Cuba

Multivariate Integration on the Hypercube

Bijan Chokoufe Nejad

Vortrag zum FOKUS-Forschungspraktikum

25 April 2013

Cuba, T. Hahn (2005), is a general purpose integrator for the hypercube $C^d = \{\boldsymbol{x} \,|\, \boldsymbol{x} \in [0,1]^d\}$,

i.e. it computes

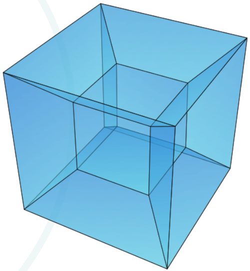
$$\mathrm{I}f = \int_0^1 \mathrm{d}^d x\, f(\boldsymbol{x}) \,.$$

Hyperrectangular regions can easily be scaled

$$\int_{a_1}^{b_1} \mathrm{d}x_1 \ldots \int_{a_n}^{b_n} \mathrm{d}x_n\, f(\boldsymbol{x})$$

$$= \prod_{i=1}^{n} \int_0^1 (b_i - a_i)\mathrm{d}y_i\, f(a_i + (b_i - a_i) y_i) \,.$$

Complicated forms can be achieved by setting $f(x) = 0$ for all x which should be excluded.



Projection of the 4-cube

`Cuba`, T. Hahn (2005), is a general purpose integrator for the hypercube $C^d = \{\boldsymbol{x} \,|\, \boldsymbol{x} \in [0,1]^d\}$,

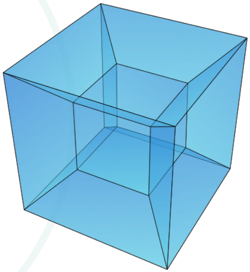
i.e. it computes

$$\mathrm{I}f = \int_0^1 \mathrm{d}^d x \, f(\boldsymbol{x}) \, .$$

Hyperrectangular regions can easily be scaled

$$\int_{a_1}^{b_1} \mathrm{d}x_1 \, \ldots \int_{a_n}^{b_n} \mathrm{d}x_n \, f(\boldsymbol{x})$$

$$= \prod_{i=1}^{n} \int_0^1 (b_i - a_i) \mathrm{d}y_i \, f(a_i + (b_i - a_i) y_i) \, .$$



Projection of the 4-cube

Complicated forms can be achieved by setting $f(\boldsymbol{x}) = 0$ for all $\boldsymbol{x}$ which should be excluded.

- Completely general adaptive integration of the hypercube
- Works for easy integrands out of the box, performance can still be improved by turning the 'knobs'
- f can be given as function of x in `C/C++`, `Fortran` or `Mathematica`
- Simple interface allows usage in a broad range of Physics and other disciplines
- Parallelization via `fork` and shared memory
- Four different algorithms for cross checking

- Completely general adaptive integration of the hypercube
- Works for easy integrands out of the box, performance can still be improved by turning the 'knobs'
- f can be given as function of x in `C/C++`, `Fortran` or `Mathematica`
- Simple interface allows usage in a broad range of Physics and other disciplines
- Parallelization via `fork` and shared memory
- Four different algorithms for cross checking

- Completely general adaptive integration of the hypercube
- Works for easy integrands out of the box, performance can still be improved by turning the 'knobs'
- f can be given as function of x in `C/C++`, `Fortran` or `Mathematica`
- Simple interface allows usage in a broad range of Physics and other disciplines
- Parallelization via `fork` and shared memory
- Four different algorithms for cross checking

- Completely general adaptive integration of the hypercube
- Works for easy integrands out of the box, performance can still be improved by turning the 'knobs'
- f can be given as function of x in `C/C++`, `Fortran` or `Mathematica`
- Simple interface allows usage in a broad range of Physics and other disciplines
- Parallelization via `fork` and shared memory
- Four different algorithms for cross checking

- Completely general adaptive integration of the hypercube
- Works for easy integrands out of the box, performance can still be improved by turning the 'knobs'
- f can be given as function of x in `C/C++`, `Fortran` or `Mathematica`
- Simple interface allows usage in a broad range of Physics and other disciplines
- Parallelization via `fork` and shared memory
- Four different algorithms for cross checking

- Completely general adaptive integration of the hypercube
- Works for easy integrands out of the box, performance can still be improved by turning the 'knobs'
- f can be given as function of x in `C/C++`, `Fortran` or `Mathematica`
- Simple interface allows usage in a broad range of Physics and other disciplines
- Parallelization via `fork` and shared memory
- Four different algorithms for cross checking

Vegas

Standard Monte Carlo (MC) estimate of the integral and the variance

$$\bar{m}\,(f) = \frac{1}{M}\sum_{k=1}^{M} f(\boldsymbol{x}_k) \quad \text{and} \quad \bar{\sigma}(f)^2 = \frac{1}{M-1}\sum_{k=1}^{M} (f(r_k) - \bar{m}\,)^2$$

gives with the central limit theorem for large M and independent random numbers $\boldsymbol{x}_k$

$$\mathrm{I}f = \bar{m} \pm \frac{\bar{\sigma}(f)}{\sqrt{M}}\,,$$

regardless of the dimension. For variance reduction, we can use importance sampling:

$$\mathrm{I}f = \int \mathrm{d}^d x\, w(\boldsymbol{x})\,\frac{f(\boldsymbol{x})}{w(\boldsymbol{x})} = \left\langle \frac{f}{w} \right\rangle_w$$

Standard Monte Carlo (MC) estimate of the integral and the variance

$$\bar{m}\,(f) = \frac{1}{M}\sum_{k=1}^{M} f(\boldsymbol{x}_k) \quad \text{and} \quad \bar{\sigma}(f)^2 = \frac{1}{M-1}\sum_{k=1}^{M} (f(r_k) - \bar{m}\,)^2$$

gives with the central limit theorem for large M and independent random numbers $\boldsymbol{x}_k$

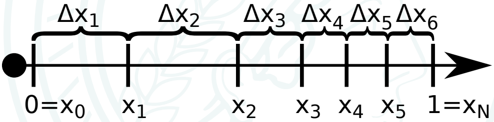
$$\mathrm{I}f = \bar{m}\, \pm \frac{\bar{\sigma}(f)}{\sqrt{M}}\ ,$$

regardless of the dimension. For variance reduction, we can use importance sampling:

$$\mathrm{I}f = \int \mathrm{d}^d x\, w(\boldsymbol{x})\, \frac{f(\boldsymbol{x})}{w(\boldsymbol{x})} = \left\langle \frac{f}{w} \right\rangle_w$$

`Vegas` constructs $w(\boldsymbol{x})$ as product of piecewise-constant weight functions to minimize $\sigma(f/w)$.

The probability for a random x in any given step is equal $1/N$

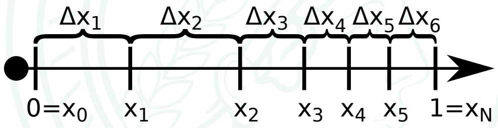


The grid of Δx_i is adapted in each iteration such that the increment density is high where the margin sum of $|f|$ is large.

Main weakness of `Vegas` is sampling of peaks which are not aligned with the coordinate axes.

`Vegas` constructs $w(\boldsymbol{x})$ as product of piecewise-constant weight functions to minimize $\sigma(f/w)$.

The probability for a random x in any given step is equal $1/N$



The grid of Δx_i is adapted in each iteration such that the increment density is high where the margin sum of $|f|$ is large.

Main weakness of `Vegas` is sampling of peaks which are not aligned with the coordinate axes.

Usually Pseudo Random Number Generators (PRNGs) are used to simulate randomness on a deterministic machine.

In `Cuba`, one can choose between `MersenneTwister` and `Ranlux` by using a non-zero `seed`.

Central limit theorem only states average convergence is $1/\sqrt{N}$.

Quasi-random numbers, which are overall evenly distributed, aim to be better than average.

Number-theoretic basis is the Koksma-Hlawka inequality

$$\left| \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}_i) - \int_{C^s} \mathrm{d}\boldsymbol{x}\, f(\boldsymbol{x}) \right| \leq V(f)\, D_N^*(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$$

Usually Pseudo Random Number Generators (PRNGs) are used to simulate randomness on a deterministic machine.

In `Cuba`, one can choose between `MersenneTwister` and `Ranlux` by using a non-zero `seed`.

Central limit theorem only states average convergence is $1/\sqrt{N}$.

Quasi-random numbers, which are overall evenly distributed, aim to be better than average.

Number-theoretic basis is the Koksma-Hlawka inequality

$$\left|\frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}_i) - \int_{C^s} \mathrm{d}\boldsymbol{x}\, f(\boldsymbol{x})\right| \le V(f)\; D_N^*(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$$

$$\left|\frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}_i) - \int_{C^s} \mathrm{d}\boldsymbol{x}\, f(\boldsymbol{x})\right| \leq V(f)\; D_N^*\big(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\big)$$

Variation in the sense of Hardy and Krause

$$V(f) = \sum_{k=1}^{s} \sum_{1\leq i_1<\cdots<i_k\leq s} V^{(k)}\left(f; i_1,\ldots,i_k\right) \quad \text{where}$$

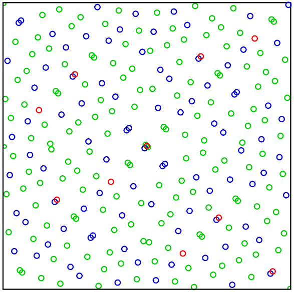
$$V^{(k)}\left(f; i_1,\ldots,i_k\right) = \int \mathrm{d}x_{i_1}\ldots\mathrm{d}x_{i_k} \left|\frac{\partial^s f}{\partial x_{i_1}\ldots\partial x_{i_k}}\right|_{x_j=1, f\neq i_1,\ldots,i_k}$$

Star-Discrepancy of a set $P = \left\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\right\}$

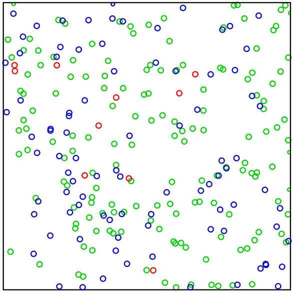
$$D_N^*(P) = \sup_{B\in J^*}\left|\frac{\#\text{ of points in } P \text{ falling into } B}{N} - \lambda(B)\right|$$

where J^* is the set of all intervals $\prod_{i=1}^{s}[0,u_i)$ and $u_i\in[0,1)$.

Sobol



Pseudo Random

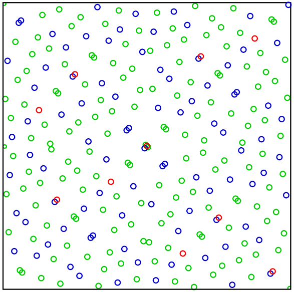


Wikimedia

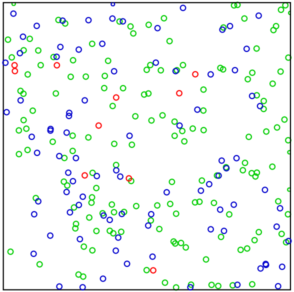
$$D_N^*(x_1, \ldots, x_N) = \mathcal{O}\left(\frac{(\log N)^s}{N}\right)$$

In general, this is not superior to PNRGs, as often claimed. Computational studies in dimension up to $s = 100$, L. Kocis *et al.* (1997), show for polynomials errors of about $\mathcal{O}\left(1/N\right)$ up to worse than $\mathcal{O}\left(1/\sqrt{N}\right)$ if the integrand has many peaks.

Sobol



Pseudo Random



Wikimedia

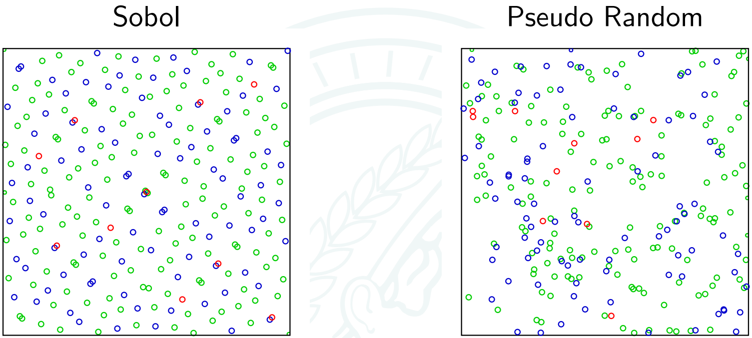
$$D_N^*(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \mathcal{O}\left(\frac{(\log N)^s}{N}\right)$$

In general, this is not superior to PNRGs, as often claimed.

Computational studies in dimension up to $s = 100$,

L. Kocis *et al.* (1997), show for polynomials errors of about $\mathcal{O}(1/N)$

up to worse than $\mathcal{O}\left(1/\sqrt{N}\right)$ if the integrand has many peaks.



Wikimedia

$$D_N^*(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \mathcal{O}\left(\frac{(\log N)^s}{N}\right)$$

In general, this is not superior to PNRGs, as often claimed.
Computational studies in dimension up to $s = 100$,
L. Kocis *et al.* (1997), show for polynomials errors of about $\mathcal{O}\left(1/N\right)$
up to worse than $\mathcal{O}\left(1/\sqrt{N}\right)$ if the integrand has many peaks.

Fairly simple: `struct state` contains all running objects

```
count niter;
bool backup;
number nsamples, neval;
Cumulants cumul[NCOMP];
Grid grid[NDIM];
```

Fixed size objects like this can easily be dumped to disc in C with

```
fd = creat(s, 0666);
Write(fd, &state, sizeof state);
```

and loaded again via

```
cint fd = open(t->statefile, O_RDONLY);
if( st.st_size == sizeof(state) )
  Read(fd, &state, sizeof state);
```

Fairly simple: `struct state` contains all running objects

```
count niter;
bool backup;
number nsamples, neval;
Cumulants cumul[NCOMP];
Grid grid[NDIM];
```

Fixed size objects like this can easily be dumped to disc in C with

```
fd = creat(s, 0666);
Write(fd, &state, sizeof state);
```

and loaded again via

```
cint fd = open(t->statefile, O_RDONLY);
if( st.st_size == sizeof(state) )
  Read(fd, &state, sizeof state);
```

Suave

`Suave` aims to avoid the dependence of factorizable integrands of `Vegas` by using a divide & conquer approach:

- Integrate the whole region $\Rightarrow I_{\text{tot}} \pm E_{\text{tot}}$
- while $\Big(E_{\text{tot}} > \max\big(\epsilon_{\text{rel}} I_{\text{tot}}, \epsilon_{\text{abs}}\big)\Big)$
- Find the region r with the largest error
- Bisect r in the dimension which minimizes the fluctuations
- Sample r_L and r_R
- $I_{\text{tot}} \mathrel{+}= I_L + I_R - I_r \qquad E^2_{\text{tot}} \mathrel{+}= E^2_L + E^2_R - E^2_r$
- end

Each subregion is sampled with a `VEGAS`-like sampling step and the grids are reused and stretched.

`Suave` aims to avoid the dependence of factorizable integrands of `Vegas` by using a divide & conquer approach:

- Integrate the whole region $\Rightarrow I_{\text{tot}} \pm E_{\text{tot}}$
- while $\Big(E_{\text{tot}} > \max(\epsilon_{\text{rel}} I_{\text{tot}}, \epsilon_{\text{abs}})\Big)$
-     Find the region r with the largest error
-     Bisect r in the dimension which minimizes the fluctuations
-     Sample r_L and r_R
-     $I_{\text{tot}} \mathrel{+}= I_L + I_R - I_r \qquad E_{\text{tot}}^2 \mathrel{+}= E_L^2 + E_R^2 - E_r^2$
- end

Each subregion is sampled with a `VEGAS`-like sampling step and the grids are reused and stretched.

In Suave the points for the two new regions are distributed according to their variance which is known as stratified sampling.

Splitting up a region r in r_A and r_B leads to

$$\mathrm{I}f = \frac{1}{2}\Big(m_{N_A}(f) + m_{N_B}(f)\Big)$$

$$\sigma^2 = \frac{1}{4}\left(\frac{\sigma_A(f)^2}{N_A} + \frac{\sigma_B(f)^2}{N_B}\right)$$

Obviously, the overall variance can be reduced by making $N_{A,B}$ proportional to $\sigma_{A,B}$.

Caveat: Using too little points in each iteration, can badly underestimate the true variance in a region.

In Suave the points for the two new regions are distributed according to their variance which is known as stratified sampling.

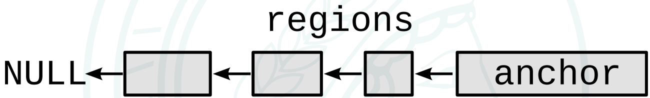
Splitting up a region r in r_A and r_B leads to

$$\mathrm{I}f = \frac{1}{2}\Big(m_{N_A}(f) + m_{N_B}(f)\Big)$$
$$\sigma^2 = \frac{1}{4}\left(\frac{\sigma_A(f)^2}{N_A} + \frac{\sigma_B(f)^2}{N_B}\right)$$

Obviously, the overall variance can be reduced by making $N_{A,B}$ proportional to $\sigma_{A,B}$.

Caveat: Using too little points in each iteration, can badly underestimate the true variance in a region.

Not so easy: Similar `struct state` for the cumulants but the number of regions isn't fixed and neither is their size, since all points are remembered.

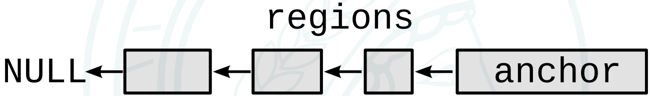


One must save the size of each region in the head of the statefile. Furthermore, the `pointers` become invalid.

On loading, the size of file has to be checked with header, otherwise `Segmentation fault` lurks around the corner.

Finally, the linked list can be recreated.

Not so easy: Similar `struct state` for the cumulants but the number of regions isn't fixed and neither is their size, since all points are remembered.



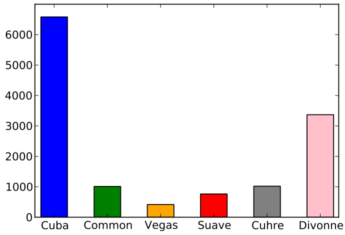
One must save the size of each region in the head of the statefile. Furthermore, the `pointers` become invalid.

On loading, the size of file has to be checked with header, otherwise `Segmentation fault` lurks around the corner.

Finally, the linked list can be recreated.

Divonne

The most complex algorithm. As indicator, consider the physical lines of source code measured with `cloc-1.56`:



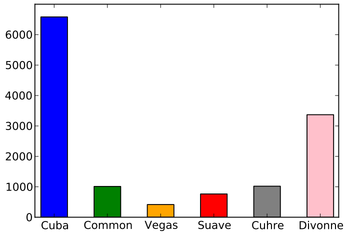
Divonne works in 3 phases

- Partitioning
- Sampling
- Refinement

and can employ 4 different samplings

- Korobov
- Sobol
- MersenneTwister/Ranlux
- Cubature

The most complex algorithm. As indicator, consider the physical lines of source code measured with `cloc-1.56`:



`Divonne` works in 3 phases

- Partitioning
- Sampling
- Refinement

and can employ 4 different samplings

- Korobov
- Sobol
- MersenneTwister/Ranlux
- Cubature

Task: Final result as list of regions with similar spread $s(R)$

$$s(R) = \mathrm{Vol}\,(R) \left(\max_{\boldsymbol{x}\in R} f(\boldsymbol{x}) - \min_{\boldsymbol{x}\in R} f(\boldsymbol{x}) \right)$$

Min/Max are sought with quasi-Newton method with finite difference approximations of 1st derivatives.

Partitioning: Find approximate isopleths $\tilde{f}$

$$(f^{\max} - \tilde{f})\,\mathrm{Vol}\,(R^{\max}) = (\tilde{f} - f^{\min})\,\mathrm{Vol}\left(R^{\min}\right)$$

which divide the region into disjoint, axis-oriented hyperrectangular regions.

Leads to a nonlinear system whose solution is only roughly approximated with a secant method.

Task: Final result as list of regions with similar spread $s(R)$

$$s(R) = \mathrm{Vol}\,(R) \left(\max_{\boldsymbol{x} \in R} f(\boldsymbol{x}) - \min_{\boldsymbol{x} \in R} f(\boldsymbol{x}) \right)$$

Min/Max are sought with quasi-Newton method with finite difference approximations of 1st derivatives.

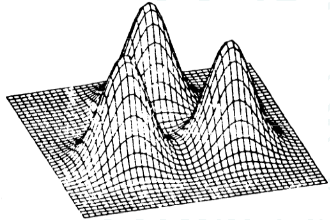
Partitioning: Find approximate isopleths $\tilde{f}$

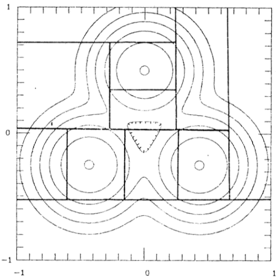
$$(f^{\max} - \tilde{f})\,\mathrm{Vol}\,(R^{\max}) = (\tilde{f} - f^{\min})\,\mathrm{Vol}\left(R^{\min}\right)$$

which divide the region into disjoint, axis-oriented hyperrectangular regions.

Leads to a nonlinear system whose solution is only roughly approximated with a secant method.

Further algorithm

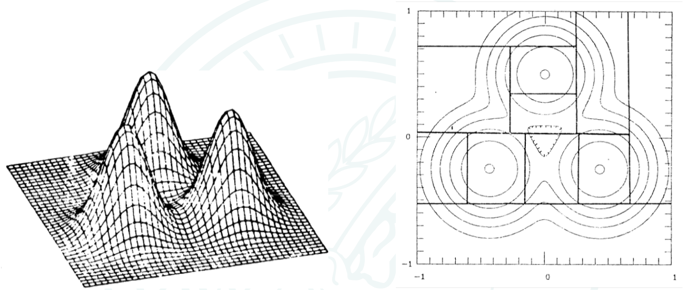




Jerome H. Friedman *et al.* (1978)

Sampling: Sample each subregion with same number of points. This number can be estimated from Phase 1.

Refinement: Subdivide or sample again if results from first phases don't agree within their error.



Jerome H. Friedman *et al.* (1978)

Sampling: Sample each subregion with same number of points. This number can be estimated from Phase 1.

Refinement: Subdivide or sample again if results from first phases don't agree within their error.

Given a periodic function on C^d, the integral can be computed as

$$\int_{C^d} f(\boldsymbol{t})\mathrm{d}\boldsymbol{t} \approx \frac{1}{M}\sum_{n=1}^{M} f\left(\frac{n}{M}\,\boldsymbol{g}\right) \quad \text{with} \quad M \geq 2\,.$$

$\boldsymbol{g} \in \mathbb{Z}^d$ is a suitably chosen lattice point.

Due to periodicity, $\frac{n}{M}\,\boldsymbol{g} = \{\frac{n}{M}\,\boldsymbol{g}\}$ (the fractional part).

Periodicity can easily be obtained with an appropriate transformation.

But why should this work? ...

Given a periodic function on C^d, the integral can be computed as

$$\int_{C^d} f(\boldsymbol{t})\mathrm{d}\boldsymbol{t} \approx \frac{1}{M}\sum_{n=1}^{M} f\left(\frac{n}{M}\,\boldsymbol{g}\right) \quad \text{with} \quad M \geq 2\,.$$

$\boldsymbol{g} \in \mathbb{Z}^d$ is a suitably chosen lattice point.

Due to periodicity, $\frac{n}{M}\,\boldsymbol{g} = \{\frac{n}{M}\,\boldsymbol{g}\}$ (the fractional part).

Periodicity can easily be obtained with an appropriate transformation.

But why should this work? ...

Consider

$$f(\boldsymbol{t}) = \sum_{\boldsymbol{h}\in\mathbb{Z}^s} c_{\boldsymbol{h}} \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{h}\boldsymbol{t}} \qquad c_{\boldsymbol{h}} = \int_{C^s} \mathrm{d}\boldsymbol{t}\, f(\boldsymbol{t}) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{h}\boldsymbol{t}}$$

$$\begin{aligned}
\frac{1}{M}\sum_{n=1}^{M} f\left(\frac{n}{M}\boldsymbol{g}\right) - \int_{C^s} \mathrm{d}\boldsymbol{t}\, f(\boldsymbol{t}) &= \frac{1}{M}\sum_{n}\sum_{\boldsymbol{h}} c_{\boldsymbol{h}} \mathrm{e}^{2\pi\mathrm{i}\frac{n}{M}\boldsymbol{h}\boldsymbol{g}} - c_0 \\
&= \frac{1}{M}\sum_{\boldsymbol{h}\neq\boldsymbol{0}} c_{\boldsymbol{h}} \underbrace{\sum_{n=1}^{M} \mathrm{e}^{2\pi\mathrm{i}\frac{n}{m}\boldsymbol{h}\boldsymbol{g}}}_{\begin{cases} = 0 & \text{for } \boldsymbol{h}\boldsymbol{g} \neq 0 \mod m \\ = M & \text{for } \boldsymbol{h}\boldsymbol{g} = 0 \mod m \end{cases}} \\
&= \sum_{\substack{\boldsymbol{h}\neq 0 \\ \boldsymbol{h}\boldsymbol{g}=0 \mod m}} c_{\boldsymbol{h}}\,,
\end{aligned}$$

which is the reason why Korobov benefits from smooth functions which can be described with a finite number of coefficients.

Consider

$$f(\boldsymbol{t}) = \sum_{\boldsymbol{h}\in\mathbb{Z}^s} c_{\boldsymbol{h}} \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{h}\boldsymbol{t}} \qquad c_{\boldsymbol{h}} = \int_{C^s} \mathrm{d}\boldsymbol{t}\, f(\boldsymbol{t}) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{h}\boldsymbol{t}}$$

$$\frac{1}{M}\sum_{n=1}^{M} f\left(\frac{n}{M}\boldsymbol{g}\right) - \int_{C^s} \mathrm{d}\boldsymbol{t}\, f(\boldsymbol{t}) = \frac{1}{M}\sum_{n}\sum_{\boldsymbol{h}} c_{\boldsymbol{h}} \mathrm{e}^{2\pi\mathrm{i}\frac{n}{M}\boldsymbol{h}\boldsymbol{g}} - c_0$$

$$= \frac{1}{M}\sum_{\boldsymbol{h}\neq\boldsymbol{0}} c_{\boldsymbol{h}} \underbrace{\sum_{n=1}^{M} \mathrm{e}^{2\pi\mathrm{i}\frac{n}{m}\boldsymbol{h}\boldsymbol{g}}}_{\begin{cases} = 0 & \text{for } \boldsymbol{h}\boldsymbol{g} \neq 0 \mod m \\ = M & \text{for } \boldsymbol{h}\boldsymbol{g} = 0 \mod m \end{cases}}$$

$$= \sum_{\substack{\boldsymbol{h}\neq 0 \\ \boldsymbol{h}\boldsymbol{g}=0 \mod m}} c_{\boldsymbol{h}}\,,$$

which is the reason why Korobov benefits from smooth functions which can be described with a finite number of coefficients.

Consider

$$f(\boldsymbol{t}) = \sum_{\boldsymbol{h}\in\mathbb{Z}^s} c_{\boldsymbol{h}} \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{h}\boldsymbol{t}} \qquad c_{\boldsymbol{h}} = \int_{C^s} \mathrm{d}\boldsymbol{t}\, f(\boldsymbol{t}) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{h}\boldsymbol{t}}$$

$$\begin{aligned}
\frac{1}{M}\sum_{n=1}^{M} f\left(\frac{n}{M}\boldsymbol{g}\right) - \int_{C^s}\mathrm{d}\boldsymbol{t}\, f(\boldsymbol{t}) &= \frac{1}{M}\sum_{n}\sum_{\boldsymbol{h}} c_{\boldsymbol{h}} \mathrm{e}^{2\pi\mathrm{i}\frac{n}{M}\boldsymbol{h}\boldsymbol{g}} - c_0 \\
&= \frac{1}{M}\sum_{\boldsymbol{h}\neq\boldsymbol{0}} c_{\boldsymbol{h}} \underbrace{\sum_{n=1}^{M} \mathrm{e}^{2\pi\mathrm{i}\frac{n}{m}\boldsymbol{h}\boldsymbol{g}}}_{\begin{cases} = 0 & \text{for } \boldsymbol{h}\boldsymbol{g} \neq 0 \mod m \\ = M & \text{for } \boldsymbol{h}\boldsymbol{g} = 0 \mod m \end{cases}} \\
&= \sum_{\substack{\boldsymbol{h}\neq\boldsymbol{0} \\ \boldsymbol{h}\boldsymbol{g}=0 \mod m}} c_{\boldsymbol{h}} \,,
\end{aligned}$$

which is the reason why Korobov benefits from smooth functions which can be described with a finite number of coefficients.

Consider

$$f(\boldsymbol{t}) = \sum_{\boldsymbol{h}\in\mathbb{Z}^s} c_{\boldsymbol{h}} \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{h}\boldsymbol{t}} \qquad c_{\boldsymbol{h}} = \int_{C^s} \mathrm{d}\boldsymbol{t}\, f(\boldsymbol{t}) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{h}\boldsymbol{t}}$$

$$\begin{aligned}
\frac{1}{M}\sum_{n=1}^{M} f\left(\frac{n}{M}\boldsymbol{g}\right) - \int_{C^s} \mathrm{d}\boldsymbol{t}\, f(\boldsymbol{t}) &= \frac{1}{M}\sum_{n}\sum_{\boldsymbol{h}} c_{\boldsymbol{h}} \mathrm{e}^{2\pi\mathrm{i}\frac{n}{M}\boldsymbol{h}\boldsymbol{g}} - c_0 \\
&= \frac{1}{M}\sum_{\boldsymbol{h}\neq\boldsymbol{0}} c_{\boldsymbol{h}} \underbrace{\sum_{n=1}^{M} \mathrm{e}^{2\pi\mathrm{i}\frac{n}{m}\boldsymbol{h}\boldsymbol{g}}}_{\begin{cases} = 0 & \text{for } \boldsymbol{h}\boldsymbol{g} \neq 0 \mod m \\ = M & \text{for } \boldsymbol{h}\boldsymbol{g} = 0 \mod m \end{cases}} \\
&= \sum_{\substack{\boldsymbol{h}\neq 0 \\ \boldsymbol{h}\boldsymbol{g}=0 \mod m}} c_{\boldsymbol{h}} \,,
\end{aligned}$$

which is the reason why Korobov benefits from smooth functions which can be described with a finite number of coefficients.

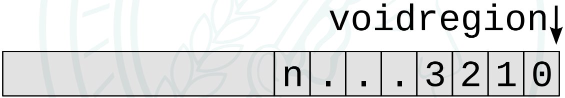
Consider

$$f(\boldsymbol{t}) = \sum_{\boldsymbol{h}\in\mathbb{Z}^s} c_{\boldsymbol{h}} \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{h}\boldsymbol{t}} \qquad c_{\boldsymbol{h}} = \int_{C^s} \mathrm{d}\boldsymbol{t}\, f(\boldsymbol{t}) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{h}\boldsymbol{t}}$$

$$\begin{aligned}
\frac{1}{M}\sum_{n=1}^{M} f\left(\frac{n}{M}\boldsymbol{g}\right) - \int_{C^s} \mathrm{d}\boldsymbol{t}\, f(\boldsymbol{t}) &= \frac{1}{M}\sum_{n}\sum_{\boldsymbol{h}} c_{\boldsymbol{h}} \mathrm{e}^{2\pi\mathrm{i}\frac{n}{M}\boldsymbol{h}\boldsymbol{g}} - c_0 \\
&= \frac{1}{M}\sum_{\boldsymbol{h}\neq\boldsymbol{0}} c_{\boldsymbol{h}} \underbrace{\sum_{n=1}^{M} \mathrm{e}^{2\pi\mathrm{i}\frac{n}{m}\boldsymbol{h}\boldsymbol{g}}}_{\begin{cases} = 0 & \text{for } \boldsymbol{h}\boldsymbol{g} \neq 0 \mod m \\ = M & \text{for } \boldsymbol{h}\boldsymbol{g} = 0 \mod m \end{cases}} \\
&= \sum_{\substack{\boldsymbol{h}\neq 0 \\ \boldsymbol{h}\boldsymbol{g}=0 \mod m}} c_{\boldsymbol{h}}\,,
\end{aligned}$$

which is the reason why Korobov benefits from smooth functions which can be described with a finite number of coefficients.

`struct` for the cumulants and fixed size regions joined together in one big chunk



Have to continue in the right phase and remember the used random numbers.

Cuhre

Cuhre is a deterministic algorithm of Berntsen, Espelid, Genz & Malik.

Integrates via one of five fully symmetric integration rules R of polynomial degree $d = 2m+1$, integrating exactly all monomials $x_1^{k_1} \ldots x_n^{k_n}$ with $\sum k_i \le d$,

$$\mathrm{I}f = \int \mathrm{d}\boldsymbol{x}\, f(\boldsymbol{x}) \simeq R[f] = \sum_{j=1}^{L} w_j f(\boldsymbol{x}_j)$$

Additionally, there are four null rules N_i of degree $2m-1$, $2m-1$, $2m-3$ and $2m-5$ for error estimation with

$$N_i[f] = \sum_{j=1}^{L} w_j^{(i)} f(\boldsymbol{x}_j), \quad i \in \{1,2,3,4\}$$

Cuhre is a deterministic algorithm of Berntsen, Espelid, Genz & Malik.

Integrates via one of five fully symmetric integration rules R of polynomial degree $d = 2m + 1$, integrating exactly all monomials $x_1^{k_1} \dots x_n^{k_n}$ with $\sum k_i \le d$,

$$\mathrm{I}f = \int \mathrm{d}\boldsymbol{x}\, f(\boldsymbol{x}) \simeq R[f] = \sum_{j=1}^{L} w_j f(\boldsymbol{x}_j)$$

Additionally, there are four null rules N_i of degree $2m - 1$, $2m - 1$, $2m - 3$ and $2m - 5$ for error estimation with

$$N_i[f] = \sum_{j=1}^{L} w_j^{(i)} f(\boldsymbol{x}_j), \quad i \in \{1, 2, 3, 4\}$$

- Same variance reduction scheme as in Suave: Globally adaptive subdivision.
- In low to moderate dimensions very accurate predictions are possible, particularly if the integrand is well approximated by polynomials.
- Memory layout is similar to Suave but with fixed size linked pools of regions

Further contributions

- Detected numerical instability in grid adaption in Vegas
- Found two Segmentation Faults in Divonne
- Fixed a bug in Cuhre which lead to negative χ^2 and wrong weights
- Updated demo files and Mathematica interfaces
- Reconsidering the `LAST` option which can lead to bad convergence and wrong results

Further contributions

- Detected numerical instability in grid adaption in Vegas
- Found two Segmentation Faults in Divonne
- Fixed a bug in Cuhre which lead to negative χ^2 and wrong weights
- Updated demo files and Mathematica interfaces
- Reconsidering the `LAST` option which can lead to bad convergence and wrong results

- Detected numerical instability in grid adaption in Vegas
- Found two Segmentation Faults in Divonne
- Fixed a bug in Cuhre which lead to negative χ^2 and wrong weights
- Updated demo files and Mathematica interfaces
- Reconsidering the `LAST` option which can lead to bad convergence and wrong results

Further contributions

- Detected numerical instability in grid adaption in Vegas
- Found two Segmentation Faults in Divonne
- Fixed a bug in Cuhre which lead to negative χ^2 and wrong weights
- Updated demo files and Mathematica interfaces
- Reconsidering the `LAST` option which can lead to bad convergence and wrong results

Further contributions

- Detected numerical instability in grid adaption in Vegas
- Found two Segmentation Faults in Divonne
- Fixed a bug in Cuhre which lead to negative χ^2 and wrong weights
- Updated demo files and Mathematica interfaces
- Reconsidering the `LAST` option which can lead to bad convergence and wrong results

T. Hahn
Comput.Phys.Commun. ***168***, (2005), 78-95
doi: 10.1016/j.cpc.2005.01.010

L. Kocis *et al.*
ACM Trans.Math.Software ***23***, (1997), 266-294
doi: 10.1145/264029.264064

J. H. Friedman *et al.*
ACM Trans.Math.Software ***7***, (1981), 76
doi: 10.1145/355934.355939