Polytechnique Montréal
Department of Mechanical Engineering
Course Number : **MEC6215**
Course Name : **Méthodes Numériques en Ingénierie**
Professor : **Sébastien Leclaire**
Homework : **Array Programming (10%)**
Date of Delivery : **September 22th 2021 23:55**

Pedagogical Guidelines

- Submit a concise report of 5 pages maximum on Moodle (excluding first page, table of contents, and references), letter format, no narrow fonts, minimum 12pt, and margin 1.9cm;
- Submit your code that can reproduce your numerical results;
- All data (report and code) must be submitted in a single zip file "YOURMATRICULE.zip" (less than 100MB).
- Your report must be perfectly understandable when printed in black and white.

Objective

- Learn the basics of array programming with ArrayFire.

1 Simple Monte Carlo Integration (Press et al. [1996])

Suppose that we pick N random points, uniformly distributed in a multidimensional volume V. Call them x_1, ..., x_N. Then the basic theorem of Monte Carlo integration estimates the integral of a function f over the multidimensional volume,

$$\int f dV \approx V\langle f\rangle \pm V\sqrt{\frac{\langle f^2\rangle - \langle f\rangle^2}{N}} \tag{1}$$

Here the angle brackets denote taking the arithmetic mean over the N sample points,

$$\langle f\rangle \equiv \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{2}$$

$$\langle f^2\rangle \equiv \frac{1}{N}\sum_{i=1}^{N} f^2(x_i) \tag{3}$$

The "plus-or-minus" term in Eq. (1) is a one standard deviation error estimate for the integral, not a rigorous bound; further, there is no guarantee that the error is distributed as a Gaussian, so the error term should be taken only as a rough indication of probable error.

Suppose that you want to integrate a function g over a region W that is not easy to sample randomly. For example, W might have a very complicated shape. No problem. Just find a region V that includes W and that can easily be sampled, and then define f to be equal to g for points in W and equal to zero for points outside of W (but still inside the sample V). You want to try to make V enclose W as closely as possible, because the zero values of f will increase the error estimate term of Eq. (1).

Application The Monte Carlo integration algorithm is a very powerful integration method. Although the method is not very precise, it is still widely used in many fields (Cools [2002]). As an example, one of these important areas is uncertainty/reliability analysis in mechanical design (Huang and Du [2005], Sadoughi et al. [2018]). The integration variables are the design parameters, the function to be integrated is the joint probability distribution expressing the probability that the design parameters occur, and the integration domain represents the safety region of the system. The integral then represents the reliability of the entire system.

Algorithm The exercises below will allow you to program step-by-step a general N-dimensional Monte Carlo integration algorithm that can be executed, thanks to the ArrayFire library, on any CPUs or GPUs supporting the OpenCL or CUDA programming standard. The first thing to note is that usually in a real Monte Carlo integration algorithm, the complete set of evaluation points x_1, ..., x_N are not generated all at once to avoid *OOM* (a computer state with out of memory). They are generated in batches, let's say for this homework in batches of **nBatch** = 100000 evaluation points. Of course, this number may depend on the computational cost related to the evaluation of the function f. Then there is an iterative process, where for each iteration **it**, the function f is evaluated **nBatch** more times. If we define the value integralApproximation(**it**) to be the integral approximation after a total of **it** $\times$ **nBatch** evaluation points, the algorithm stops when:

$$|\text{integralApproximation}(\textbf{it}) - \text{integralApproximation}(\textbf{it-1})| < \textbf{absTol} \tag{4}$$

where the absolute tolerance **absTol** is provided before the algorithm start. Naturally, to avoid an infinite loop, it is necessary to specify a maximum number of iterations **itMax** for the algorithm. More advanced stopping criterion are also possible which could use a relative tolerance or even a combination of absolute and relative tolerance. However, one must be careful with a relative tolerance if the value of the integral is zero.

Exercises **Use the MS Visual Studio solution "MAIN_MONTECARLO.sln"** already provided in the homework statement. Fill in the blanks within the main file **"MAIN_MONTECARLO.cpp"** and functions to answer the following questions step-by-step. You may also notice that the correct value of each integral are also given and printed by the program so you can verify the progress of your work. To generate random numbers with ArrayFire, use the function *af::randu*. Please also visit **"arrayfire.org"** for more information about the different functions you can use with this library.

1. Compute the integral:

$$\int_{-4}^{5}\int_{-1}^{3} \left(x^2 - y^2\right)\, dx\, dy \tag{5}$$

   with an 2D Monte Carlo algorithm. To do this task you only have to **complete the function "myFunctionEx1"**, compile and execute the program. The function "monteCarloIntegral2" is already correct.

2. Compute an approximation of the number "pi" using a 2D Monte Carlo algorithm. You must use the fact that the surface of a disk is given by $S = \pi R^2$ and that the disk is centered

in $(0,0)$ with a radius of $1/2$. The bounding box used by the Monte Carlo algorithm is the boundary of the square domain $(x,y) \in [-1/2,1/2] \times [-1/2,1/2]$. This exercise is a typical example that may be found on the Web. To do this task you only need to **complete the function "myFunctionEx2"**. The function "monteCarloIntegral2" is again already correct.

3. Compute the integral:

$$\int_{-7}^{8}\int_{-4}^{5}\int_{-1}^{3}\left(x^2 - y^2 + z^3\right)\,dx\,dy\,dz \tag{6}$$

with an 3D Monte Carlo algorithm. To do this task you need to **complete the functions "myFunctionEx3" and "monteCarloIntegral3"**. To code the new "monteCarloIntegral3" function, you can reuse code from the "monteCarloIntegral2" function.

4. Compute the number "pi" using a 3D Monte Carlo algorithm. You must use the fact that the volume of a sphere is given by $V = 4\pi R^3/3$ and that the sphere is centered in $(0,0,0)$ with a radius of $1/2$. The bounding box used by the Monte Carlo algorithm is the boundary of the cubic domain $(x,y,z) \in [-1/2,1/2] \times [-1/2,1/2] \times [-1/2,1/2]$. To do this task you need to **complete the functions "monteCarloIntegral3" (already completed at step 3) and "myFunctionEx4"**.

5. Compute the volume of a torus:

$$V = \iiint_{\mathbb{D}} (1)\,dx\,dy\,dz \tag{7}$$

with an 3D Monte Carlo algorithm. The domain occupy by the torus is $\mathbb{D} = \{(x,y,z) \in \mathbb{R}^3 : a^2 - (c - \sqrt{x^2+y^2})^2 > z^2\}$ with $a = c = 1$. To do this task you need to **complete the functions "monteCarloIntegral3" (already completed at step 3) and "myFunctionEx5"**.

6. Compute the integral:

$$\int_{-1}^{1}\int_{1}^{2}\int_{-1}^{2}\int_{-1}^{3}\left(x^2 - y^2 + z^{3/2} - t^2\right)\,dx\,dy\,dz\,dt \tag{8}$$

with an 4D Monte Carlo algorithm. To do this task you need to **complete the functions "monteCarloIntegral4" and "myFunctionEx6"**.

7. Compute the time-average temperature $\bar{T}$ of a torus on the same domain $\mathbb{D}$ as in step 5 with an 4D Monte Carlo algorithm:

$$\bar{T} = \frac{1}{t_{\text{end}}}\int_0^{t_{\text{end}}}\iiint_{\mathbb{D}} T(x,y,z,t)\,dx\,dy\,dz\,dt \tag{9}$$

where $t_{\text{end}} = 6\pi$ and the temperature inside the torus is given by $T(x,y,z,t) = x^2y^2z^2(\cos(t/3)+1)$. To do this task you need to **complete the functions "monteCarloIntegral4" (already completed at step 6) and "myFunctionEx7"**.

By the end of step 7 above, you should understand how to do an integral with the Monte Carlo method. Now the real task of this homework begin and the goal is to generalize the Monte Carlo algorithm to an **N-dimensional and vectorized function** using array programming with ArrayFire. Note that $1/t_{\text{end}}$ is not part of the integrand in Eq. (9) and is not to be processed inside the Monte Carlo method.

A. Recompute the 3D integral of step 3 above. However, this time, to do this task you need to **complete the function "monteCarloIntegralN" and "myFunctionExA"**. You may note the following changes: The function pointer passed to the "monteCarloIntegralN" function, only accept one input *af::array* argument. This "input" argument needs to be a single array of size **nBatch** by **nDim** so that the function is able to evaluate many points at each call. The value **nDim** is the number of dimension of the problem. Also, the bounding box for the Monte Carlo algorithm are regroup in two **row** arrays, one for the minimal coordinates **xMin** and another one for the maximal coordinates **xMax**. To program the "monteCarloIntegralN" function, you may need to use many ArrayFire functions or array methods for array programming such as *.elements()*, *af::product<T>*, *af::tile*, *af::span*, *af::pow*, *af::sqrt*, *af::cos*, etc. However, note that more than one solution is possible.

B. Recompute the 4D integral of step 6 above, but with the function "monteCarloIntegralN". To do this task you need to **complete the functions "monteCarloIntegralN" (already completed at step A) and "myFunctionExB"**.

C. Recompute the 4D integral of step 7 above, but again with the function "monteCarloIntegralN". To do this task you need to **complete the functions "monteCarloIntegralN" (already completed at step A) and "myFunctionExC"**.

Congratulations, you now have a Monte Carlo algorithm able to compute very complicated integrals in any number of dimensions. Besides, your code may be executed on CPUs or GPUs.

In your report, show the values computed by your program. Mainly discuss your new "monteCarloIntegralN" function. Check and verify the performance of your "monteCarloIntegralN" function against "monteCarloIntegral4" and if you have access to an OpenCL or CUDA backend, please use it. Discuss of the advantages and inconvenients of array programming using ArrayFire.

References

Ronald Cools. Advances in multidimensional integration. *Journal of Computational and Applied Mathematics*, 149(1):1 – 12, 2002. ISSN 0377-0427. doi: httpsdoi.org10.1016S0377-0427(02)00517-4.

Beiqing Huang and Xiaoping Du. Uncertainty analysis by dimension reduction integration and saddlepoint approximations. *Journal of Mechanical Design*, 128(1):26–33, 03 2005. ISSN 1050-0472. doi: 10.11151.2118667. URL `httpsdoi.org10.11151.2118667`.

William H. Press, Saul a. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in Fortran 77 the Art of Scientific Computing. Second Edition*, volume 1. 1996. ISBN 052143064X.

Mohammad Kazem Sadoughi, Meng Li, Chao Hu, Cameron A. MacKenzie, Soobum Lee, and Amin Toghi Eshghi. A high-dimensional reliability analysis method for simulation-based design under uncertainty. *Journal of Mechanical Design*, 140(7), 04 2018. ISSN 1050-0472. doi: 10.11151.4039589. URL `httpsdoi.org10.11151.4039589`. 071401.