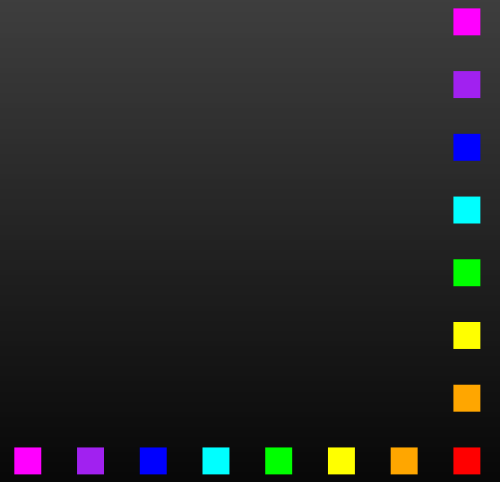
The CUBA Library

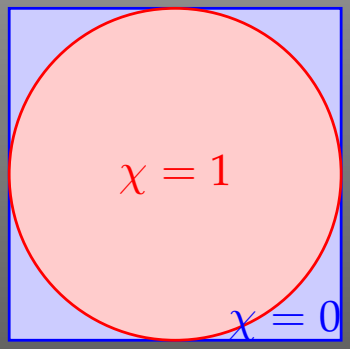
Thomas Hahn

Max-Planck-Institut für Physik
München



Why is multidimensional integration difficult?

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
$\frac{\mathrm{Vol}\, S_d}{\mathrm{Vol}\, C_d}$	.785	.164	.0025	1.5×10^{-28}	1.9×10^{-70}

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

Integrals

CUBA can do only Riemann integrals of the form

$$\mathbf{I}f := \int_0^1 \mathrm{d}^d x\, f(\vec{x})$$

where it is assumed that $f(\vec{x})$ is given as a function or subroutine that can be sampled at arbitrary points $\vec{x}_i \in [0,1]^d$.

This is not a serious restriction since most integrands can easily be transformed to the unit hypercube:

$$\int_{a_1}^{b_1} \cdots \int_{a_d}^{b_d} \mathrm{d}^d x\, f(\vec{x}) = \int_0^1 \mathrm{d}^d y\, f(\vec{x}) \prod_{i=1}^{d} (b_i - a_i)\,,$$

$$\text{where} \quad x_i = a_i + (b_i - a_i) y_i\,.$$

Overview of the CUBA Routines

Routine	Basic method	Type	Variance reduction
Vegas	Sobol sample *or* MT sample	quasi MC pseudo MC	importance sampling
Suave	Sobol sample *or* MT sample	quasi MC pseudo MC	globally adaptive subdivision + importance sampling
Divonne	Korobov sample *or* Sobol sample *or* MT sample *or* cubature rules	lattice MC quasi MC pseudo MC deterministic	stratified sampling, aided by methods from numerical optimization
Cuhre	cubature rules	deterministic	globally adaptive subdivision

- Very similar invocation (easily interchangeable)
- Fortran, C/C++, Mathematica interface provided
- Can integrate vector integrands

Deterministic vs. Monte Carlo Methods

Deterministic

Use a **Quadrature Formula**

$$\mathrm{I}f \approx \mathrm{Q}_n f := \sum_{i=1}^{n} w_i f(\vec{x}_i)$$

with specially chosen **Nodes** $\vec{x}_i$ and **Weights** w_i.

Error estimation e.g. by **Null Rules** N_m which give zero for functions Q_n integrates exactly and thus measure errors due to "higher terms."

Monte Carlo

Take the **Statistical Average** over random samples $\vec{x}_i$

$$\mathrm{I}f \approx \mathrm{M}_n f := \frac{1}{n} \sum_{i=1}^{n} f(\vec{x}_i)\,.$$

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

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

Construction of Polynomial Rules

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

$$\mathrm{I}\, b_i \stackrel{!}{=} \mathrm{Q}_n b_i \quad \Leftrightarrow \quad \sum_{k=1}^{n} w_k b_i(\vec{x}_k) = \int_0^1 \mathrm{d}^d x\, b_i(\vec{x})\,.$$

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

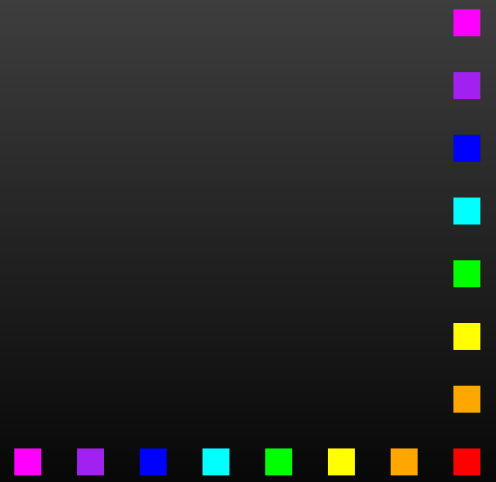
Additional assumptions (e.g. Symmetries) are often necessary to solve this system.

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

Globally Adaptive Subdivision

If an error estimate is available, global adaptiveness is easy to implement:

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



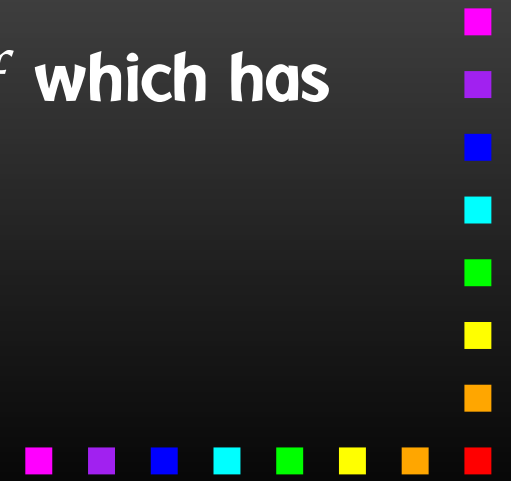
Importance Sampling

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

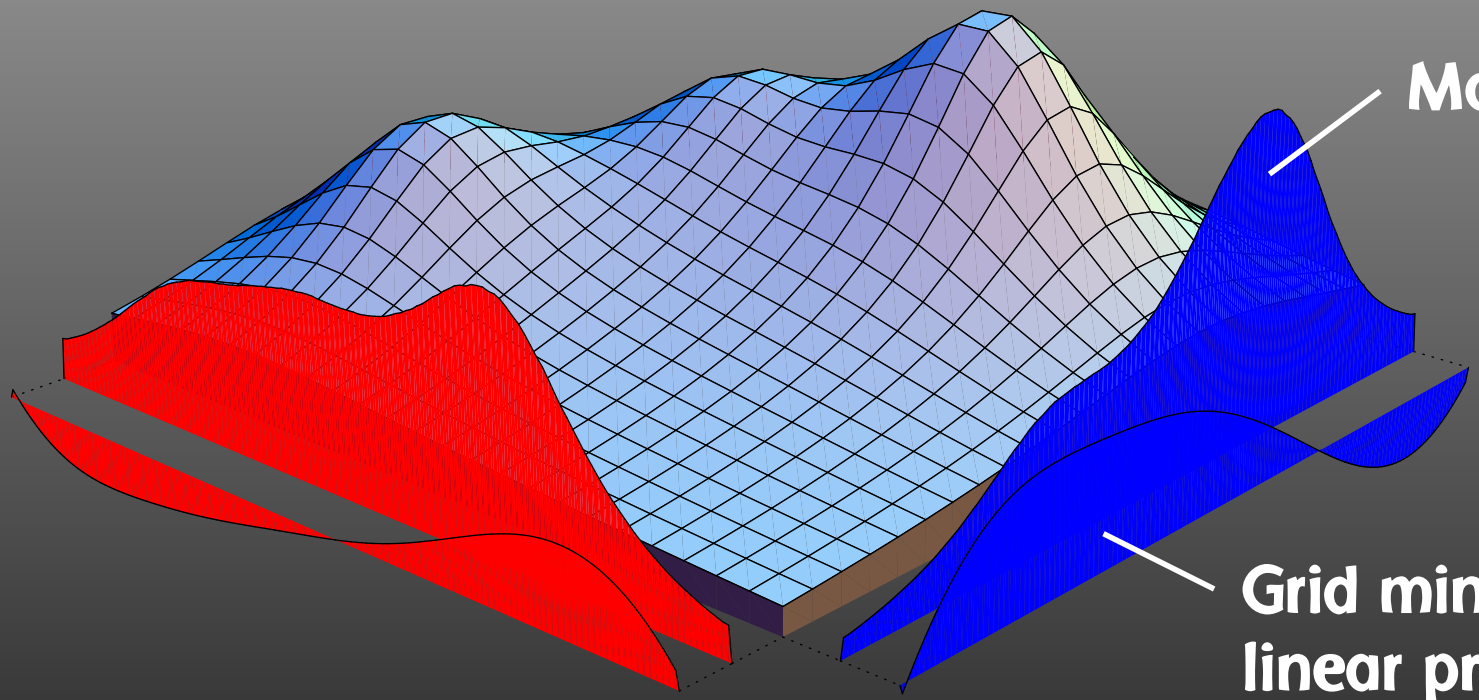
$$\mathrm{I}f = \int_0^1 \mathrm{d}^d x\, w(\vec{x})\, \frac{f(\vec{x})}{w(\vec{x})}\,, \qquad w(\vec{x}) > 0\,, \quad \mathrm{I}\,w = 1\,.$$

- 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 known to be $w(\vec{x}) = |f(\vec{x})|/\mathrm{I}f$ which has $\sigma_w(f/w) = 0$.



Importance Sampling in Vegas



The grid shows the progression along the respective axis. Progression is slow (i.e. many points are sampled) where the grid's value is small.

Stratified Sampling

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
Integral	$\mathbf{I}f \approx \mathbf{M}_n f$	$\mathbf{I}f \approx \frac{1}{2}(\mathbf{M}^a_{n/2} f + \mathbf{M}^b_{n/2} f)$
Variance	$\frac{\sigma^2 f}{n}$ $= \frac{1}{2n}\left(\sigma_a^2 f + \sigma_b^2 f\right) +$ $\frac{1}{4n}\left(\mathbf{I}_a f - \mathbf{I}_b f\right)^2$	$\frac{1}{4}\left(\frac{\sigma_a^2 f}{n/2} + \frac{\sigma_b^2 f}{n/2}\right)$ $= \frac{1}{2n}\left(\sigma_a^2 f + \sigma_b^2 f\right)$

The optimal reduction of variance is for $n_a/n_b = \sigma_a f/\sigma_b f$.
Thus: Split up the integration region into parts with equal variance, then sample all parts with same number of points.
But: naive splitting causes a 2^d increase in regions!

Number-Theoretic Methods

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

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

$$|\mathrm{Q}_n f - \mathrm{I} f| \leqslant V(f)\, D^*(\vec{x}_1, \ldots, \vec{x}_n)\,.$$

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

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

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

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.

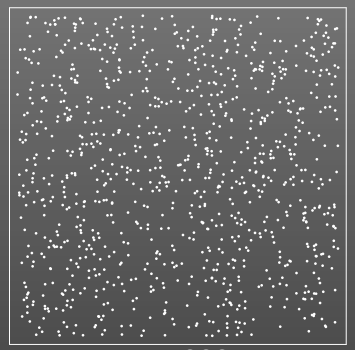
- **Halton Sequences,**
- **Sobol Sequences,**
- **Faure Sequences.**

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.

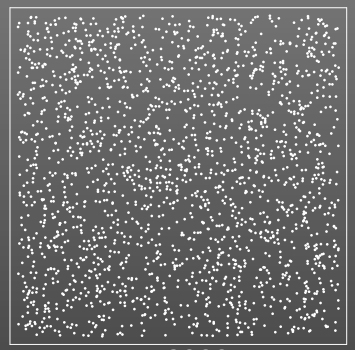
Example: CUBA's Vegas and Suave use Sobol sequences.

Comparison of Sequences

Mersenne Twister
Pseudo-Random Numbers



n = 1000

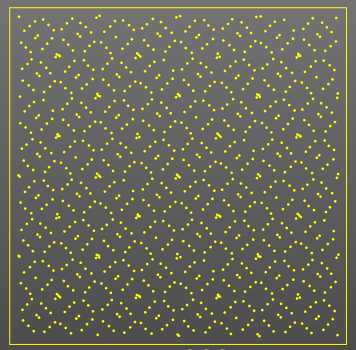


n = 2000

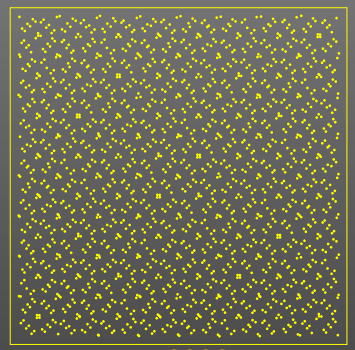
n = 3000

n = 4000

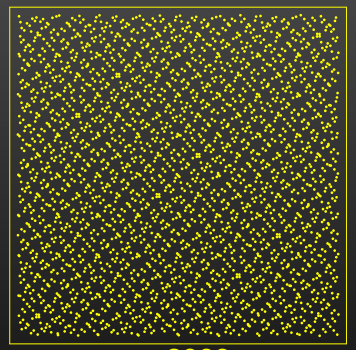
Sobol
Quasi-Random Numbers



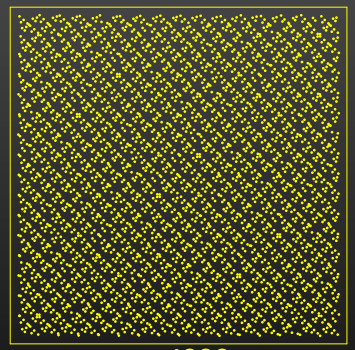
n = 1000



n = 2000



n = 3000



n = 4000

Lattice Methods

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

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

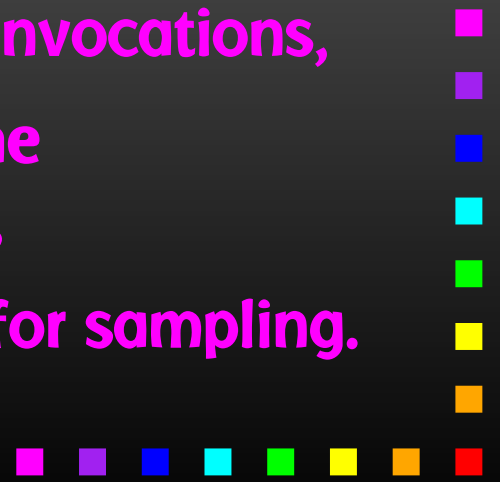
Construction principle for $\vec{z}$: knock out as many low-order **"Bragg reflections"** as possible in the error term:

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

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.

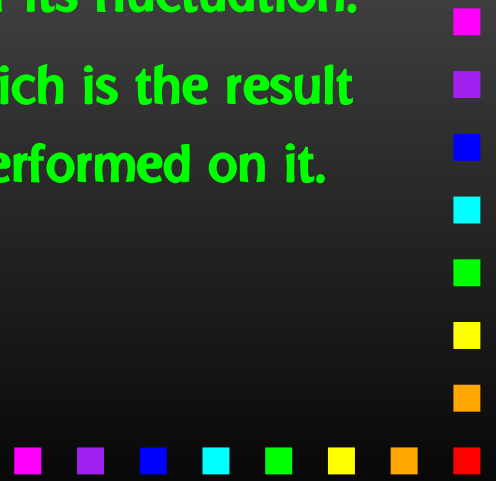
Vegas Implementation in Cuba

- Monte Carlo algorithm.
- Variance reduction: importance sampling.
- Algorithm:
  - Iteratively build up a piecewise constant weight function, represented on a rectangular grid.
  - Each iteration consists of a sampling step followed by a refinement of the grid.
- Vegas can memorize its grid for subsequent invocations,
- Vegas can save its internal state such that the calculation can be resumed e.g. after a crash,
- Choice of quasi- or pseudo-random numbers for sampling.



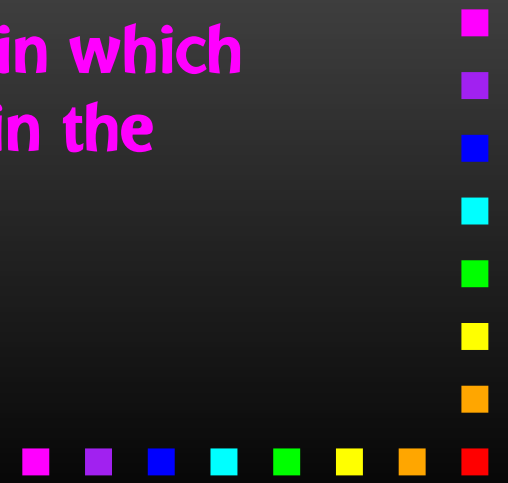
Suave Implementation in CUBA

- Monte Carlo algorithm.
- Variance reduction: Vegas-style importance sampling combined with globally adaptive subdivision.
- Algorithm:
  - ▷ Until the requested accuracy is reached, bisect the region with the largest error along the axis in which the fluctuations of the integrand are reduced most.
  - ▷ Prorate the number of new samples in each half for its fluctuation.
  - ▷ Vegas grid is kept across divisions, i.e. a region which is the result of $n-1$ subdivisions has had n Vegas iterations performed on it.
- Hybrid Vegas/Miser algorithm.
- Somewhat memory intensive.



Divonne Implementation in CUBA

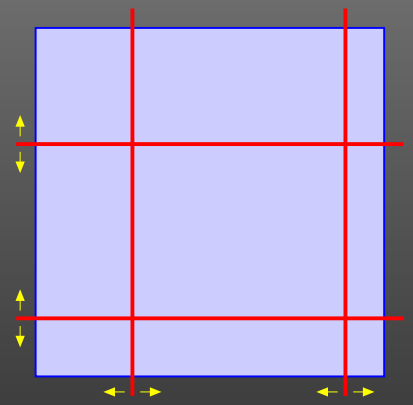
- Monte Carlo algorithm (+ cubature rules for comparison).
- Variance reduction: Stratified sampling.
- 3-Phase Algorithm:
  Partitioning - Sampling - Refinement.
- Original algorithm extended by Refinement Phase.
- The user can point out extrema for tricky integrands.
- For integrands which cannot be sampled too close to the border, a 'safety distance' can be prescribed in which values will be extrapolated from two points in the interior.



Divonne Algorithm

- **PHASE 1 – Partitioning**
  - ▷ For each subregion, 'actively' determine $\sup f$ and $\inf f$ using methods from numerical optimization.
  - ▷ Move 'dividers' around until all subregions have approximately equal spread, defined as

$$\text{Spread}(r) = \frac{1}{2}\,\text{Vol}(r)\left(\sup_{\vec{x}\in r} f(\vec{x}) - \inf_{\vec{x}\in r} f(\vec{x})\right).$$



- **PHASE 2 – Sampling**

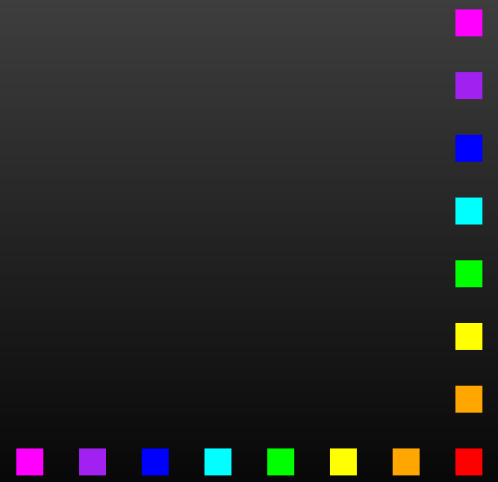
  Sample the subregions independently with the same number of points each. The latter is extrapolated from the results of Phase 1.

- **PHASE 3 – Refinement**

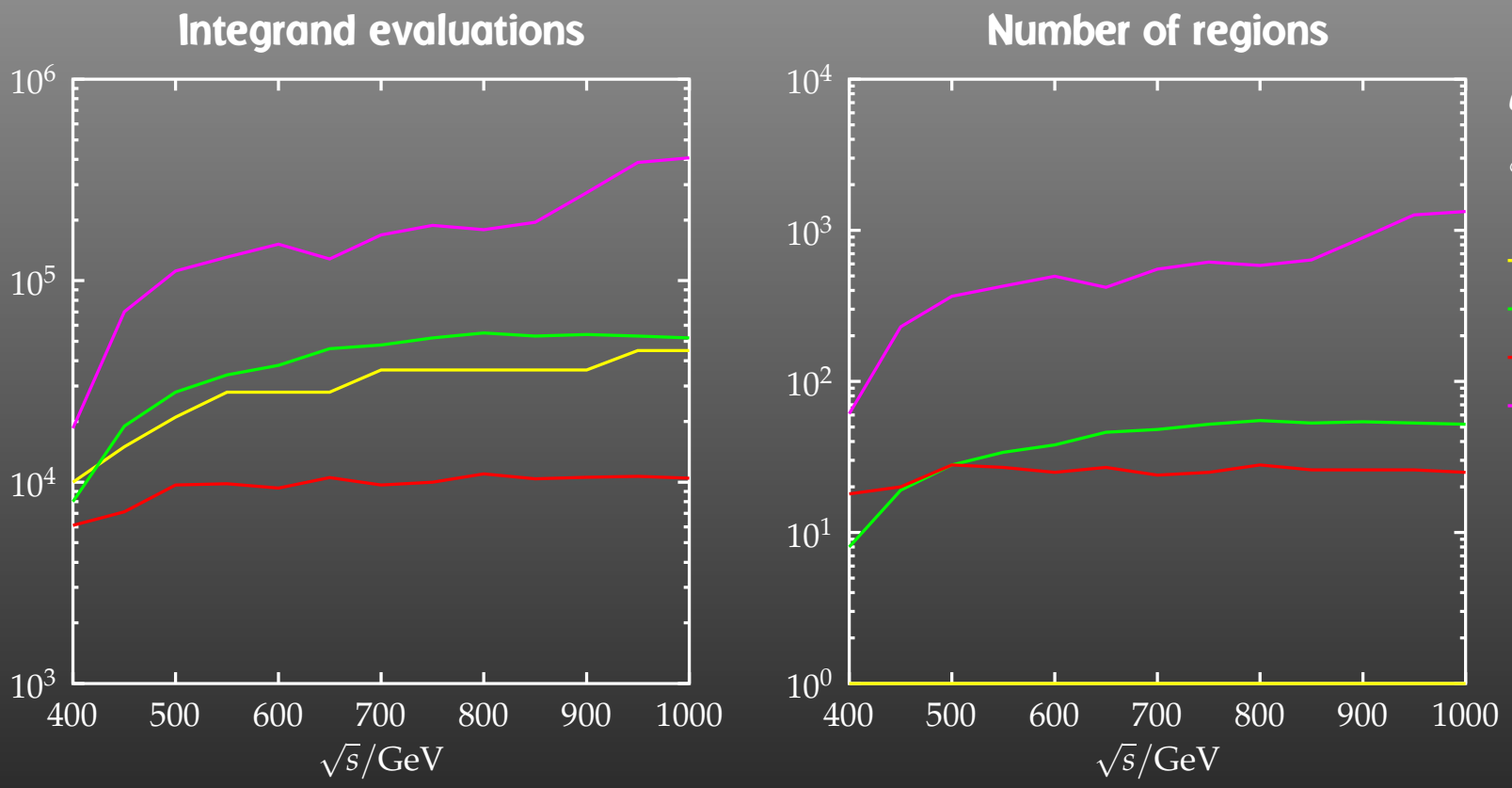
  Further subdivide or sample again if results from Phase 1 and 2 do not agree within their error.

Cuhre Implementation in CUBA

- Deterministic algorithm (uses Genz-Malik cubature rules of polynomial degree).
- Variance reduction: Globally adaptive subdivision.
- Algorithm:
  - Until the requested accuracy is reached, bisect the region with the largest error along the axis with the largest fourth difference.
- Consistent interface only, same as original DCUHRE (TOMS Algorithm 698).



Test Run



Above all: Very important to have several methods for cross-checking the results!

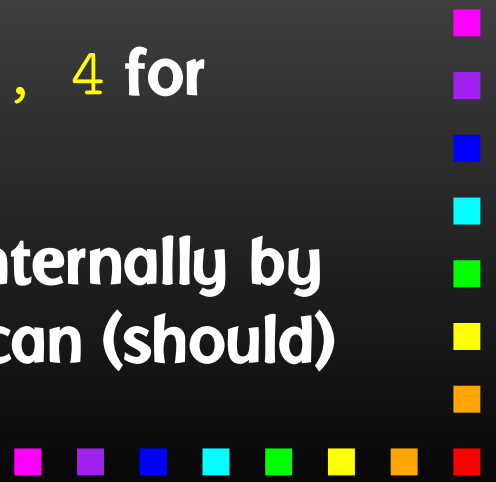
CUBA Chooser

CUBA includes a "one-stop interface" which further simplifies the invocation of the CUBA routines:

```
      subroutine Cuba(method, ndim, ncomp, integrand,
     &     integral, error, prob)
      integer method, ndim, ncomp
      external integrand
      double precision integral(ncomp)
      double precision error(ncomp)
      double precision prob(ncomp)
```

The user merely has to choose `method = 1, 2, 3, 4` for Vegas, Suave, Divonne, Cuhre.

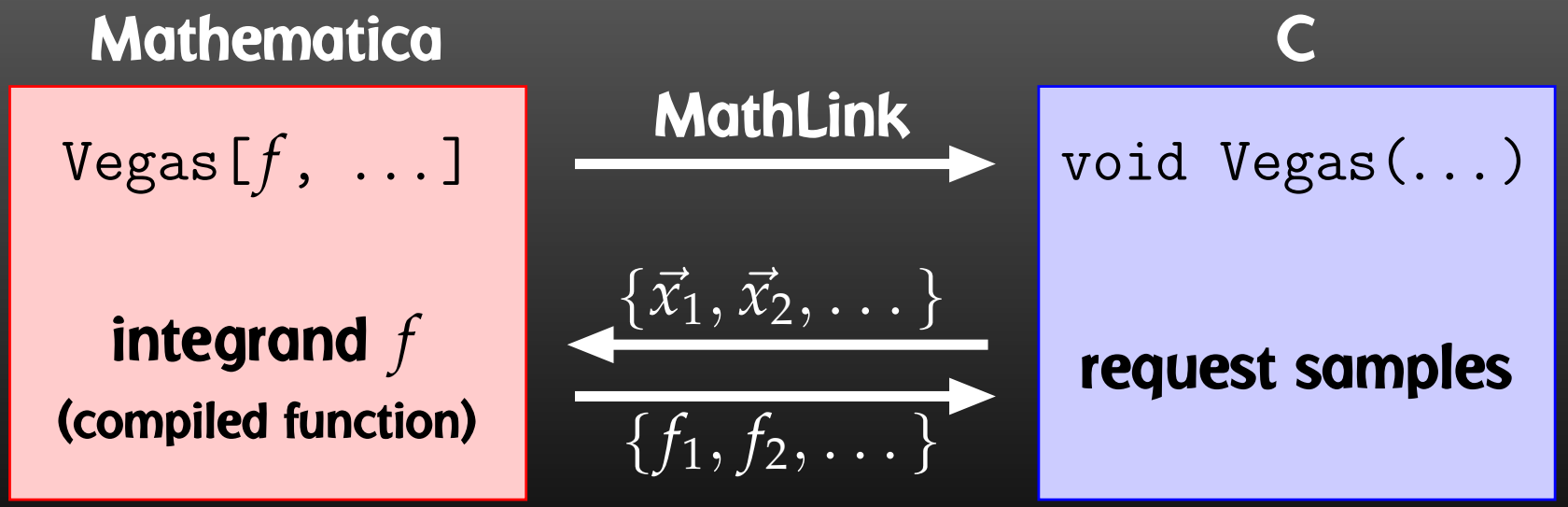
All other integration parameters are determined internally by the routine, i.e. this is not a finished product, but can (should) be modified by the user.



Mathematica interface

- **Used almost like** `NIntegrate`.
- **The integrand is evaluated completely in Mathematica. Can do things like**

```
Cuhre[Zeta[x y], {x,2,3}, {y,4,5}]
```

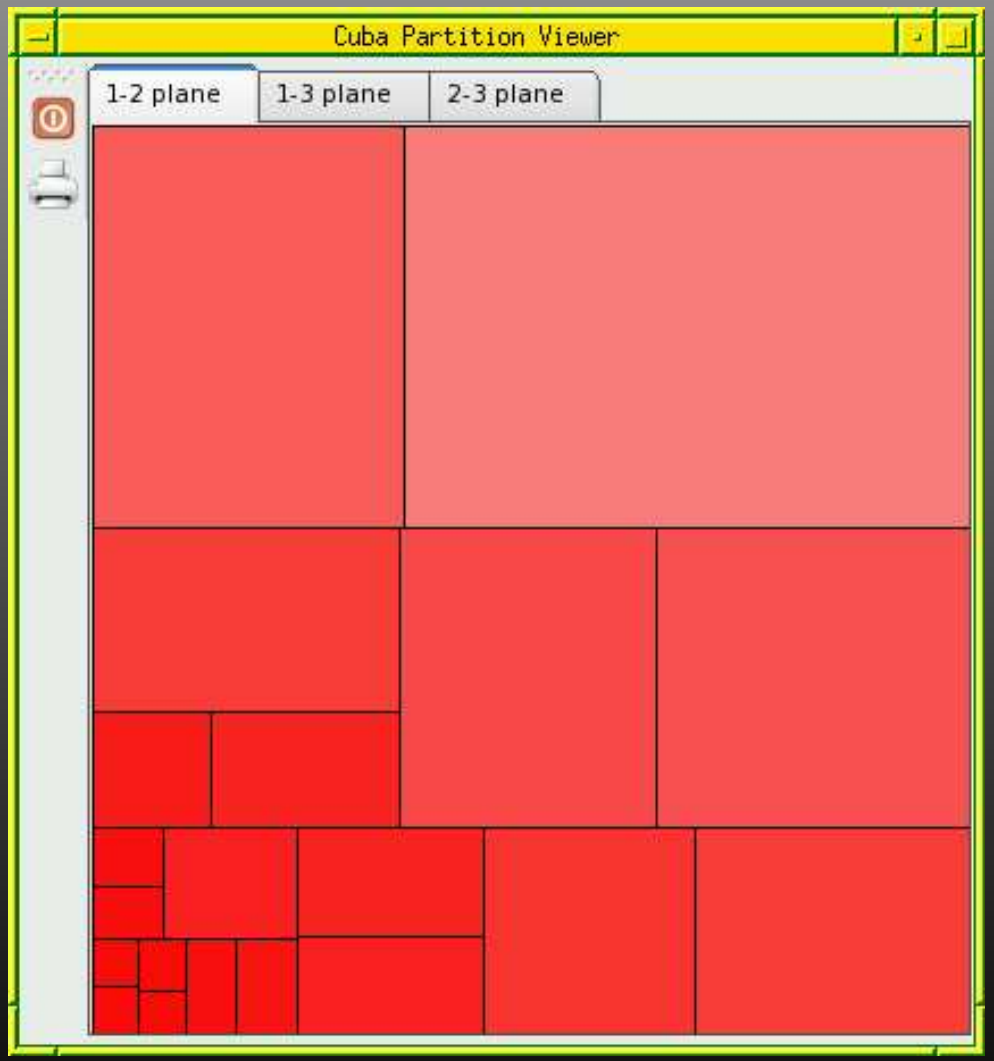


Partition Viewer

CUBA's Partition Viewer visualizes the partition taken by the integration algorithm.

Verbosity level 3 must be chosen and the output piped through `partview`:

```
myprog | partview 1 2
```



Summary

- CUBA is a library for multidimensional numerical integration written in C.
- Four independent algorithms: Vegas, Suave, Divonne, and Cuhre have similar invocations and can be exchanged easily for testing.
- All routines can integrate vector integrands.
- CUBA has a Fortran, C/C++, and Mathematica interface.
- The package includes additional tools, such as one-stop invocation and a partition viewer.
- Available at http://www.feynarts.de/cuba (LGPL) and easy to build (autoconf).