PARALLEL QUASI-MONTE CARLO ALGORITHMS FOR SOME LINEAR ALGEBRA PROBLEMS

Aneta Karaivanova
E-mail: anet@parallel.bas.bg
Home Page: http://parallel.bas.bg/~anet

Department of Parallel Algorithms
IPP - Bulgarian Academy of Sciences
Sofia, Bulgaria

Typeset by $\mathcal{AMS}$-TEX

Outline of The talk

1. Formulation of the Problem
- Solving Systems via Neumann Series
- Computing Extremal Eigenvalues: Power Method
- Computing Extremal Eigenvalues: Resolvent Method

2. Monte Carlo Methods

3. Quasirandom Numbers
- Discrepancy facts

4. MCMs for Matrix-Vector Products
- Error Analysis

5. QMCMs for Matrix-Vector Products
- Analysis of Quasi-MCMs

6. Mathematical Expectations

7. MC and QMC estimations

5. Some Numerical Results

Formulation of the Problem

Given $A \in R^{n \times n}$, $f \in R^n$ consider MCMs and QMCMs to solve the problems:

- Find the solution or the scalar product of a given vector and the solution and of a system of linear algebraic equations:

$$x = Ax + f$$

- Find some of the elements of the inverse matrix $C = A^{-1}$
- Find (λ, x), $x \in R^n$, such that

$$Ax = \lambda x$$

Solving SLAE via Neumann Series

- Assume a SLAE in the form $x = Ax + f$, all the eigenvalues of A lie within the unit circle.
- Given x_0, consider the sequence:

$$x^{(k)} = Ax^{(k-1)} + f, \;\; k = 1, 2, \ldots .$$

- The approximate solution is the truncated Neumann series

$$x^{(k)} = f + Af + A^2 f + \ldots + A^{(k-1)} f + A^k x^{(0)}, \;\; k > 0$$

with a truncation error of $x^{(k)} - x = A^k (x^{(0)} - x)$.

- We can estimate (h, x), where h is a given vector as:

$$(h, x) \approx h^T f + h^T Af + h^T A^2 f + \ldots + h^T A^{(k-1)} f + h^T A^k x^{(0)}$$

- We can use MCM and QMCM for computing matrix-vector products

Computing Extremal Eigenvalues: Power Method

- The eigenvalue problem:

$$A \in R^{n \times n}, \quad u \in R^n$$

$$|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_{n-1}| > |\lambda_n|.$$

$$Au = \lambda u$$

- The power method:

$$x^m = Ax^{m-1}/||Ax^{m-1}||$$

$$\lambda^{(m)} = \frac{(h, A^m f)}{(h, A^{m-1} f)} \underset{m \to \infty}{\longrightarrow} \lambda_{max}$$

- We can apply MCM and QMCM for computing matrix-vector products

Computing Extremal Eigenvalues: The Resolvent Method

- The resolvent matrix: $R_q = [I - qA]^{-1} \in \mathbb{R}^{n \times n}$

$$[I - qA]^{-m} = \sum_{i=1}^{\infty} q^i C_{m+i-1}^i A^i, \;\; |q\lambda| < 1$$

- Connection between eigenvalues:

$$\mu = \frac{1}{1 - q\lambda}$$

  $q > 0$, μ_{max} corresponds to λ_{max},

  $q < 0$, μ_{max} corresponds to λ_{min}

- Power method for the resolvent matrix:

$$\mu^{(m)} = \frac{(h, [I - qA]^{-m} f)}{(h, [I - qA]^{-(m-1)} f)} \underset{m \to \infty}{\longrightarrow} \mu = \frac{1}{1 - q\lambda}.$$

- The Resolvent estimation:

$$\lambda = \frac{1}{q}\left(1 - \frac{1}{\mu}\right) = \frac{\sum_{i=1}^{\infty} q^{i-1} C_{i+m-2}^{i-1} (h, A^i f)}{\sum_{i=0}^{\infty} q^i C_{i+m-1}^i (h, A^i f)}.$$

Monte Carlo Methods

- J is a quantity to be estimated via a MCM
- θ is a random variable (RV) with $E[\theta] = J$
- θ_N is the estimator with N samples
- $err_N \approx \sigma(\theta)N^{-1/2}$ is the overall error

◇ How do we reduce this error?

- Variance reduction methods: $\sigma(\theta)$
  - ▶ Antithetic variates
  - ▶ Control variates
  - ▶ Stratification
  - ▶ Importance sampling
- Using more powerful random numbers: $N^{-1/2}$
  - ▶ Quasi-MCMs

Quasirandom Numbers

- QRNs are constructed to minimize a measure of their deviation from uniformity called *discrepancy* (built with numerical integration in mind)
- **Definition**: The *star discrepancy* D_N^* of $x_1, \ldots, x_N \in [0,1)$ (measure of uniformity):

$$D_N^* = D_N^*(x_1, \ldots, x_N) =$$

$$\sup_{0 \le u \le 1} \left| \frac{1}{N} \sum_{n=1}^{N} \chi_{[0,u)}(x_n) - u \right|,$$

where χ is the characteristic function

- **Theorem** (Koksma, 1942): if $f(x)$ has bounded variation $V(f)$ on $[0,1]$ and $x_1, \ldots, x_N \in [0,1]$ with star discrepancy D_N^*, then:

$$\left| \frac{1}{N} \sum_{n=1}^{N} f(x_n) - \int_0^1 f(x)\, dx \right| \le V(f) D_N^*$$

Discrepancy Facts

◇ Real random numbers have (the law of the iterated logarithm):

$$D_N^* = O(N^{-1/2}(\log\log N)^{-1/2})$$

◇ Klaus F. Roth (Fields medalist in 1958) proved the following lower bound in 1954 for the star discrepancy of N points in s dimensions:

$$D_N^* \geq O(N^{-1}(\log N)^{\frac{s-1}{2}})$$

◇ Sequences (indefinite length) and point sets have different "best discrepancies" at present:

- Sequence: $D_N^* \leq O(N^{-1}(\log N)^{s-1})$
- Point set: $D_N^* \leq O(N^{-1}(\log N)^{s-2})$

Some Types of QRNs

- ◇ Must choose point sets (finite #) or sequences (infinite #) with small D_N^*
- ◇ Often used is the *van der Corput sequence* in base b: $x_n = \Phi_b(n-1), n = 1, 2, \ldots,$ where for $b \in \mathbb{Z}, b \geq 2$:

$$\Phi_b\left(\sum_{j=0}^{\infty} a_j b^j\right) = \sum_{j=0}^{\infty} a_j b^{-j-1} \quad \text{with}$$

$$a_j \in \{0, 1, \ldots, b-1\}$$

For v.d. Corput sequ. $ND_N^* \leq \frac{\log N}{3 \log 2} + O(1)$

- ◇ With $b = 2$, we get $\{\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8} \ldots\}$
- ◇ With $b = 3$, we get $\{\frac{1}{3}, \frac{2}{3}, \frac{1}{9}, \frac{4}{9}, \frac{7}{9}, \frac{2}{9}, \frac{5}{9}, \frac{8}{9} \ldots\}$

Some Types of QRNs (Cont.)

- Other small D_N^* points sets and sequences:

1. Halton sequence: $\mathbf{x}_n = (\Phi_{b_1}(n-1), \ldots, \Phi_{b_s}(n-1))$, $n = 1, 2, \ldots$, $D_N^* = O\left(N^{-1}(\log N)^s\right)$ if $b_1, \ldots, b_s$ pairwise relatively prime

2. Hammersley point set: $\mathbf{x}_n = \left(\frac{n-1}{N}, \Phi_{b_1}(n-1), \ldots, \Phi_{b_{s-1}}(n-1)\right)$, $n = 1, 2, \ldots, N$, $D_N^* = O\left(N^{-1}(\log N)^{s-1}\right)$ if $b_1, \ldots, b_{s-1}$ are pairwise relatively prime

3. Ergodic dynamics: $\mathbf{x}_n = \{n\alpha\}$, where $\alpha = (\alpha_1, \ldots, \alpha_s)$ is irrational and $\alpha_1, \ldots, \alpha_s$ are linearly independent over the rationals then for almost all $\alpha \in \mathbb{R}^s$, $D_N^* = O(N^{-1}(\log N)^{s+1+\epsilon})$ for all $\epsilon > 0$

Some Types of QRNs (Cont.)

1. The Soboĺ sequence

- ◇ Use recursions with a primitive binary polynomial to define the direction numbers v_i
- ◇ The Soboĺ sequence is defined as:

$$s_n = \bigoplus_{i \geq 0} a_{i+1}(n) v_i$$

- ◇ For speed of implementation, we use Gray-code ordering

2. The Faure sequence:

$$x_n^{(k)} = \sum_{i=0}^{\infty} a_{i+1}(n) q^{-(i+1)}, k = 1$$

$$x_n^{(k)} = \sum_{j=0}^{\infty} c_{j+1} q^{-(j+1)}, k \geq 2$$

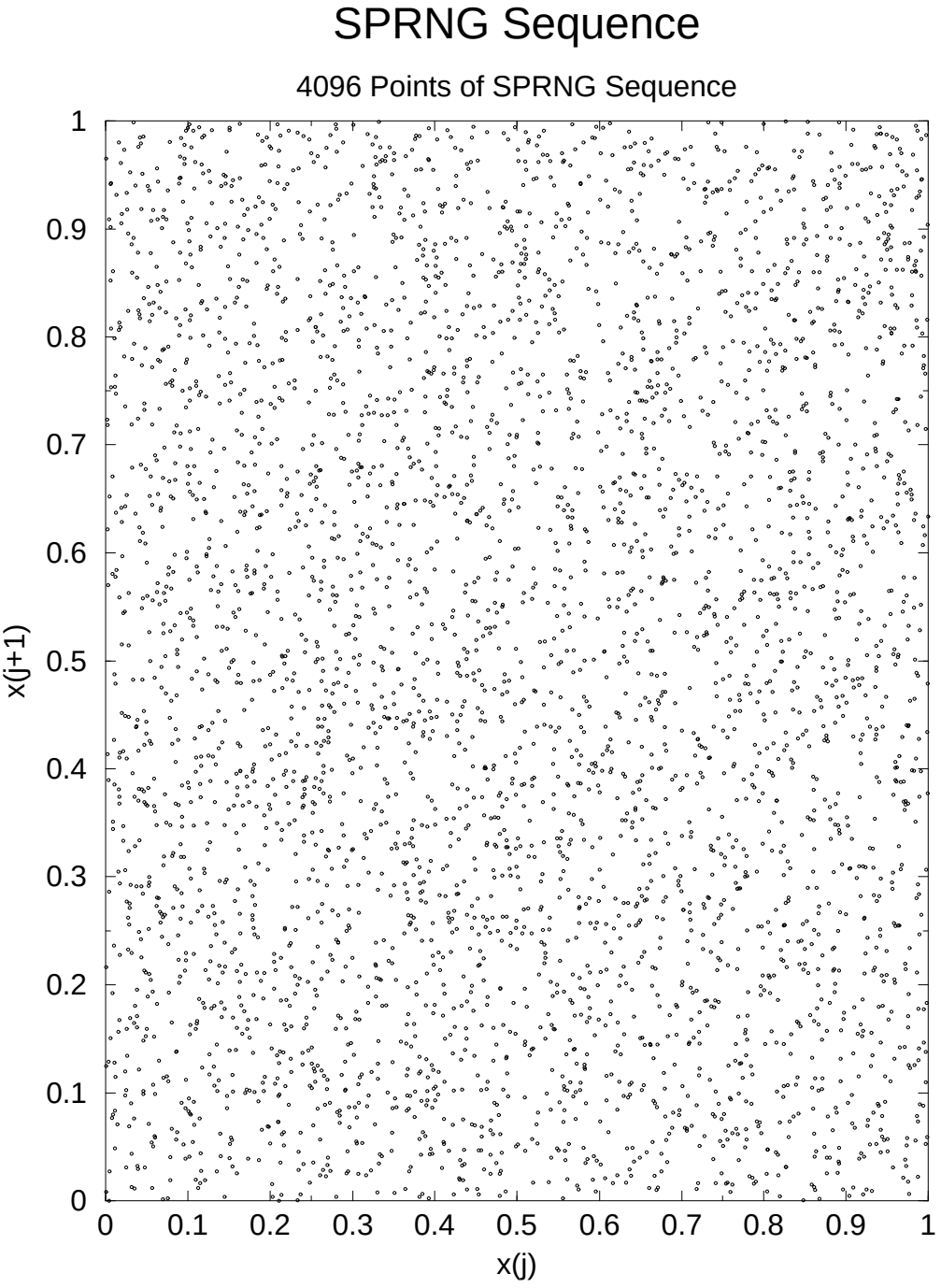
where

$$c_j = \left[D_N^* \sum_{i \geq j} (k-1)^{i-j} \frac{i!}{(i-j)!j!} a_i(n) \right] (mod \ \ q),$$

$j \geq 1$, q is a prime ($q \geq s \geq 2$).

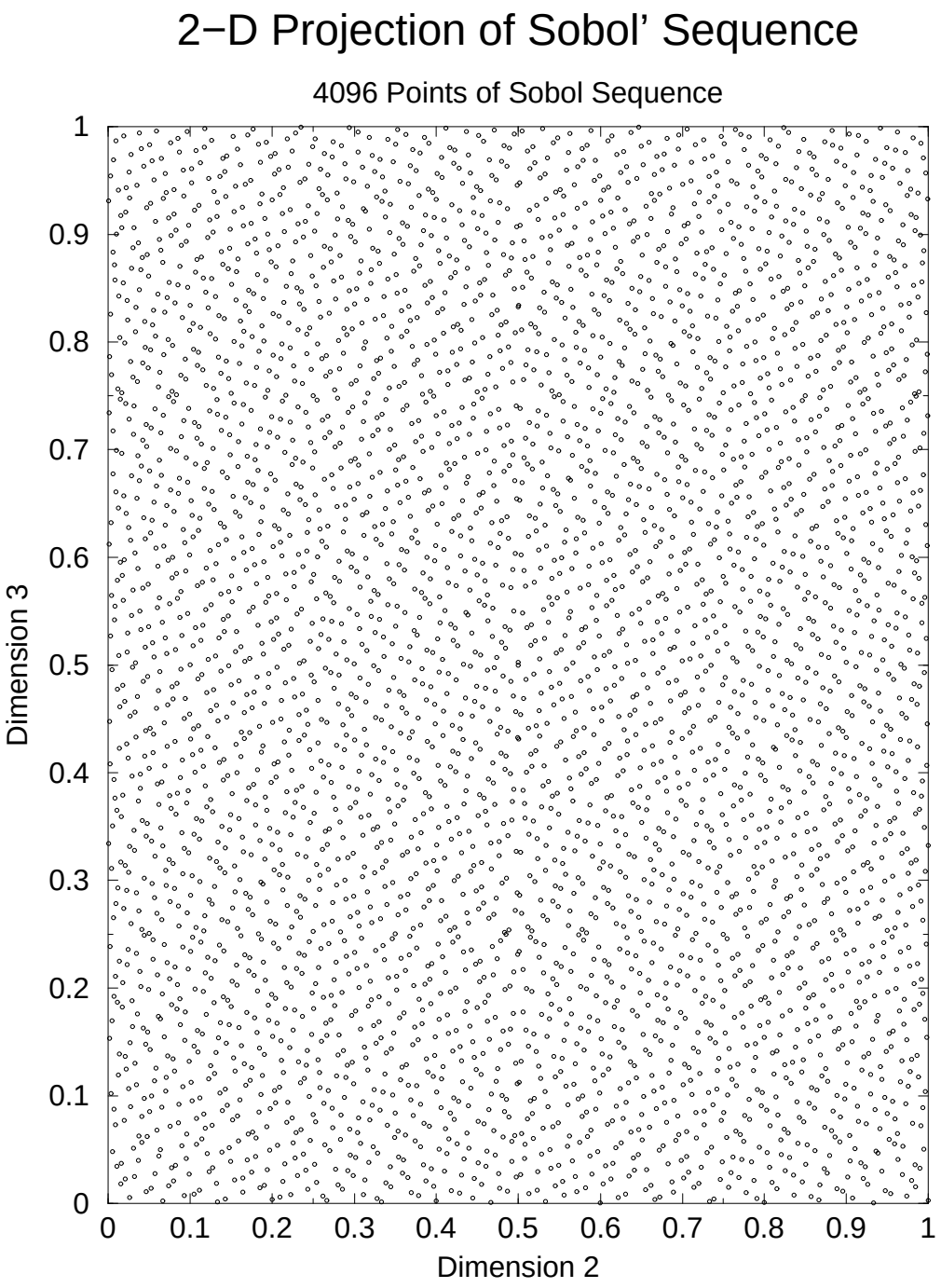
A Picture is Worth a Thousand Words

- 4096 pseudorandom tuples



A Picture is Worth a Thousand Words

- 4096 quasirandom tuples



MCM for Matrix-Vector Products (cont.)

- Consider $(h, A^m f) = h^T A^m f$

Markov chain: $k_0 \to k_1 \to \ldots \to k_m$

- Possible choices for initial and transition densities:

$$p_\alpha = \frac{1}{n}, \;\; p_{\alpha\beta} = \frac{1}{n}, \quad \text{(crude Monte Carlo)}$$

$$p_\alpha = \frac{|h_\alpha|}{\sum_{\alpha=1}^n |h_\alpha|}; \;\; p_{\alpha\beta} = \frac{|a_{\alpha\beta}|}{\sum_{\beta=1}^n |a_{\alpha\beta}|}, \;\; \alpha = 1, \ldots, n,$$

(importance sampling)

Random variable:

$$\theta = \frac{h_{k_0}}{p_{k_0}} W_m f_{k_m}$$

where $W_0 = 1, \;\; W_j = W_{j-1} \frac{a_{k_{j-1} k_j}}{p_{k_{j-1} k_j}}, j = 1, \ldots, m$

$$E[\theta] = h^T A^m f \approx \frac{1}{N} \sum_{s=1}^{N} (\theta)_s$$

with a typical statistical error of size $O(Var(\theta)^{1/2} N^{-1/2})$.

Quasi-MCM for Matrix-Vector Products

- $h^T A^i f$ is really a $(i+1)$-dimensional integral
  - Define the sets $G = [0, n)$ and $G_i = [i-1, i),\ \ i = 1, \ldots, n$, and likewise define the piecewise continuous functions $f(x) = f_i,\ \ x \in G_i, i = 1, \ldots, n$, $a(x, y) = a_{ij}, x \in G_i, y \in G_j,\ \ i, j = 1, \ldots, n$ and $h(x) = h_i,\ \ x \in G_i, i = 1, \ldots, n$.
  - Choose: $p(x) = p_i,\ \ x \in G_i\ \ (\sum_{i=1}^{n} p_i = 1)$, $p(x, y) = p_{ij},\ \ x \in G_i,\ y \in G_j,\ (\sum_{j=1}^{n} p_{ij} = 1, i = 1, \ldots, n)$.
  - Define a random trajectory:

$$T_i = (y_0 \to y_1 \to \ldots \to y_i),$$

    which can be interpreted as a point in $G \times \ldots \times G = G^{i+1}$ with density:

$$p_i(y_0, y_1, \ldots y_i) = p(y_0)p(y_0, y_1) \ldots p(y_{i-1}, y_i).$$

Analysis of the Quasi-MCM

- Integral representation: $(h, A^i f) = E[W_i^\star f_{k_i}] =$

$$\int_{G_0} \dots \int_{G_i} p_i(y_0, y_1, \dots y_i) h(y_0) a(y_0, y_1) \dots$$

$$a(y_{i-1}, y_i) f(y_i) dy_0 \dots dy_i.$$

- Error bound:

$$|h_N^T A_N^m f_N - \frac{1}{N} \sum_{s=1}^{N} h(y_0) a(y_1, y_2) \dots a(y_{m-1}, y_m) f(y_m)|$$

$$\leq |h|^T |A|^m |f| D_N^* \{y_0, y_1, \dots, y_m\}.$$

- Comparison of MCM and QMCM errors:
  - ▶ Products of two factors (first depends on A, second - on the sequence). The order is $N^{-1/2}$ for MCM and $(log^{m+1} N) N^{-1}$ for QMCM
  - ▶ Probabilistic error bound for MCM, worst-case bound (inequality) for QMCM
- Computational complexity for MCM and QMCM is the same: $O((m+1)N)$, where N is the number of chains, $m+1$ is the length of a single Markov chain

The mathematical expectations

- The matrix-vector products:

$$(h, A^m f) = E\left[W_m f_{k_m}\right], \quad i = 1, 2, \ldots .$$

$$(h, [I - qA]^{-m} f) = E[\sum_{i=0}^{\infty} q^i C_{i+m-1}^i W_i f(x_i)].$$

- Scalar product (h, x) for $x = Ax + f$ and given vector h

$$(h, x) = E\left[\sum_{i=0}^{\infty} W_i f_{k_i}\right]$$

- Extremal eigenvalues:

$$\lambda_{max} \approx \frac{E[W_m f_{k_m}]}{E[W_{m-1} f_{k_{m-1}}]}$$

$$\lambda_{min} = \frac{1}{q}\left(1 - \frac{1}{\mu}\right) \approx \frac{E[\sum_{i=1}^{\infty} q^{i-1} C_{i+m-2}^{i-1} W_i f(x_i)]}{E[\sum_{i=0}^{\infty} q^i C_{i+m-1}^i W_i f(x_i)]}.$$

The MC and QMC estimations

- Scalar product of the solution $(x = Ax + f)$

$$(h, x) \approx \frac{1}{N} \sum_{s=1}^{N} \sum_{i=0}^{l} s(W_i f_{k_i})_s$$

- Elements of the inverse matrix $A^{-1} = C = \{c_{rr'}\}$

$$c_{rr'} \approx \frac{1}{N} \sum_{s=1}^{N} \left[\sum_{(j|k_j=r')} W_j \right]_s$$

Computational complexity: lN

Convergence:

$$O\left(\frac{||A||^l ||r^{(0)}||}{1 - ||A||} + \sigma N^{-1/2} \right)$$

$$O\left(\frac{||A||^l ||r^{(0)}||}{1 - ||A||} + (log^l N) N^{-1} \right)$$

The MC estimations (cont.)

- Largest eigenvalue:

$$\lambda_{max} \approx \frac{\sum_{s=1}^{N}(W_m f_{k_m})_s}{\sum_{s=1}^{N}(W_{m-1} f_{k_{m-1}})_s}$$

Computational complexity: mN_w

Convergence:

$$O\left(\|\frac{\lambda_2}{\lambda_1}\|^m + \sigma N^{-1/2}\right)$$

$$O\left(\|\frac{\lambda_2}{\lambda_1}\|^m + (log^m N)N^{-1}\right)$$

- Smallest eigenvalue:

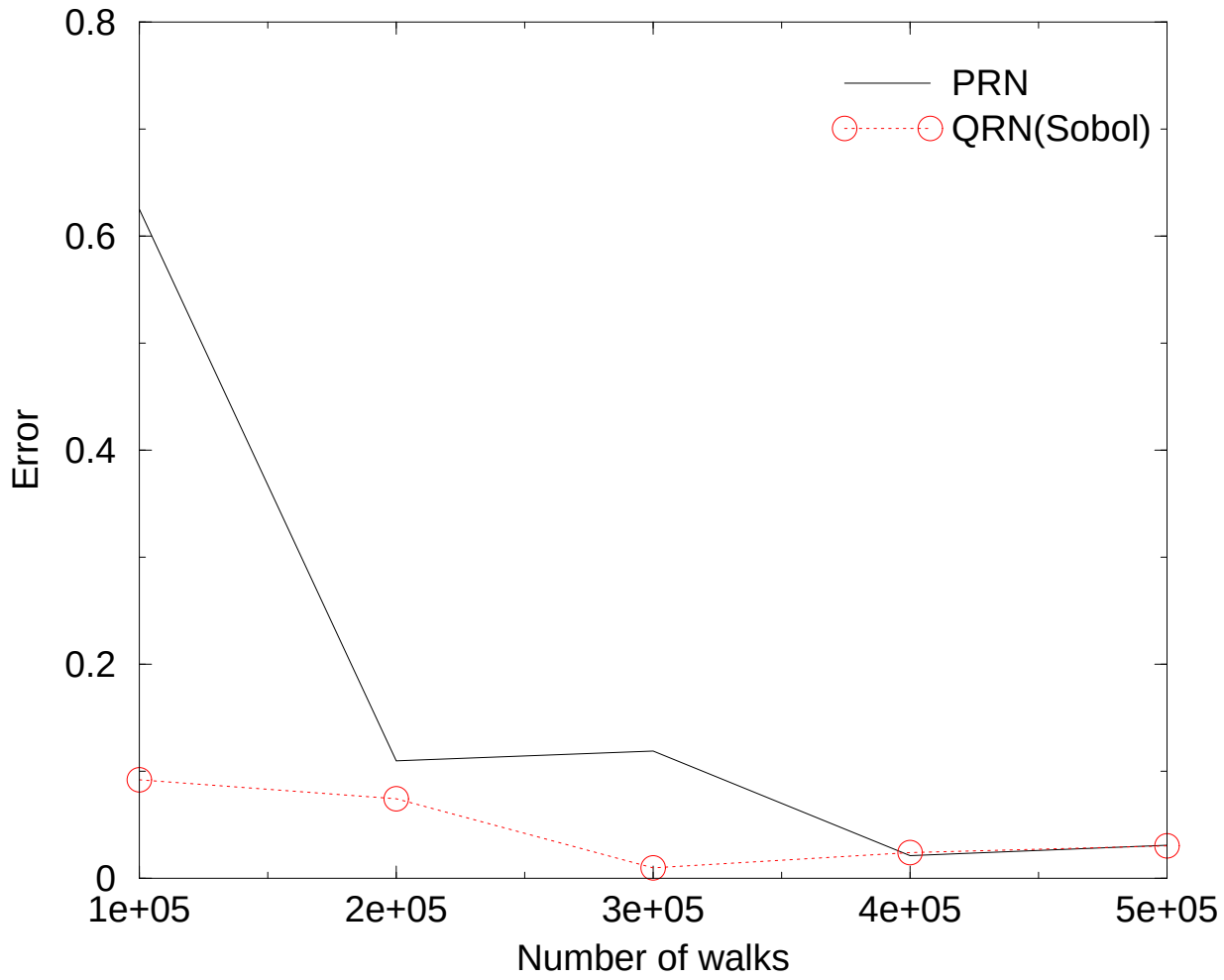
$$\lambda_{min} = \approx \frac{\sum_{s=1}^{N}([\sum^{l}}{\sum_{s=1}^{N}([\sum_{i=0}^{l} q^i C_{i+m-1}^{i-1} W_{i+1} f(x_{i+1})])_s \\ {\scriptstyle i=0}\, q^i C_{i+m-1}^{i} W_i f(x_i)])_s}.$$

Computational complexity: $4lN_w$

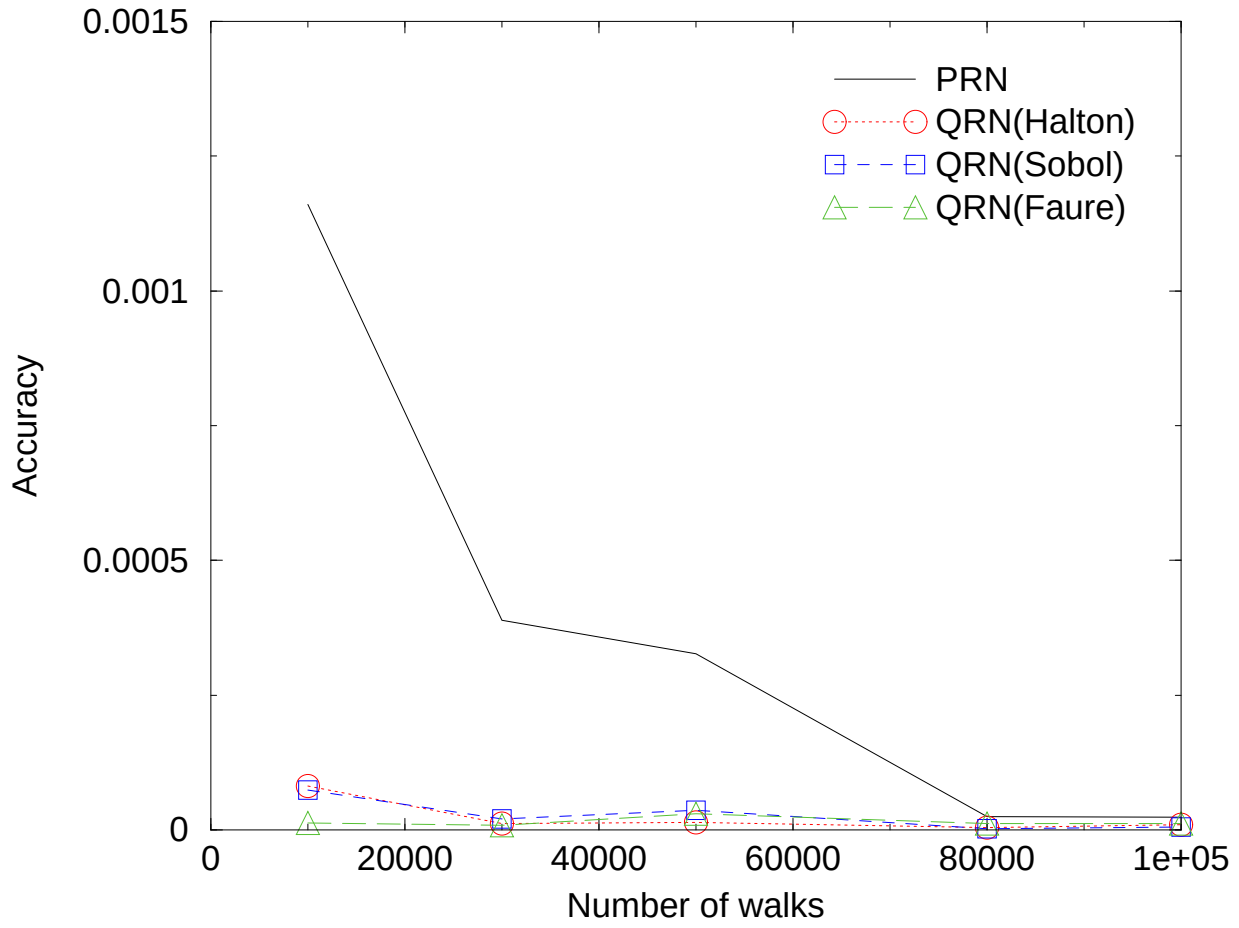
Convergence: $\left(\|\frac{\mu_2}{\mu_1}\|^m + \sigma N^{-1/2}\right)$

$$O\left(\|\frac{\mu_2}{\mu_1}\|^m + (log^l N)N^{-1}\right)$$

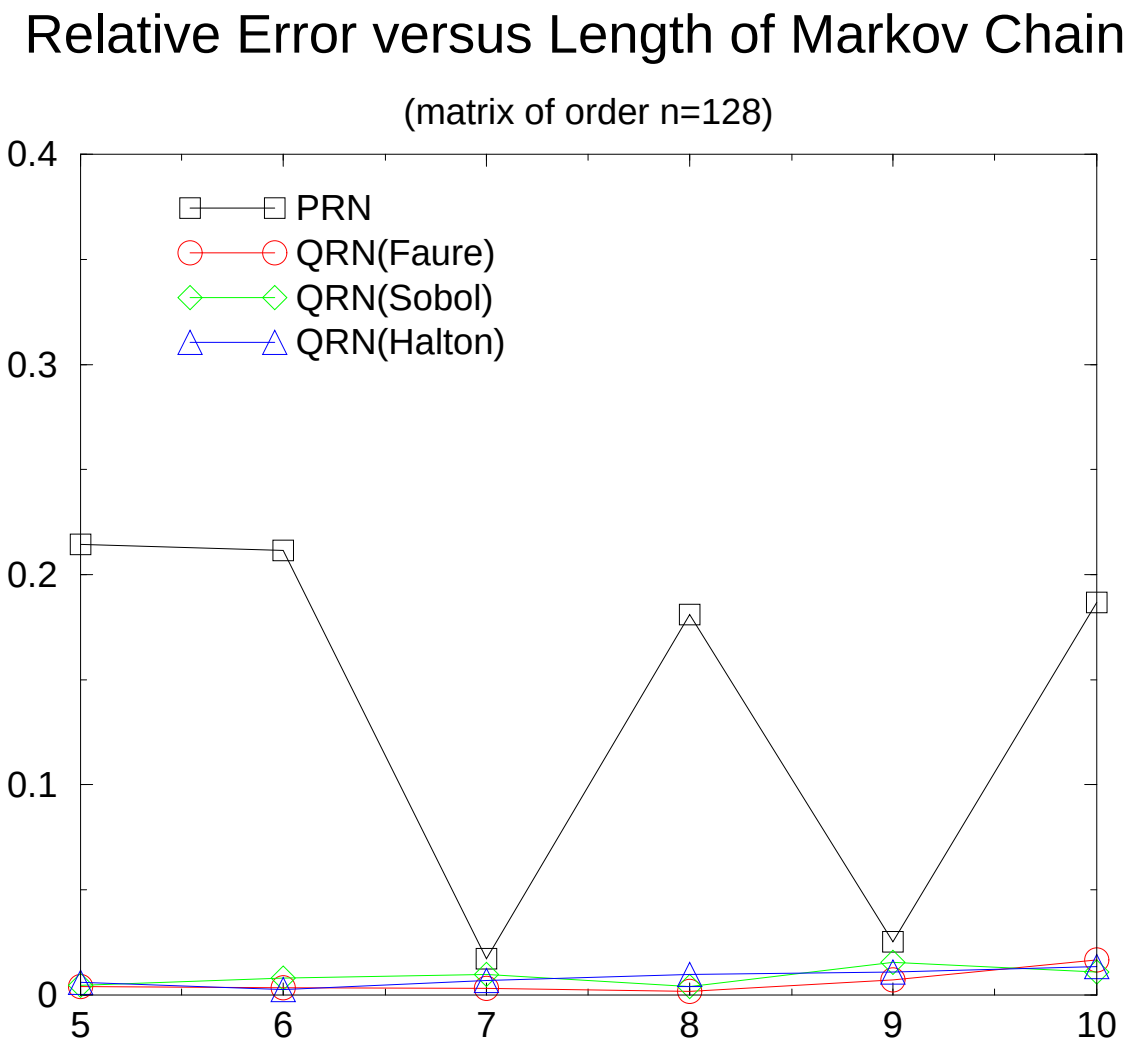
Numerical Results: Convergence: computing (g, x), where x is the solution of a system with 2000 equations.



Numerical Results (Cont.): Convergence: computing one component, x_{64}, of the solution for a system with 1024 equations.

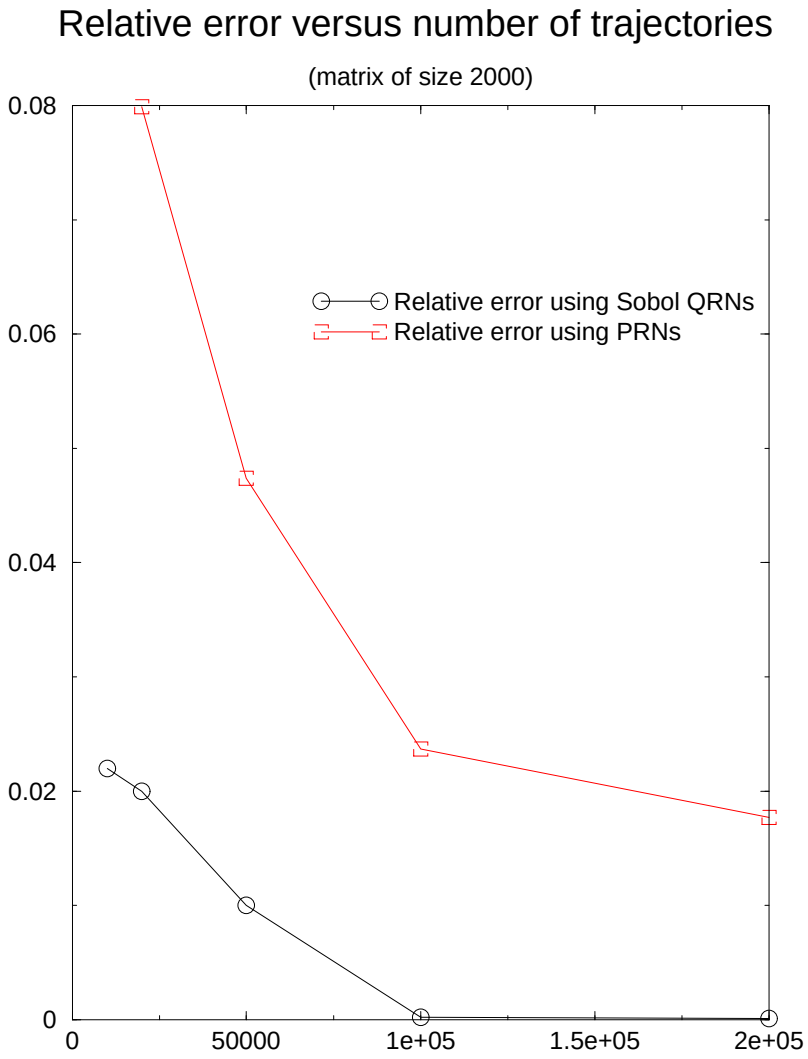


Numerical Results (Cont.): Effect of Increasing Matrix Powers (Relative Errors in $h^T A^k f$)

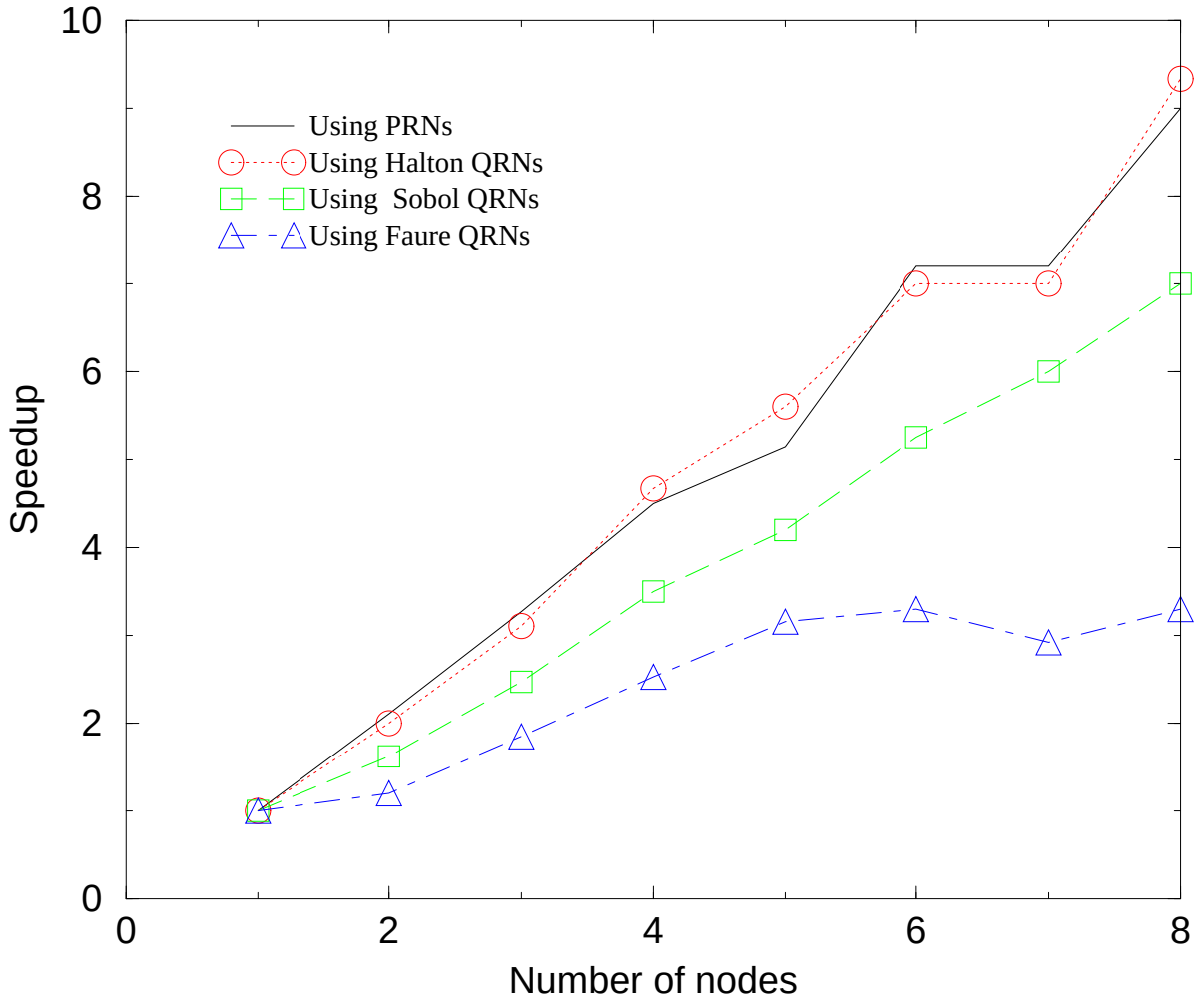


Numerical Results

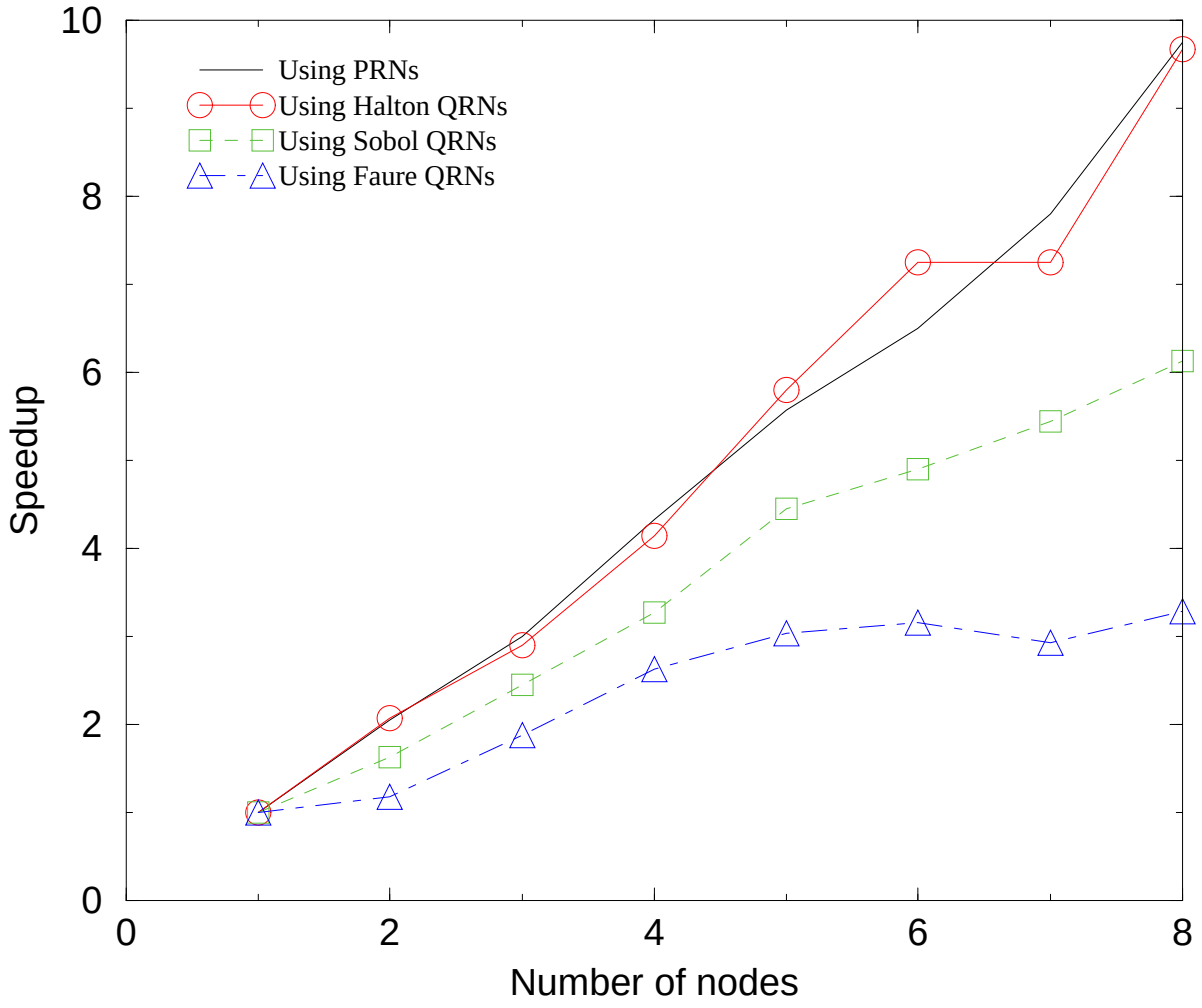
- Relative errors of the MCM and the quasi-MCM (using the Soboĺ sequence) for computing the dominant eigenvalue of a sparse, square, matrix of size 2000



Numerical Results (Cont.): Speedup: solving system with 1024 equations



Numerical Results (Cont.): Speedup: solving system with 2000 equations



Numerical Results(Cont.): Parallel efficiency

Table 1: Sparse system with 1024 equations: MPI times, parallel efficiency and the estimated value of one component of the solution using PRNs, Halton, Soboĺ and Faure QRNs.

	1 pr.	2 pr.	3 pr.	4 pr.	5 pr.	6 pr.	7 pr.	8 pr.
PRN								
T(s)	36	17	11	8	7	5	5	4
Eff		1.05	1.09	1.125	1.02	1.2	1.05	1.125
Sol	1.000031	1.000055	.999963	.999966	1.000024	1.000029	.999979	.999990
Halton								
T(s)	28	14	9	6	5	4	4	3
Eff		1	1.03	1.16	1.12	1.16	1	1.16
Sol	0.999981	0.999981	0.999981	0.999981	0.999981	0.999981	0.999981	0.999981
Sobol								
T(s)	42	25	17	12	10	8	7	6
Eff		0.84	0.82	0.875	0.84	0.875	0.857	0.875
Sol	0.999983	0.999983	0.999983	0.999983	0.999983	0.999983	0.999983	0.999983
Faure								
T(s)	76	63	41	30	24	23	26	23
Eff		0.6	0.62	0.63	0.63	0.55	0.42	0.41
Sol	0.999919	0.999919	0.999919	0.999919	0.999919	0.999919	0.999919	0.999919

Numerical Results(Cont.): Parallel efficiency (Cont.)

Table 1: Sparse system with 2000 equations: MPI times and parallel efficiency and using PRNs, Halton, Sobol and Faure QRNs.

	1 pr.	2 pr.	3 pr.	4 pr.	5 pr.	6 pr.	7 pr.	8 pr.
PRNs								
T(s)	39	19	13	9	7	6	5	4
Eff		1.02	1	1.08	1.11	1.08	1.11	1.21
Halton								
T(s)	29	14	10	7	5	4	4	3
Eff		1.03	0.97	1.03	1.16	1.21	1.03	1.21
Sobol								
T(s)	49	30	20	15	11	10	9	8
Eff		0.82	0.82	0.82	0.89	0.82	0.78	0.76
Faure								
Time (s)	79	67	42	30	26	25	27	24
Efficiency		0.59	0.63	0.66	0.61	0.53	0.42	0.41

Conclusions and Future Directions

1. Conclusions from the presented work

- We have shown that QRNs can improve convergence of *discrete* Markov chain problems
  - ▶ Have seen faster convergence
  - ▶ Have also seen smoother convergence with i
- QMCMs and MCMs have the same computational complexity
- QMCMs preserve the parallel efficiency of MCMs

2. Future work

- Application of results with discrete Markov chains
  - ▶ Better QRN generation tools
- Study the use of c.u.d. sequences