Monte Carlo Simulations

Created by Rida Assaf (with Dr. E. de Doncker)
College of Engineering and Applied Sciences
Western Michigan University, Kalamazoo MI 49008

Problem and Applications

We present an efficient approach for Monte Carlo integration on GPUs (Graphics Processing Units).

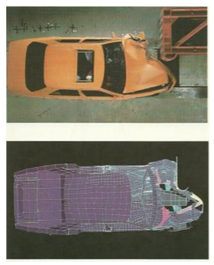
A Monte Carlo simulation uses the mean value of N sample point evaluations, which corresponds to an approximation of the integral:

$$\frac{1}{N} \int f(x_1, x_2, \ldots x_d)\, dx_1\, dx_2 \ldots dx_d$$

For a function with d parameters (coordinates) $x_1 \ldots x_d$

Some applications of MC simulations:

- Modeling in Computer Aided Design(e.g. Automotive Safety)
- Finite elements – using tessellations
- Molecular modeling
- Particle physics
- Finance (cash flow, mortgage obligations)
- Psychology/Biometrics (e.g. analysis of taste testing)
- Statistics



Methods

For testing, we used the Tesla M2090 Nvidia GPU card with the following specifications:

- Peak double precision floating point performance : 665 Gigaflops
- Peak single precision floating point performance : 1331 Gigaflops
- Memory Bandwidth(EEC off) : 177 GB/sec
- Memory size(GDDR5) : 6 GigaBytes
- CUDA cores : 512

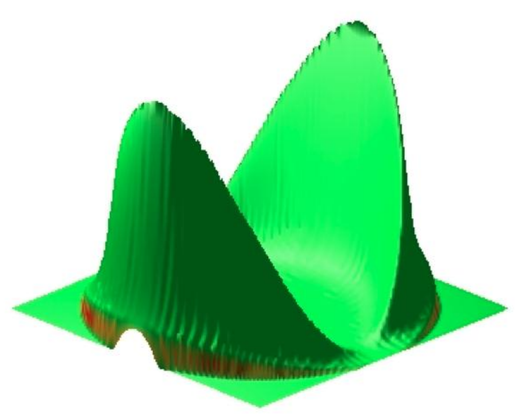
Example tests

The functions below (which we call the “Dice” functions) are of interest for modeling the behaviour of particle interactions in nuclear physics.

- Parallelization is needed in view of the singular behavior and computational expense as large sets of these are needed.
- Most problems have more parameters, therefore requiring higher dimensions. We use 2D functions for problem visualization. In the plots below, ε = 0.1

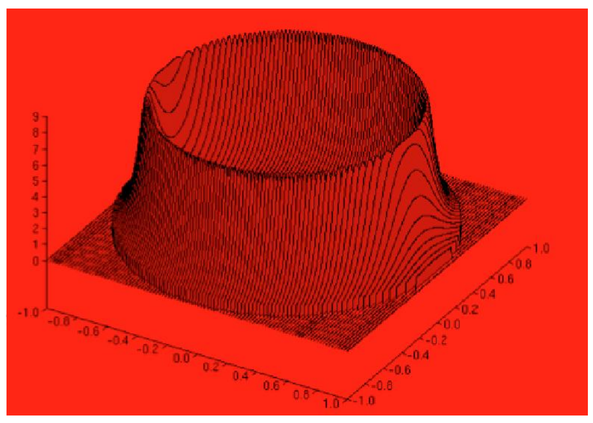
DICE2

$$\int_{-1}^{1}\int_{-1}^{1} dx_1 dx_2 \frac{\varepsilon x_2^2 \theta(1 - x_1^2 - x_2^2)}{(x_1^2 + x_2^2 - a^2)^2 + \varepsilon^2}$$



DICE3B

$$\int_{-1}^{1}\int_{-1}^{1} dx_1 dx_2 \frac{\varepsilon \sqrt{x_1^2 + x_2^2}\, \theta(1 - x_1^2 - x_2^2)}{(x_1^2 + x_2^2 - a^2)^2 + \varepsilon^2}$$



Results/Conclusions

Our simulation program is coded in CUDA C and contains three main parts:

- The Curand library is used to generate N random points on the GPU.
- A GPU kernel is launched to evaluate the Dice functions over these points.
- The Dice functions are represented as CUDA device functions.

The program achieves very good speedups(up to ~181, with respect to sequential execution) when tested on different problems. The table below shows the speedups and relative errors for different number of points used to integrate the Dice functions with ε = 1.0e-6. The Dice3 problem is a 3D version of Dice3b.

#point (Million)	10	50	100	150
Dice2	Speedup: 47 R.Error: 0.022	Speedup: 116 R.Error: 0.01	Speedup: 143 Error: 0.032	Speedup: 155 Error: 0.032
Dice3B	Speedup: 54 R.Error: 0.025	Speedup: 128 R.Error: 0.013	Speedup: 156 R.Error: 0.013	Speedup: 168 R.Error: 0.004
Dice3	Speedup: 63 R.Error: 0.071	Speedup: 143 R.Error: 0.055	Speedup: 170 R.Error: 0.055	Speedup: 181 R.Error: 0.001

In future work, the approach will be combined with multi-core and Distributed computations, on the cluster of the High Performance Computational Science Laboratory (HPCS), Department of Computer Science WMU.

References

- de Doncker, E., Kaugars, K., Cucos, L., Zanny, R.: Current status of the ParInt package for parallel multivariate integration. Proc. of Computational Particle Physics Symposium (CPP 2001),110–119
- De Doncker, E, Assaf, R: Integral Computations in Stochastic Geometry. (CGA 2013). 2013.
- Li, Shujun: Online Support for Multivariate Integration.(PhD Dissertation, Western Michigan University, 2005).
- Supercomputing and the Transformation of Science, W. J. Kaufmann III and L. L. Smarr, Publ. Scientific American Library, 1993.