Quasi-Monte Carlo Integration on GRIDS: Using Blocked Substreams

Heinz Hofbauer* and Andreas Uhl† and Peter Zinterhof‡

Abstract. *The splitting of Quasi-Monte Carlo (QMC) point sequences into blocks or interleaved substreams has been suggested to raise the speed of distributed numerical integration and to lower to traffic on the network. The usefulness of this approach in GRID environments is discussed. After specifying requirements for using QMC techniques in GRID environments in general we review and evaluate the proposals made in literature so far. In numerical integration experiments we investigate the ability of blocking, and to a lower extend leaping, to deal with the special requirements in the GRID, with regard to the Sobol', Halton, Faure, Niederreiter-Xing, and Zinterhof sequences.*

1. Introduction

High dimensional numerical integration problems may require a significant amount of computations. Therefore, substantial effort has been invested in finding techniques for performing these computations on all kinds of parallel architectures (see [4, 11, 12, 24] for an exhaustive overview). In order to minimize the communication amount within a parallel system, each processing element (PE) requires its own source of integration nodes. Therefore, the aim is to investigate techniques for using separately initialized and disjoint sets of integration nodes on a single PE.

Currently, the most efficient numerical techniques for evaluating high-dimensional integrals are based on Monte Carlo and quasi-Monte Carlo techniques [7]. Whereas in the Monte Carlo (MC) case the integration nodes are produced by a random number generator (RNG), low-discrepancy point sets and sequences (e.g. (t,m,s)-nets or (t,s)-sequences [18]) are employed in quasi-Monte Carlo (QMC) algorithms. QMC techniques improve the probabilistic error bounds of MC techniques especially in higher dimensions. Nevertheless, these techniques are related [5] since a full period random number sequence may be seen as a low-discrepancy point set (e.g. a rank-1 lattice rule in the case of a linear congruential generator) as well.

GRID environments exhibit challenging properties for numerical integration techniques. This type of computing facility potentially shows extreme heterogeneity in terms of PE speed and network connections, moreover the available computing resources may change in time even during ongoing computations (e.g. new machines may become available or others may get lost due to network problems or maintainance shutdowns, other users may start computations on the same hardware, etc.). Li et al. [14] discuss the use of a GRID service called Integration Service for solving multivariate integration problems, the PARINT package [4] is used as integration engine.

*Department or Computer Sciences, University of Salzburg, Salzburg, Austria, email: hhofbaue@cosy.sbg.ac.at
†Department or Computer Sciences, University of Salzburg, Salzburg, Austria, email: uhl@cosy.sbg.ac.at
‡Department or Computer Sciences, University of Salzburg, Salzburg, Austria, email: peter.zinterhof@sbg.ac.at

Therefore, it is not only difficult but impossible to predict the amount of integration nodes required on a single PE under such conditions. Additionally, in practice it is usually not possible to determine a priori the number of integration nodes N necessary to meet a given error requirement. As a consequence, it is of great importance that N may be increased without losing previously calculated function values. It is clear that techniques for providing integration nodes on the single PEs need to be very flexible under these circumstances.

One possibility is to generate separately initialized and disjoint substreams of a given (sequential) sequence of integration nodes. Among other suggestions made in literature in this context, we have investigated leaped substream parallelization techniques for (t,s)-sequences in previous work [21, 22, 6] and have discovered several shortcomings and problems when applied in parallel and distributed QMC integration. In [10] we further investigated the shortcomings of leaped substreams with special regard to an application in the GRID.

In this work, we extend the focus of our work on parallelization of improper integration in the GRID by the following points:

- So far, only the quality of using leaping in heterogenous environments (even though extreme cases) has been investigated. Here we additionally treat how well splitting methods are able to cope with the special properties of the GRID.
- So far, only leaping has been examined closer. Here we also take into account blocking.

In Section 2 we discuss strategies for using QMC techniques in GRID environments. Section 3 shortly reviews the QMC computation of improper integrals (which is being used as application case). Section 4 is the main part of this work where we report and discuss experimental integration results using various QMC node sets. Section 5 concludes this work and provides outlook to future work in this direction.

2. QMC Techniques in GRID Environments

GRID environments exhibit a potentially high heterogeneity in terms of network capacity (i.e. bandwidth and latency) and computing speed (memory capacity, cache sizes, processor speed). In addition to that, these environments are error prone with respect to broken network links or failing PEs. Moreover, additional resources may become available during an ongoing simulation which should be used to optimize resource consumption. As a consequence, the following requirements should be met by a QMC technique employed in a GRID environment:

- Variety in computing speed requires dynamic load balancing capability.
- Variety in network capacity requires load balancing strategies without central organization and a minimal number of control messages exchanged among the computing nodes.
- Failure in hardware resources requires tolerance to lost partial results.
- Additional resources becoming available require a possibility to assign workload to these resources (i.e. by redistributing or redefining workload).

In addition to that, error bounds and computation results should preferably carry over from sequential execution. If the the QMC point sets differ between sequential and parallel execution, the quality of the results needs to be investigated thoroughly. Reproducibility is as well an important issue to be considered.

So far, two entirely different strategies have been discussed in literature to employ QMC sequences in parallel and distributed environments.

1. Splitting a given QMC sequence into separately initialized and disjoint parts which are then used independently on the PEs. This strategy comes in two flavors:
   - **Blocking**: p disjoint contiguous blocks of maximal length l of the original sequence are used on the PEs. This is achieved by simply using a different starting point on each PE (e.g., PE_i, $i = 0, \ldots, p-1$, generates the vectors $x_{il}, x_{il+1}, x_{il+2}, \ldots, x_{il+l-1}$). In case a large number of smaller blocks is used index j is assigned dynamically to PE_i which generates the vectors $x_j, x_{j+1}, \ldots, x_{j+l-1}$ (where j is incremented in steps of size l to avoid overlap).
   - **Leaping**: interleaved streams of the original sequence are used on the PEs. Each PE skips those points consumed by other PEs (*leap-frogging*) (e. g. employing p PEs, PE_i, $i = 0, \ldots, p-1$, generates the vectors $x_i, x_{i+p}, x_{i+2p}, \ldots$).
2. Using inherently independent sequences on the different PEs (denoted as "parametrization" which can be realized for example by randomization of given QMC sequences). Parametrization will not be discussed due to the limited scope of the paper.

Blocking has been suggested to be used in many application focused papers. Mascagni and Karaivanova [16] propose to use disjoint contiguous blocks from Halton, Faure, and Sobol' sequences in the context of solving sparse systems of linear algebraic equations. Numerical experiments are carried out on a homogeneous cluster using static load distribution. In a second paper [15] the same authors use the suggested techniques for computing extremal eigenvalues, again a QMC sequence is "neatly broken into same-sized subsequences" by blocking. The authors point out that this simple strategy can not be employed in general for all types of simulation settings. Alexandrov et al. [1] use scrambled Sobol' and Halton sequences to solve certain linear algebra systems. They discuss static and dynamic load balancing and point out the importance of efficient dynamic load balancing in GRID environments. Load balancing is done by dynamically distributing chunks (i.e. blocks) of relatively small size to avoid unevenly sized chunks and techniques for efficiently generating non-adjacent chunks an a single PE are discussed. Tests are carried out on homogeneous and heterogeneous systems, in the latter case MPICH over Globus-2 GRID software is used. Wan et al. [28] present a parallel strategy for pricing multidimensional american options. In a first stage, the QMC sequence is generated by independently computing equally sized blocks on the PEs using static load distribution. For the second stage, the stochastic mesh method which involves a backward recursion, two strategies for data distribution are compared which both correspond to distributing the original sequence in blocks of different size in different manner across the PEs. Test are conducted on a SGI Onyx machine. Schürer [23] employs equally sized blocks of (t,m,s)-nets on the PEs when comparing QMC integration techniques to adaptive cubature rules. A SGI Power Challenge is used as a test platform. In previous work [21] we have conducted experiments with blocking Niederreiter (t,s)-sequences where large disjoint blocks are used on the PEs. Good reliability of the results has been observed in homogeneous and (simulations of) heterogeneous environments (tests conducted on a SGI Power Challenge). We have

also provided theoretical evidence for this good behavior by showing that discrepancy estimates of arbitrary blocks do not degrade as compared to estimates of entire (t,s)-sequences [22].

Leaping has been discussed much more controversial in literature. Bromley [2] describes a leapfrog parallelization technique to break up the Sobol' sequence into interleaved substreams in an efficient manner. We have generalized this idea to all types of binary digital (t,s)-sequences [22] in earlier work. Based on these techniques, Li and Mullen [13] use a leap frog scheme for (t,m,s)-nets to solve financial derivative problems. However, severe problems occur with leapfrog parallelization especially in case of processor speed heterogeneity which results in QMC point sets which do not correspond to sequential computation. First results showed that single (t,s)-sequence substreams with leaps of the form 2^n lead to extremely poor numerical integration results whereas this is not the case for leaps of the form 2^{n+1} [21]. Using leaped substream parallelization in a heterogeneous processor speed environment therefore may lead to severely degraded results as compared to sequential execution when this form of leaping is employed. Different PEs consume a different amount of integration nodes and so the poor results of using single substreams are propagated to the parallel results if no synchronization among PEs is performed [6, 21, 22]. We have also provided theoretical evidence for the observed effects by showing the discrepancy estimated of leaped substreams to be significantly larger as compared to the original sequences [22]. It has also turned out that not only 2^n type substreams are affected by poor quality but these effects occur for many forms of leaps and are highly unpredictable [6, 22].

Based on the requirements for a QMC technique to be useful in GRID environments stated before we try to assess the respective suitedness of the two splitting QMC techniques proposed in literature.

Blocking: Two flavors of blocking are discussed. In the first variant, the QMC sequence is partitioned into small blocks which are dynamically distributed among the PEs. Whereas this technique uses QMC node sets almost identical to sequential execution and can handle all types of changing resource scenarios and heterogeneity quite well, it requires the frequent exchange of control messages and is therefore not suited for GRID environments. The validity of this assessment of course depends on the relation between block size and the communication possibilities in the actual GRID environment. In the second blocking variant, one large block is assigned to each PE at the start of the computation. Since the number of QMC points required on each PE is not known a priori the block size needs to be chosen large enough to avoid a PE to exceed the number of available points in its block (exceeding the number would then result in overlap of the blocks which of course degrades the final result). On the other hand, if the blocks have been selected much too large, a significant amount of points may not be consumed on slow PEs and the overall point set used exhibits large "gaps" as compared to the sequential case which potentially threatens result accuracy (although the results available so far concerning this effect do not seem to be very severe). Choosing the block size appropriately is therefore a critical issue in this approach.

In addition to that, the possibility of failing nodes or nodes which become available during computation has to be considered. With both big and small block sizes it is possible to assign blocks to nodes which become available, in the case of big blocks the new machines get a single big block following the last block assigned. In the case of small blocks the request for a block is treated like every other request and the next block is given out to whichever node requests it. When a machine fails, the block it currently processed will be unfinished leaving a gap in the otherwise continuous point sequence. With small block sizes this is not a problem since the block can be flagged unfinished and assigned

in the usual fashion. With big blocks the matter becomes rather more difficult, since the big block approach is chosen specifically in such a fashion that no reassigning of blocks happens. Thus if a node fails the block is essentially a gap in the point sequence which can't be reassigned. Even if a new node would become available assigning the unused block can be a risky. The block can be assigned without problem if there are no partial results since the whole block would become available to the new node, which is of the same size as if a new block would be assigned. However, if there are partial results and we don't want to loose these partial results, which would be the same as if no results are present, it would be possible to use the remaining part of the block. While this seems a valid approach at first it should be considered that the computation nodes in the GRID do not all have the same speed, thus the new node could end up needing a whole block and only receiving a partial block. This would result in an overlapping of blocks which could adversely affect the result.

While using big blocks is clearly more efficient than using small blocks, in terms of network usage and delay of calculation, the drawbacks, gaps and overlaps, can negatively affect the integration. When using small blocks, the network efficiency of using big blocks or leaping can't be achieved. However, the delay of calculation can be eliminated, or at least reduced by a great amount, by carefully scheduling the assignment of new blocks. There are two possible ways to deal with this, one is to assign two block to each node, when a block is finished a new block is requested and the other assigned block is worked on. This approach would result in block sized gaps towards the end of the calculation, when each spare block is not processed when the overall calculation stops. The second approach is to prerequest a block before the current block is finished, this assumes each node keeps track of it's speed of computation and the round trip time (rtt) to the root node. Then when it sets out the request for a new block when the rtt equals the time it needs to process the remaining points in it's current block. This would result in the timely arrival of a new block when the current block is done. While this will likely be off at the beginning of the overall process which results in a block being assigned to early, and could lead to the same problem as with two blocks being assigned, this does not pose a problem for a careful chosen size of blocks. When the block size is small enough so that several blocks will be processed by each node before the overall calculation is finished the nodes have enough time to adapt the estimated rtt and speed of calculation to prefetch new blocks just in time, which alleviates the above mentioned problem. However while these methods have nearly the same speedup as leaping the load on the network will be higher, and this can't be reduced by a great amount since the nodes need a number of previous results for rtt to correctly estimate it.

This will be discussed in more details in the context of the results in Section 4.

Leaping: Contrasting to the blocking case, there is no need to specify a number of points required on each PE since each substream may deliver an infinite number of points in principle. Therefore, there is no danger of running short of points. Also, substream overlap can not occur. In the case of homogeneous environments where care is taken that each PE consumes an equal share of QMC points, a result identical to sequential execution is easily obtained. Note that this is not the case for blocking due to the problems with choosing a good block size (except in case the number of points required is known in advance – which is rarely the case). The situation changes drastically in heterogeneous environments: in case of different PEs consuming a different amount of QMC points it has been shown that depending on the type of load imbalance more or less severe degradations in result accuracy are observed. The same considerations (with even more pronounced effects) of course apply as well if a PE fails. Also the failing of a PE can't be alleviated since the stream can't be redistributed to other nodes, the only possibility would be to assign the stream, from where it broke off, to a node which

becomes available at a later time. However, even when a node becomes available and the stream is continued the stream would be short by an amount of points equivalent to the time it was unused, leading to imbalance between the streams. When additional resources become available the classical leaping scenario can not handle this situation properly – usually a QMC node set is partitioned into J interleaved substreams if J PEs are available. The is no additional substream available in this scenario. A way to handle this situation is to partition a given QMC point set into $I > J$ substreams in case of J PEs are available. The $I - J$ substreams are not used by default but kept as additional work share in case additional PEs become available. However, neither empirical nor theoretical results are available so far to assess the quality of corresponding results. The use of leaping in GRID environments may be therefore accompanied with problematic side effects which endanger a flexible and transparent use. This will be discussed in more details in the context of the results in Section 4. In addition to that, the most important advantage of leaped substream parallelization as compared to blocking (i.e. in case of synchronized execution the used point set correspond to the sequential case) does not apply in GRID environments due to the heterogeneity.

3. QMC Computation of Improper Integrals

The basic concept of any method for numerical integration is to approximate the integral by a finite sum, such that

$$I(f) := \int_{I^s} f(x)dx \approx \frac{1}{N}\sum_{n=1}^{N} f(x_n) =: I'_N(f) \tag{1}$$

where x_n are suitably chosen and I^s is the unit interval. To identify suitable, i.e. uniformly distributed, point sets x_n the star discrepancy is defined as

$$D_N^* := D_N^*(x_1,\ldots,x_n) = \sup_{J\in\mathcal{F}} \left\| \frac{\#\{x|x\in J\}}{N} - m(J) \right\| \tag{2}$$

where $\mathcal{F}$ is the family of all subintervals of the form $J = \prod_{i=1}^{s}[0,t_i) \in I^s$ with volume $m(J)$. The approximation error

$$E_N(f) := |I'_N(f) - I(f)| \tag{3}$$

depends on D_N^* and the variation $V(f)$ of the function f in the following way (Koksma-Hlawka inequality [18]):

$$E_N(f) \leq V(f)D_N^* \tag{4}$$

Consequently, point sets exhibiting low discrepancy values are attractive candidates to be used in numerical integration. Using low discrepancy sequences as point set x_n is denoted QMC approach.

QMC techniques have been considered to evaluate improper integrals in the last years only – for numerical integration of such types of integrands we use a generic method introduced recently by

Zinterhof [31] (which can employ any type of integration nodes in principle but delivers decent error bounds for low-discrepancy integration node sets). Zinterhof specifies for a function $f(x)$ and a $B > 0$ the functions $f_B(x)$ and $\hat{f}_B(x)$ as

$$f_B(x) = \begin{cases} f(x) & |f(x)| \leq B \\ 0 & |f(x)| > B \end{cases} \tag{5}$$

$$\hat{f}_B(x) = \begin{cases} 0 & |f(x)| \leq B \\ f(x) & |f(x)| > B. \end{cases} \tag{6}$$

The Class $C(\beta,\gamma)$ of s-variate functions $f(x_1,\ldots,x_s)$, $0 \leq x_i \leq 1$, $i = 1,\ldots,s$, consists of all functions which fulfill

$$\text{a)} \quad I(|\hat{f}_B|) = O(B^{-\beta}) \text{ for some } \beta > 0 \tag{7}$$

$$\text{b)} \quad V(f_B) = O(B^{\gamma}) \text{ for some } \gamma \geq 1. \tag{8}$$

It is shown that if $f(x) \in C(\beta,\gamma)$, $\mathfrak{x} = (x_1,\ldots,x_s)$, and if the discrepancy of the set of nodes is D_N^*, then for $B = {D_N^*}^{-1/(\beta+\gamma)}$ the estimate

$$I(f) = \frac{1}{N}\sum_{n=1}^{N} f_B(\mathfrak{x}_n) + O({D_N^*}^{\beta/(\beta+\gamma)}) \tag{9}$$

holds, where $I(f) = I(f_B) + I(\hat{f}_B)$. Finally Zinterhof shows that when the truncation parameter B takes on the form

$$B = {D_N^*}^{-1/(\beta+\gamma)} \tag{10}$$

the integration error attains its minimum of

$$E_N \leq K(f){D_N^*}^{\beta/(\beta+\gamma)} \tag{11}$$

where $K(f)$ is a constant depending on f. Consequently, the numerical strategy is to compute $\frac{1}{N}\sum_{n=1}^{N} f_B(\mathfrak{x}_n)$ upon selection of a suitable B.

4. Experiments: Parallelization Effects on Integration

4.1. QMC Node Sets

For generating the Sobol', Halton, Faure and Niederreiter-Xing sequences we use the implementation of the "High-dimensional Integration Library" HIntLib[1]. The descriptions of the Sobol', Faure, and Niederreiter-Xing sequences are taken from [25].

[1] Available at: http://www.cosy.sbg.ac.at/~rschuer/hintlib/

4.1.1. Sobol' Sequence

The Sobol' sequences [27] are digital (t_s, s)-sequences over F_2, where

$$t_s = \sum_{i=1}^{s} (\deg p_i - 1), \tag{12}$$

with $p_1 = x \in F_2[x]$ and p_{i+1} denoting the ith primitive polynomial over F_2 ordered by degree.

Sobol' sequences were the first known constructions yielding (t,s)-sequences for arbitrary dimensions s. They were introduced long before the theory of (t,s)-sequences over arbitrary finite fields F_b was established in [19]. However, they only exist for $b = 2$, and even in this case, the resulting t parameter is not optimal for $s > 3$. For $s > 7$, even the Niederreiter sequence, which is equally easy to implement, yields lower t-values.

For $s = 1$ the Sobol' sequence (defined by the polynomial $p_1 = x$) is a $(0,1)$-sequence identical to the van der Corput sequence in base 2.

We use the implementation of construction 6 in [17].

4.1.2. Faure Sequence

The Faure sequences are digital $(0,s)$-sequences over F_b with b denoting a prime number (original case) or a prime power (general case) greater or equal to s. The case for b prime was shown by Faure [8], the general result is due to Niederreiter [19, Theorem 6.2].

The s infinite generator matrices $C^{(1)}, \ldots, C^{(s)}$ over F_b are defined by $C^{(i)} = (c_{jr}^{(i)})_{j,r>0}$ with

$$c_{jr}^{(i)} = \binom{r}{j} \alpha_i^{r-j}, \tag{13}$$

where $\alpha_1, \ldots, \alpha_s$ denote s distinct elements from F_b and the conventions $\alpha^0 = 1$ for all $\alpha \in F_b$ and $\binom{r}{j} = 0$ for $j > r$ are used.

For $\alpha = 1$, the resulting matrix is the infinite Pascal matrix modulo the characteristic of F_b; for $\alpha = 0$, it is the infinite identity matrix. If $s = 1$ and $\alpha_1 = 0$, the resulting $(0,1)$-sequence is identical to the van der Corput sequence in the same base.

Sequences with the same parameters can also be obtained using Niederreiter sequences or Niederreiter-Xing sequences with rational function fields.

We use the implementation of construction 8 in [17].

4.1.3. Halton Sequence

The construction of the Halton sequence was introduced in [9]. For a dimension $s > 0$, let $b_1, \ldots, b_s$ be integers ≥ 2. Then the Halton sequence in the bases sequence $b_1, \ldots, b_s$ is defined as $x_0, x_1, \ldots$ with

$$x_n = (\Phi_{b_1}(n), \ldots, \Phi_{b_s}(n)) \in I^s \; \forall n \geq 0, \tag{14}$$

where $\Phi_b(n)$ is the radical inverse function. The radical inverse function in base b is defined as

$$\Phi_b(n) = \sum_{i=0}^{\infty} a_i(n) b^{-j-1} \; \forall n \geq 0, \tag{15}$$

where $a_i(n)$ is the ith digit in the digit expansion of n in base b.

4.1.4. Niederreiter-Xing Sequence

In [20] and [29] Niederreiter and Xing develop two methods for creating a digital (t,s)-sequence over F_b based on an algebraic function field with full constant field F_b, genus t, and containing at least $s+1$ rational places. Niederreiter-Xing sequence construction III is constructive, assuming that defining equations for the function field are given and that $s+1$ rational places are known. We use the implementation of construction 18 in [3].

4.1.5. Zinterhof Sequence

Zinterhof sequences ([30]) are a special case of Weyl sequences. Weyl sequences are defined by

$$x_n = (\{n\theta_1\}, \{n\theta_2\}, \ldots, \{n\theta_s\}) \; n = 1,2,3,\ldots$$

where s is the dimension and $\{x\}$ is the fractional part of x. It is well known that a Weyl sequence is uniformly distributed iff θ_i are independent irrational numbers. An important issue with respect to their quality in terms of uniformity of distribution is the amount or degree of irrationality of the employed starting vector $\Theta = (\theta_1, \ldots, \theta_s)$. For the Zinterhof sequence we set $\theta_i = e^{1/i}$ and consequently:

$$x_n = (\{ne^{1/1}\}, \ldots, \{ne^{1/s}\}) \; n = 1,2,3,\ldots. \tag{16}$$

The Zinterhof sequence has been shown to exhibit excellent distribution behavior and they excel by their ease of construction and implementation even for non-specialists.

4.2. Test Functions

A widely used test function for numerical integration of improper integrals, which was first used by Sobol' in 1973 [26], is

$$t(x) = \frac{1}{x_1^{\alpha_1} \cdots x_s^{\alpha_s}}, \tag{17}$$

where s is the dimension and $0 < \alpha_i < 1 \; \forall i \in 1, \ldots, s$. We use a slightly less general version of this test function of the form

$$f(x) = \prod_{i=1}^{s} \frac{1}{x_i^{\alpha}} \tag{18}$$

where s is the dimension and $0 < \alpha < 1$. For $f(x)$ clearly $\int_0^1 f(x)dx = \left(\frac{1}{1-\alpha}^s\right)$ and the value of α determines the severity of the singularity.

4.3. Results

For each of the above point sets the following test where executed:
Gap: For this test big blocks were used which were laid out such that there is an unused gap between the blocks. The gap is 20% the size of a block.
Overlap: For the overlap test big blocks are used with a size set to generate less than the overall number of points and no acquisition of new blocks. This resulted in an overlap where about 30% of one block overlaps the following block.
Streamsave: For this test small blocks were used with a point size of 75 and between block there is a gap with size 25. This test can also be interpreted as using synchronized streams (leaping) where the last 25 of the 100 overall streams are reserved for nodes which become available during computation. The synchronization of the streams was used to get a better grasp of the influence of missing streams, this is somewhat artificial since leaping performs somewhat erratic in a heterogenous environment (see [10]).

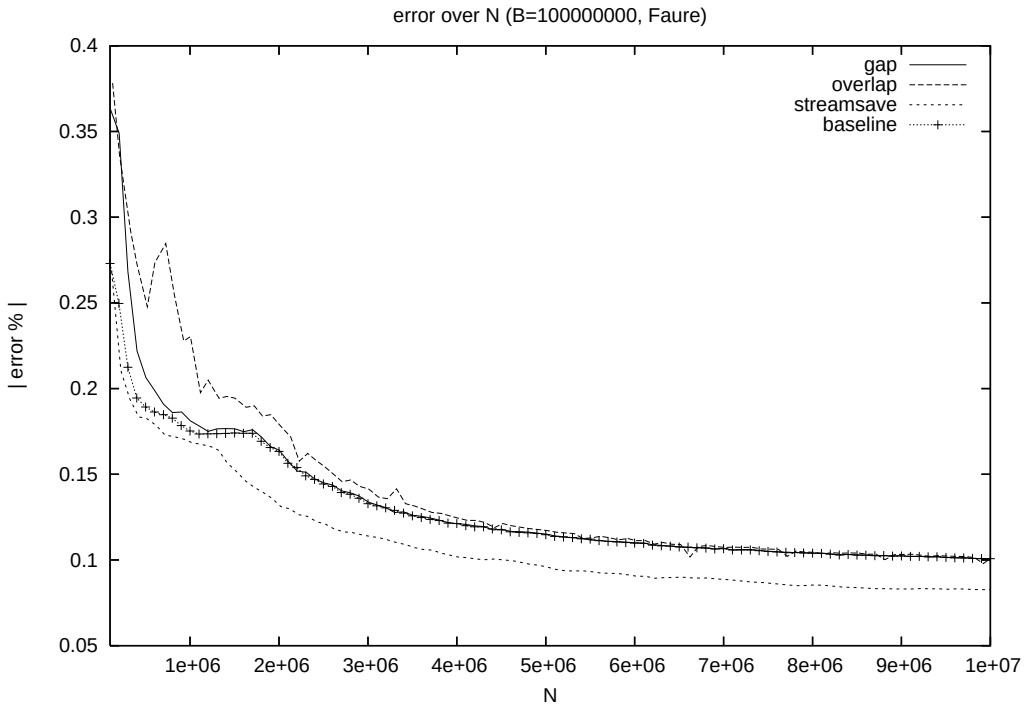

Figure 1. Comparison of overlap, gaps and stream save.

Additionally, a baseline run, calculation of exactly N contiguous points, is given for comparison. The results from the test is given in Fig. 1 to Fig. 5, the range of x axis is the same for all graphics but the range of the y axis is chosen so as to discern the effects of a single graph instead of keeping the same range for comparison.

Overall for these test all point sets except Halton behave well, meaning toward higher N they are smooth and difference between the other cases and baseline become negligible and the streamsave case does exceptionally well, even with Halton.

However, apart form the overall poor performance of Halton for gaps and overlaps another effect can be discerned. All functions, even the otherwise well behaved Niederreiter/Xing sequence show some fluctuation of the results. When comparing this to the baseline run for each function we see

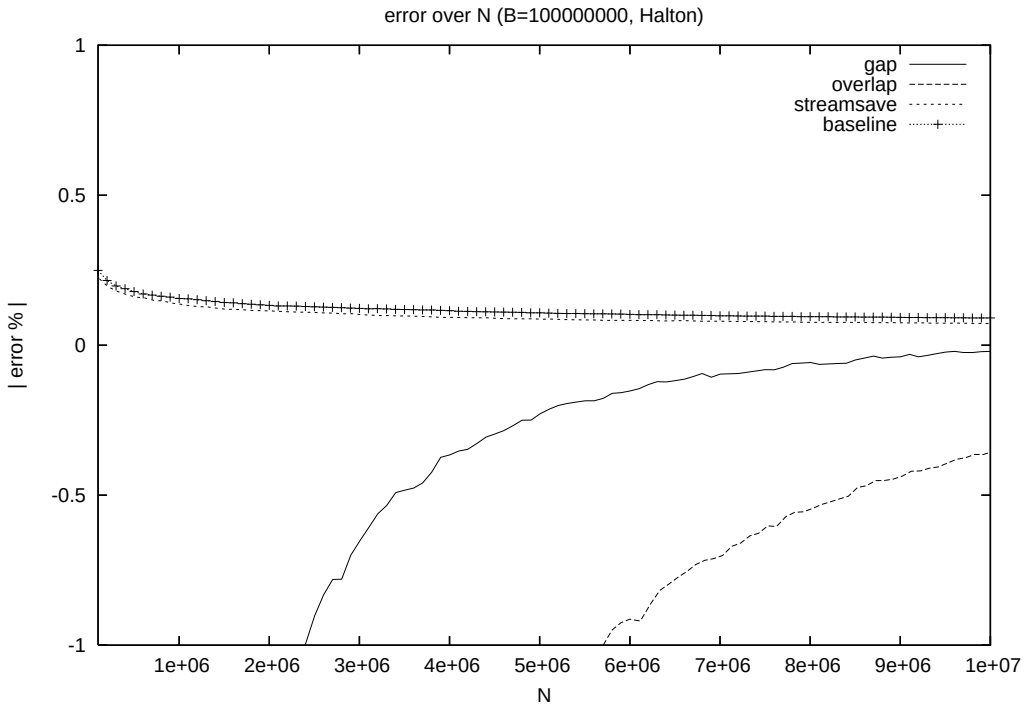

Figure 2. Comparison of Halton sequence regarding overlap, gaps and stream save.

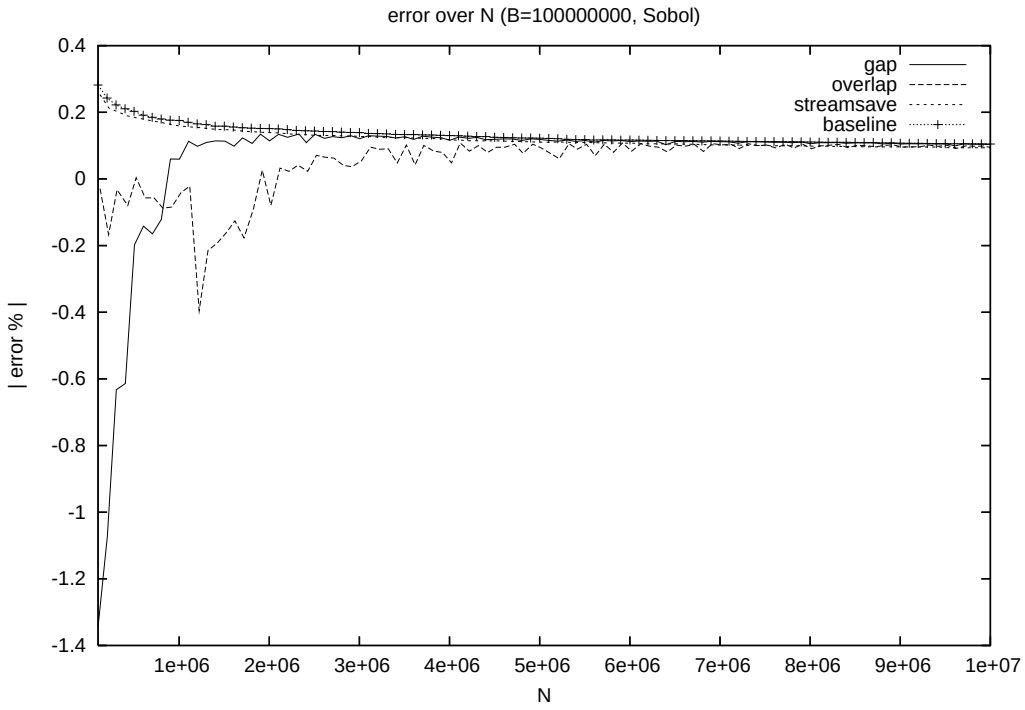

Figure 3. Comparison of Sobol' sequence regarding overlap, gaps and stream save.

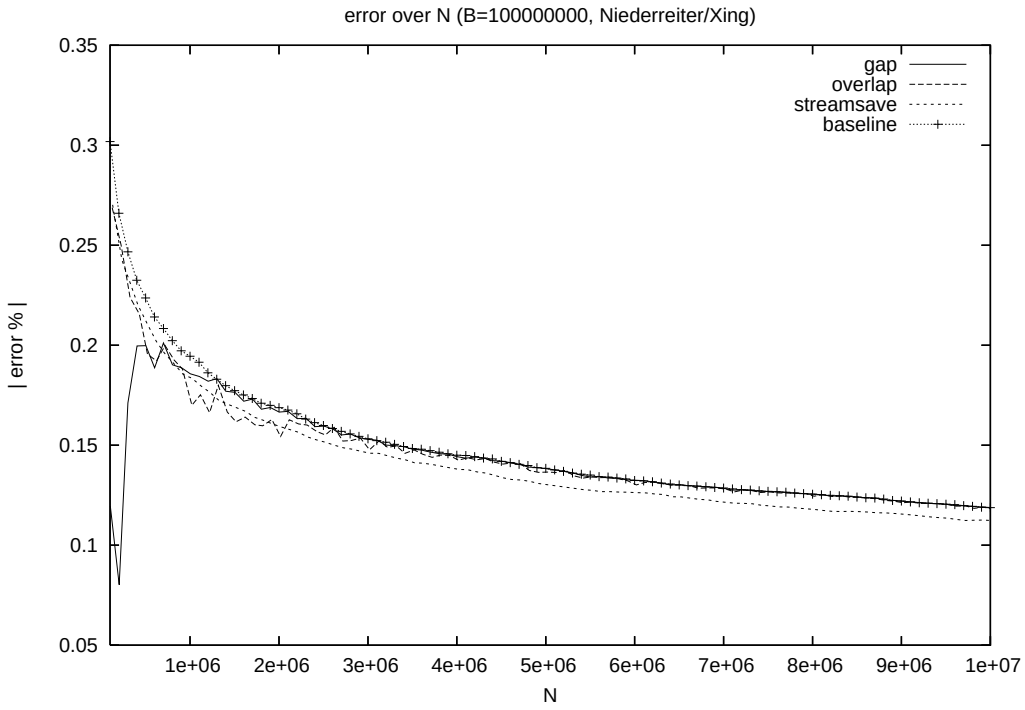

Figure 4. Comparison of Niederreiter/Xing sequence regarding overlap, gaps and stream save.

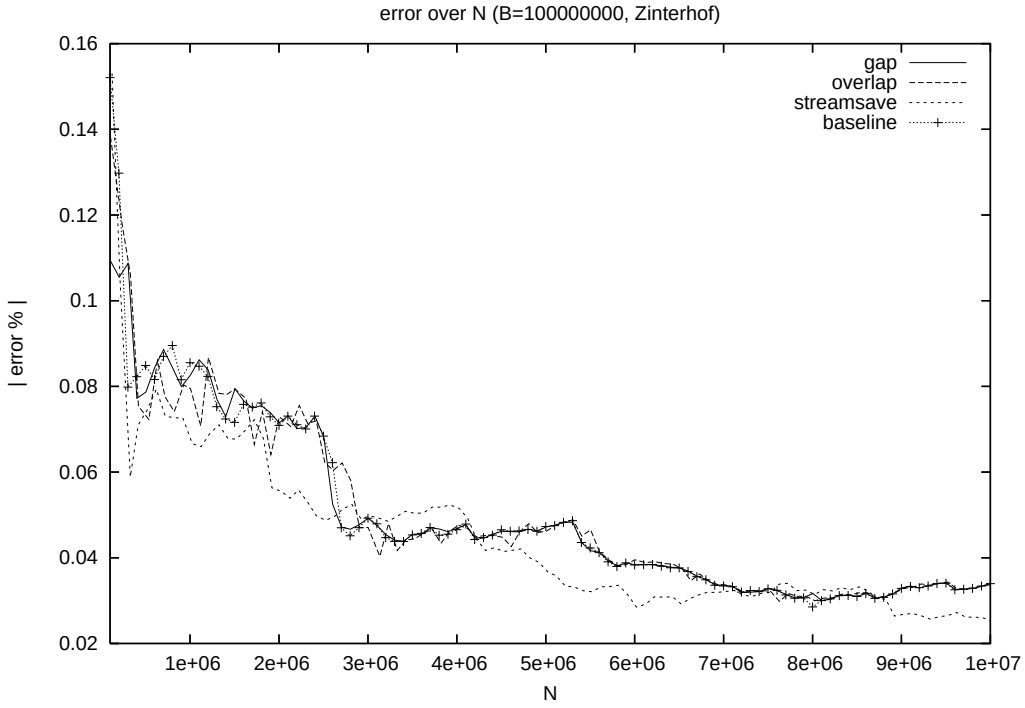

Figure 5. Comparison of Zinterhof sequence regarding overlap, gaps and stream save.

that Niederreiter/Xing-, Zinterhof- and Faure sequences behave basically the same as the baseline for high N while showing a bit of fluctuation for lower values of N. On the other hand, the Sobol function shows this behavior for all but the highest values of N where it finally becomes smooth like the baseline.

These erratic behavior can become problematic during actual integration, since in an application case the fluctuation of the result would be used as a stop criterion, i.e. if the result stabilizes the calculation is considered finished. The fluctuation as exhibited by the Sobol sequence can lead to an calculation of more points than are necessary, and thus more time consumption.

From these results big blocks should only be used when no gaps or overlaps occur or when there is a certainty that a high number of points N has to be used. Also even when there is a certainty for high N a knowledge of the point set is necessary to prevent usage of point sets which show effects like Halton for our test function.

5. Conclusion and Future Work

Small block sizes can achieve the same speedup as leaping when disregarding network load. Big blocks on the other hand have the potential of the same speedup and low network use as leaping. However, big blocks can lead to a higher number of processed points when the block size is not chosen carefully. In fact a good choice of block size for big block is nearly impossible since the amount of points N used for the integration is not known a priori, and even if it is known the acquisition of new nodes during runtime can result in gaps.

In the, rather fictional, case of synchronized streams for leaping the possibility of saving an amount of streams for the acquisition of new nodes is given.

Overall the usage of small blocks with a proper method of preventing delay due to a lack of points while requesting a new block is preferable, unless the streams for leaping are synchronized or the behavior of the point set concerning gaps or overlaps is known. When the properties of the sequences concerning gaps and overlaps are known to behave well, the Zinterhof and Niederreiter/Xing sequences seem to qualify, the authors would suggest that big blocks are the preferred method of parallelization. Additionally, blocking can inherently deal with the possibility for faulty or newly available machines during computation.

Acknowledgments

The work described in this paper is partially supported by the Austrian Grid Project, funded by the Austrian BMBWK (Federal Ministry for Education, Science and Culture) under contract GZ 4003/2-VI/4c/2004.

References

[1] V. Alexandrov, E. Atanassov, and I. Dimov. Parallel quasi Monte Carlo methods for linear algebra problems. *Monte Carlo Methods and Applications*, 10(3-4):213–219, 2004.

[2] B.C. Bromley. Quasirandom number generators for parallel Monte Carlo algorithms. *Journal of Parallel and Distributed Computing*, **38**:101–104, 1996.

[3] Andrew T. Clayman, K. Mark Lawrence, Gary L. Mullen, Harald Niederreiter, and N. J. A. Sloane. Updated tables of parameters of (t,m,s)-nets. *Journal of Combinatorial Designs*, 7:381–393, 1999.

[4] E. deDoncker, R. Zanny, and K. Kaugars. Distributed numerical integration algorithms and applications. In *Proceedings of the 4th World Multiconference on Systemics, Cybernetics, and Informatics (SCI'00)*, pages 244–249, 2000.

[5] K. Entacher, P. Hellekalek, and P. L'Ecuyer. Quasi-Monte Carlo node sets from linear congruential generators. In H. Niederreiter and J. Spanier, editors, *Monte Carlo and Quasi-Monte Carlo Methods 1998*, pages 188–198. Springer, 2000.

[6] K. Entacher, T. Schell, W. Ch. Schmid, and A. Uhl. Defects in parallel Monte Carlo and quasi-Monte Carlo integration using the leap-frog technique. *Parallel Algorithms and Applications*, 18(1–2):27–47, 2003.

[7] G. Evans. *Practical numerical integration*. Wiley, Chichester, 1993.

[8] Henry Faure. Discrépance de suites associées à un système de numération (en dimension s). *Acta Arithmetica*, 41:337–351, 1982.

[9] J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimension integrals. *Numer. Math.*, 2:84–90, 1960. Berichtigung, ibid., (1960), p. 196.

[10] H. Hofbauer, A. Uhl, and P. Zinterhof. Quasi monte carlo integration in grid environments: Further leaping effects. submitted to Parallel Processing Letters, 12 2005.

[11] A.R. Krommer and C.W. Überhuber. *Numerical Integration on Advanced Computer Systems*, volume 848 of *Lecture Notes in Computer Science*. Springer, Berlin, 1994.

[12] A.R. Krommer and C.W. Überhuber. *Computational integration*. SIAM, Philadelphia, 1998.

[13] J.X. Li and G.L. Mullen. Parallel computing of a quasi-Monte Carlo algorithm for valuing derivatives. *Parallel Computing*, 26(5):641–653, 2000.

[14] S. Li, K. Kaugars, and E. deDoncker. Grid-based numerical integration and visualization. In *Sixth International Conference on Computational Intelligence and Multimedia Applications (ICCIMA'05)*, pages 260–265. IEEE Computer Society Press, 2005.

[15] M. Mascagni and A. Karaivanova. A parallel quasi-Monte Carlo method for computing extremal eigenvalues. In K. T. Fang, F. J. Hickernell, and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 369–380. Springer-Verlag, 2002.

[16] M. Mascagni and A. Karaivanova. A parallel quasi-Monte Carlo method for solving systems of linear equations. In P. Sloot et al., editors, *The 2002 International Conference on Computational Science - ICCS 2002*, volume 2330 of *Lecture Notes in Computer Science*, pages 598–608. Springer Verlag, Berlin, Germany, May 2002.

[17] Gary L. Mullen, Arijit Mahalanabis, and Harald Niederreiter. Tables of (t,m,s)-net and (t,s)-sequence parameters. In Harald Niederreiter and P. J.-S. Shiue, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, volume Lecture Notes in Statistics of *106*, pages 58–86. Springer-Verlag, 1995.

[18] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Number 63 in CBMS-NSF Series in Applied Mathematics. SIAM, Philadelphia, 1992.

[19] Harald Niederreiter. Point sets and sequences with small discrepancy. *Monatshefte für Mathematik*, 104:273–337, 1987.

[20] Harald Niederreiter and Chaoping Xing. Low-discrepancy sequences and global function fields with many rational places. *Finite Fields and Their Applications*, 2:241–273, 1996.

[21] W. Ch. Schmid and A. Uhl. Parallel quasi-Monte Carlo integration using (t,s)-sequences. In P. Zinterhof, M. Vajtersic, and A. Uhl, editors, *Parallel Computation. Proceedings of ACPC'99*, volume 1557 of *Lecture Notes on Computer Science*, pages 96–106. Springer-Verlag, 1999.

[22] W. Ch. Schmid and A. Uhl. Techniques for parallel quasi-Monte Carlo integration with digital sequences and associated problems. *Mathematics and Computers in Simulation*, 55:249–257, 2001.

[23] R. Schürer. Parallel high-dimensional integration: quasi-Monte Carlo versus adaptive cubature rules. In V. N. Alexandrov, J. J. Dongarra, B. A. Juliano, R. S. Renner, and C. J. K. Tan, editors, *The 2001 International Conference on Computational Science - ICCS 2001*, volume 2073 of *Lecture Notes in Computer Science*, pages 1262–1271, San Francisco, CA, USA, May 2001. Springer Verlag, Berlin, Germany.

[24] R. Schürer and A. Uhl. An evaluation of adaptive numerical integration algorithms on parallel systems. *Parallel Algorithms and Applications*, 18(1–2):13–26, 2003.

[25] Rudol Schürer and Wolfgang Ch. Schmid. Mint - the database of optimal (t, m, s)-net parameters. online: `http://mint.sbg.ac.at/`.

[26] I.M. Sobol'. Calculation of improper integrals using uniformly distributed sequences. *Soviet Math Dokl.*, 14(3):734–738, July 1973.

[27] Ilya M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

[28] J.W.L. Wan, K. Lai, A.W. Kolkiewicz, and K.S. Tan. A parallel quasi Monte Carlo approach to pricing multidimensional americal options. *International Journal of High Performance Computing and Networking*, 2006. To appear.

[29] Chaoping Xing and Harald Niederreiter. A construction of low-discrepancy sequences using global function fields. *Acta Arithmetica*, 71(1):87–102, 1995.

[30] Peter Zinterhof. Einige zahlentheoretische methoden zur numerischen Quadratur und Interpolation. *Sitzungsberichte der Österreichischen Akademie der Wissenschaften, math.-nat.wiss. Klasse Abt. II*, 177:51–77, 1969.

[31] Peter Zinterhof. High dimensional improper integration procedures. In Civil Engineering Faculty Technical University of Košice, editor, *Proceedings of Contributions of the 7th International Scientific Conference*, pages 109–115, Hroncova 5, 04001 Košice, SLOVAKIA, May 2002. TULIP.