Array-RQMC for Markov Chains with Random Stopping Times

Pierre L'Ecuyer
Maxime Dion
Adam L'Archevêque-Gaudet

Informatique et Recherche Opérationnelle, Université de Montréal

1. Markov chain setting, Monte Carlo, classical RQMC.
2. Array-RQMC: preserving the low discrepancy of the chain's states.
3. Least-squares Monte Carlo for optimal stopping times.
4. Examples.

Monte Carlo for Markov Chains

Setting: A Markov chain with state space $\mathcal{X} \subseteq \mathbb{R}^\ell$, evolves as

$$X_0 = x_0, \qquad X_j = \varphi_j(X_{j-1}, \mathbf{U}_j),\ j \geq 1,$$

where the $\mathbf{U}_j$ are i.i.d. uniform r.v.'s over $(0,1)^d$. Want to estimate

$$\mu = \mathbb{E}[Y] \quad \text{where} \quad Y = \sum_{j=1}^{\tau} g_j(X_j)$$

and τ is a stopping time w.r.t. the filtration $\mathcal{F}\{(j, X_j),\ j \geq 0\}$.

Monte Carlo for Markov Chains

Setting: A Markov chain with state space $\mathcal{X} \subseteq \mathbb{R}^\ell$, evolves as

$$X_0 = x_0, \qquad X_j = \varphi_j(X_{j-1}, \mathbf{U}_j),\ j \geq 1,$$

where the $\mathbf{U}_j$ are i.i.d. uniform r.v.'s over $(0,1)^d$. Want to estimate

$$\mu = \mathbb{E}[Y] \quad \text{where} \quad Y = \sum_{j=1}^{\tau} g_j(X_j)$$

and τ is a stopping time w.r.t. the filtration $\mathcal{F}\{(j, X_j),\ j \geq 0\}$.

Ordinary MC: For $i = 0, \ldots, n-1$, generate $X_{i,j} = \varphi_j(X_{i,j-1}, \mathbf{U}_{i,j})$, $j = 1, \ldots, \tau_i$, where the $\mathbf{U}_{i,j}$'s are i.i.d. $\mathrm{U}(0,1)^d$. Estimate μ by

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{\tau_i} g_j(X_{i,j}) = \frac{1}{n} \sum_{i=1}^{n} Y_i.$$

Classical RQMC for Markov Chains

Put $\mathbf{V}_i = (\mathbf{U}_{i,1}, \mathbf{U}_{i,2}, \ldots)$. Estimate μ by

$$\hat{\mu}_{\mathrm{rqmc},n} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{\tau_i} g_j(X_{i,j})$$

where $P_n = \{\mathbf{V}_0, \ldots, \mathbf{V}_{n-1}\} \subset (0,1)^s$ has the following properties:
(a) each point $\mathbf{V}_i$ has the uniform distribution over $(0,1)^s$;
(b) P_n has low discrepancy.

Dimension is $s = \inf\{s' : \mathbb{P}[d\tau \leq s'] = 1\}$.
For a Markov chain, the dimension s is often very large!

Array-RQMC for Markov Chains

[Lécot, Tuffin, L'Ecuyer 2004, 2008]

Simulate n chains in parallel. At each step, use an RQMC point set P_n to advance all the chains by one step, while inducing global negative dependence across the chains.

Intuition: The empirical distribution of $S_{n,j} = \{X_{0,j}, \ldots, X_{n-1,j}\}$, should be a more accurate approximation of the theoretical distribution of X_j, for each j, than with crude Monte Carlo. The discrepancy between these two distributions should be as small as possible.

Array-RQMC for Markov Chains

[Lécot, Tuffin, L'Ecuyer 2004, 2008]

Simulate n chains in parallel. At each step, use an RQMC point set P_n to advance all the chains by one step, while inducing global negative dependence across the chains.

Intuition: The empirical distribution of $S_{n,j} = \{X_{0,j}, \ldots, X_{n-1,j}\}$, should be a more accurate approximation of the theoretical distribution of X_j, for each j, than with crude Monte Carlo. The discrepancy between these two distributions should be as small as possible.

Then, we will have small variance for the (unbiased) estimators:

$$\mu_j = \mathbb{E}[g_j(X_j)] \approx \frac{1}{n}\sum_{i=0}^{n-1} g_j(X_{i,j}) \quad \text{and} \quad \mu = \mathbb{E}[Y] \approx \frac{1}{n}\sum_{i=0}^{n-1} Y_i \,.$$

Array-RQMC for Markov Chains

[Lécot, Tuffin, L'Ecuyer 2004, 2008]

Simulate n chains in parallel. At each step, use an RQMC point set P_n to advance all the chains by one step, while inducing global negative dependence across the chains.

Intuition: The empirical distribution of $S_{n,j} = \{X_{0,j}, \ldots, X_{n-1,j}\}$, should be a more accurate approximation of the theoretical distribution of X_j, for each j, than with crude Monte Carlo. The discrepancy between these two distributions should be as small as possible.

Then, we will have small variance for the (unbiased) estimators:

$$\mu_j = \mathbb{E}[g_j(X_j)] \approx \frac{1}{n}\sum_{i=0}^{n-1} g_j(X_{i,j}) \quad \text{and} \quad \mu = \mathbb{E}[Y] \approx \frac{1}{n}\sum_{i=0}^{n-1} Y_i\,.$$

How can we preserve low-discrepancy of $X_{0,j}, \ldots, X_{n-1,j}$ when j increases? Can we quantify the variance improvement?

To simplify, suppose each X_j is a uniform r.v. over $(0,1)^\ell$.

Select a discrepancy measure D for the point set $S_{n,j} = \{X_{0,j}, \ldots, X_{n-1,j}\}$ over $(0,1)^\ell$, and a corresponding measure of variation V, such that

$$\mathrm{Var}[\hat{\mu}_{\mathrm{rqmc},j,n}] = \mathbb{E}[(\hat{\mu}_{\mathrm{rqmc},j,n} - \mu_j)^2] \leq \mathbb{E}[D^2(S_{n,j})]\ V^2(g_j).$$

To simplify, suppose each X_j is a uniform r.v. over $(0,1)^\ell$.

Select a discrepancy measure D for the point set $S_{n,j} = \{X_{0,j}, \ldots, X_{n-1,j}\}$ over $(0,1)^\ell$, and a corresponding measure of variation V, such that

$$\mathrm{Var}[\hat{\mu}_{\mathrm{rqmc},j,n}] = \mathbb{E}[(\hat{\mu}_{\mathrm{rqmc},j,n} - \mu_j)^2] \leq \mathbb{E}[D^2(S_{n,j})]\ V^2(g_j).$$

If D is defined via a reproducing kernel Hilbert space, then, for some random ξ_j (that generally depends on $S_{n,j}$),

$$\begin{aligned}\mathbb{E}[D^2(S_{n,j})] &= \mathrm{Var}\left[\frac{1}{n}\sum_{i=1}^{n}\xi_j(X_{i,j})\right] = \mathrm{Var}\left[\frac{1}{n}\sum_{i=1}^{n}(\xi_j \circ \varphi_j)(X_{i,j-1}, \mathbf{U}_{i,j}))\right] \\ &\leq \mathbb{E}[D^2_{(2)}(Q_n)] \cdot V^2_{(2)}(\xi_j \circ \varphi_j)\end{aligned}$$

for some other discrepancy $D_{(2)}$ over $(0,1)^{\ell+d}$, where $Q_n = \{(X_{0,j-1}, \mathbf{U}_{0,j}), \ldots, (X_{n-1,j-1}, \mathbf{U}_{n-1,j})\}$.

Heuristic: Under appropriate conditions, we should have $V_{(2)}(\xi_j \circ \varphi_j) < \infty$ and $\mathbb{E}[D^2_{(2)}(Q_n)] = O(n^{-\alpha+\epsilon})$ for some $\alpha \geq 1$.

In the points $(X_{i,j-1}, \mathbf{U}_{i,j})$ of Q_n, the $\mathbf{U}_{i,j}$ can be defined via some RQMC scheme, but the $X_{i,j-1}$ cannot be chosen; they are determined by the history of the chains.

The idea is to select a low-discrepancy point set

$$\tilde{Q}_n = \{(\mathbf{w}_0, \mathbf{U}_0), \ldots, (\mathbf{w}_{n-1}, \mathbf{U}_{n-1})\},$$

where the $\mathbf{w}_i \in [0,1)^\ell$ are fixed and the $\mathbf{U}_i \in (0,1)^d$ are randomized, and then define a bijection between the states $X_{i,j-1}$ and the $\mathbf{w}_i$ so that the $X_{i,j-1}$ are "close" to the $\mathbf{w}_i$ (small discrepancy between the two sets).

Bijection defined by a permutation π_j of $S_{n,j}$.

In the points $(X_{i,j-1}, \mathbf{U}_{i,j})$ of Q_n, the $\mathbf{U}_{i,j}$ can be defined via some RQMC scheme, but the $X_{i,j-1}$ cannot be chosen; they are determined by the history of the chains.

The idea is to select a low-discrepancy point set

$$\tilde{Q}_n = \{(\mathbf{w}_0, \mathbf{U}_0), \ldots, (\mathbf{w}_{n-1}, \mathbf{U}_{n-1})\},$$

where the $\mathbf{w}_i \in [0,1)^\ell$ are fixed and the $\mathbf{U}_i \in (0,1)^d$ are randomized, and then define a bijection between the states $X_{i,j-1}$ and the $\mathbf{w}_i$ so that the $X_{i,j-1}$ are "close" to the $\mathbf{w}_i$ (small discrepancy between the two sets).

Bijection defined by a permutation π_j of $S_{n,j}$.

State space in $\mathbb{R}^\ell$: same algorithm essentially.

Array-RQMC algorithm

$X_{i,0} \leftarrow x_0$, for $i = 0, \ldots, n-1$;
for $j = 1, 2, \ldots, \max_i \tau_i$ **do**
  Randomize afresh $\{\mathbf{U}_{0,j}, \ldots, \mathbf{U}_{n-1,j}\}$ in $\tilde{Q}_n$;
  $X_{i,j} = \varphi_j(X_{\pi_j(i),j-1}, \mathbf{U}_{i,j})$, for $i = 0, \ldots, n-1$;
  Compute the permutation π_{j+1} (sort the states);
end for
Estimate μ by the average $\bar{Y}_n = \hat{\mu}_{\mathrm{rqmc},n}$.

Array-RQMC algorithm

$X_{i,0} \leftarrow x_0$, for $i = 0, \ldots, n-1$;
for $j = 1, 2, \ldots, \max_i \tau_i$ **do**
  Randomize afresh $\{\mathbf{U}_{0,j}, \ldots, \mathbf{U}_{n-1,j}\}$ in $\tilde{Q}_n$;
  $X_{i,j} = \varphi_j(X_{\pi_j(i),j-1}, \mathbf{U}_{i,j})$, for $i = 0, \ldots, n-1$;
  Compute the permutation π_{j+1} (sort the states);
end for
Estimate μ by the average $\bar{Y}_n = \hat{\mu}_{\mathrm{rqmc},n}$.

Theorem: The average $\bar{Y}_n$ is an unbiased estimator of μ.

Can estimate $\mathrm{Var}[\bar{Y}_n]$ by the empirical variance of m indep. realizations.

Mapping chains to points

Multivariate sort:

Sort the states (chains) by first coordinate, in n_1 packets of size n/n_1.

Sort each packet by second coordinate, in n_2 packets of size $n/n_1 n_2$.

$\vdots$

At the last level, sort each packet of size n_ℓ by the last coordinate.

Choice of $n_1, n_2, ..., n_\ell$?

Mapping chains to points

Multivariate sort:
Sort the states (chains) by first coordinate, in n_1 packets of size n/n_1.

Sort each packet by second coordinate, in n_2 packets of size $n/n_1 n_2$.

$\vdots$

At the last level, sort each packet of size n_ℓ by the last coordinate.

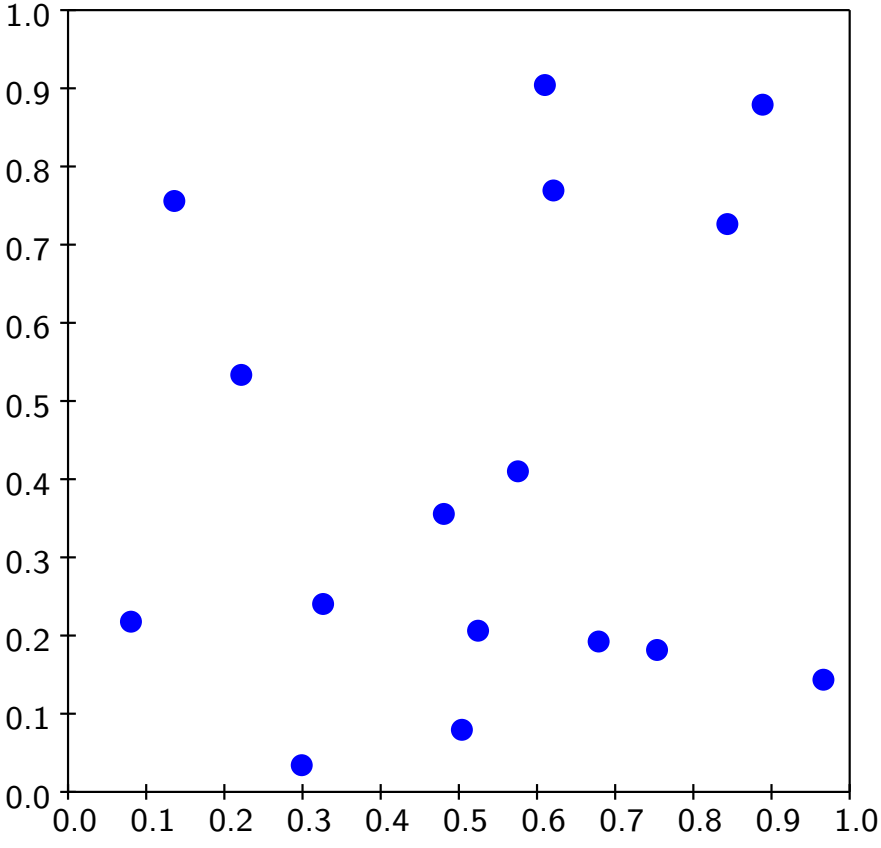
Choice of $n_1, n_2, ..., n_\ell$?

Generalization:
Define a sorting function $v : \mathcal{X} \to [0,1)^c$ and apply the multivariate sort (in c dimensions) to the transformed points $v(X_{i,j})$.

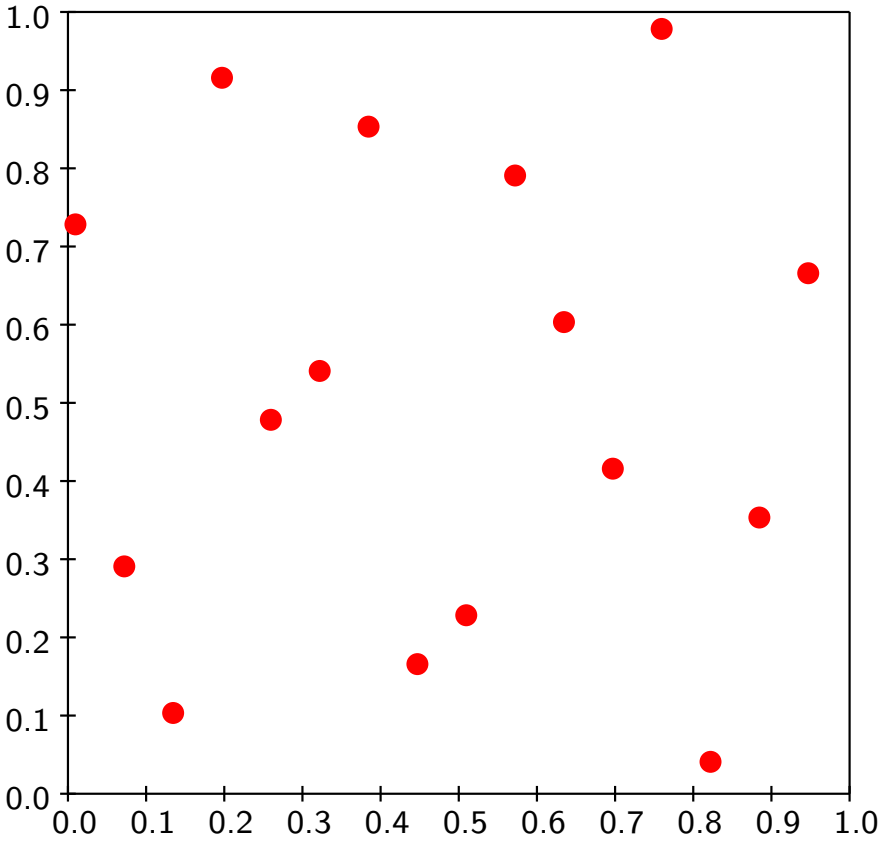
Choice of v: Two states mapped to nearby values of v should be approximately equivalent.

A (4,4) mapping

States of the chains

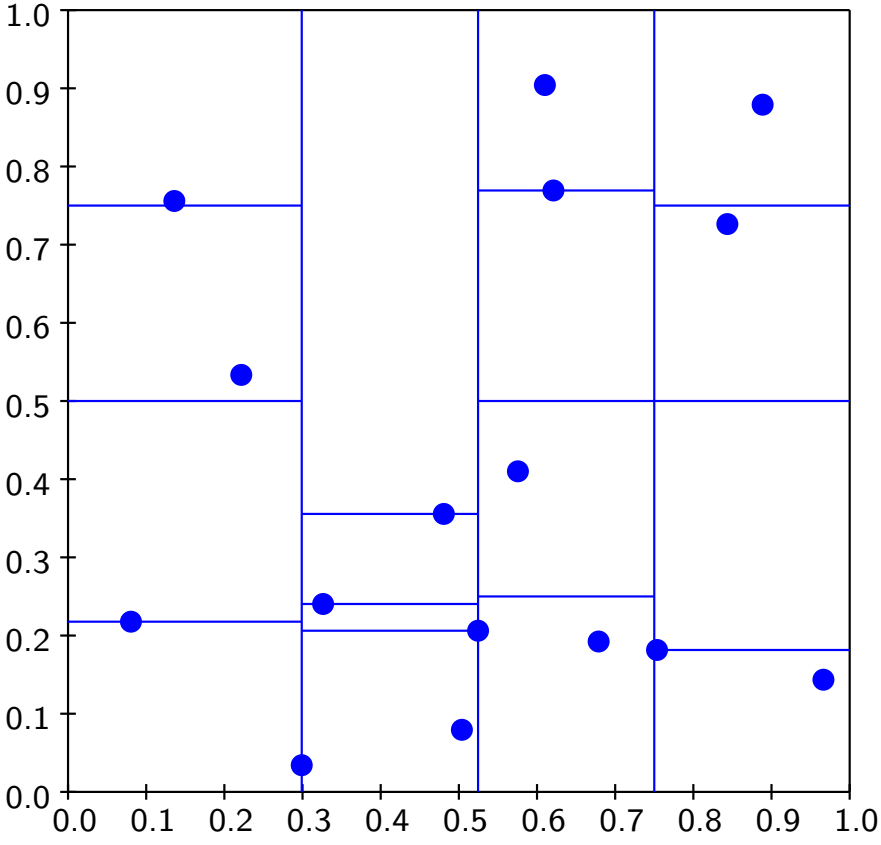


Sobol' net in 2 dimensions with digital shift

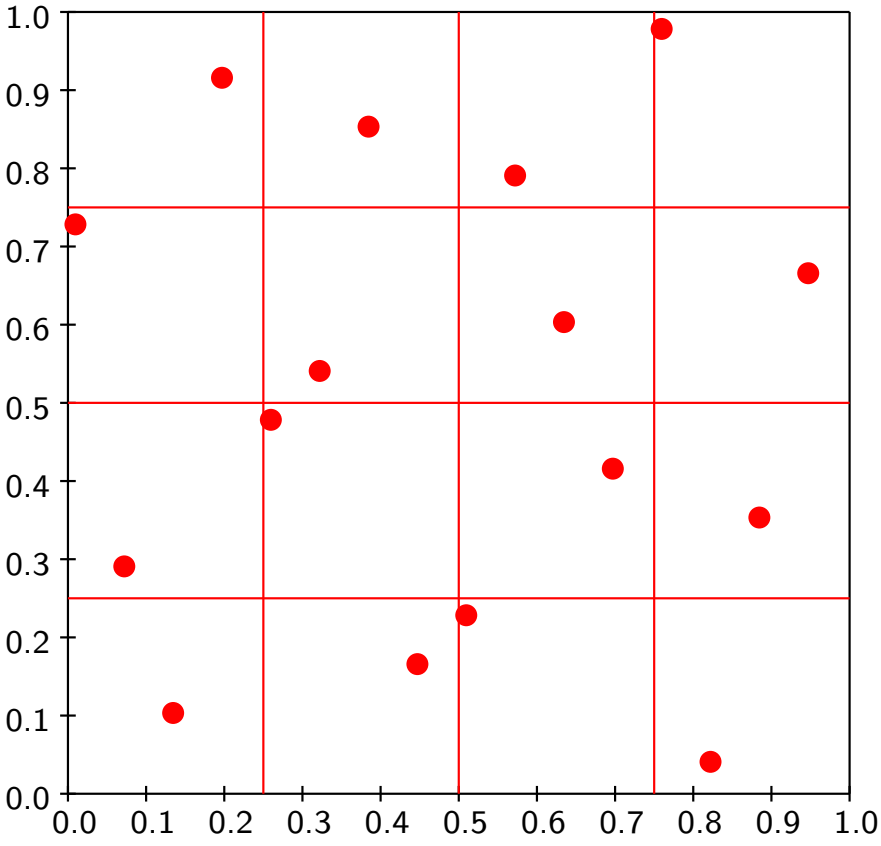


A (4,4) mapping

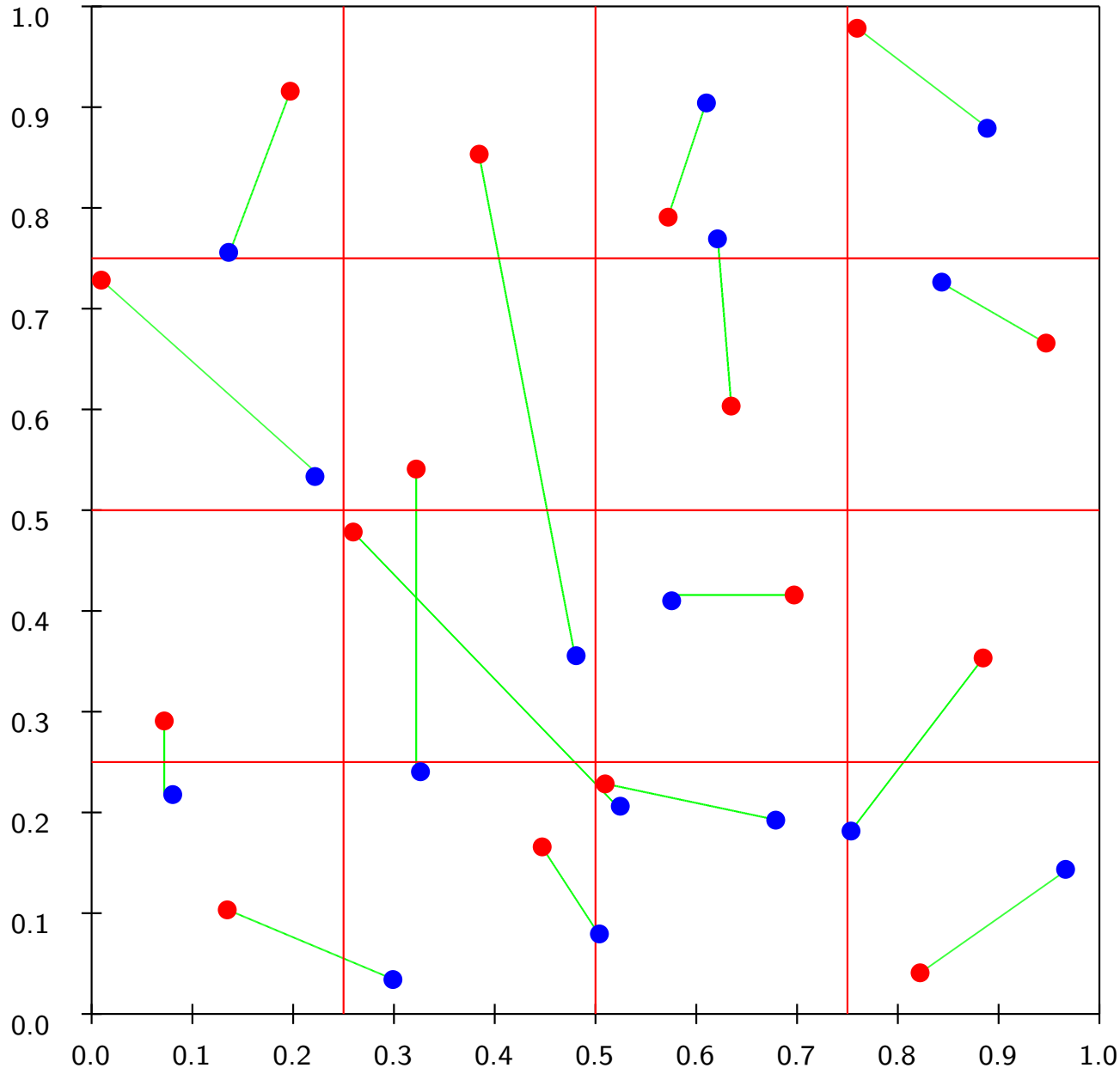
States of the chains



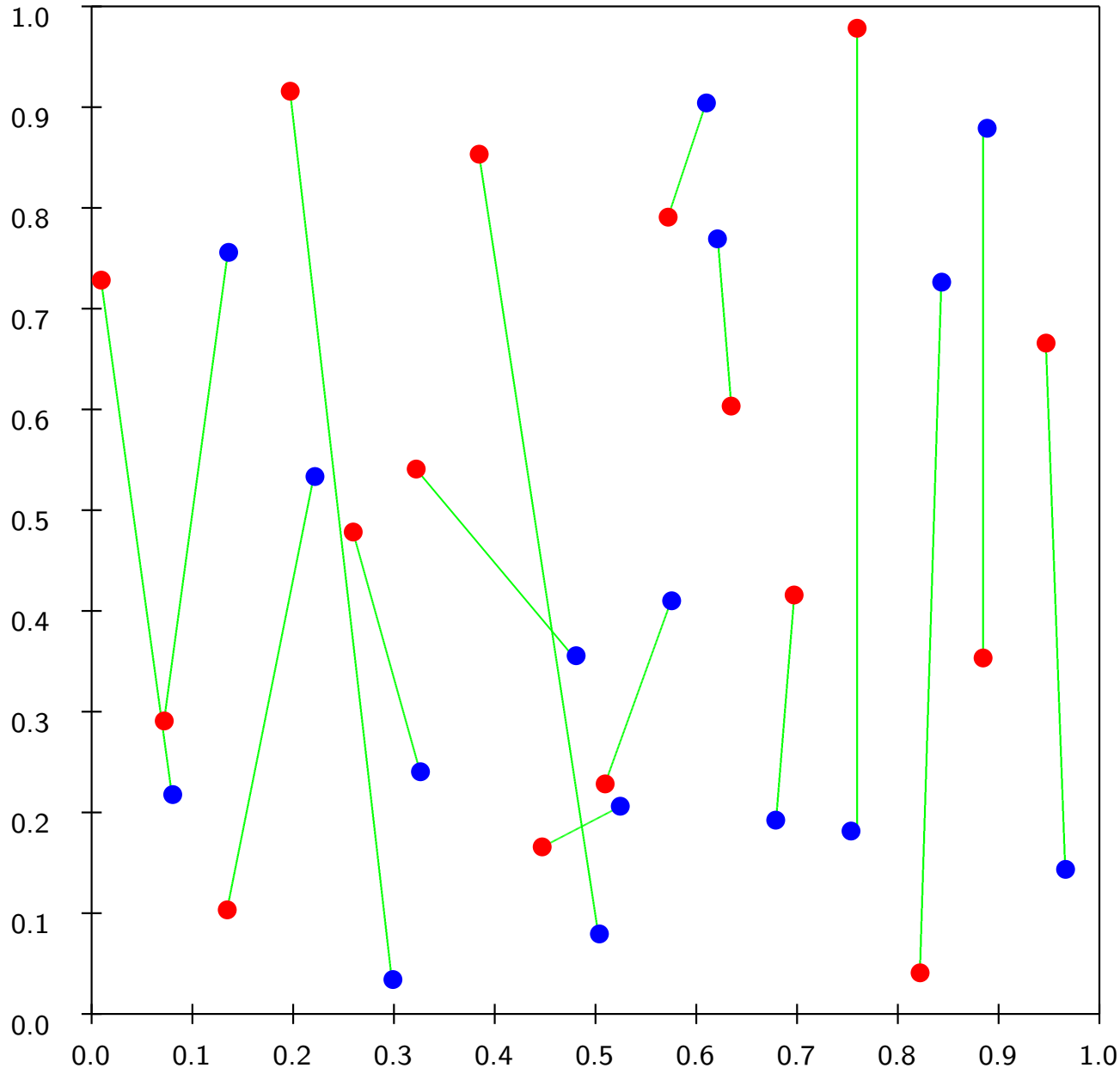
Sobol' net in 2 dimensions with digital shift



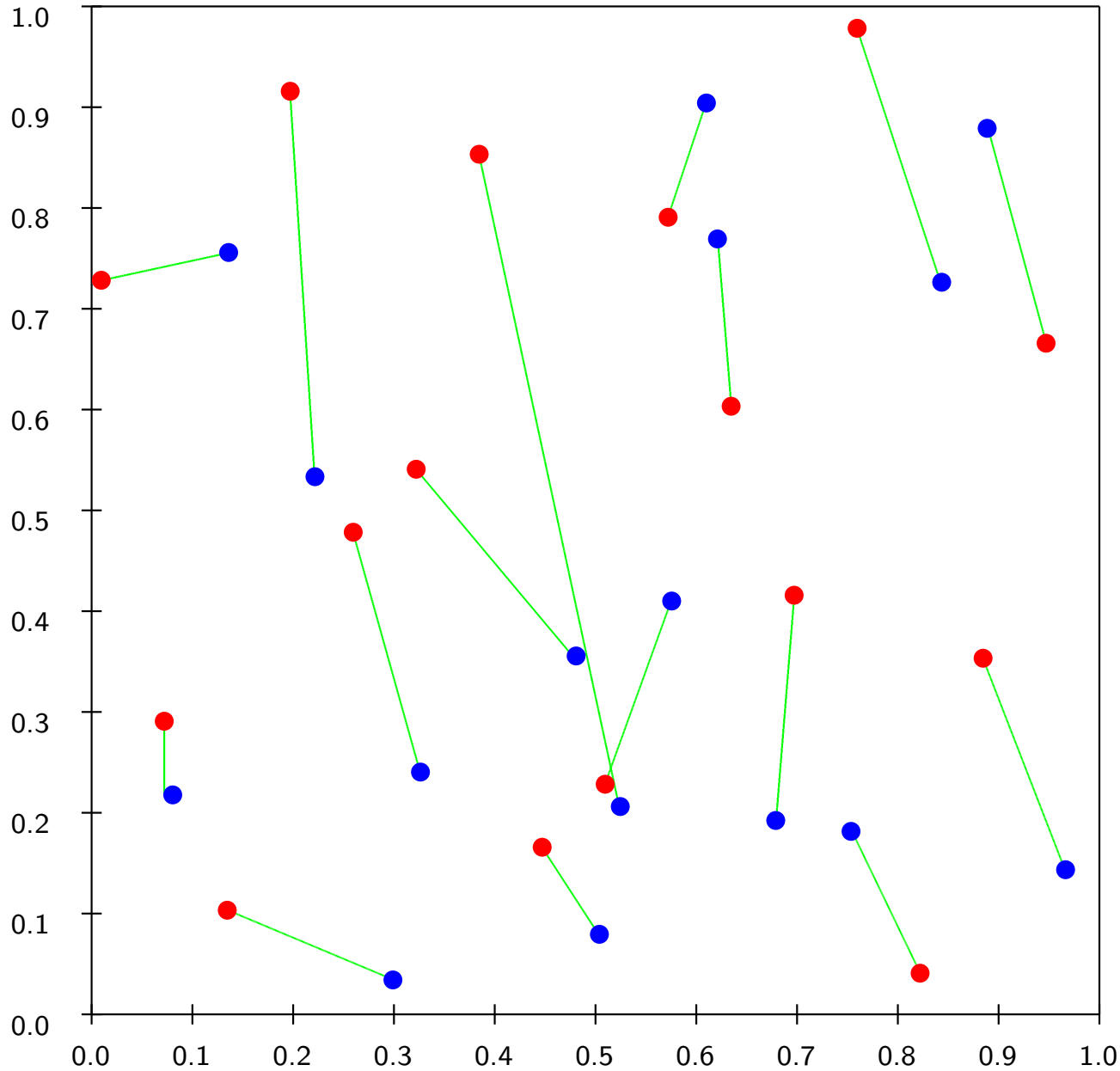
A (4,4) mapping



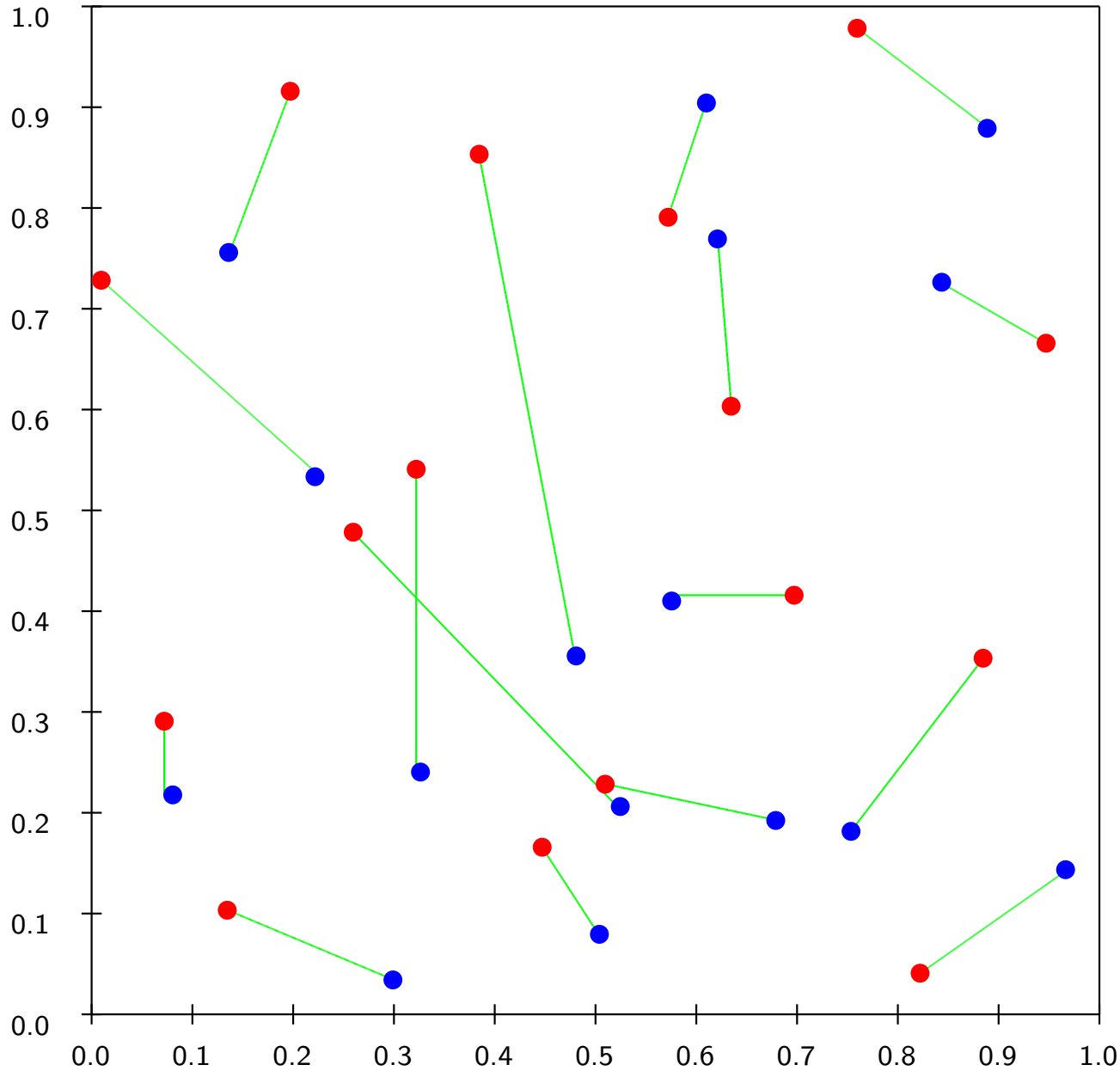
A (16,1) mapping, sorting along first coordinate



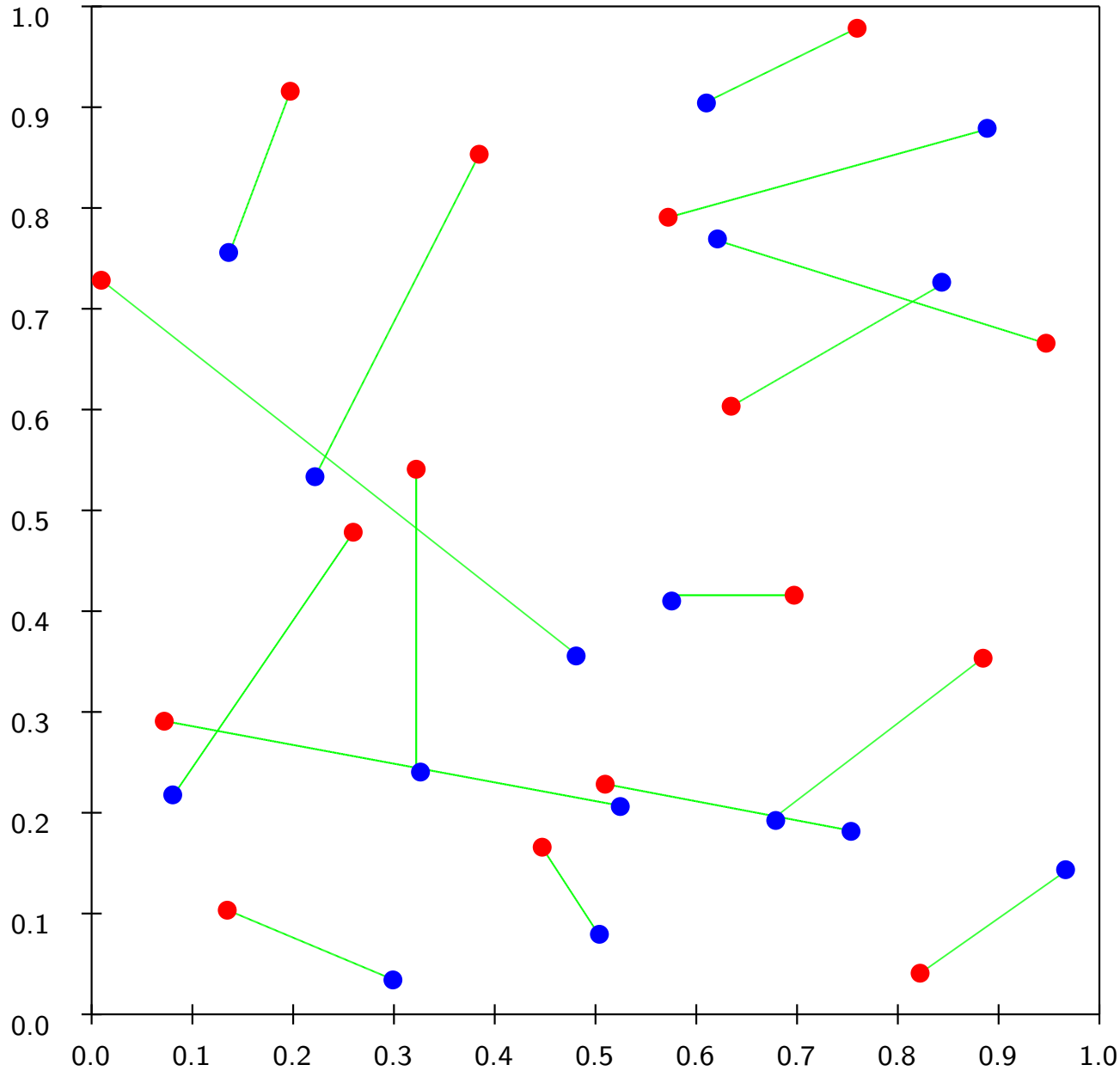
A (8,2) mapping



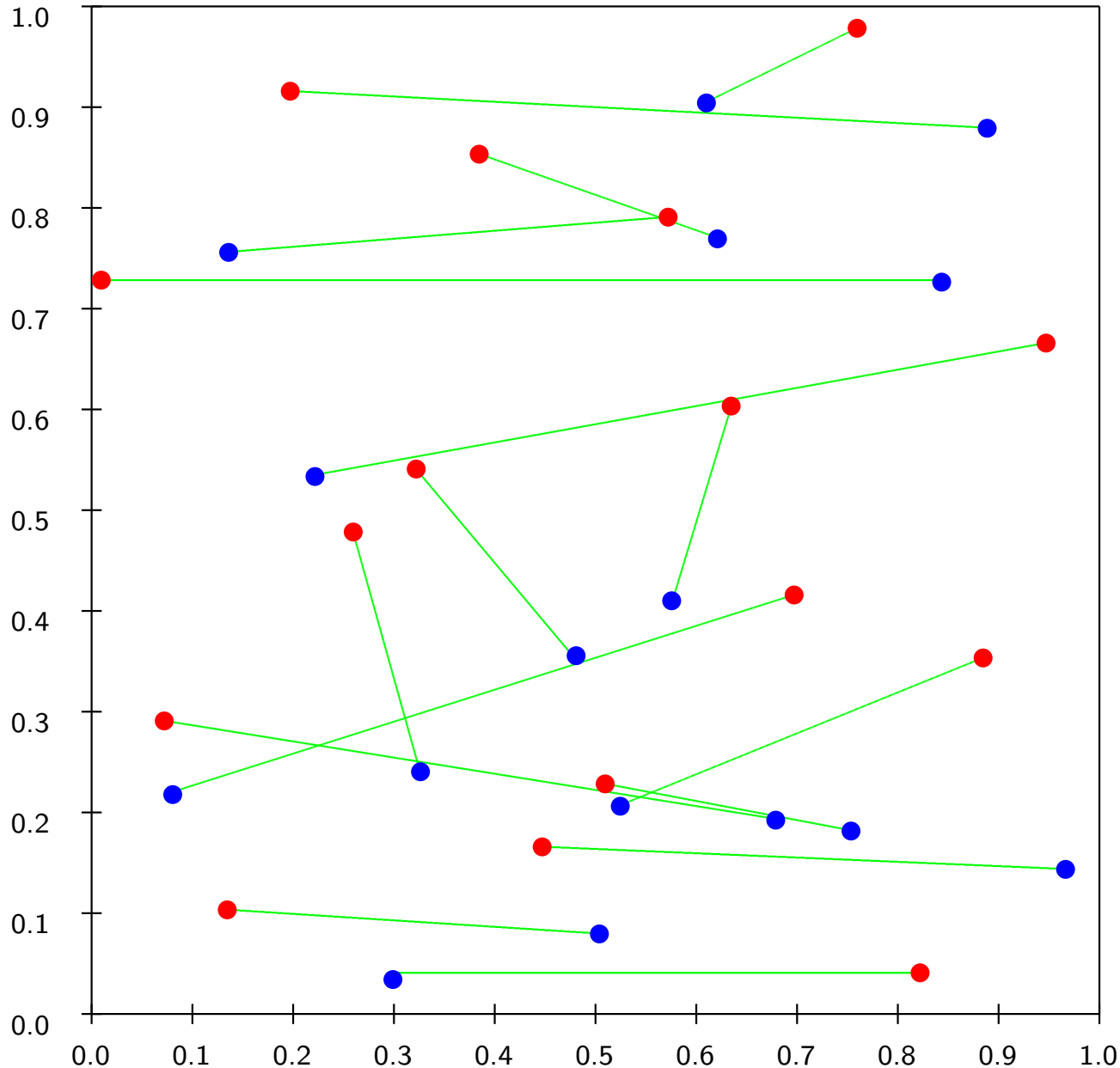
A (4,4) mapping



A (2,8) mapping



A (1,16) mapping, sorting along second coordinate



Dynamic programming for optimal stopping times

Suppose the stopping time τ is a decision determined by a stopping policy $\pi = (\nu_0, \nu_1, \ldots, \nu_{T-1})$ where $\nu_j : \mathcal{X} \to \{\text{stop now}, \text{ wait}\}$.
Suppose also that must stop at or before step T.

Dynamic programming for optimal stopping times

Suppose the stopping time τ is a decision determined by a stopping policy $\pi = (\nu_0, \nu_1, \ldots, \nu_{T-1})$ where $\nu_j : \mathcal{X} \to \{\text{stop now}, \text{ wait}\}$.
Suppose also that must stop at or before step T.

Dynamic programming equations:

$$
\begin{aligned}
V_T(x) &= g_T(x), \\
Q_j(x) &= \mathbb{E}[V_{j+1}(X_{j+1}) \mid X_j = x], && \text{(continuation value)} \\
V_j(x) &= \max[g_j(x),\, Q_j(x)], && \text{(optimal value)} \\
\nu_j^*(x) &= \begin{cases} \text{stop now} & \text{if } g_j(x) \geq Q_j(x) \\ \text{wait} & \text{otherwise,} \end{cases} && \text{(optimal decision)}
\end{aligned}
$$

for $j = T-1, \ldots, 0$ and all $x \in X$.

Hard to solve when the state space is large and multidimensional.

Can approximate Q_j with a small set of basis functions.
$\{\psi_k : \mathcal{X} \to \mathbb{R},\ 1 \le k \le m\}$:

$$\tilde{Q}_j(x) = \sum_{k=1}^{m} \beta_{j,k} \psi_k(x)$$

where $\boldsymbol{\beta}_j = (\beta_{j,1}, \ldots, \beta_{j,m})^{\mathrm{t}}$ can be determined by least-squares regression, using an approximation $W_{i,j}$ of $Q_j(x_{i,j})$ at a set of points $x_{i,j}$.

We solve

$$\min_{\boldsymbol{\beta}_j \in \mathbb{R}^m} \sum_{i=1}^{n} \left(\tilde{Q}_j(x_{i,j}) - W_{i,j+1} \right)^2 .$$

A set of representative states $x_{i,j}$ at each step j can be generated by Monte Carlo, or RQMC, or array-RQMC.

Regression-based least-squares Monte Carlo

Tsistiklis and Van Roy (2000) (TvR);

Simulate n indep. trajectories of the chain $\{X_j, j = 0, \ldots, T\}$,
and let $X_{i,j}$ be the state for trajectory i at step j;

$W_{i,T} \leftarrow g_T(X_{i,T}), \quad i = 1, \ldots, n;$

for $j = T - 1, \ldots, 0$ **do**

Compute the vector $\boldsymbol{\beta}_j$ that minimizes

$$\sum_{i=1}^{n} \left(\sum_{k=1}^{m} \beta_{j,k} \psi_k(X_{i,j}) - W_{i,j+1} \right)^2 .$$

$W_{i,j} \leftarrow \max[g_j(X_{i,j}), \tilde{Q}_j(X_{i,j})], \quad i = 1, \ldots, n;$

end for

return $\hat{Q}_0(\mathbf{x}_0) = (W_{1,0} + \cdots + W_{n,0})/n$ as an estimate of $Q_0(\mathbf{x}_0)$;

Regression-based least-squares Monte Carlo

Tsistiklis and Van Roy (2000) (TvR);

Simulate n indep. trajectories of the chain $\{X_j, j = 0, \ldots, T\}$,
and let $X_{i,j}$ be the state for trajectory i at step j;

$W_{i,T} \leftarrow g_T(X_{i,T}), \quad i = 1, \ldots, n;$

for $j = T - 1, \ldots, 0$ **do**

Compute the vector $\boldsymbol{\beta}_j$ that minimizes

$$\sum_{i=1}^{n} \left(\sum_{k=1}^{m} \beta_{j,k} \psi_k(X_{i,j}) - W_{i,j+1} \right)^2 .$$

$W_{i,j} \leftarrow \max[g_j(X_{i,j}), \tilde{Q}_j(X_{i,j})], \quad i = 1, \ldots, n;$

end for

return $\hat{Q}_0(\mathbf{x}_0) = (W_{1,0} + \cdots + W_{n,0})/n$ as an estimate of $Q_0(\mathbf{x}_0)$;

Longstaff and Schwartz (2001) (LSM): Define $W_{i,j}$ instead by

$$W_{i,j} = \begin{cases} g_j(X_{i,j}) & \text{if } g_k(X_{j,k}) \geq \tilde{Q}_j(X_{i,j}); \\ W_{i,j+1} & \text{otherwise .} \end{cases}$$

Example: a simple put option

Asset price obeys GBM $\{S(t),\, t \geq 0\}$ with drift (interest rate) $\mu = 0.05$, volatility $\sigma = 0.08$, initial value $S(0) = 100$.

For American version, exercise dates are $t_j = j/16$ for $j = 1, \ldots, 16$.
Payoff at t_j: $g_j(S(t_j)) = e^{-0.05 t_j} \max(0, K - S(t_j))$, where $K = 101$.

European version: Can exercise only at $t_{16} = 1$.

Example: a simple put option

Asset price obeys GBM $\{S(t),\, t \geq 0\}$ with drift (interest rate) $\mu = 0.05$, volatility $\sigma = 0.08$, initial value $S(0) = 100$.

For American version, exercise dates are $t_j = j/16$ for $j = 1, \ldots, 16$.
Payoff at t_j: $g_j(S(t_j)) = e^{-0.05 t_j} \max(0, K - S(t_j))$, where $K = 101$.

European version: Can exercise only at $t_{16} = 1$.

One-dimensional state $X_j = S(t_j)$. Sorting for array-RQMC is simple.

Basis functions for regression-based MC:
polynomials $\psi_k(x) = (x - 101)^{k-1}$ for $k = 1, \ldots, 5$.

Example: a simple put option

Asset price obeys GBM $\{S(t),\, t \geq 0\}$ with drift (interest rate) $\mu = 0.05$, volatility $\sigma = 0.08$, initial value $S(0) = 100$.

For American version, exercise dates are $t_j = j/16$ for $j = 1, \ldots, 16$.
Payoff at t_j: $g_j(S(t_j)) = e^{-0.05 t_j} \max(0, K - S(t_j))$, where $K = 101$.

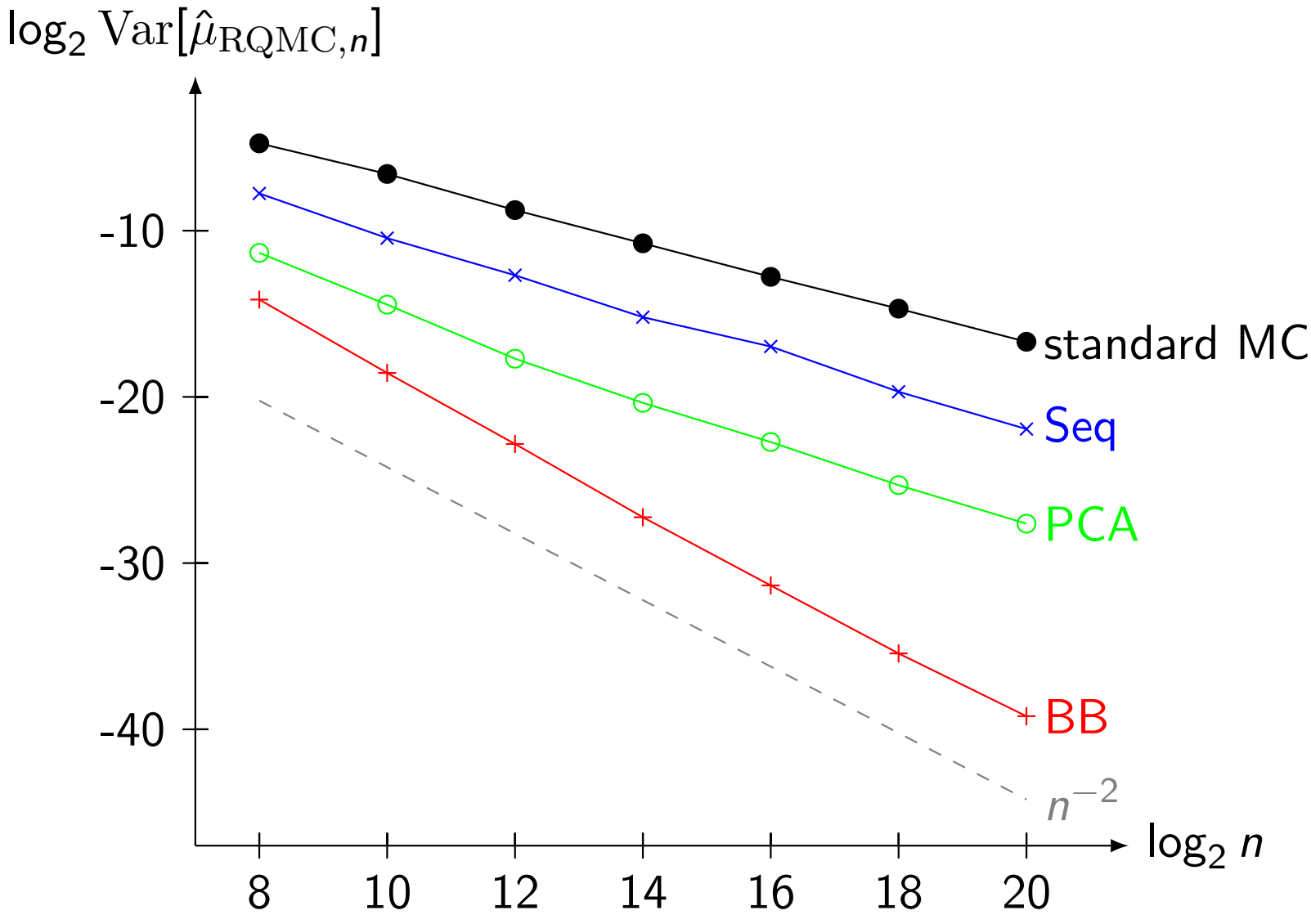
European version: Can exercise only at $t_{16} = 1$.

One-dimensional state $X_j = S(t_j)$. Sorting for array-RQMC is simple.

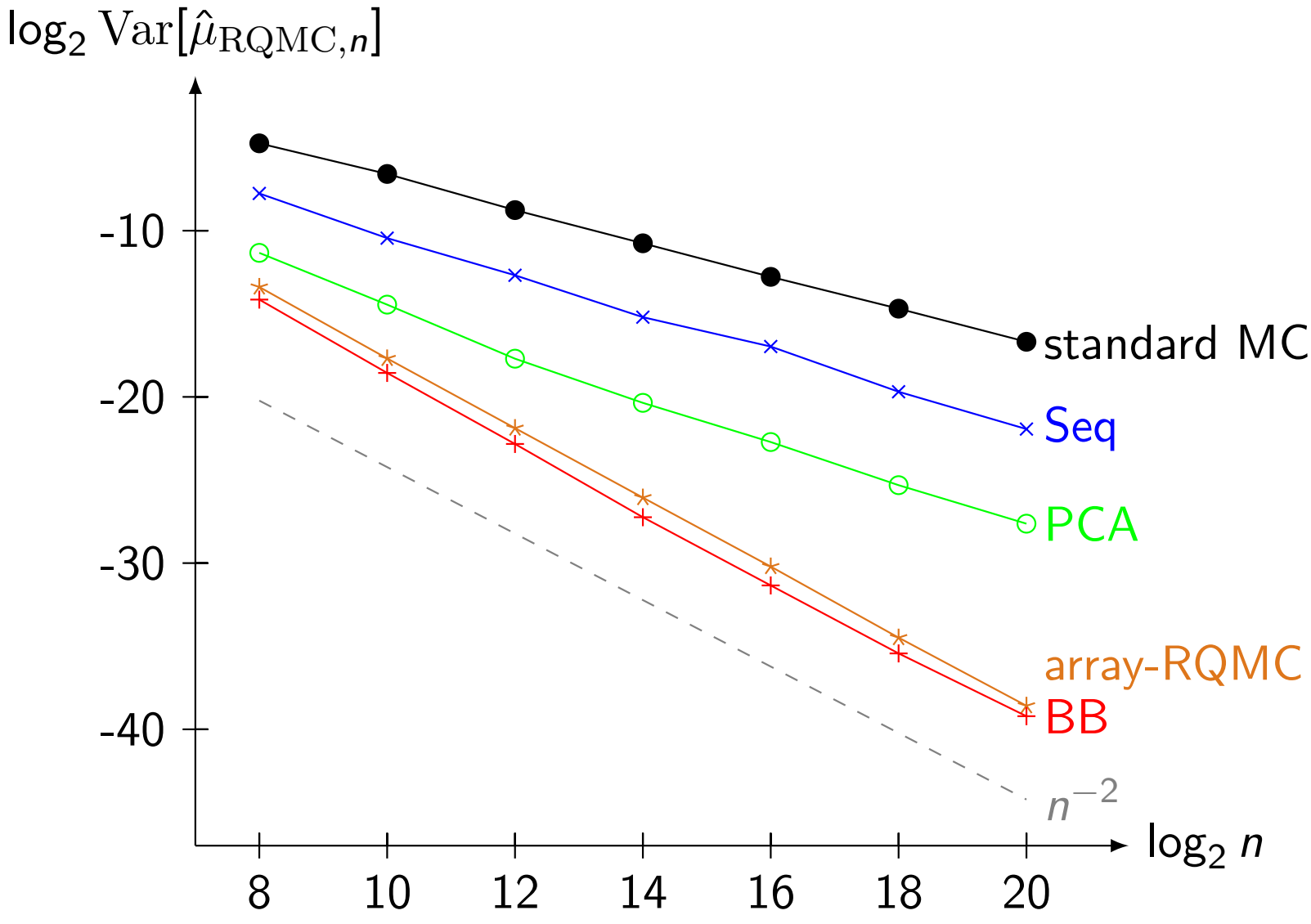
Basis functions for regression-based MC:
polynomials $\psi_k(x) = (x - 101)^{k-1}$ for $k = 1, \ldots, 5$.

For RQMC and array-RQMC, we use Sobol' nets with a linear scrambling and a random digital shift, for all the results reported here.
Results are very similar for randomly-shifted lattice rule + baker's transformation.

European version of put option.

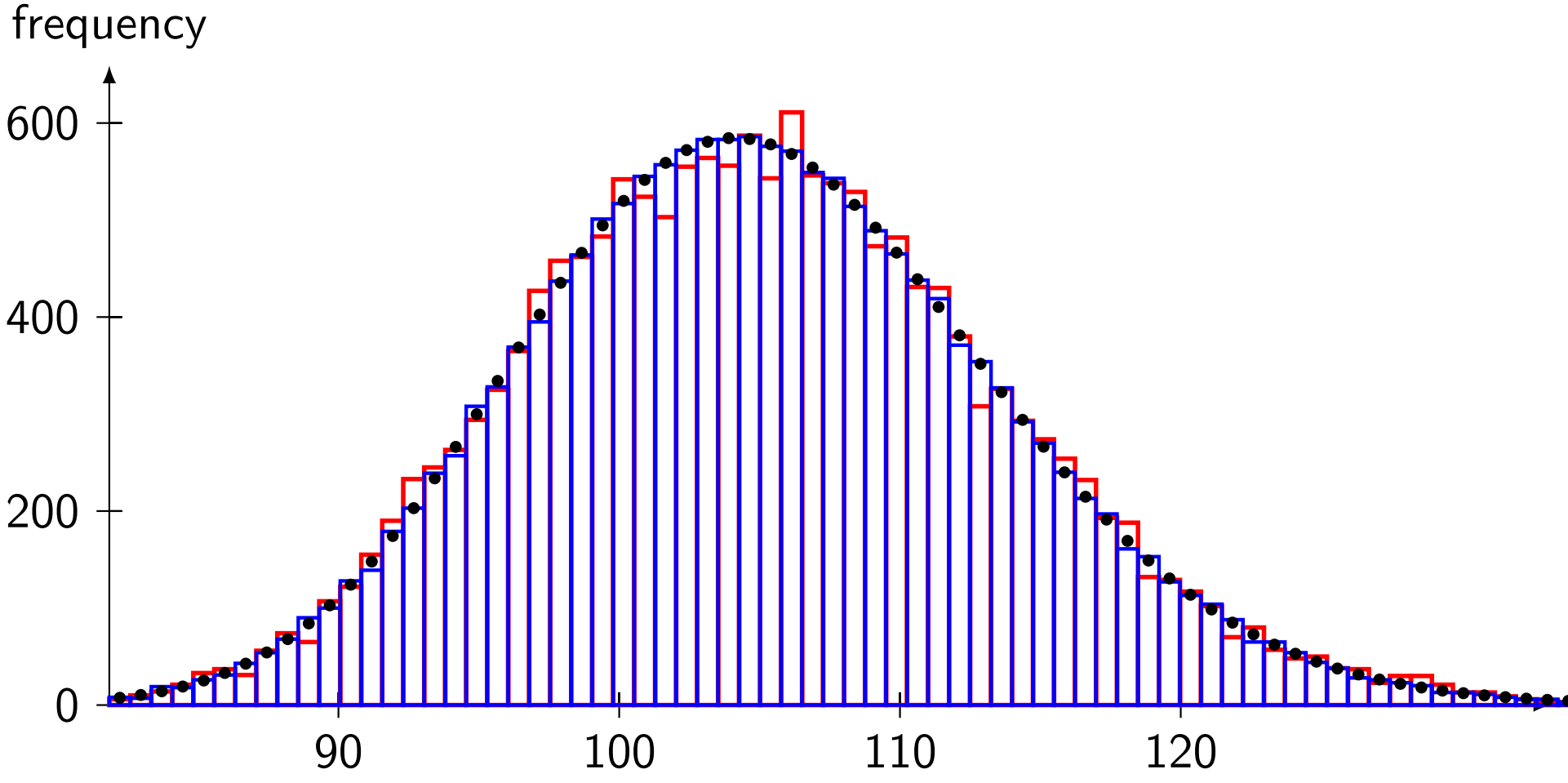


European version of put option.



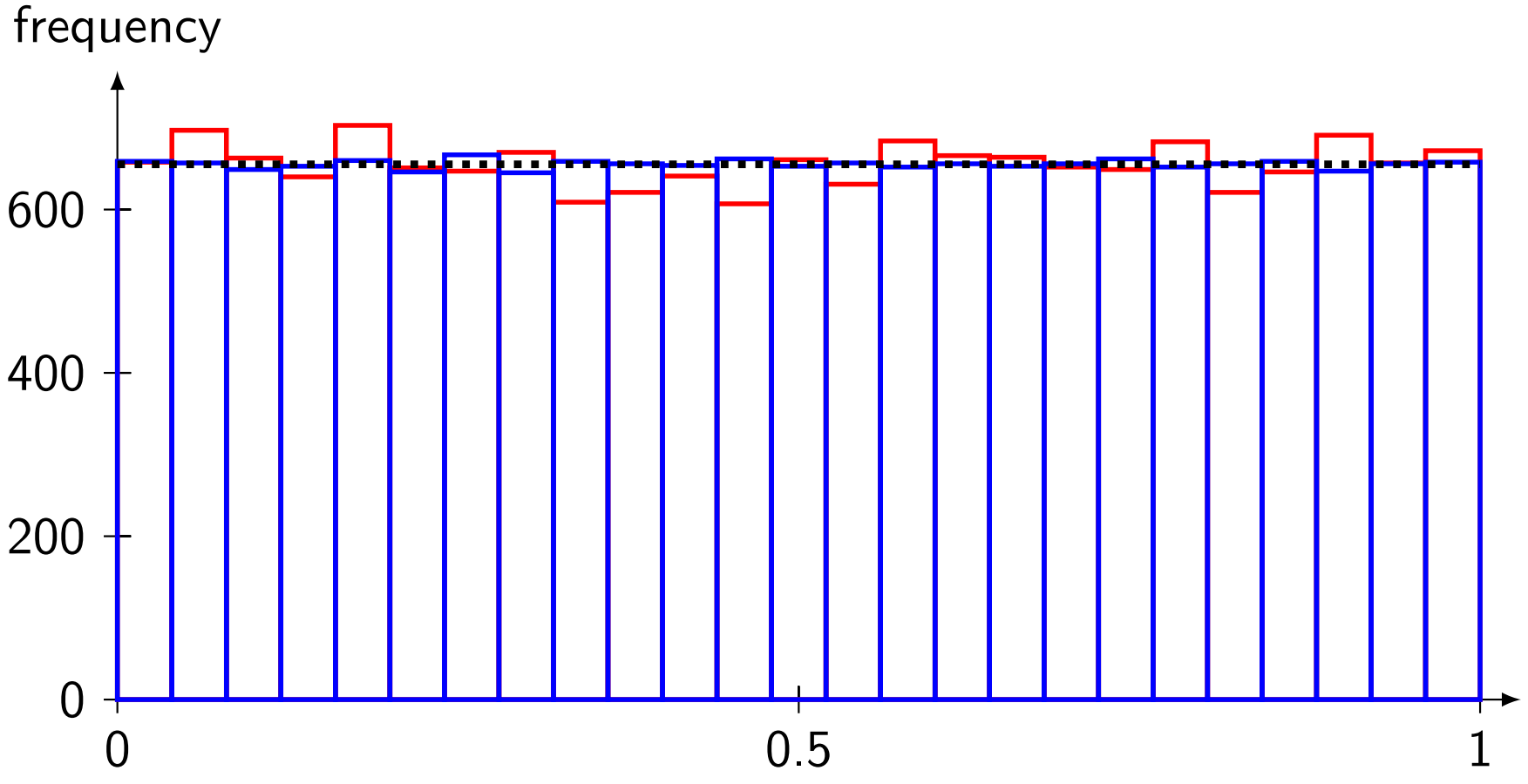
Histogram of states at step 16

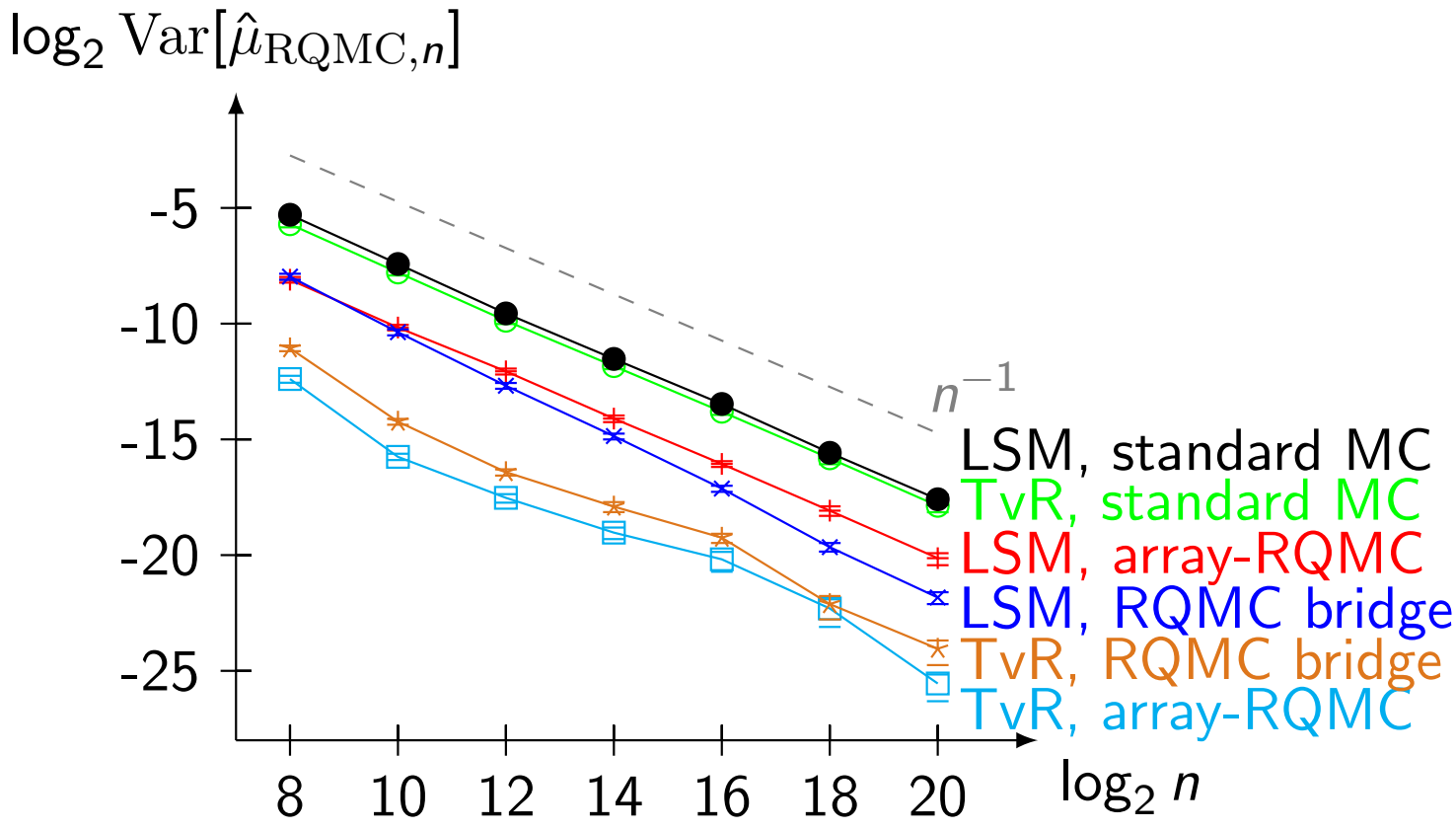
States for array-RQMC with $n = 2^{14}$ in blue and for MC in red.
Theoretical dist.: black dots.



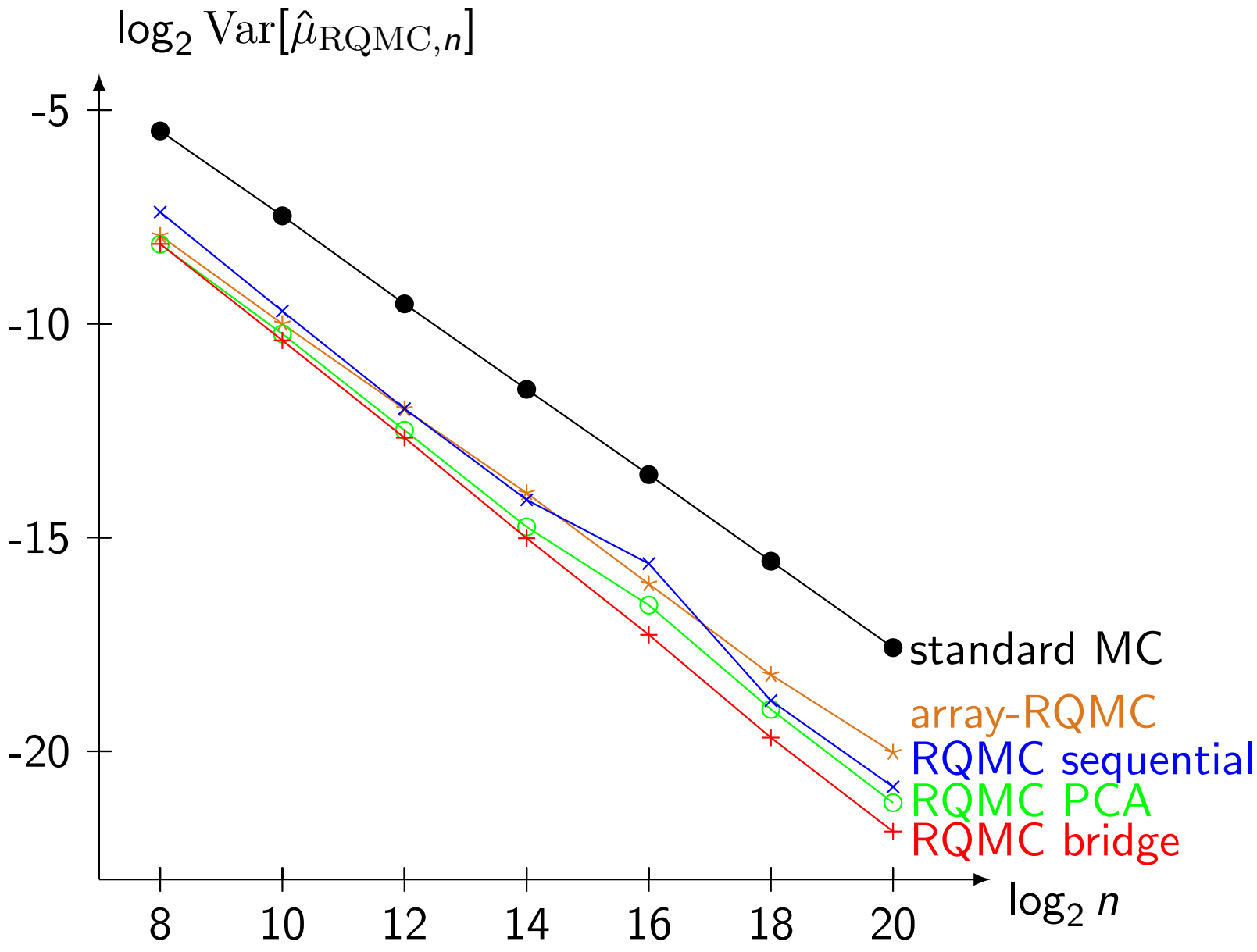
Histogram after transformation to uniforms (applying the cdf).

States for array-RQMC with $n = 2^{14}$ in blue and for MC in red.
Theoretical dist. is uniform (black dots).

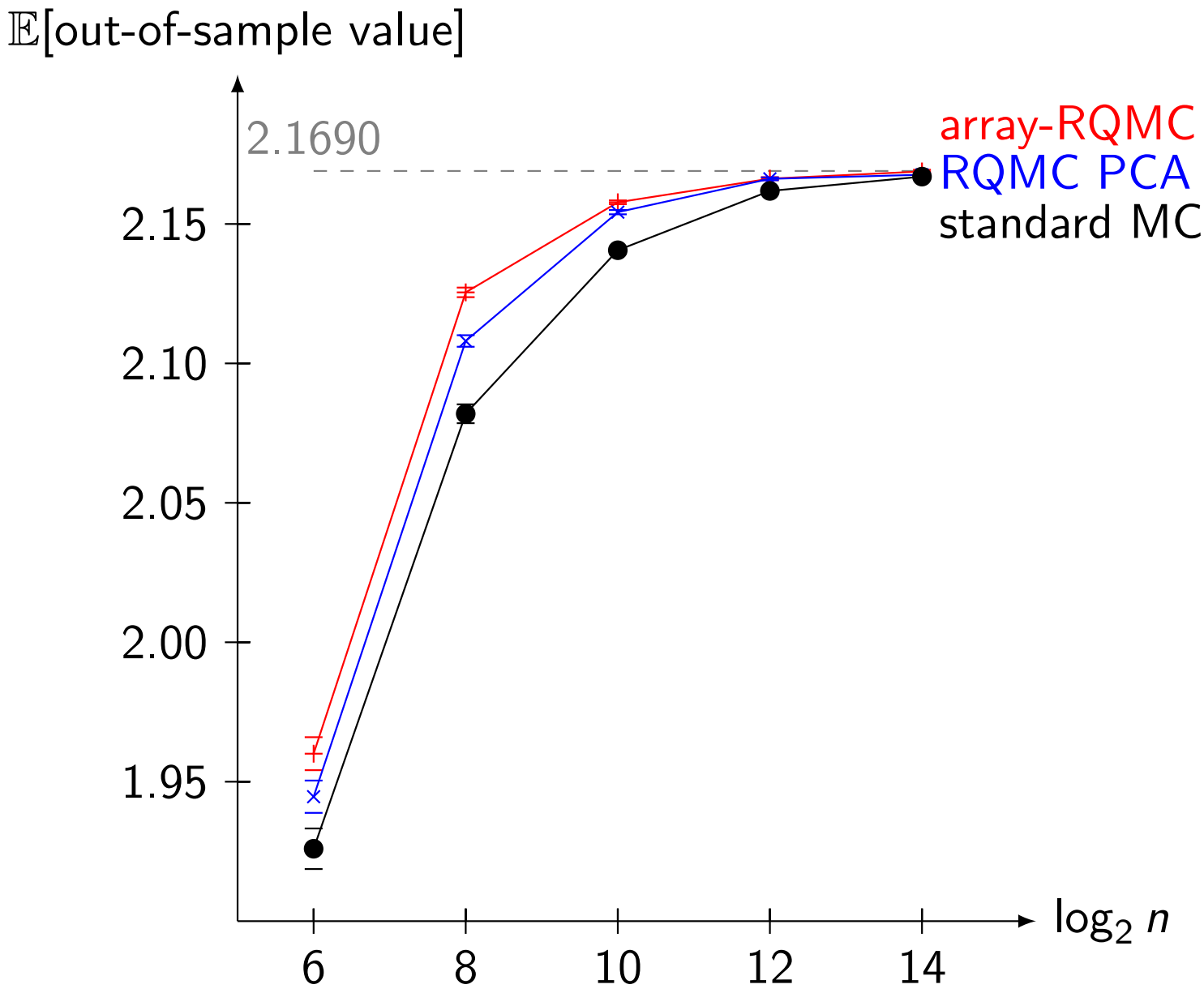



$\log_2 \mathrm{Var}[\hat{\mu}_{\mathrm{RQMC},n}]$
-5
-10
-15
-20
-25
8
10
12
14
16
18
20
$\log_2 n$
n^{-1}
LSM, standard MC
TvR, standard MC
LSM, array-RQMC
LSM, RQMC bridge
TvR, RQMC bridge
TvR, array-RQMC

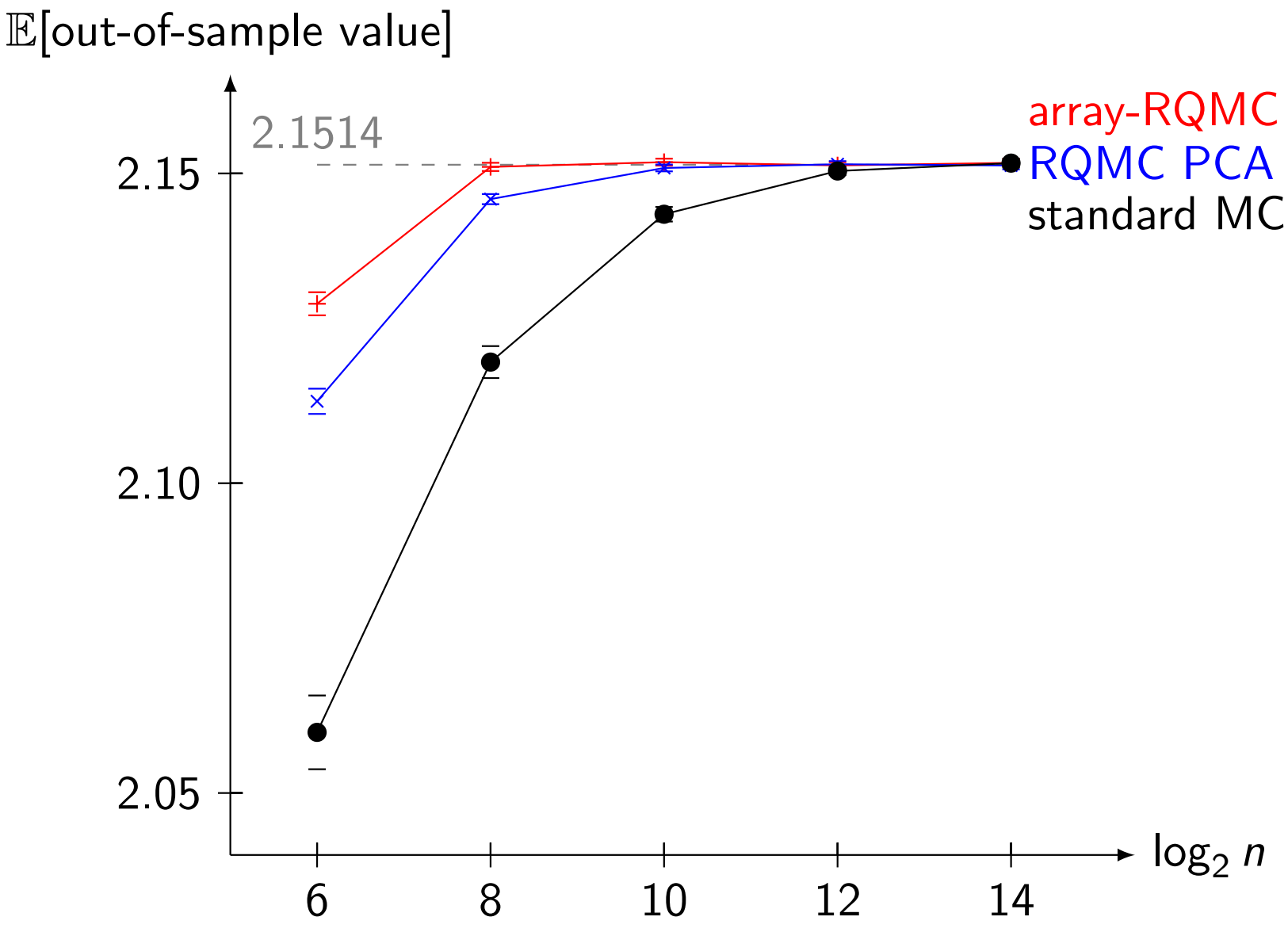
American put option: estimation for a fixed policy.



American put option: out-of-sample value for policy obtained from LSM.



American put option: out-of-sample value for policy obtained from TvR.



Example: Asian Option

Given observation times $t_1, t_2, \ldots, t_s$, suppose

$$S(t_j) = S(t_{j-1}) \exp[(r - \sigma^2/2)(t_j - t_{j-1}) + \sigma(t_j - t_{j-1})^{1/2}\Phi^{-1}(U_j)],$$

where $U_j \sim U[0,1)$ and $S(t_0) = s_0$ is fixed.

State is $X_j = (S(t_j), \bar{S}_j)$, where $\bar{S}_j = \frac{1}{j}\sum_{i=1}^{j} S(t_i)$.

Transition:

$$(S(t_j), \bar{S}_j) = \varphi(S(t_{j-1}), \bar{S}_{j-1}, U_j) = \left(S(t_j), \frac{(j-1)\bar{S}_{j-1} + S(t_j)}{j}\right).$$

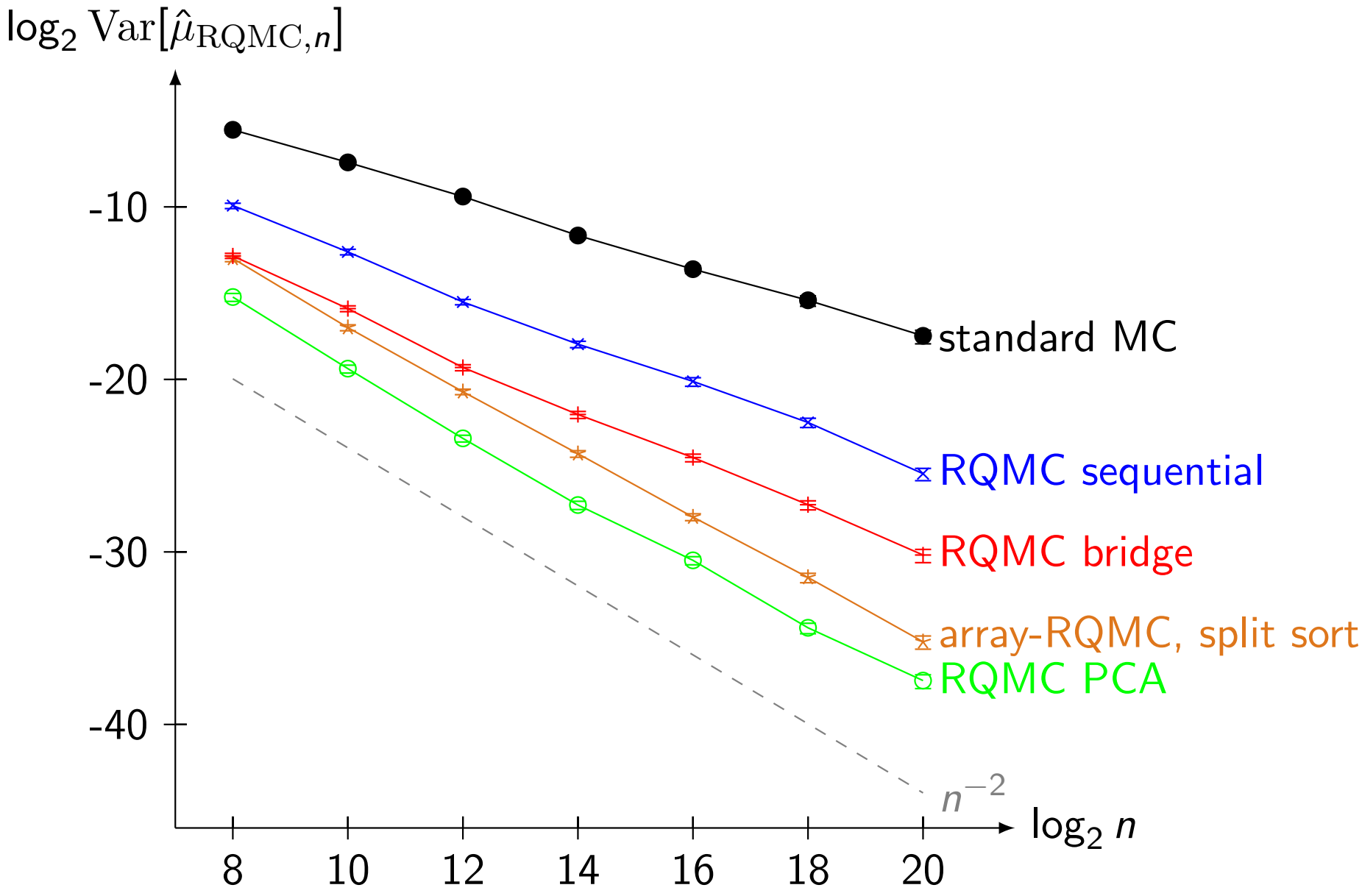
Payoff at step j is $\max\left[0, \bar{S}_j - K\right]$.

We use the two-dimensional sort at each step; we first sort in n_1 packets based on $S(t_j)$, then sort the packets based on $\bar{S}_j$.

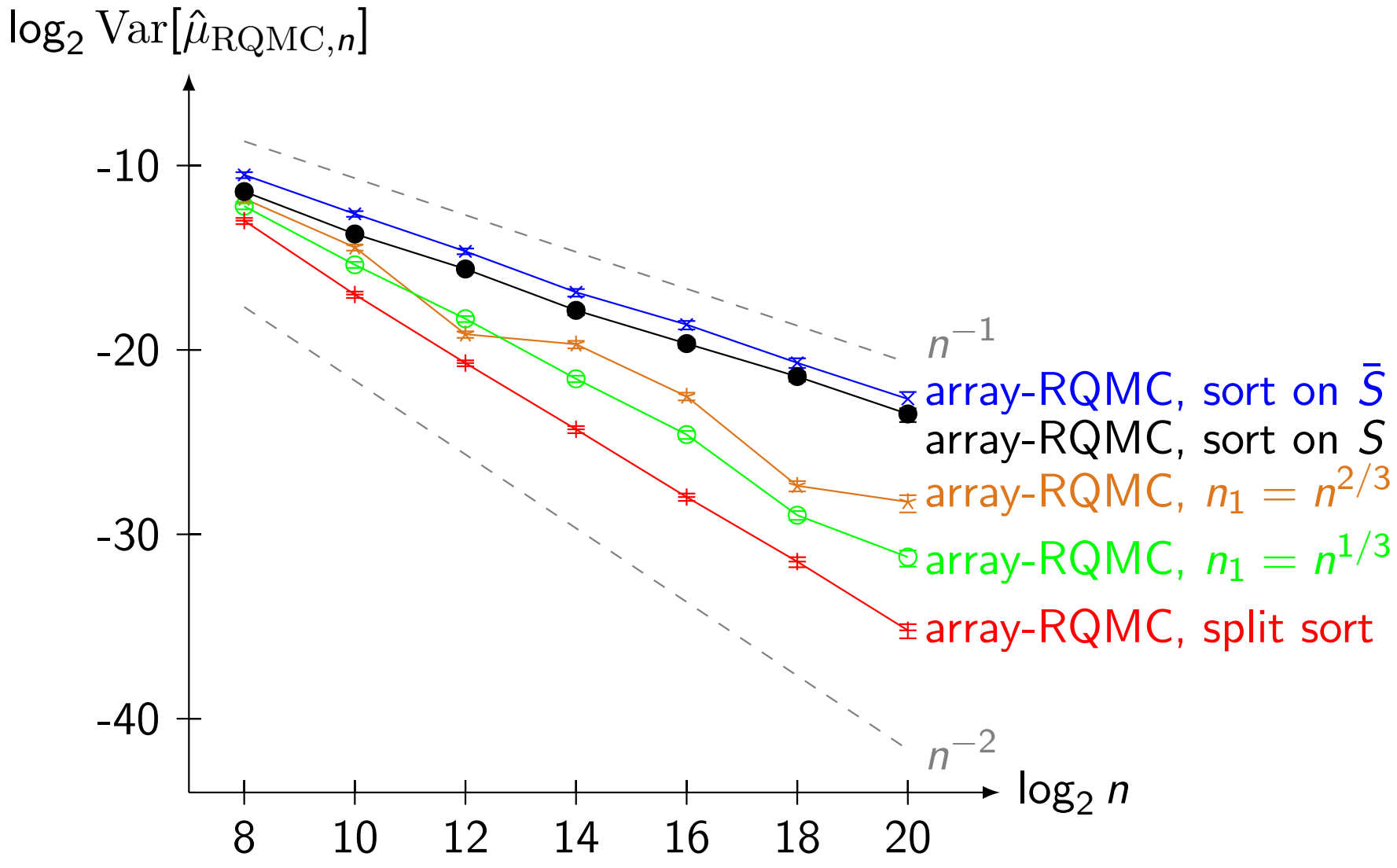
GBM with parameters: $S(0) = 100$, $K = 100$, $r = 0.05$, $\sigma = 0.15$, $t_j = j/52$ for $j = 0, \ldots, s = 13$.

Basis functions to approximate the continuation value: polynomials of the form $g(S, \bar{S}) = (S - 100)^k (\bar{S} - 100)^m$, for $k, m = 0, \ldots, 4$ and $km \leq 4$. Also broken polynomials $\max(0, S - 100)^k$ for $k = 1, 2$, and $\max(0, S - 100)(\bar{S} - 100)$.

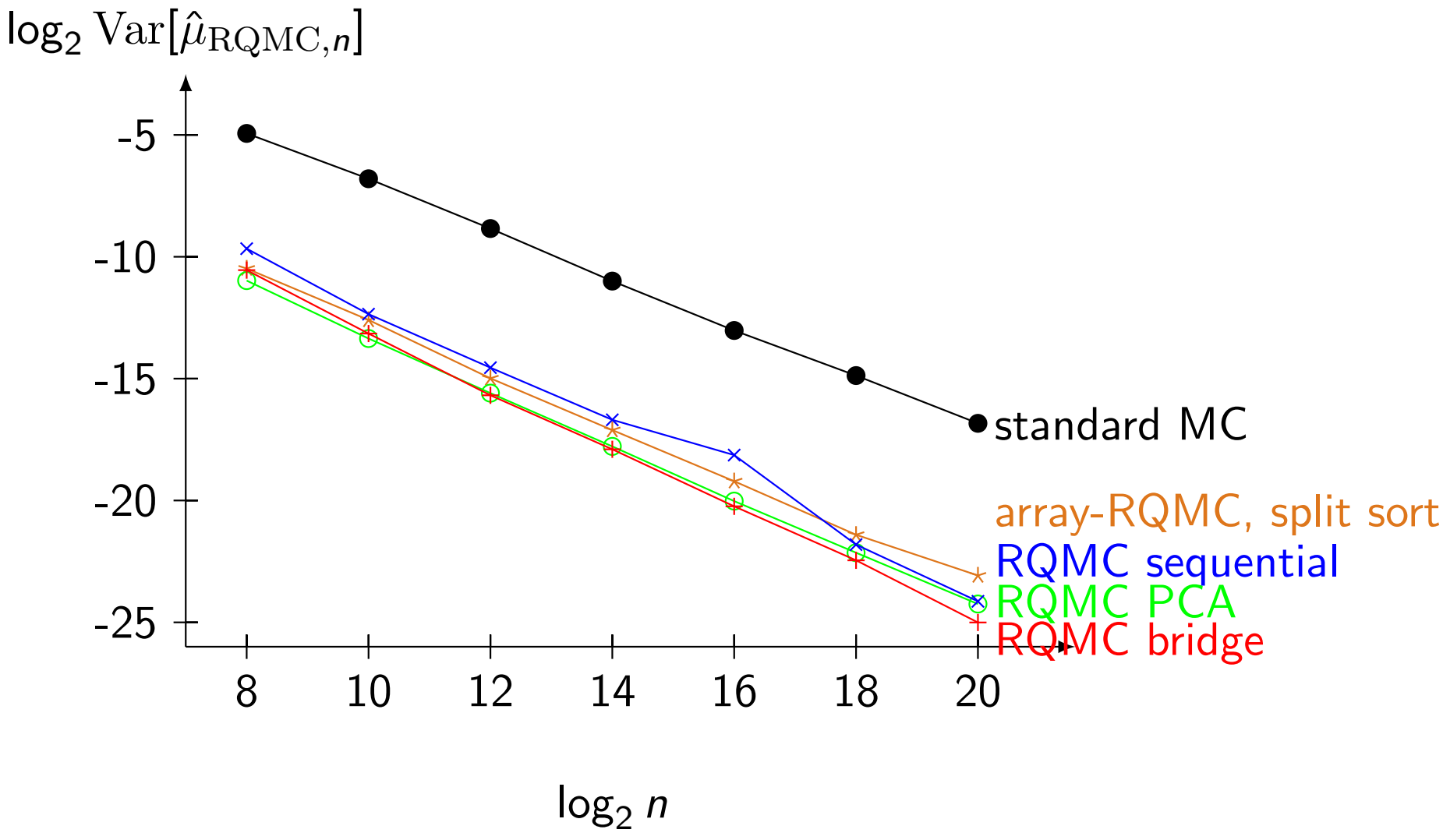
European version of Asian call option



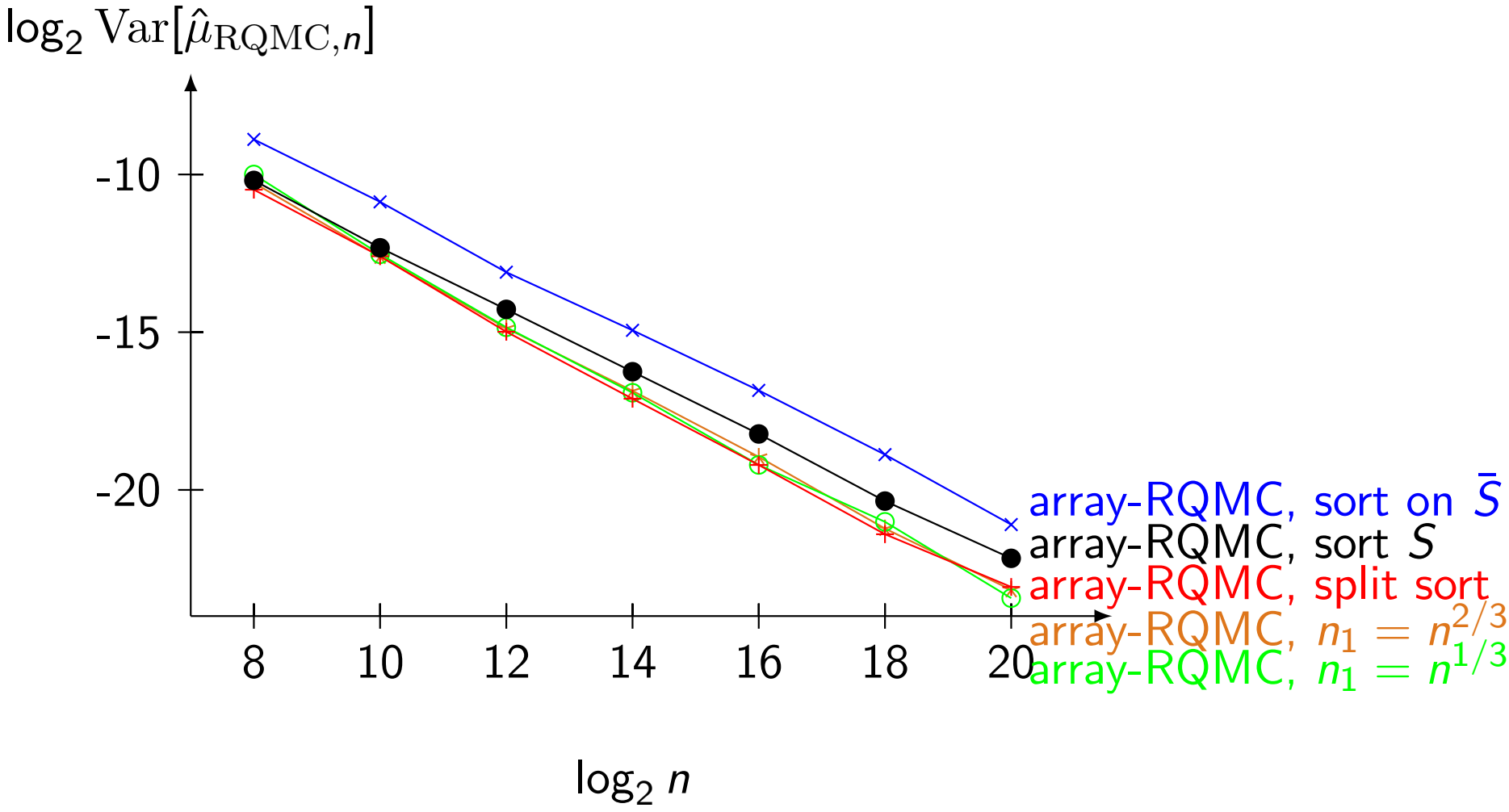
European version, sorting strategies for array-RQMC.



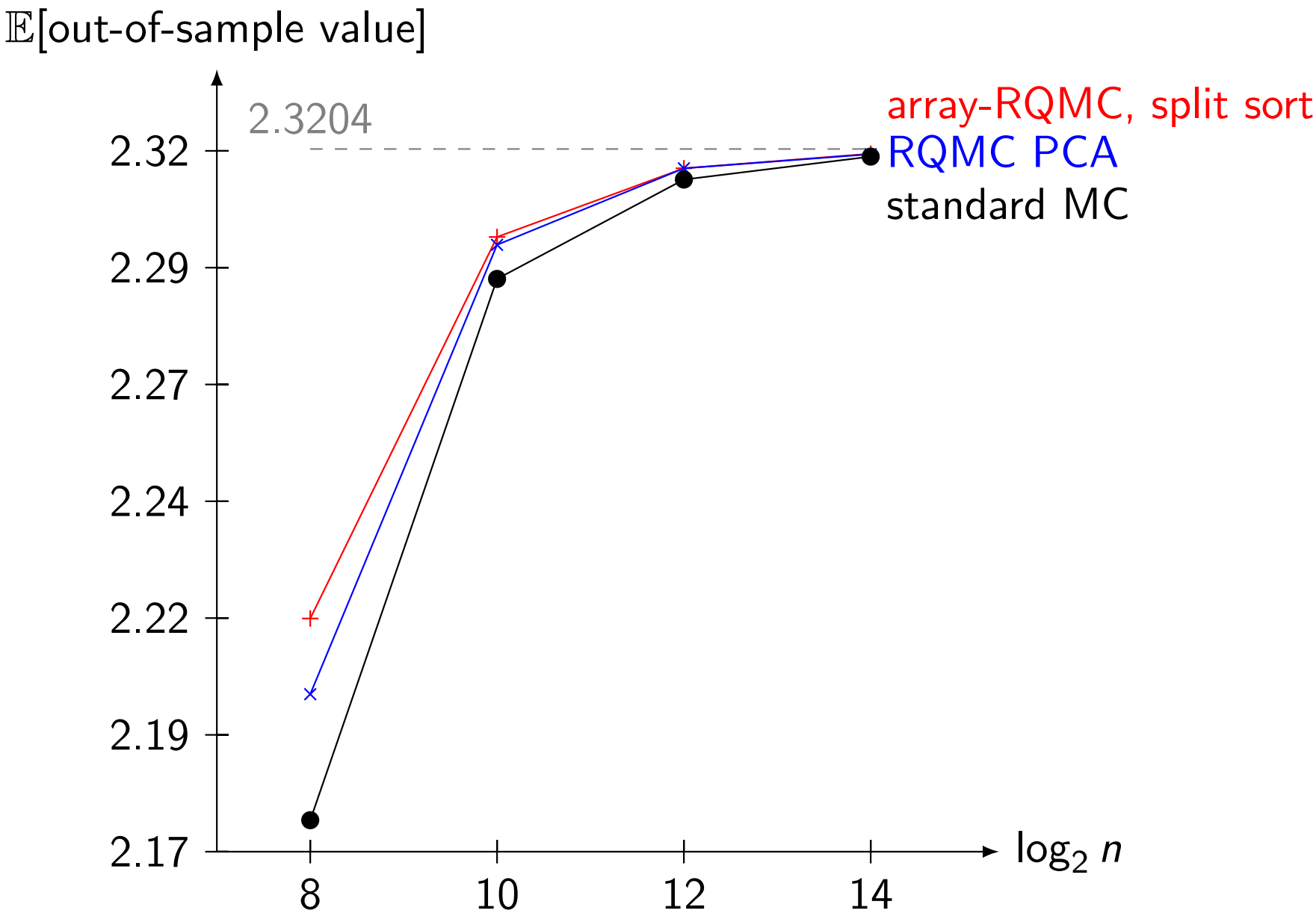
American-style Asian option with a fixed policy.



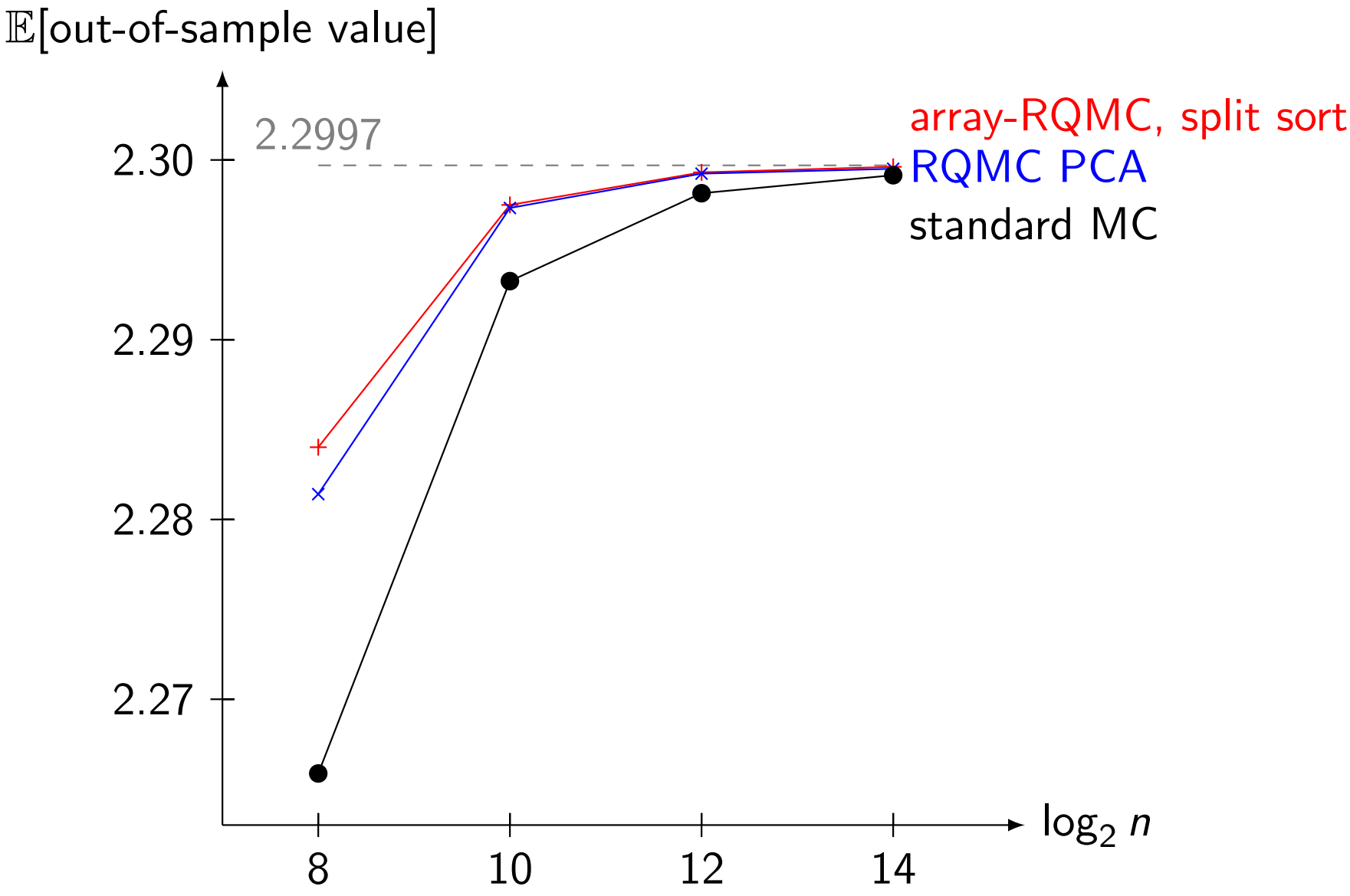
Fixed policy, choices of array-RQMC sorting.



Out-of-sample value of policy obtained from LSM.



Out-of-sample value of policy obtained from TvR.



Call on the maximum of 5 assets

Five indep. asset prices obeys a GBM with $s_0 = 100$, $r = 0.05$, $\sigma = 0.2$. The assets pay a dividend at rate 0.10, which means that the effective risk-free rate can be taken as $r' = 0.05 - 0.10 = -0.05$.

Exercise dates are $t_j = j/3$ for $j = 1, \ldots, 9$.

State at t_j is $X_j = (S_{j,1}, \ldots, S_{j,5})$.

Call on the maximum of 5 assets

Five indep. asset prices obeys a GBM with $s_0 = 100$, $r = 0.05$, $\sigma = 0.2$. The assets pay a dividend at rate 0.10, which means that the effective risk-free rate can be taken as $r' = 0.05 - 0.10 = -0.05$.

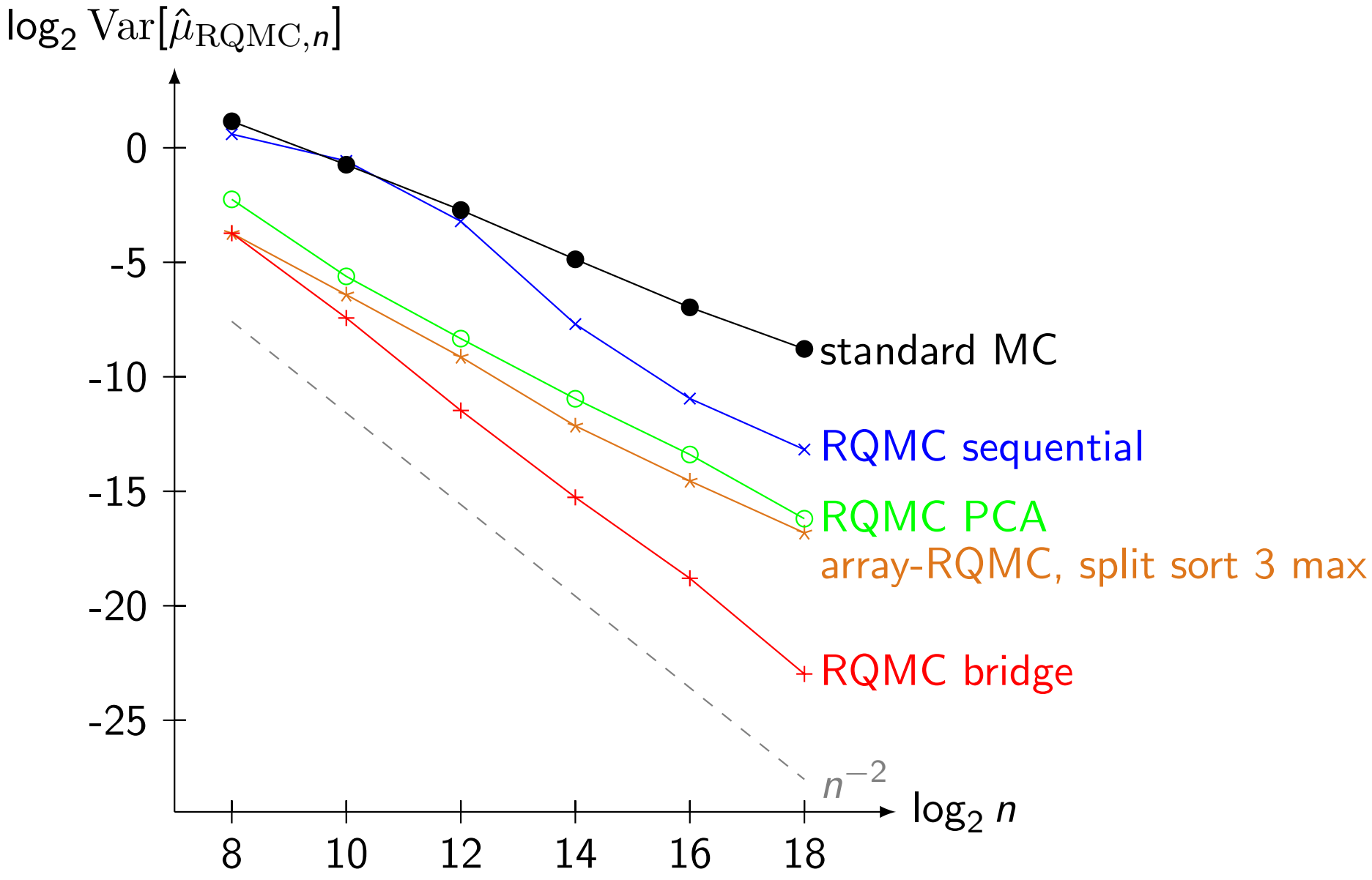
Exercise dates are $t_j = j/3$ for $j = 1, \ldots, 9$.

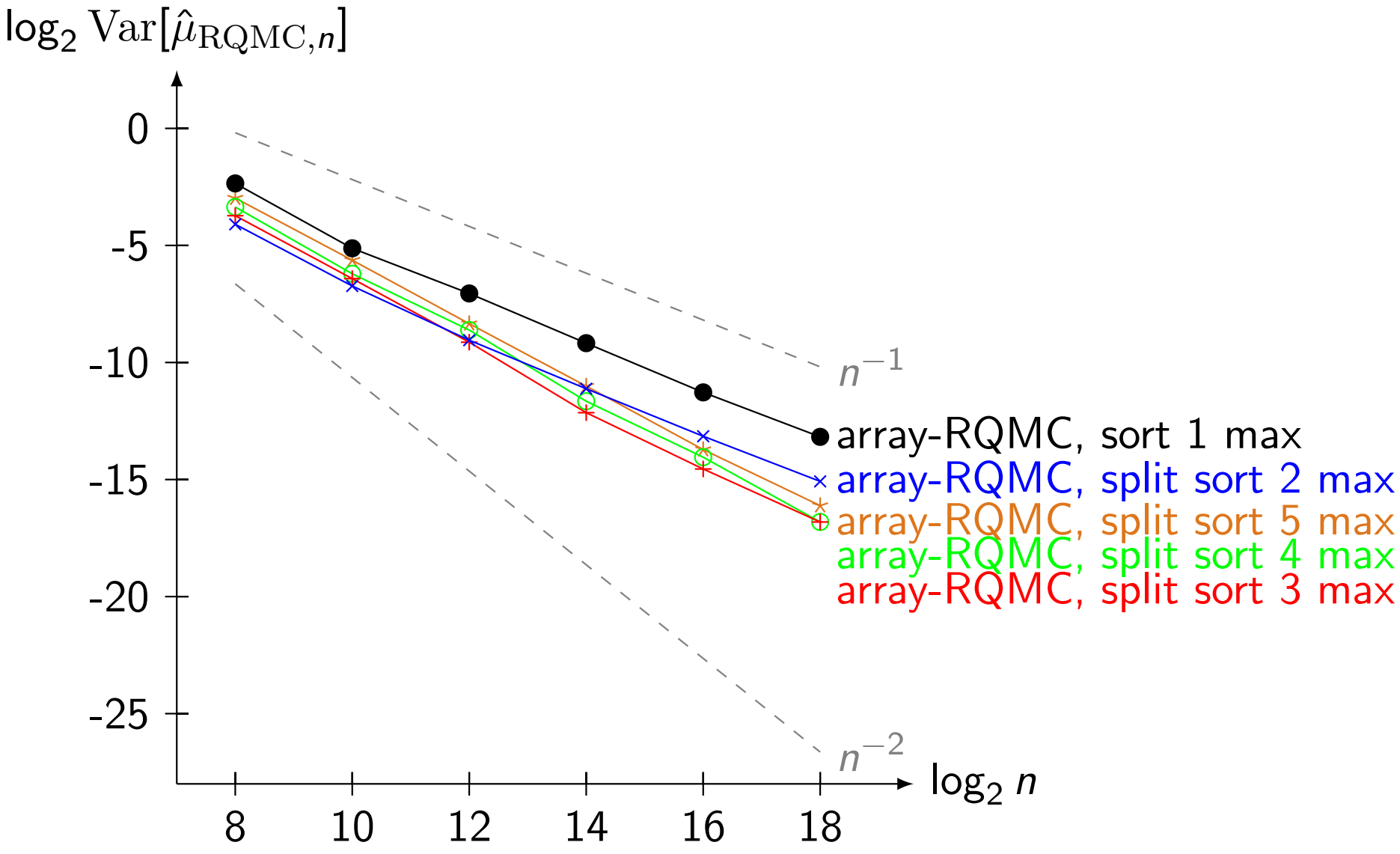
State at t_j is $X_j = (S_{j,1}, \ldots, S_{j,5})$.

Basis functions for regression: 19 polynomials in the $S_{j,(\ell)} - 100$, where $S_{j,(1)}, \ldots, S_{j,(5)}$ are the asset prices sorted in increasing order.

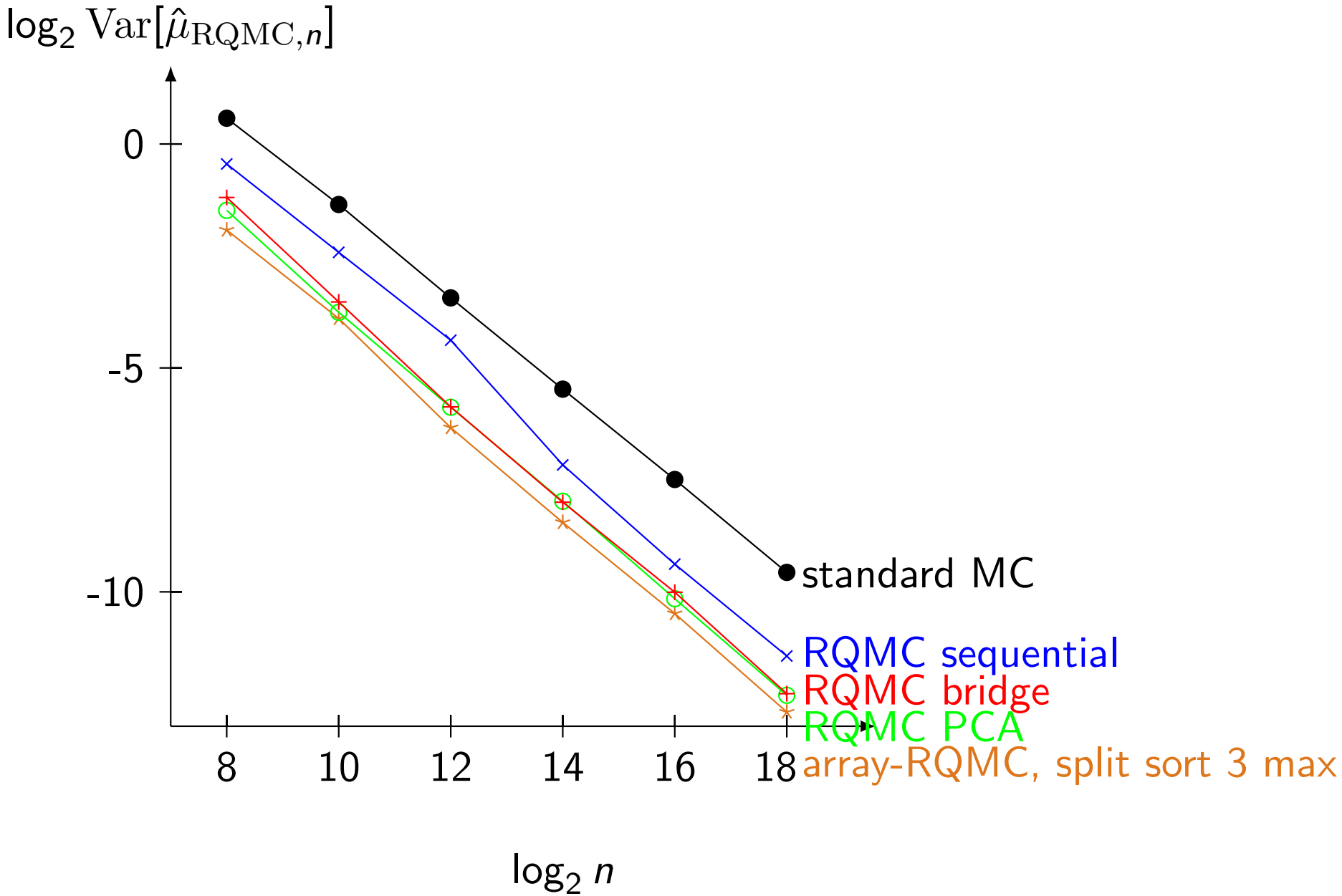
For array-RQMC, we sort on the m largest asset prices.
At each step we generate the next value first for the maximum, then for the second largest, and so on.

European version

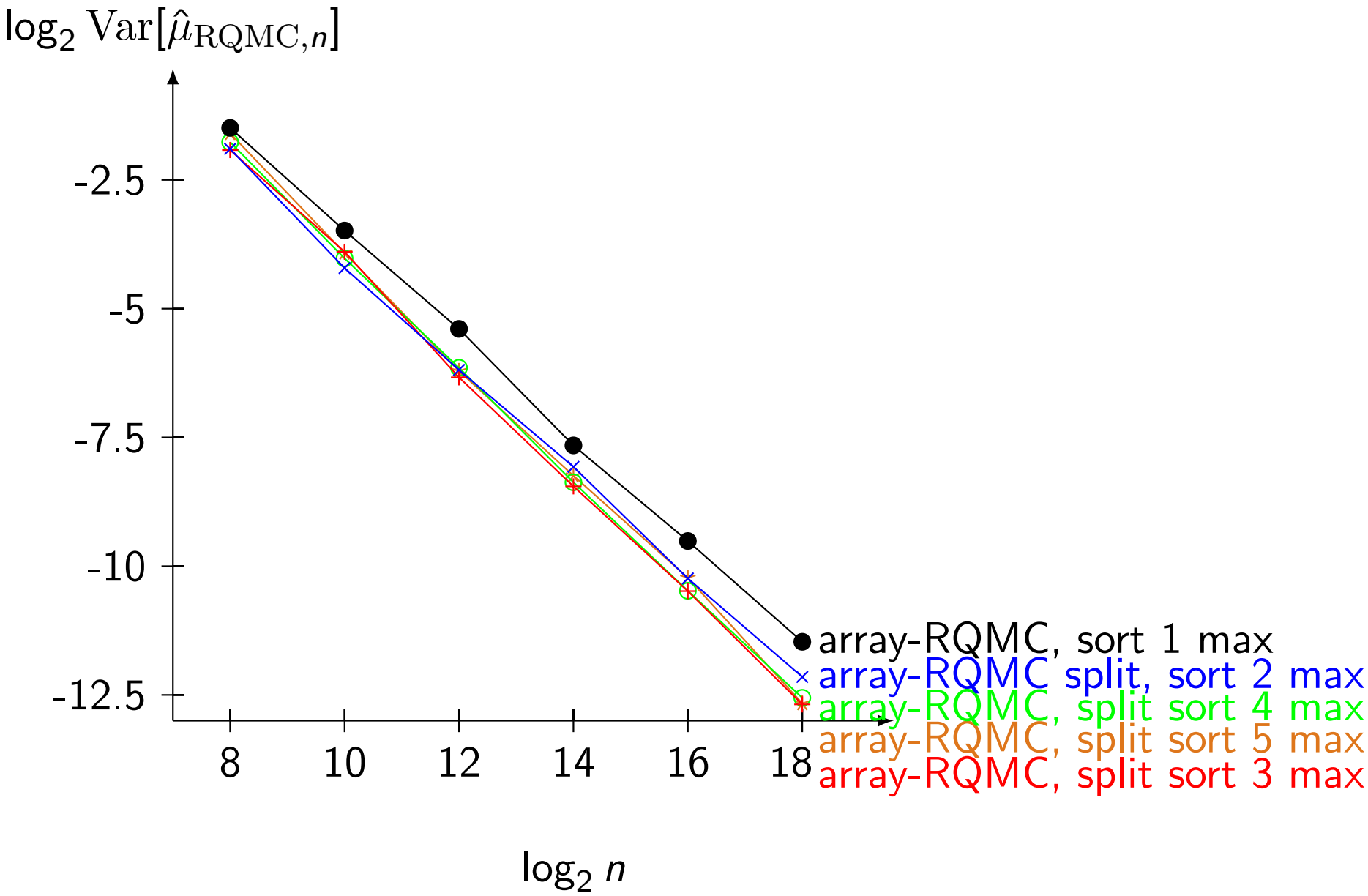



$\log_2 \mathrm{Var}[\hat{\mu}_{\mathrm{RQMC},n}]$
0
-5
-10
-15
-20
-25
8
10
12
14
16
18
$\log_2 n$
n^{-1}
n^{-2}
array-RQMC, sort 1 max
array-RQMC, split sort 2 max
array-RQMC, split sort 5 max
array-RQMC, split sort 4 max
array-RQMC, split sort 3 max

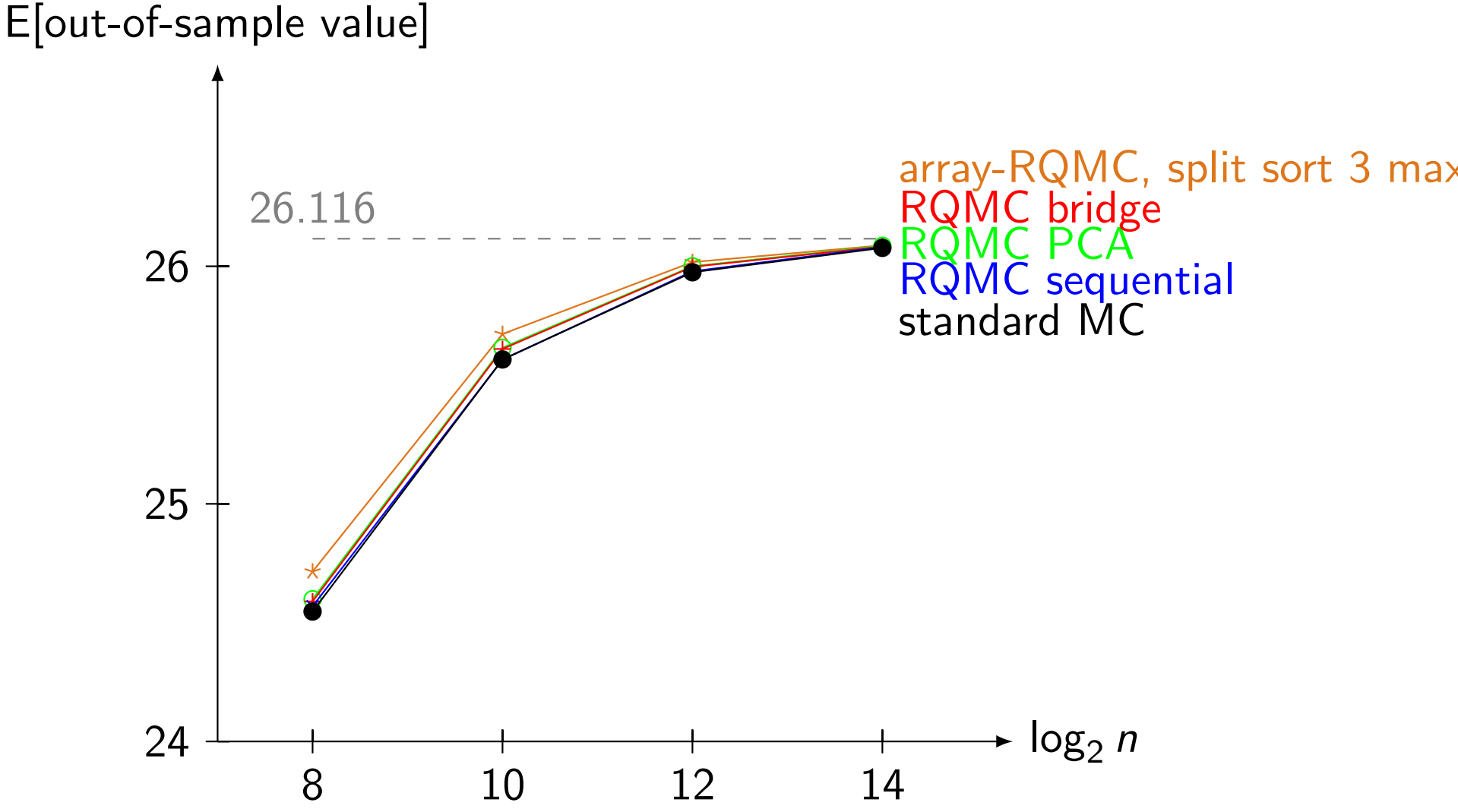
American version, fixed policy



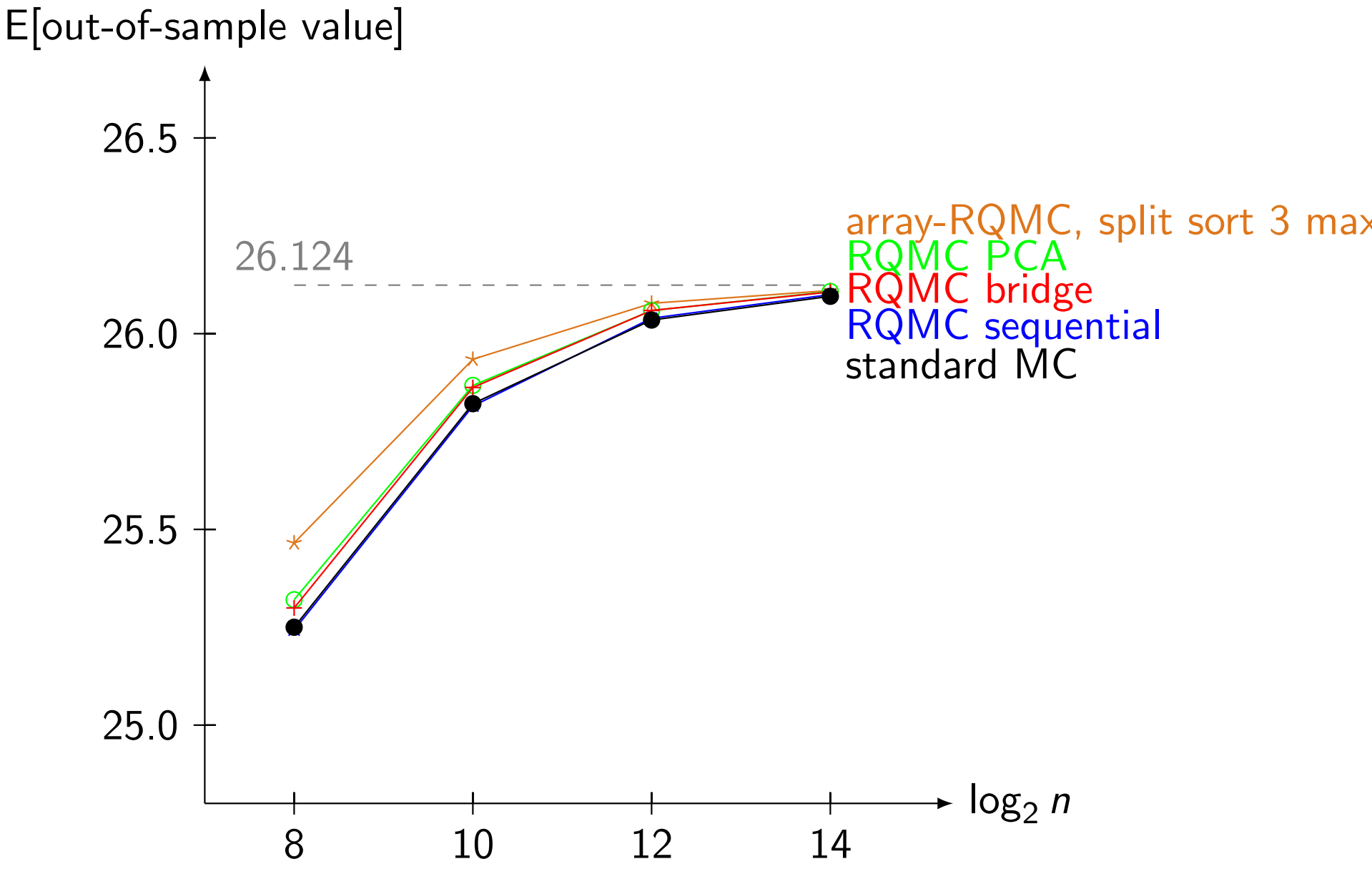
Fixed policy.



Out-of-sample value of policy obtained from LSM.



Out-of-sample value of policy obtained from TvR.



Conclusion

Empirical results are excellent for fixed number of steps.

More modest but still interesting for random stopping time.

Proving the observed convergence rates seems difficult; we need help!