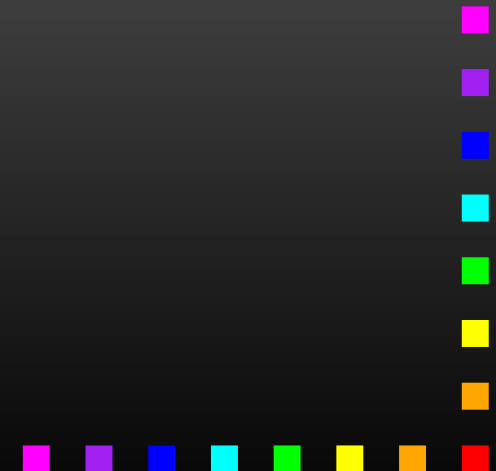


# Numerical Integration

Thomas Hahn

Max-Planck-Institut für Physik  
München



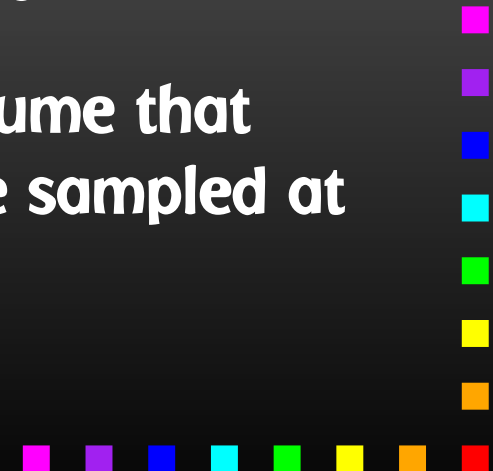
# Restrictions

Integration is a wide field. We will concentrate here on Riemann integrals of the form

$$If := \int_0^1 d^d x f(\vec{x}) w(\vec{x}).$$

The **Weight Function**  $w(\vec{x})$  is generally known analytically and is used to **absorb characteristics of the integrand** which are difficult to treat otherwise, e.g. peaks or oscillatory behaviour.

For the purposes of numerical integration, we assume that  $f(\vec{x})$  is given as a function/subroutine that can be sampled at arbitrary points  $\vec{x}_i \in [0, 1]^d$ .



# Quadrature Formulas

Task: Find a **Quadrature Formula**

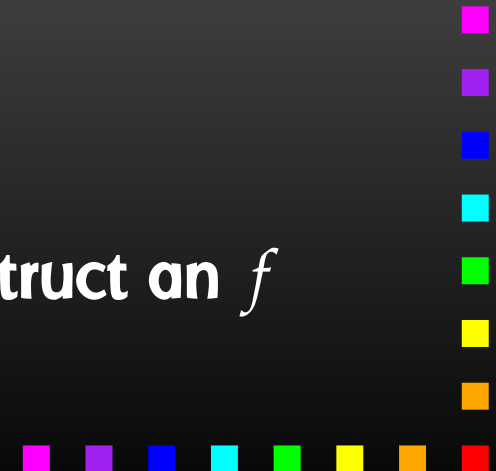
$$Q_n f := \sum_{i=1}^n w_i f(\vec{x}_i)$$

with **Nodes (sampling points)**  $\vec{x}_i$  and **Weights**  $w_i$ .

$Q_n f$  should approximate  $I f$  for a large class of functions with as small an **Error**  $E_n f$  as possible:

$$I f = Q_n f + E_n f, \quad E_n f \text{ "small."}$$

**But:** For a given  $Q_n$ , it is always possible to construct an  $f$  such that  $E_n f$  becomes arbitrarily large!



# Terminology

A **Quadrature Rule** is the vector of nodes and weights,

$$R_n = \left\{ \begin{pmatrix} \vec{x}_1 \\ w_1 \end{pmatrix}, \dots, \begin{pmatrix} \vec{x}_n \\ w_n \end{pmatrix} \right\}$$

and of course completely specifies  $Q_n$ .

- **Open Rule** = all  $\vec{x}_i$  lie strictly inside the unit hypercube.
- **Closed Rule** = some  $\vec{x}_i$  lie on the boundary.
- **Positive Rule** = all  $w_i$  are positive.

(Weights with alternating sign are generally considered inferior because numerical cancellations may occur when adding up.)



# Interpolatory Formulas (1D)

Idea: **Approximate**  $f$  by a polynomial  $p^{(n-1)}$  and integrate the latter. Works as far as  $f$  is well approximated by polynomials.

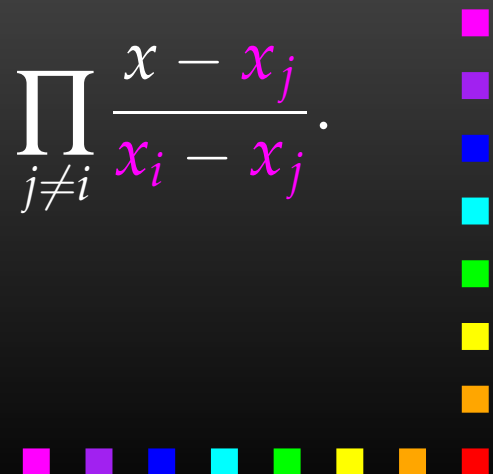
We impose that  $p^{(n-1)}$  **interpolates**  $f$  at  $n$  given points  $x_i$ :

$$p^{(n-1)}(x_i) \stackrel{!}{=} f(x_i), \quad i = 1, \dots, n.$$

The polynomial thus specified is unique and can explicitly be given in terms of **Lagrange Polynomials**  $\ell_{n-1,i}$ :

$$p^{(n-1)}(x) = \sum_{i=1}^n f(x_i) \ell_i^{(n-1)}(x), \quad \text{where} \quad \ell_i^{(n-1)}(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}.$$

By construction,  $\ell_i^{(n-1)}(x_j) = \delta_{ij}$ .



# Interpolatory Formulas (1D)

Weights:

$$\begin{aligned} \mathbf{I} p^{(n-1)} &= \int_0^1 dx w(x) p^{(n-1)}(x) \\ &= \sum_{i=1}^n f(x_i) \underbrace{\int_0^1 dx w(x) \ell_i^{(n-1)}(x)}_{w_i}. \end{aligned}$$

From the practical point of view, the weight function  $w(x)$  must be chosen such that these integrals can be computed.

The **Degree of  $Q_n$**  is the degree of the highest polynomial integrated exactly by  $Q_n$ .



# Newton-Cotes (Rectangle) Rules (1D)

The simplest case: **take equidistant**  $x_i$ .

For  $w(x) = 1$  the lowest-order rules are:

**Open rules:**  $x_i = \frac{i}{n+1}$ , e.g.

$$Q_1 f = f\left(\frac{1}{2}\right),$$

$$Q_2 f = \frac{1}{2} \left( f\left(\frac{1}{3}\right) + f\left(\frac{2}{3}\right) \right),$$

$$Q_3 f = \frac{1}{3} \left( 2f\left(\frac{1}{4}\right) - f\left(\frac{1}{2}\right) + 2f\left(\frac{3}{4}\right) \right).$$

**Closed rules:**  $x_i = \frac{i-1}{n-1}$ , e.g.

$$Q_2 f = \frac{1}{2} \left( f(0) + f(1) \right),$$

$$Q_3 f = \frac{1}{6} \left( f(0) + 4f\left(\frac{1}{2}\right) + f(1) \right),$$

$$Q_4 f = \frac{1}{8} \left( f(0) + 3f\left(\frac{1}{3}\right) + 3f\left(\frac{2}{3}\right) + f(1) \right).$$

By construction,  $\deg Q_n \geq n - 1$ , but cannot generally be expected to be larger than  $n - 1$  because the  $x_i$  are prescribed and only the  $n$   $w_i$  have been determined.

The Newton-Cotes rules by themselves are not nearly as powerful as the Gauss rules, but they have the advantage that they can be **compounded** easily, e.g. in **Romberg Rules**.



## Gauss Rules (1D)

Choose the  $x_i$  such that  $Q_n$  has the **highest possible degree**.  
With  $2n$  degrees of freedom ( $n x_i$  and  $n w_i$ ), we ought to achieve  $\deg Q_n = 2n - 1$ .

By Euclid's GCD algorithm we write

$$p^{(2n-1)}(x) = q^{(n)}(x) r^{(n-1)}(x) + s^{(n-1)}(x), \quad q^{(n)}(x) = \prod_{i=1}^n (x - x_i)$$

$$\begin{aligned} \mathbb{I} p^{(2n-1)} &= \boxed{\int_0^1 dx w(x) q^{(n)}(x) r^{(n-1)}(x)} + \int_0^1 dx w(x) s^{(n-1)}(x) \\ &= \sum_{i=1}^n w_i \underbrace{q^{(n)}(x_i)}_{=0} r^{(n-1)}(x_i) + \sum_{i=1}^n w_i s^{(n-1)}(x_i) + \mathbf{E}_n p^{(2n-1)} \end{aligned}$$

If the boxed term were zero, the error term would vanish!





# Gauss Rules (1D)

$$\int_0^1 dx w(x) q^{(n)}(x) r^{(n-1)}(x) = \langle q^{(n)} | r^{(n-1)} \rangle = 0 \quad \Leftrightarrow \quad q^{(n)} \perp r^{(n-1)}$$

Choose **Orthogonal Polynomials** for the  $q^{(n)}$ , then

$$r^{(n-1)} = \sum_{i=0}^{n-1} \langle q^{(i)} | r^{(n-1)} \rangle q^{(i)} \quad \text{and} \quad \langle q^{(n)} | r^{(n-1)} \rangle = 0$$

because  $\langle q^{(n)} | q^{(i < n)} \rangle = 0$ .

Since  $q^{(n)}(x) = \prod_{i=1}^n (x - x_i)$ , the  $x_i$  are just the **zeros of the orthogonal polynomials  $q^{(n)}$ !**



# Gauss Rules (1D)

Gauss rules for particular weight functions have special names, hinting at the orthogonal polynomials used:

$$w(x) = 1$$

**Gauss-Legendre Rules**

$$w(x) = 1/\sqrt{1-x^2}$$

**Gauss-Chebyshev Rules**

$$w(x) = x^\alpha e^{-x}$$

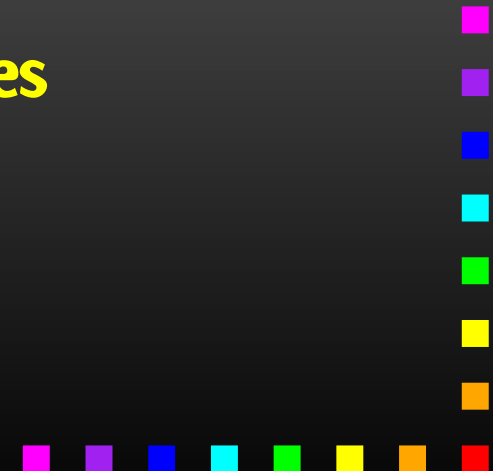
**Gauss-Laguerre Rules**

$$w(x) = e^{-x^2}$$

**Gauss-Hermite Rules**

$$w(x) = (1-x)^\alpha (1+x)^\beta$$

**Gauss-Jacobi Rules**



## Error Estimate and Embedded Rules (1D)

So far we have **no error estimate** for our integration rule.

Idea: **Compare the results** of two rules,  $Q_n$  and  $Q_{n+m}$ .

Use **Embedded Rules** with  $\{x_i\}_{i=1}^n \subset \{x_i\}_{i=1}^{n+m}$  for economy.

But: e.g. Gauss rules of different  $n$  have no common  $x_i$ .

Seek to add new points  $x_{n+1}, \dots, x_{n+m}$  such that  $Q_{n+m}$  is exact for polynomial of maximum degree expectable from #df,  $p^{(n+2m-1)}$ . This leads to

$$\int_0^1 dx w(x) v^{(n)}(x) q^{(m)}(x) r^{(m-1)}(x) = 0$$

with  $v^{(n)}(x) = \prod_{i=1}^n (x - x_i)$ ,  $q^{(m)}(x) = \prod_{i=1}^m (x - x_{n+i})$ .

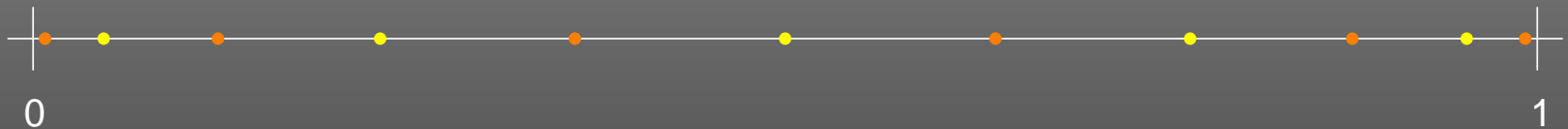
**Not all  $w(x)$  have solutions since a scalar product with weight function  $w(x) v^{(n)}(x)$  cannot in general be defined.**



# Embedded Rules and Null Rules

Some common **Embedded Rules** in  $d = 1$  are:

- **Gauss-Kronrod Rules** add one point between every two points of a Gauss rule  $R_n$  so that  $R_n \subset R_{n+n+1}$ , e.g.



- **Patterson Rules** add points on top of the Gauss-Kronrod rules, e.g. there exists a set of rules  $R_1 \subset R_3 \subset R_5 \subset R_7 \subset R_{15} \subset R_{31} \subset R_{63} \subset R_{127} \subset R_{255}$ .

A **Null Rule**  $N_m$  is associated with an integration rule  $Q_n$  (usually  $m < n$ ) and is engineered to give **zero for all functions that are integrated exactly** by  $Q_n$ .

$N_m$  measures the “higher terms” of the integrand that are not exactly integrated by  $Q_n$  and is also used for error estimation.



# Adaptive Algorithms

With an error estimate available, adaptiveness can easily be implemented:

1. **Integrate the entire region:**  $I_{\text{tot}} \pm E_{\text{tot}}$ .
2. **while**  $E_{\text{tot}} > \max(\varepsilon_{\text{rel}} I_{\text{tot}}, \varepsilon_{\text{abs}})$
3.     **Find the region**  $r$  **with the largest error.**
4.     **Bisect (or otherwise cut up)**  $r$ .
5.     **Integrate each subregion of**  $r$  **separately.**
6.      $I_{\text{tot}} = \sum I_i, E_{\text{tot}} = \sqrt{\sum E_i^2}$ .
7. **end while**

**Examples: QUADPACK's QAG, CUBA's Cuhre, Suave.**



# Extrapolation

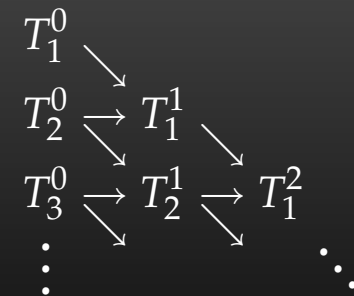
**Extrapolation** can be used to **accelerate convergence** if the functional behaviour of the error term is known.

Example: for the Newton-Cotes formulas the error can be shown to vary with the spacing  $h$  of the nodes as

$$Q_n f - I f = c_2 h^2 + c_4 h^4 + \dots, \quad h = \frac{1}{n}.$$

Use **Richardson's Extrapolation** to eliminate  $k$  powers of  $h$ :

$$T_m^k := \frac{4^k T_{m+1}^{k-1} - T_m^{k-1}}{4^k - 1}, \quad T_m^0 := Q_{2^{m-1}} f.$$

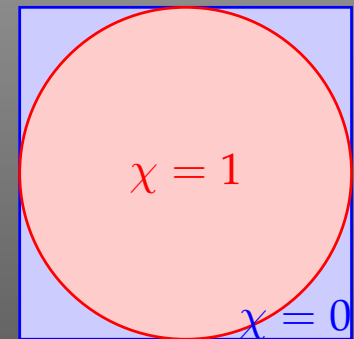


These are known as **Romberg Formulas**.



# Curse of Dimension

Imagine computing the volume of the  $d$ -dim. sphere  $S_d$  by integrating its characteristic function  $\chi = \theta(1 - \|x\|_2)$  inside the surrounding hypercube  $C_d = [-1, 1]^d$ .



The following table gives the ratio of the volumes:

$d$	2	5	10	50	100
$\frac{\text{Vol } S_d}{\text{Vol } C_d}$	.785	.164	.0025	$1.5 \times 10^{-28}$	$1.9 \times 10^{-70}$

This ratio can in a sense be thought of as the **chance that a general-purpose integrator will find the sphere at all!**

# Product Formulas

Easiest method: **Iterate 1-dimensional rules**, e.g.

$$\int_0^1 d^3x f(\vec{x}) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \sum_{k=1}^{n_z} w_i^{(x)} w_j^{(y)} w_k^{(z)} f(x_i^{(x)}, x_j^{(y)}, x_k^{(z)}).$$

But: the **number of samples increases as**  $N = \prod_{i=1}^d n_i \sim n^d$ .

Consider e.g. the Newton-Cotes rules, where the error term is  $\mathcal{O}(h^2) = \mathcal{O}(n^{-2})$ . Convergence is thus:

$$Q_n f - I f = \mathcal{O}(N^{-2/d}).$$

Even for moderate dimensions (say  $d \gtrsim 5$ ), this convergence rate is usually **too slow to be useful**.





# Construction of Polynomial Rules

Select **orthogonal basis of functions**  $\{b_1, \dots, b_m\}$  (usually monomials) with which most  $f$  can (hopefully) be approximated sufficiently and impose that **each  $b_i$  be integrated exactly by  $Q_n$** :

$$\mathbb{I} b_i \stackrel{!}{=} Q_n b_i \quad \Leftrightarrow \quad \sum_{k=1}^n w_k b_i(\vec{x}_k) = \int_0^1 d^d x w(\vec{x}) b_i(\vec{x}).$$

These are  **$m$  Moment Equations** for  $nd + n$  unknowns  $\vec{x}_i, w_i$ , and a formidable, in general nonlinear, system of equations.

Additional assumptions (e.g. **Symmetries**) are often necessary to solve this system. If a unique solution exists,  $Q_n$  is an **Interpolatory Rule**.

**Example: the Genz-Malik rules used in CUBA's Cuhre.**



# Monte Carlo Methods

Idea: Interpret  $f$  as a **Random Variable** and estimate  $\mathbf{I}f$  by the **Statistical Average** over independent, identically distributed samples  $\vec{x}_i \in [0, 1]^d$

$$\mathbf{I}f \approx \mathbf{M}_n f = \frac{1}{n} \sum_{i=1}^n f(\vec{x}_i) \quad (w(\vec{x}) = 1 \text{ here}).$$

The **Standard Deviation** is a probabilistic estimate of the integration error:

$$\sigma(\mathbf{M}_n f) = \sqrt{\mathbf{M}_n(f^2) - (\mathbf{M}_n f)^2}.$$

From  $\sigma(\mathbf{M}_n f) = \frac{\sigma(f)}{\sqrt{n}}$ , convergence is  $\mathbf{M}_n f - \mathbf{I}f = \mathcal{O}(n^{-1/2})$ .

**Not particularly fast, but independent of the dimension  $d$ !**

# Variance Reduction

**Variance Reduction** = Methods for accelerating convergence.

In **Importance Sampling** one introduces a weight function:

$$If = \int_0^1 d^d x w(\vec{x}) \frac{f(\vec{x})}{w(\vec{x})}, \quad w(\vec{x}) > 0, \quad \int w = 1.$$

- One must be able to sample from the distribution  $w(\vec{x})$ ,
- $f/w$  should be “smooth,” such that  $\sigma_w(f/w) < \sigma(f)$ ,  
e.g.  $w$  and  $f$  should have the same peak structure.

The ideal choice is  $w(\vec{x}) = |f(\vec{x})|/If$  which has  $\sigma_w(f/w) = 0$ .

**Example: Vegas uses a piecewise constant weight function which is successively refined, thus coming closer to  $|f(\vec{x})|/If$ .**

# Variance Reduction

**Stratified Sampling** works by sampling subregions. Consider:

	$n$ samples in total region $r_a + r_b$	$n_a = n/2$ samples in $r_a$ , $n_b = n/2$ samples in $r_b$
<b>Integral</b>	$I f \approx \mathbf{M}_n f$	$I f \approx \frac{1}{2} (\mathbf{M}_{n/2}^a f + \mathbf{M}_{n/2}^b f)$
<b>Variance</b>	$\frac{\sigma^2 f}{n}$ $= \frac{1}{2n} (\sigma_a^2 f + \sigma_b^2 f) + \frac{1}{4n} (\mathbf{I}_a f - \mathbf{I}_b f)^2$	$\frac{1}{4} \left( \frac{\sigma_a^2 f}{n/2} + \frac{\sigma_b^2 f}{n/2} \right)$ $= \frac{1}{2n} (\sigma_a^2 f + \sigma_b^2 f)$

The optimal reduction of variance is for  $n_a/n_b = \sigma_a f / \sigma_b f$ .

But: splitting each dimension causes a  $2^d$  increase in regions!

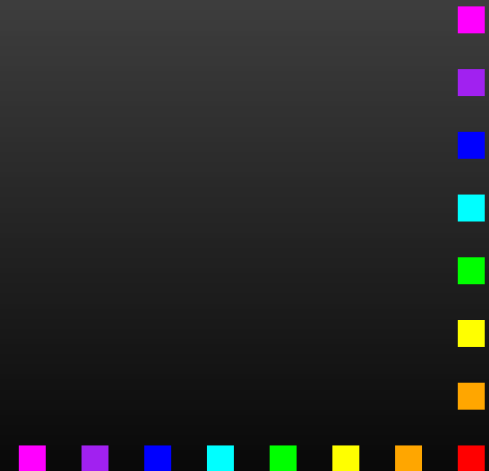
Example: Miser uses Recursive Stratified Sampling.



# Variance Reduction

Importance Sampling and Stratified Sampling are complementary:

- **Importance Sampling** puts most points where the magnitude of the integrand  $|f|$  is largest,
- **Stratified Sampling** puts most points where the variance of  $f$  is largest.



# Number-Theoretic Methods

The basis for the number-theoretical formulas is the **Koksma-Hlawka Inequality**:

The error of every  $Q_n f = \frac{1}{n} \sum_{i=1}^n f(\vec{x}_i)$  is bounded by

$$|Q_n f - I f| \leq V(f) D^*(\vec{x}_1, \dots, \vec{x}_n).$$

where  $V$  is the “**Variation in the sense of Hardy and Krause**” and  $D^*$  is the **Discrepancy** of the sequence  $\vec{x}_1, \dots, \vec{x}_n$ ,

$$D^*(\vec{x}_1, \dots, \vec{x}_n) = \sup_{r \in [0,1]^d} \left| \frac{\nu(r)}{n} - \text{Vol } r \right|,$$

where  $\nu(r)$  counts the  $\vec{x}_i$  that fall into  $r$ . For an **Equidistributed Sequence**,  $\nu(r)$  should be proportional to  $\text{Vol } r$ .



# Low-Discrepancy Sequences and Quasi-Monte Carlo

Cannot do much about  $V(f)$ , but can sample with **Low-Discrepancy Sequences** a.k.a. **Quasi-Random Numbers** which have discrepancies significantly below the pseudo-random numbers used in ordinary Monte Carlo, e.g.

- **Halton Sequences,**
- **Sobol Sequences,**
- **Faure Sequences.**

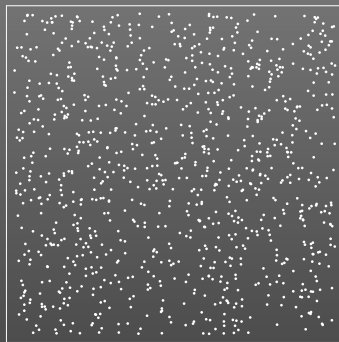
These **Quasi-Monte Carlo Methods** typically achieve convergence rates of  $\mathcal{O}(\log^{d-1} n/n)$  which are much better than the  $\mathcal{O}(1/\sqrt{n})$  of **ordinary Monte Carlo**.

**Example: CUBA's Vegas and Suave use Sobol sequences.**

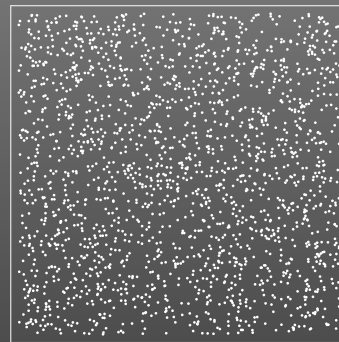


# Comparison of Sequences

## Mersenne Twister Pseudo-Random Numbers



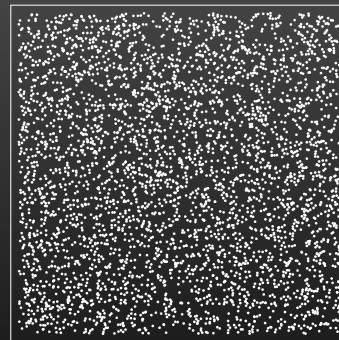
n = 1000



n = 2000

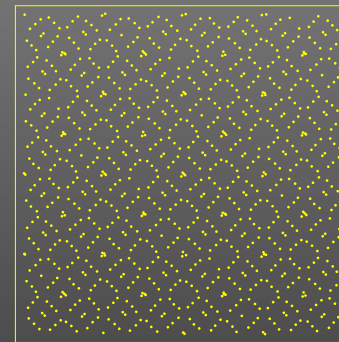


n = 3000

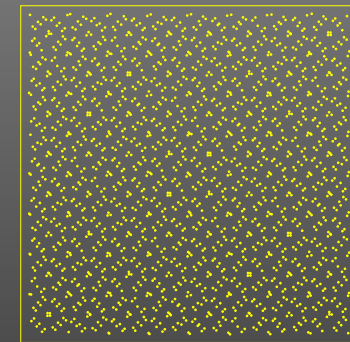


n = 4000

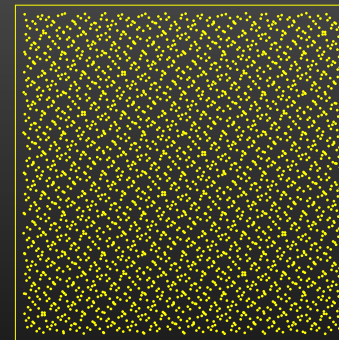
## Sobol Quasi-Random Numbers



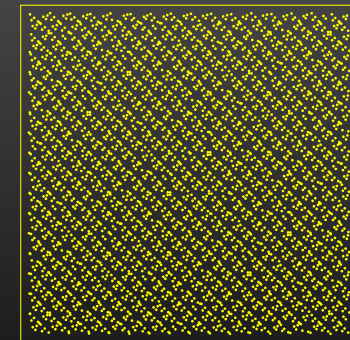
n = 1000



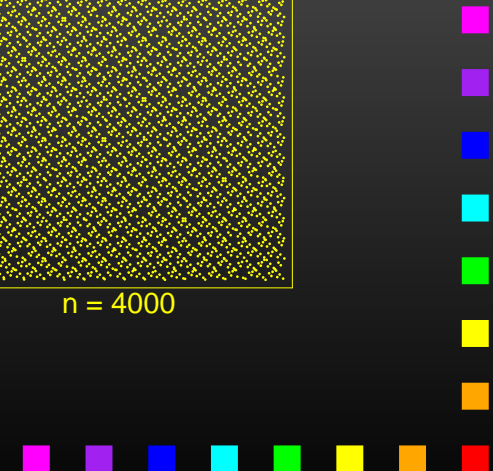
n = 2000



n = 3000



n = 4000





# Lattice Methods

**Lattice Methods** require a **periodic integrand**, usually obtained by applying a **Periodizing Transformation** (e.g.  $x \rightarrow 3x^2 - 2x^3$ ). Sampling is done on an **Integration Lattice**  $L$  spanned by a carefully selected integer vector  $\vec{z}$ :

$$Q_n f = \frac{1}{n} \sum_{i=0}^{n-1} f\left(\left\{\frac{i}{n}\vec{z}\right\}\right), \quad \{x\} = \text{fractional part of } x.$$

Construction principle for  $\vec{z}$ : knock out as many low-order **“Bragg reflections”** as possible in the error term:

$$Q_n f - \mathbf{I}f = \sum_{\vec{k} \in \mathbb{Z}^d} \tilde{f}(\vec{k}) Q_n e^{2\pi i \vec{k} \cdot \vec{x}} - \tilde{f}(\vec{0}) = \sum_{\vec{k} \in L^\perp, \vec{k} \neq \vec{0}} \tilde{f}(\vec{k}),$$

where  $L^\perp = \{\vec{k} \in \mathbb{Z}^d : \vec{k} \cdot \vec{z} = 0 \pmod{n}\}$  is the **Reciprocal Lattice**. Method: extensive computer searches.



# Summary

- **Univariate Integration** is pretty straightforward. **Gauss or Gauss-Kronrod/Patterson** are the rules of choice, often in an **adaptive algorithm**.  
There are, of course, many **special routines**, e.g. for oscillatory or singular integrands.
- **Multivariate Integration** is not at all straightforward. Most multivariate integrators use a more or less empirical hodge-podge of **Quadrature Rules, Monte Carlo, quasi-Monte Carlo, and Lattice Techniques**, garnished with one or more **Variance Reduction Methods**.  
There are several “good” algorithms on the market, all of which have their PROS and CONS, but there is in general no “best” algorithm.



# Software

- **QUADPACK**, <http://www.netlib.org/quadpack>  
**Fortran**. Standard for 1D problems. Well tested. Well documented in QUADPACK book (Springer), but book out of print, README file.
- **NAG (Numerical Algorithms Group)**, <http://www.nag.co.uk>  
**Fortran and C**. Commercial, well documented and tested.
- **IMSL**, <http://www.imsl.com>  
(Intl. Math and Statistical Libraries, now Visual Numerics)  
**Fortran, C, C#, Java**. Commercial, well documented and tested.
- **CERNLIB**, <http://wwwinfo.cern.ch/asd/>  
**Fortran**. Somewhat aged, well documented.
- **CUBA**, <http://www.feynarts.de/cuba>, [hep-ph/0404043](http://hep-ph/0404043)  
**Fortran, C/C++, Mathematica**. Quite new, well documented.



## Books

- A.R. Krommer, C.W. Ueberhuber  
**Computational Integration**  
SIAM Press, 1998.
- P.J. Davis, P. Rabinowitz  
**Methods of Numerical Integration**, Second Edition  
Academic Press, 1984.
- M. Evans, T. Swartz  
**Approximating Integrals via Monte Carlo and  
Deterministic Methods**  
Oxford University Press 2000.
- W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery  
**Numerical Recipes**, Second Edition  
Cambridge University Press, 1992.

