Introduction to CUDA Parallel Programming
CUDA平行計算導論

https://ceiba.ntu.edu.tw/1092Phys8061_CUDA

Professor Ting-Wai Chiu (趙挺偉)
Email: twchiu@phys.ntu.edu.tw
Physics Department
National Taiwan University

This lecture will cover:

- Monte Carlo Simulation, Importance Sampling,
- Metropolis Algorithm
- Ising Model
- Monte Carlo Simulation of 2D Ising Model

Introduction

Computer simulations have become an integral part of contemporary basic and applied sciences, and have been serving as a bridge between experimental and theoretical approaches. For the theorists, it is often difficult to obtain exact analytic solutions, or numerical solutions by direct methods. Thus, to use Monte Carlo simulation with importance sampling turns out is the only viable way to obtain numerical solutions. For the experimentalists, it is necessary to simulate the physical parameters of the detectors before they are built. Moreover, some experiments are too expensive/difficult to perform in practice, thus must rely on numerical simulations to provide the answers.

Introduction

Systems with a large number of degrees of freedom are often of interest in physics.

Among these are ∞ many field variables in a region of space-time, the many electrons in an atom, or the many atoms in a rock. To investigate such systems often involves the evaluation of integrals of very high dimension.

Exact analytic solutions of these integrals have not been known except for a few simple models.

Even straightforward evaluations using trapezoidal or Simpson's rule are often out of question !

Introduction

For example, the classical partition function of a gas of N atoms at a temperature T interacting through a pairwise potential ϕ is proportional to

$$Z = \int d^3 r_1 \cdots d^3 r_N \exp\left(-\frac{1}{kT} \sum_{i<j} \phi\left(r_{ij} \right) \right)$$

Suppose that each coordinate takes 10 different values, the integrand must be evaluated at 10^{3N} points, which is an impossible task except for very small values of N.

Monte Carlo Simulation

The essential idea of Monte Carlo integration is not to evaluate the integrand at each point of the 3N dimensional space, but to sample a point with a probability proportional to the Boltzmann distribution in the integrand.

For limited computing resources, Monte Carlo methods can estimate this multiple integral much better than the direct integration.

$$< O >= \frac{1}{Z}\int d^3 r_1 \cdots d^3 r_N \; O(r_1,\cdots,r_N)\exp\left(-\frac{1}{kT}\sum_{i<j}\phi\left(r_{ij}\right)\right)$$

$$\approx \frac{1}{M}\sum_{k=1}^{M} O(\{r_1,\cdots,r_N\}_k) + O\left(\frac{1}{\sqrt{M}}\right)$$

1-Dimensional Monte-Carlo Integration

Consider the integral

$$I = \int_0^1 f(x)\, dx$$

A way of evaluating I is to choose N points at random in the interval [0, 1], where each point has equal probability being selected.

$$I \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

1-D Monte-Carlo Integration (cont)

This is called simple sampling. The uncertainty of this evaluation can be estimated using central limit theorem. In the limit $N \to \infty$, the distribution of I is a Gaussian distribution with variance σ_{I}^2 equal to $\sigma_{\mathrm{f}}^2 / N$, where σ_{f}^2 is the variance of f(x),

$$\sigma_I^2 \approx \frac{1}{N}\sigma_f^2 = \frac{1}{N}\left[\frac{1}{N}\sum_{i=1}^{N} f_i^{\,2} - \left(\frac{1}{N}\sum_{i=1}^{N} f_i\right)^2\right]$$

Two important aspects of Monte Carlo integration:

(1) σ_I decreases as $\frac{1}{\sqrt{N}}$

(2) σ_I is proportional to σ_f

1-D Monte-Carlo Integration (cont)

Now (2) suggests that σ_{I} will be minimum if f is as smooth as possible. In the limit f is a constant, $\sigma_{\mathrm{f}} = 0$, then the integral can be evaluated exactly by only one point. This suggests that if we introduce a weight function w(x) satisfying

$$\int_0^1 dx\, w(x) = 1$$

and choose w(x) such that it behaves approximately as f, and transform the integral I as the following

$$\begin{aligned} I &= \int_0^1 dx\, w(x)\, \frac{f(x)}{w(x)} \\ &= \int_0^1 dy\, \frac{f(x(y))}{w(x(y))}, \qquad y(x) \equiv \int_0^x w(t)\, dt \\ &= \int_0^1 dy\, F(y) \end{aligned}$$

1-D Monte-Carlo Integration (cont)

Then the uncertainty σ_I can be minimized, since the transformed function $F = \frac{f}{w}$ is very smooth over the entire interval.

Sampling $y \in [0,1]$ uniformly amounts to sampling $x \in [0,1]$ with probability distribution w(x), hence more points are sampled at those locations where f is large.

This is called importance sampling, the essential idea of Monte Carlo simulations .

$$I = \int_0^1 dx\, w(x)\, \frac{f(x)}{w(x)} = \int_0^1 dy\, \frac{f\left(x(y)\right)}{w\left(x(y)\right)} \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f\left(x(y_i)\right)}{w\left(x(y_i)\right)}$$

where $dy = dx\, w(x)$, and $y(x) = \int_0^x w(t)\, dt$

1-D Monte-Carlo Integration (cont)

Example $f(x) = \dfrac{1}{1+x^2}$ $I = \int_0^1 f(x)\,dx = \dfrac{\pi}{4}$

$$\text{choose} \quad w(x) = \frac{4-2x}{3}$$

$$y = \int_0^x w(x)\,dx = \frac{x(4-x)}{3} \quad \Rightarrow \quad x = 2 - \sqrt{4-3y}$$

Computational results show that using the importance sampling does yield much smaller error than using the simple sampling !

1-D Monte-Carlo Integration (cont)

Example $\quad f(x) = \dfrac{1}{1+x^2} \qquad I = \int_0^1 f(x)\,dx = \dfrac{\pi}{4}$

$$\text{choose} \quad w(x) = \frac{4-2x}{3}$$

$$y = \int_0^x w(x)\,dx = \frac{x(4-x)}{3} \quad \Rightarrow \quad x = 2 - \sqrt{4-3y}$$

Computational results show that using the importance sampling does yield much smaller error than using the simple sampling !

```
Monte-Carlo integration of one-dimensional integral
How many random numbers to be generated ? 10000

Simple sampling:      0.7857936029 +/- 0.0016056079
Importance sampling:  0.7855253386 +/- 0.0001992327
Exact solution:       0.7853981634
```

1-D Monte-Carlo Integration (cont)

Example $\quad f(x) = \dfrac{1}{1+x^2} \qquad I = \int_0^1 f(x)\,dx = \dfrac{\pi}{4}$

$$\text{choose} \quad w(x) = \frac{4-2x}{3}$$

$$y = \int_0^x w(x)\,dx = \frac{x(4-x)}{3} \quad \Rightarrow \quad x = 2 - \sqrt{4-3y}$$

Computational results show that using the importance sampling does yield much smaller error than using the simple sampling !

In general, it is difficult to find an appropriate weight function w(x) such that x can be solved analytically in terms of y. Except for a few special cases, it is impractical to generalize this scheme to evaluate integrals in higher dimensions.

Metropolis Algorithm

The Algorithm of Metropolis et al.
(Metropolis, Rosenbluth, Rosenbluth, Teller, Teller, 1953)

Metropolis algorithm is a very general scheme to produce random variables with a given probability distribution of arbitrary form. It only requires the calculation of weight function at each step of updating.

Suppose that we want to generate variables $\{\vec{x}\}$ with probability distribution $w(\vec{x})$ in a multi-dimensional space. The Metropolis algorithm generates a sequence of points $\{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_{n-1}, \vec{x}_n, \cdots\}$ according to the following rule:

Metropolis Algorithm

At each point, say $\vec{x}_n$, a new trial step $\vec{x}_t$ is generated at random but uniformly within a neighborhood of $\vec{x}_n$.
This trial step $\vec{x}_t$ is then accepted as $\vec{x}_{n+1}$ if

$$\eta \le \frac{w(\vec{x}_t)}{w(\vec{x}_n)}$$

where η is a random number uniformly distributed in [0,1].
The probability of accepting the transition is

$$A(\vec{x}_n \to \vec{x}_t) = \min\left(1, \frac{w(\vec{x}_t)}{w(\vec{x}_n)}\right)$$

In general, any algorithm with $T(\vec{x} \to \vec{y})$ and $A(\vec{x} \to \vec{y})$ satisfying the detailed balance

$$\frac{T(\vec{y} \to \vec{x})A(\vec{y} \to \vec{x})}{T(\vec{x} \to \vec{y})A(\vec{x} \to \vec{y})} = \frac{w(\vec{x})}{w(\vec{y})}$$

will work, and eventually leads to the equilibrium probability distribution $w(\vec{x})$. However, we require an algorithm which is

1. An efficient computation.
2. Acceptance probability $A(\vec{x} \to \vec{y})$ is high.
3. Weak correlation between successive configurations.

METROPOLIS ALGORITHM:

$$T(\vec{x} \to \vec{y}) = T(\vec{y} \to \vec{x})$$

$$A(\vec{x} \to \vec{y}) = \min\left(1, \frac{w(\vec{y})}{w(\vec{x})}\right)$$

Problems of Metropolis Algorithm

The drawback of Metropolis algorithm is that the probability of accepting a new configuration is very small unless the new one is very close to the old one.

In either case, this leads to very strong correlated sequences of configurations and thus extremely long relaxation and correlation times.

This becomes a serious problem especially at the critical point (phase transition point).
This is called critical slow down .

Metropolis algorithm to generate $\{x_0, \cdots, x_{N-1}; x_i \in (0,1)\}$ with probability distribution function $w(x)$

(1) Initially, to generate a random $x_{\text{old}} \in (0,1)$ with uniform deviate.
and compute $w_{\text{old}} = w(x_{\text{old}})$. Set $i = 0$.

(2) To generate a new random $x_{\text{new}} \in (0,1)$ with uniform deviate,
and compute $w_{\text{new}} = w(x_{\text{new}})$.

(3) If $w_{\text{new}} \geq w_{\text{old}}$, then accept x_{new}, and set $x_{\text{old}} = x_{\text{new}}, w_{\text{old}} = w_{\text{new}}$;
otherwise, generate a random number $r \in (0,1)$ with uniform deviate.
If $r < w_{\text{new}} / w_{\text{old}}$, then accept x_{new}, and set $x_{\text{old}} = x_{\text{new}}, w_{\text{old}} = w_{\text{new}}$.

(4) Set $x[i] = x_{\text{old}}$, $i \to i+1$.
(Note that if x_{new} is rejected, x_{old} is saved in the sequence.)

(5) If $i < N$, go to (2); otherwise exit.

Pseudocode of using Metropolis algorithm to generate $\{x_0, \cdots, x_{N-1}\}$ with probability distribution function $w(x)$

```
x_old = gsl_rng_uniform(rng);
w_old = w(x_old);
for(i=0;i<N;i++) {
    x_new = gsl_rng_uniform(rng);
    w_new = w(x_new);
    if(w_new >= w_old) {
      w_old = w_new;
      x_old = x_new;
    }
    else {
       r = gsl_rng_uniform(rng);
       if(r < w_new/w_old ) {
         w_old = w_new;
         x_old = x_new;
       }
    }
    x[i] = x_old;       /* Note that if x_new is rejected, x_old is saved in the sequence. */
}
```

Ising Model

https://en.wikipedia.org/wiki/Ising_model

Consider spins on a D-dimensional lattice (torus) with energy

$$E(\sigma) = -J \sum_{<i,j>} \sigma_i \sigma_j - B \sum_i \sigma_i$$

$$\sigma_i = \begin{cases} +1 & \text{spin up} \quad \uparrow \\ -1 & \text{spin down} \quad \downarrow \end{cases}$$

J : coupling strength

B : external magnetic field

where σ_i denotes the spin variable at the site $i = (i_1, i_2, \cdots, i_D)$, B is the external magnetic field, J is the coupling strength between nearest neighboring spins, and $< i, j >$ denotes that i and j are nearest neighboring sites.

Ising Model

Partition function $Z = \sum_{\{\sigma\}} \exp\left(-E(\sigma)/kT\right)$

$K \equiv J/kT$

$h \equiv B/kT$

$$= \sum_{\{\sigma\}} \exp\left(K \sum_{<ij>} \sigma_i \sigma_j + h \sum_i \sigma_i \right)$$

If we can evaluate the partition function, then we can obtain all statistical properties of the system.

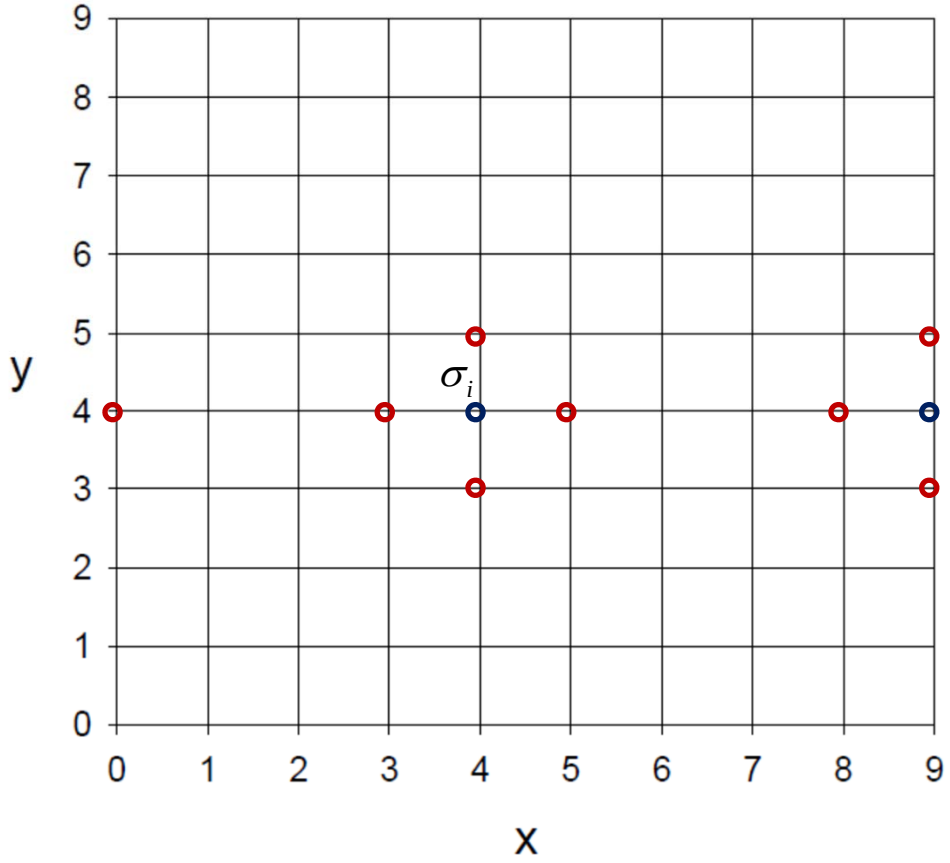
magnetization density

$$\langle M \rangle = \frac{1}{N_{\text{sites}}} \left\langle \sum_i \sigma_i \right\rangle = \frac{1}{N_{\text{sites}}} \frac{1}{Z} \sum_{\{\sigma\}} \sum_i \sigma_i e^{-E(\sigma)/KT} = \frac{1}{N_{\text{sites}}} \frac{\partial}{\partial h} \ln\left(Z\right)$$

energy density

$$\langle E \rangle = \frac{1}{N_{\text{sites}}} \frac{1}{Z} \sum_{\{\sigma\}} E(\sigma) e^{-E(\sigma)/kT} = \frac{-J}{N_{\text{sites}}} \frac{\partial}{\partial K} \ln Z - B \langle M \rangle$$

Ising Model on the 2D Lattice with periodic b.c.



2D Lattice with periodic b.c. = Torus

Metropolis Algorithm for MC of Spin Systems

(a) The new spin σ_i' is selected with an arbitrary prob. distribution satisfying $\mathrm{T}(\sigma_\mathrm{i} \to \sigma_i') = \mathrm{T}(\sigma_i' \to \sigma_\mathrm{i})$

(b) Compute the change in energy $\Delta E = E(\sigma_\mathrm{i}') - E(\sigma_\mathrm{i})$

(c) If $\Delta E \le 0$, then σ_i' is accepted, otherwise generate a random number $r \in (0,1)$ with uniform deviate.

If $r \le \exp(-\Delta E / kT)$, then σ_i' is accepted.

$$A(\sigma_i \to \sigma_i') = \min\left(1, \frac{\exp[-E(\sigma')/kT]}{\exp[-E(\sigma)/kT]}\right)$$

(d) Repeat the same procedure for another spin

Monte Carlos Simulation of 2D Ising Model with Metropolis Algorithm

```
for(j1=0; j1<ny; j1++) {
  for(i1=0; i1<nx; i1++) {
    k = i1 + j1*nx;
    old_spin = spin[k];
    new_spin = -old_spin;
    k1 = fw[i1] + j1*nx;     // right
    k2 = i1 + fw[j1]*nx;     // top
    k3 = bw[i1] + j1*nx;     // left
    k4 = i1 + bw[j1]*nx;     // bottom
    spins = spin[k1] + spin[k2] + spin[k3] + spin[k4];
    de = -(new_spin - old_spin)*(spins + B);
    if((de <= 0.0) || (gsl_rng_uniform(rng) < exp(-de/T))) {
           // if (new energy <= old energy) or (r < exp(-dE/T))
           // accept the new spin
      spin[k] = new_spin;          // accept the new spin;
    }
  }
} // The complete CPU code is in twqcd80:/home/cuda_lecture_2021/Ising2D
```

Monte Carlos Simulation of 2D Ising Model with Metropolis Algorithm (cont)

After thermalization, the probability of getting a spin configuration $\{\sigma\}$ is proportional to $\exp\{-E(\sigma)/kT\}$, then the expectation value of any observable can be measured by averaging over the configurations.

$$\langle O \rangle = \frac{1}{Z}\sum_{\{\sigma\}} O(\{\sigma\})\, e^{-E(\sigma)/kT} \approx \frac{1}{M}\sum_{k=1}^{M} O(\{\sigma\}_k) + O\left(\frac{1}{\sqrt{M}}\right)$$

magnetization density

$$\langle M \rangle = \frac{1}{N_{\text{sites}}}\left\langle \sum_i \sigma_i \right\rangle \approx \frac{1}{M}\sum_{k=1}^{M} \frac{1}{N_{\text{sites}}}\left(\sum_i \sigma_i\right)_k + O\left(\frac{1}{\sqrt{M}}\right)$$

energy density

$$\langle E \rangle = \frac{1}{N_{\text{sites}}}\frac{1}{Z}\sum_{\{\sigma\}} E(\sigma) e^{-E(\sigma)/kT} \approx \frac{1}{M}\sum_{k=1}^{M} \frac{1}{N_{\text{sites}}} E(\{\sigma\}_k) + O\left(\frac{1}{\sqrt{M}}\right)$$