalg X_2)(X_4 \amalg X_5) + (1 - X_3)((X_1X_4) \amalg (X_2X_5))$$



Example – Bridge structure (cont.)

We assume that the component state variables are independent and that $P(X_i = 1) = p_i$, $i \in C$. We also let $\boldsymbol{p} = (p_1, \ldots, p_5)$.

By considering the structure function ϕ we get that:

$$h(\boldsymbol{p}) = p_3(p_1 \amalg p_2)(p_4 \amalg p_5) + (1 - p_3)((p_1 p_4) \amalg (p_2 p_5))$$

Assume more specifically that:

$$\boldsymbol{p} = (0.6, 0.4, 0.5, 0.4, 0.6)$$

By inserting this $\boldsymbol{p}$ into the reliability function $h(\boldsymbol{p})$ we get the true system reliability:

$$h = h(0.6, 0.4, 0.5, 0.4, 0.6) = 0.5$$



Example – Bridge structure (cont.)

We start out by using crude Monte Carlo simulation, and sample independent binary vectors:

$$\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N \stackrel{d}{=} \boldsymbol{X}$$

For $j = 1, \ldots, N$, the vector $\boldsymbol{X}_j$ is generated by generating a vector $\boldsymbol{U}_j$ of independent uniformly distributed variables such that:

$$U_{ij} \sim R[0,1], \quad i = 1, \ldots, n, \quad j = 1, \ldots, N.$$

Then the vector $\boldsymbol{X}_j = (X_{1j}, \ldots, X_{nj})$ is obtained as:

$$X_{ij} = \mathrm{I}(U_{ij} \leq p_i), \quad i = 1, \ldots, n, \quad j = 1, \ldots, N.$$

This implies that $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ get the correct distributions. In particular:

$$P(X_{ij} = 1) = P(U_{ij} \leq p_i) = p_i, \quad i = 1, \ldots, n, \quad j = 1, \ldots, N.$$



Example – Bridge structure (cont.)

For each vector $\boldsymbol{X}_j$ the resulting value of the structure function is calculated. The system reliability h is then estimated by:

$$\hat{h}_{MC} = \frac{1}{N} \sum_{r=1}^{N} \phi(\boldsymbol{X}_r)$$

In order to monitor the convergence of the simulations, we introduce:

$$\hat{h}_{MC}^{(m)} = \frac{1}{m} \sum_{r=1}^{m} \phi(\boldsymbol{X}_r), \quad m = 1, \ldots, N.$$

Thus, $\hat{h}_{MC}^{(m)}$ is the crude Monte Carlo estimate based on the m first simulations.

By plotting $\hat{h}_{MC}^{(m)}$ as a function of m, we get a convergence curve for the crude Monte Carlo estimate.



Example – Bridge structure (cont.)

```
################################################
#                                              #
#    Estimate system reliability using         #
#    crude Monte Carlo simulations.            #
#                                              #
################################################

import matplotlib.pyplot as plt
import numpy as np

# Number of simulations
num_sims = 1000

# Interval between h_hat calculations
h_interv = 10

# Random number generator seed. Set to 'None' for a random sequence
seed_num = 1234
gen = np.random.default_rng(seed = seed_num)

# Save plot to file or not?
save_plot = True

# Name of system
sys_name = "bridge"

# Number of components:
n = 5

# Component reliabilities
px = [0.0, 0.6, 0.4, 0.5, 0.4, 0.6]      # P(X_i = px[i]), i = 1, .., n. px[0] is not used
```



Example – Bridge structure (cont.)

```
# Component reliabilities
px = [0.0, 0.6, 0.4, 0.5, 0.4, 0.6]      # P(X_i = px[i]), i = 1, .., n. px[0] is not used

def coprod(x, y):
    return x + y - x*y

def phi(xx):
    return xx[3] * coprod(xx[1], xx[2]) * coprod(xx[4], xx[5]) + (1-xx[3]) * coprod(xx[1] * xx[4], xx[2] * xx[5])

def hh(pp):
    return pp[3] * coprod(pp[1], pp[2]) * coprod(pp[4], pp[5]) + (1-pp[3]) * coprod(pp[1] * pp[4], pp[2] * pp[5])

X = np.zeros(n + 1, dtype = int)        # The component state variables (X[0] is not used)

I = []
H = []
H_hat = []
T = 0

# Calculate the true system reliability
h = hh(px)
```



Example – Bridge structure (cont.)

```
# Calculate the true system reliability
h = hh(px)

for sim in range(num_sims):
    U = gen.uniform(0.0, 1.0, n+1)     # Uniform variables (U[0] is not used)
    for m in range(1, n+1):
        if U[m] <= px[m]:
            X[m] = 1
        else:
            X[m] = 0
    T += phi(X)
    if (sim+1) % h_interv == 0:
        h_hat = T / (sim+1)
        I.append(sim+1)
        H.append(h)
        H_hat.append(h_hat)

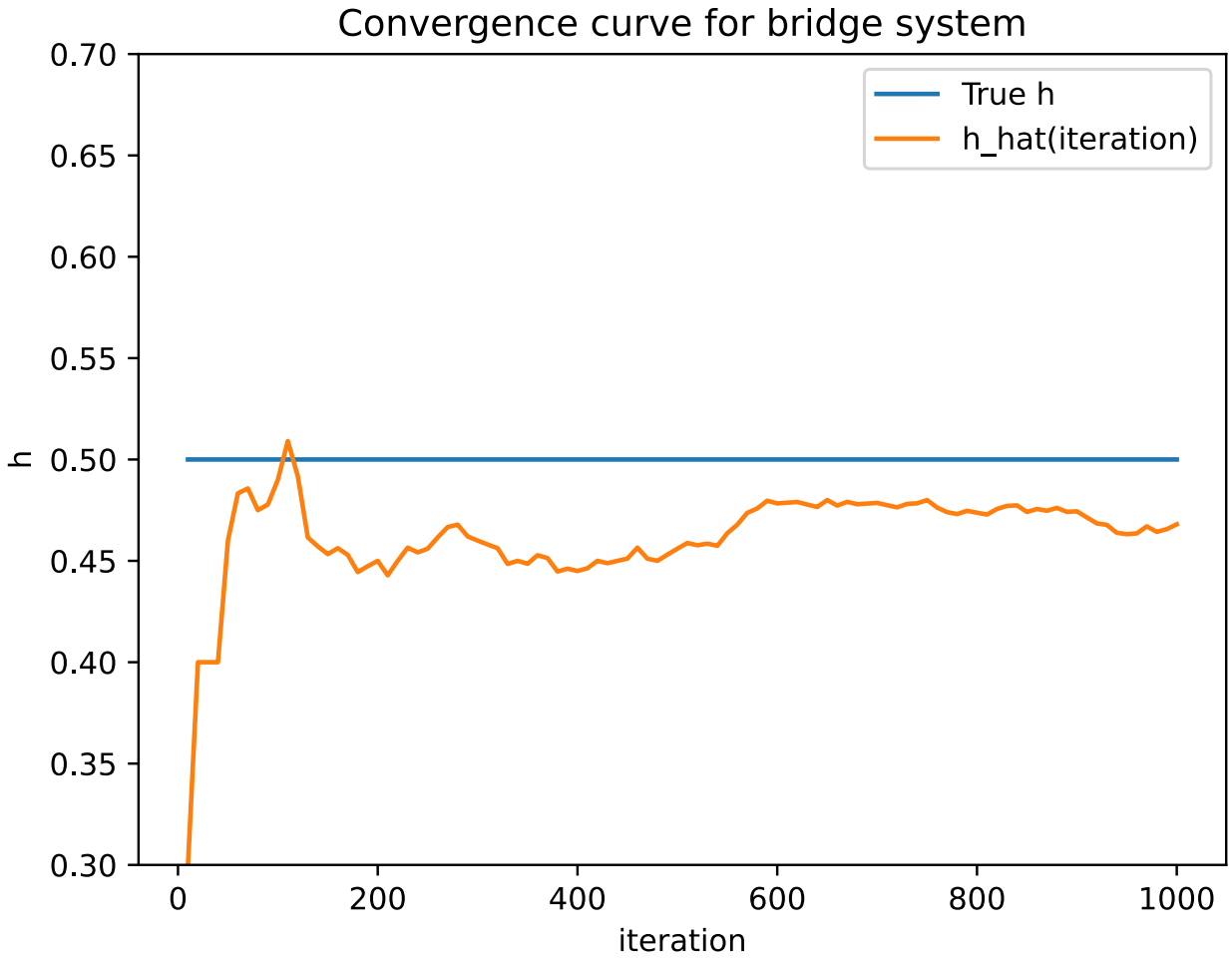
# Estimate final system reliability
h_hat = T / num_sims

print("h = ", h, ", h_hat = ", h_hat)

plt.plot(I, H, label='True h')
plt.plot(I, H_hat, label='h_hat(iteration)')
plt.ylim(h-0.2, h+0.2)
plt.xlabel('iteration')
plt.ylabel('h')
plt.title('Convergence curve for ' + sys_name + " system")
plt.legend()
if save_plot:
    plt.savefig(FOLDER + "/" + sys_name + ".pdf")
plt.show()
```



Example – Bridge structure (cont.)



RESULT: `h = 0.5 , h_hat = 0.468`

Example – Bridge structure (cont.)

We then proceed by using conditional Monte Carlo simulation, and introduce:

$$S = S(\boldsymbol{X}) = \sum_{i=1}^{5} X_i \in \{0, 1, \ldots, 5\}$$

We also introduce the partial sums:

$$S_m = \sum_{i=m}^{5} X_i, \quad m = 1, \ldots, 5.$$

For the conditional Monte Carlo simulation we need the distributions of $S_1, \ldots, S_5$, where $S = S_1$. These distributions are derived recursively:

$$P(S_5 = 1) = p_5, \quad P(S_5 = 0) = (1 - p_5)$$

$$P(S_4 = s) = P(S_5 = s - 1)p_4 + P(S_5 = s)(1 - p_4), \quad s = 0, 1, 2$$

$$\ldots$$

$$P(S_1 = s) = P(S_2 = s - 1)p_1 + P(S_2 = s)(1 - p_1), \quad s = 0, 1, \ldots, 5$$



Example – Bridge structure (cont.)

Given component reliabilities:

$$\boldsymbol{p} = (0.6, 0.4, 0.5, 0.4, 0.6)$$

we get the following distribution for $S = S_1$:

$$P(S = 0) = 0.0288$$
$$P(S = 1) = 0.1536$$
$$P(S = 2) = 0.3176$$
$$P(S = 3) = 0.3176$$
$$P(S = 4) = 0.1536$$
$$P(S = 5) = 0.0288$$



Example – Bridge structure (cont.)

The minimal path and cut sets of the bridge structure are:

- Minimal paths: $P_1 = \{1, 4\}$, $P_2 = \{1, 3, 5\}$, $P_3 = \{2, 3, 4\}$, $P_4 = \{2, 5\}$
- Minimal cuts: $K_1 = \{1, 2\}$, $K_2 = \{1, 3, 5\}$, $K_3 = \{2, 3, 4\}$, $K_4 = \{4, 5\}$

The size of the smallest minimal path set is $d = 2$, while the size of the smallest minimal cut set is $c = 2$.

We let:

$$\theta_s = E[\phi | S = s], \quad s = 0, 1, \ldots, 5,$$

and note that since $d = 2$ and $n - c = 5 - 2 = 3$, we have:

$$\begin{aligned} \theta_s &= 0 \quad \text{for } s = 0, 1, & (s < d) \\ \theta_s &= 1 \quad \text{for } s = 4, 5, & (s > n - c) \end{aligned}$$

Thus, we only need to estimate θ_2 and θ_3.



Example – Bridge structure (cont.)

In order to estimate θ_2 and θ_3 we should ideally sample two sets of independent binary vectors:

$$\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{N_2} \overset{d}{=} (\boldsymbol{X}|S=2)$$
$$\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_{N_3} \overset{d}{=} (\boldsymbol{X}|S=3)$$

where:

$$N_s \approx N \cdot \frac{P(S=s)}{P(2 \leq S \leq 3)} = N \cdot \frac{0.3176}{0.3176+0.3176} = \frac{N}{2}, \quad s = 2, 3.$$

However, we would like to plot a convergence curve as well, and then it is more convenient (and asymptotically equivalent) to merge the two sequences into one sequence of independent binary vectors $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$, where:

$$\boldsymbol{X}_j \overset{d}{=} (\boldsymbol{X}|S=S'_j), \quad j = 1, \ldots, N,$$

and where $S'_1, \ldots, S'_N$ are independent random variables such that:

$$P(S'_j = s) = \frac{P(S=s)}{P(2 \leq S \leq 3)} = \frac{1}{2}, \quad s = 2, 3, \quad j = 1, \ldots, N.$$



Example – Bridge structure (cont.)

In general if $S \in \{0, 1, \ldots, n\}$, and we want to sample S' from the conditional distribution of S given that $d \leq S \leq n - c$, i.e.,

$$P(S' = s') = \frac{P(S = s')}{P(d \leq S \leq n - c)}, \quad s = d, \ldots, n - c,$$

this can be done by sampling a uniformly distributed variable $U \sim [0, Q]$, where:

$$Q = P(d \leq S \leq n - c) = \sum_{s=d}^{n-c} P(S = s)$$

From this it follows that if $0 \leq a \leq b \leq Q$, then:

$$P(a < U \leq b) = \int_a^b Q^{-1} du = \frac{b - a}{Q}$$



Example – Bridge structure (cont.)

We then construct the random variable S' as follows:

$$S' = \begin{cases} d & \text{if } 0 < U \leq P(S = d) \\ d+1 & \text{if } 0 < U - P(S = d) \leq P(S = d+1) \\ \dots & \\ s' & \text{if } 0 < U - \sum_{s=d}^{s'-1} P(S = s) \leq P(S = s') \\ \dots & \\ n-c-1 & \text{if } 0 < U - \sum_{s=d}^{n-c-2} P(S = s) \leq P(S = n-c-1) \\ n-c & \text{otherwise} \end{cases}$$



Example – Bridge structure (cont.)

Note that the formula for transforming U to S' may also be written as:

$$S' = \begin{cases} d & \text{if } 0 < U \leq P(S = d) \\ d+1 & \text{if } P(S = d) < U \leq P(S = d) + P(S = d+1) \\ \dots & \\ s' & \text{if } \sum_{s=d}^{s'-1} P(S = s) < U \leq \sum_{s=d}^{s'} P(S = s) \\ \dots & \\ n-c-1 & \text{if } \sum_{s=d}^{n-c-2} P(S = s) < U \leq \sum_{s=d}^{n-c-1} P(S = s) \\ n-c & \text{otherwise} \end{cases}$$



Example – Bridge structure (cont.)

From this it follows that:

$$P(S' = s') = P(\sum_{s=d}^{s'-1} P(S = s) \leq U \leq \sum_{s=d}^{s'} P(S = s))$$

$$= \frac{P(S = s')}{P(d \leq S \leq n - c)} \quad v = d, \ldots, n - c$$

Thus, the random variable S' gets the desired distribution.



Example – Bridge structure (cont.)

The resulting convergence curve estimates of θ_2 and θ_3 can then be expressed as:

$$\hat{\theta}_s^{(m)} = \frac{\sum_{r=1}^{m} \phi(\boldsymbol{X}_r)\, \mathrm{I}(S_r' = s)}{\sum_{r=1}^{m} \mathrm{I}(S_r' = s)}, \quad s = 2, 3, \quad m = 1, \ldots, N.$$

NOTE: For small values of m it may happen that the denominator is zero in which case the estimator $\hat{\theta}_s^{(m)}$ is undefined. However, as m grows, this problem vanishes.

The corresponding convergence curve estimates of the system reliability is:

$$\hat{h}_{CMC}^{(m)} = \hat{\theta}_2^{(m)} P(S = 2) + \hat{\theta}_3^{(m)} P(S = 3) + P(S > 3), \quad m = 1, \ldots, N,$$

where we have used the fact that we know that $\theta_0 = \theta_1 = 0$, while $\theta_4 = \theta_5 = 1$.



Example – Bridge structure (cont.)

```
################################################
#                                              #
#    Estimate system reliability using         #
#    conditional Monte Carlo simulations.      #
#                                              #
################################################

import matplotlib.pyplot as plt
import numpy as np

# Number of simulations
num_sims = 1000

# Interval between h_hat calculations
h_interv = 10

# Random number generator seed. Set to 'None' for a random sequence
seed_num = 1234
gen = np.random.default_rng(seed = seed_num)

# Save plot to file or not?
save_plot = True

# Name of system
sys_name = "bridge"

# Number of components:
n = 5

# Length of shortest path
d = 2

# Length of shortest cut
c = 2
```



Example – Bridge structure (cont.)

```
# Component reliabilities
px = [0.0, 0.6, 0.4, 0.5, 0.4, 0.6]      # P(X_i = px[i]), i = 1, .., n. px[0] is not used


def coprod(x, y):
    return x + y - x*y

def phi(xx):
    return xx[3] * coprod(xx[1], xx[2]) * coprod(xx[4], xx[5]) + (1-xx[3]) * coprod(xx[1] * xx[4], xx[2] * xx[5])

def hh(pp):
    return pp[3] * coprod(pp[1], pp[2]) * coprod(pp[4], pp[5]) + (1-pp[3]) * coprod(pp[1] * pp[4], pp[2] * pp[5])


# Compute the distributions of S_1, ... , S_n, where S_m = X_m + ... + X_n, m = 1, ..., n
ps = np.zeros([n+1, n+1])                  # P(S_m = s) = ps[m,s], s = 0, 1, ..., (n-m+1). ps[0,s] is not used

ps[n,0] = 1.0 - px[n]                      # P(S_n = 0) = 1 - P(X_n = 1)
ps[n,1] = px[n]                            # P(S_n = 1) = P(X_n = 1)

for j in range(1, n):
    m = n-j
    ps[m,0] = ps[m+1,0] * (1.0 - px[m])                                  # P(S_m = 0) = P(S_{m+1} = 0) * P(X_m = 0)

    for s in range(1, n-m+1):
        ps[m,s] = ps[m+1,s-1] * px[m] + ps[m+1,s] * (1.0 - px[m])        # P(S_m = s) = P(S_{m+1} = s-1) * P(X_m = 1)
                                                                         # + P(S_{m+1} = s) * P(X_m = 0)
    ps[m,n-m+1] = ps[m+1,n-m] * px[m]                                    # P(S_m = n-m+1) = P(S_{m+1} = n-m) * P(X_m = 1)

# Print the distribution of S = S_1
for s in range(n+1):
    print("P(S = " + str(s) + ") =", ps[1,s])
```



Example – Bridge structure (cont.)

```
# Calculate pdc = P(d <= S_1 <= n-c)
pdc = 0
for s in range(d, n-c+1):
    pdc += ps[1,s]

# Calculate pcn = P(n-c < S_1 <= n)
pcn = 0
for s in range(n-c+1, n+1):
    pcn += ps[1,s]

# Sample S_1 from the set {d, ... , n-c}
def sampleS():
    u = gen.uniform(0.0, pdc)
    for s in range(d, n-c):
        if u < ps[1,s]:
            return s
        else:
            u -= ps[1,s]
    return n-c

X = np.zeros(n+1, dtype = int)        # The component state variables (X[0] is not used)
T = np.zeros(n+1, dtype = int)        # T[s] counts random path sets of size s = d, 1, ..., n-c
V = np.zeros(n+1, dtype = int)        # V[s] counts random sets of size s = d, 1, ..., n-c

I = []
H = []
H_hat = []
```



Example – Bridge structure (cont.)

```
# Calculate the true system reliability
h = hh(px)

# Run the simulations
for sim in range(num_sims):
    s = sampleS()
    V[s] += 1
    U = gen.uniform(0.0, 1.0, n+1)     # Uniform variables (U[0] is not used)
    sumx = 0
    for m in range(1, n):
        if sumx < s:
            p = px[m] * ps[m+1, s-sumx-1] / ps[m, s-sumx]
            if U[m] <= p:
                X[m] = 1
            else:
                X[m] = 0
            sumx += X[m]
        else:
            X[m] = 0
    if sumx < s:
        X[n] = 1
    else:
        X[n] = 0
    T[s] += phi(X)
    if (sim+1) % h_interv == 0:
        h_hat = pcn
        for s in range(d, n-c+1):
            if V[s] > 0:
                h_hat += ps[1,s] * T[s] / V[s]
        I.append(sim+1)
        H.append(h)
        H_hat.append(h_hat)
```



Example – Bridge structure (cont.)

```
# Print the estimated conditional reliabilities, theta_s, s = d, ..., n-c
for s in range(d, n-c+1):
    print("theta_" + str(s) + " =", T[s] / V[s])

# Estimate final system reliability
h_hat = pcn

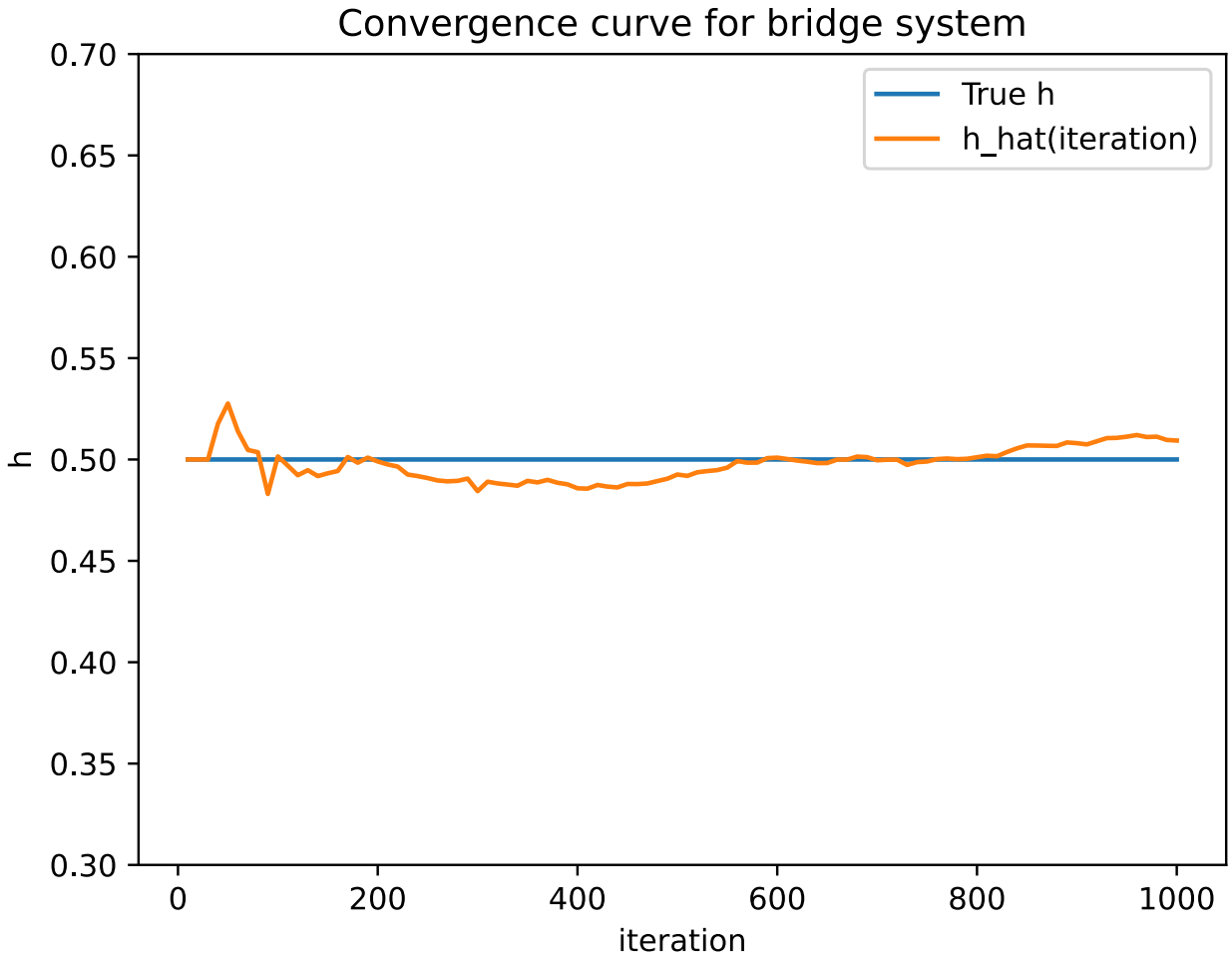
for s in range(d, n-c+1):
    h_hat += ps[1,s] * T[s] / V[s]

print("h = ", h, ", h_hat = ", h_hat)

plt.plot(I, H, label='True h')
plt.plot(I, H_hat, label='h_hat(iteration)')
plt.ylim(h - 0.2, h + 0.2)
plt.xlabel('iteration')
plt.ylabel('h')
plt.title('Convergence curve for ' + sys_name + " system")
plt.legend()
if save_plot:
    plt.savefig("cmcgen/" + sys_name + ".pdf")
plt.show()
```



Example – Bridge structure (cont.)



RESULT: `h = 0.5 , h_hat = 0.5093457820748705`