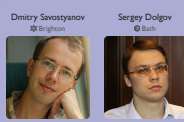


Parallel cross interpolation for high-precision calculation of high-dimensional integrals

Author and Presenter: Dmitry Savostyanov (University of Brighton, UK)

Co-author: Sergey Dolgov (University of Bath, UK)

- References**
- [1] N. Metropolis and S. Ulam, *J. American Stat. Assoc.*, 1951.
 - [2] H. Niederreiter, *Bull. AMS*, 1978.
 - [3] W. J. Morokoff and R. E. Caflisch, *J. Comp. Phys.*, 1995.
 - [4] J. Dick, F. Y. Kuo, Q. T. L. Gu, D. Nagev, and C. Schwab, *SIAM J. Num. Anal.*, 2014.
 - [5] S. A. Gonchar, M. L. Zemanovskii, and E. E. Tyrtynskov, *Mathematical Notes*, 1997.
 - [6] G. Poole and L. Nauli, *J. Comput. Appl. Math.*, 2009.
 - [7] T. T. Wu, B. M. McCoy, C. A. Tracy, and E. Brannock, *Phys. Rev. B*, 1976.
 - [8] D. H. Bailey, J. M. Borowitz, and R. E. Crandall, *J. Phys. A: Math. Gen.*, 2006.



Using **cross interpolation**, we reconstruct a multivariate function from a small number of **adaptively chosen** samples. By separating the variables, we reduce storage and CPU time, e.g. for numerical integration. We obtain results with hundred accurate digits and observe **sub-exponential convergence!**

Time to switch from Monte Carlo and qMC?

Parallel cross interpolation for high-precision calculation of high-dimensional integrals

High-dimensional integration

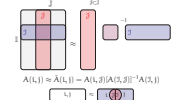
$$\int_{[0,1]^d} f(x) dx \approx 1 \approx \sum_{k=1}^N w_k T(k)$$

Existing methods

- Tensor product grids (naïve approach) — costs $N_{tot} = n^d$, bad convergence $e \approx 1 - 1/n, \sim N_{tot}^{-1/d}$
- Monte Carlo! (1930-40s, top 10 algorithm of the 20th century) — random samples, very slow convergence $e \sim N_{tot}^{-1/2}$
- Quasi MC (1980-90s) — optimised lattices, slow convergence $e \sim N_{tot}^{-1/d}$ with $0.5 < \gamma < 1$.
- High-order qMC²⁰ (2010s) — heavily optimised lattices, convergence $e \sim N_{tot}^{-1/d}$

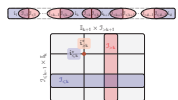
Cross interpolation for matrices

- Consider tensor product grid (but do not compute)
- Compute a few "fibers" (fixes all variables but one)
- Choose fibers adaptively, using the maximum-volume principle: $\beta_1, \beta_2 = \arg \max \{ \det A[\beta_1, \beta_2] \}$



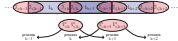
A practical algorithm for increasing the volume is Gaussian elimination with partial pivoting, e.g. rook^P : $(\beta_1, \beta_2) = \arg \max \{ \det A[\beta_1, \beta_2] \}$

Cross interpolation for tensors



Parallel cross interpolation for tensors

- Parallelisation over dimension — 3-d process is responsible for expanding the cross $\{ \beta_{1,2}, \beta_{3,4} \}$
- Distributed-memory MPI scaling up to $(d-1)$ CPUs.
- Shared-memory OpenMP scaling up to 16 threads.



Numerical example

Magnetic susceptibility in 2D Ising model²¹: $kT\chi_{xx}(T) = C_{xx}(1 - T/T_c)^{-2.5} + C_{xx}(1 - T/T_c)^{-3.5} + O(1)$

$$C_{xx} \sim C_{xx} \rightarrow \sum_{k=1}^N \frac{m(d_k)}{2^{d_k}}$$

$$C_{xx} \sim C_{xx} \rightarrow \sum_{k=1}^N \frac{m(d_k)}{2^{d_k}}$$

using susceptibility integrals²²:

$$C_{xx} = 2 \int_{\beta_1, \beta_2} \frac{1}{A_{xx}(\beta_1, \beta_2, \dots, \beta_d)} d\beta_1 \dots d\beta_d$$

$$D_{xx} = 2 \int_{\beta_1, \beta_2} \frac{A_{xx}(\beta_1, \beta_2, \dots, \beta_d)}{A_{xx}(\beta_1, \beta_2, \dots, \beta_d)} d\beta_1 \dots d\beta_d$$

$$A_{xx} = \prod_{i=1}^d \left(\frac{1 + \sum_{k=1}^N \frac{m(d_k)}{2^{d_k}} \delta_{i, d_k} \right)^2$$

$$B_{xx} = \left(1 + \sum_{k=1}^N \frac{m(d_k)}{2^{d_k}} \right)^2$$

Numerical methods

Full tensor product quadrature

