Parallel and Adaptive Numerical Integration on Hypercubes

Rudolf Schrer
The Problem

Given a function $f : C^s \rightarrow \mathbb{R}$, with $C^s$ denoting the unit hypercube $[0, 1]^s \subseteq \mathbb{R}^s$.

Calculate an approximation $Qf$ for

$$If := \int_{C^s} f(x) dx$$
The function $f$ is considered to be a *black box*, which means that

- $f(x)$ can be calculated for arbitrary $x \in C^s$ by an algorithm

However, we don’t assume any further knowledge about $f$, i. e. no knowledge about

- discontinuities
- derivations (existence, value)
- algebraic structure
- $f$ decomposing into functions of lower dimensionality
What happens, if the region of integration $B$ is not $C^s$?

• Transformation (requires a continuously differentiable, one-to-one transform $\phi$ from $\overline{B}$ to $B$)

$$
\int_B f(x)dx = \int_B \left| \text{det } J_\phi(x) \right| \cdot f(\phi(x))d\overline{x}
$$

• Characteristic function

$$
\int_B f(x)dx = \int_{C^s} c_B f(x)dx
$$

Problems:

• Using a characteristic function makes integrand discontinuous

• For high dimensions, there may be a huge difference in the volume
Basic principle for Numerical Integration:

\[ Q_n f = \sum_{i=1}^{n} w_i f(x_i) \]

- \(|Q_n f - I f| \to 0\) for \( n \to \infty \). Fast!
- All \( x_i \) should be inside \( C^s \)
- \( \sum_{i=1}^{n} |w_i| \) should be as small as possible

If some \( w_i \) or \( x_i \) depend on previously calculated \( f(x_j) \) with \( j < i \), the algorithm is called adaptive.
Two Problems

- Achieve maximum precision with a minimum of integrand evaluations

- Distribute integrand evaluations on multiple processing nodes
Content

- Quasi-Random numbers
- Multi-dimensional Cubature Rules
- Adaptive Algorithms using these Rules
- Testing
- Parallelization
Monte-Carlo approach

\[ Q_n f = \frac{1}{n} \sum_{i=1}^{n} f(x_i) \quad x_i \ldots \text{random points} \]

Does this work at all?  
*Strong law of large numbers* guarantees that this procedure converges almost surely for all bounded Lebesgue-integrable functions.

This approach seems naive at first glance, but is hard to beat in high dimensions! To be precise, it follows from the *central limit theorem* that the expected integration error is \( O\left(\frac{1}{\sqrt{n}}\right) \).

How do we get random numbers anyway?
What about a simple lattice?

I. M. Sobol (1979):

It is a common belief that the most uniformly distributed set of $n = m^s$ points in the $s$-dimensional unit cube is a rectangular lattice.

[...]

We conclude that the projections of a good set of points on every $s'$-dimensional face of $C^s$ must be good $s'$-dimensional sets ($1 \leq s' < s$).
Discrepancy of a point set

The star-discrepancy $D_n^*$ of $n$ points $x_1, \ldots, x_n$ is defined by

$$D_n^* = \sup_{E=[0,t_1] \times \ldots \times [0,t_s]} \left| \frac{\#(x_i \in E)}{n} - \text{Vol}(E) \right|$$

In 1961 E. Hlawka proofed the following theorem (Koksma-Hlawka inequality):

$$\left| \frac{1}{n} \sum_{i=1}^{n} f(x_i) - \int_{C_s} f(x) dx \right| \leq V(f) \cdot D_n^*$$

Strategy: Look for sequences with $O(D_n^*)$ small for large $n$. 
Low-Discrepancy Sequences

- 1960 Halton found a sequence with $D_n = O\left(\frac{\log^s n}{n}\right)$. However, the constant is very bad for large $s$.

- It is a widely held believed that this order of magnitude is best possible.

- 1967 Sobol published a sequence based on his theory on $LP_r$-sequences and $P_r$-nets.

- 1987 and 1988 Niederreiter published his sequence, based on $(t, s)$-sequences and $(t, m, s)$-nets. Constant decreases exponentially.

- In 1993 Tezuka generalized Niederreiter’s work, covering Niederreiter, Sobol, and other sequences including a new one invented by him, with even better theoretical properties.
Status of my implementation

- Monte-Carlo sequence using the built-in PRNG.

- Niederreiter (base 2 only) and Sobol sequence for arbitrary precision and dimension

- A small number-theoretical library to accomplish this without Mathematica or hard-coded tables

- Generating \((t, m, s)\)-nets from \((t, s)\)-sequences
What’s missing?

- Real random numbers for the Monte-Carlo sequence generator
- Faure, Halton, and some other sequences
- Generalized Niederreiter sequences with arbitrary base. Would require much work on a number-theoretical library. Practical results are questionable.
- Better algorithm for summing up samples.
Parallelization

• Trivial because it is easy to jump to arbitrary elements in the sequence.

• Implementation without load balancing:
  - Node $n$ processes the $n$th subsequence
  - There is only one Reduce for collecting data

• Implementation with load balancing:
  - Master hands out ranges of numbers to the other nodes for processing
  - If non-blocking calls are used, the master can perform work, too.
Cubature Rules

Take advantage of the smoothness of $f$!

- Every continuous function can be approximated by polynomials (Weierstrass, 1885)

- The integral of a polynomial can be calculated analytically

Gauss Formulas:

- $n$ sampling points for degree $2n - 1$

- All weights positive and all points inside

- Convergence guaranteed for all Riemann integrable functions
Embedded Rules

- Two rules $Q_n$ and $Q_m$ of different degree ($n > m$)

- Points of lower-degree rule $\subseteq$ Points of higher-degree rule
  So no extra function evaluation are done

- For “most” $f$,

$$|Q_n(f) - Q_m(f)| \geq |Q_n(f) - \int f|$$

which gives us an upper bound on the integration error.
This approach does not scale to multiple dimensions!

- Product rules have too many points ($n^s$ for a degree $2n - 1$ Product Gauss Rule)
- Building simple rules is guessing
- Rules for arbitrary dimension are rare
- Problems with complex weights, nodes outside the region...
- Different rules required for different regions
- Convergence for all continuous functions is not guaranteed
- A pain to implement
Implemented Rules

- Midpoint (deg 1, \( n = 1, \ |w_i| = 1, \) perfect!)

- Octahedron
  (deg 3, \( n = 2s, \ |w_i| = 1, \) perfect!)

- Hammer, Stroud 58 (deg 5, \( n = 2s^2 + 1 \))

- Stroud 68 (deg 5, \( n = 3s^2 + 3s + 1 \))

- Genz 80: (deg 7, \( n = O(2^s), \ |w_i| = O(s) \))

- Dobrodeev 70
  (deg 7, \( n = O(s^3), \ |w_i| = O(s^2) \))
Remarks

- Degree is not the most important thing (Genz does much better than Dobrodeev)

- Genz is $O(2^s)$, but for $s \leq 10$ it uses less points than Dobrodeev

- Genz is already an embedded Rule, while the other rules have to be combined to get an embedded rule

Further plans:

- Optimize Embedded rules. 4th difference could be integrated into rules.

- There are some additional rules I want to try: A degree 9 rule from Stenger63, Genz83 / Cools94, Genz84, Keast79.
Adaptive Algorithms

Cubature Rules provide an error estimation. This allows an adaptive algorithm:

1. Calculate estimation for result and error for the whole region. Store region in collection.

2. Take region with highest error from collection

3. Split into subregions / Choose better rule

4. Calculate estimation for result and error for these subregions

5. Store them in the collection

6. Goto 2
Adaptive Algorithms (Variations)

- Termination? Check #Function evaluations, absolute error, relative error,...

- Don’t store regions with small error

- Splitting

- Use low-degree rule to adapt; use high-degree rule to calculate final result
Test functions

Should have the following properties (Genz84):

- A particular attribute
- Analytically computable to high precision
- Fast to evaluate
- Clearly identifiable affective and unaffec-
tive parameters
- Do testing by collecting statistical data

Many functions presented in papers do not fit into this scheme! :-(

22
Implemented test functions

- I’ve implemented many functions from QRN-papers. However, most of them are inadequate to test adaptive cubature rule algorithms. The remaining functions have to be parameterized.

- Only Genz Function 1 and 5 are implemented. Still working on 2, 3, 4, and 6...

- Other interesting functions are always welcome! :-)

Genz Function 1 (Oscillatory)

\[ f(x) = \cos \left( 2\pi u_1 + \sum_{i=1}^{s} a_i x_i \right) \]
Genz Function 5 ($C_0$ Function)

\[ f(x) = e^{-\sum_{i=1}^{s} a_i|x_i-u_i|} \]

### Graphs

- **$C_0, s=2$**
- **$C_0, s=5$**
- **$C_0, s=10$**
- **$C_0, s=20$**

Each graph plots the number of correct digits against the number of integrand evaluations for different methods: Genz, Monte Carlo, Niederreiter, Sobol.
Parallel Adaptive Algorithms

- Master-Slave
- One node responsible for region management
- Master does not work
- Bottle-neck
- Startup
- Inferior to sequential algorithm
Decentralized design

- No dedicated master

- Every node maintains its own region collection

- Nodes talk to its “neighbors” to exchange “bad” regions

- Different topologies

- Problem: Global error estimates? Termination?

- Performance different than sequential algorithm