3rd Campus on Numerical

Lattice Quantum Field Theory

Numerical Integration and Harmonic Oscillator

Xiaonu Xiong

Part I: Numerical Integration and Harmonic Oscillator

Category of numerical integration algorithm

- **Deterministic algorithm**

  Return ***identical*** results with the same setup, predictable

  - example: Rectangle approximation, Trapezium approximation
    😄 : high performance in low dimension, reliable error estimation
    😢 : poor performance in high dimension

- **Non-Deterministic algorithm**

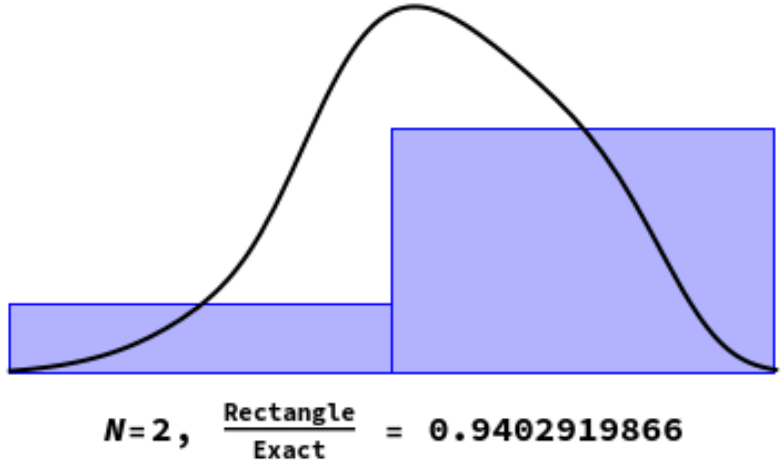
  Return ***different*** results with the same setup, unpredictable

  - example: MISER algorithm, Vegas algorithm
    😄 : high performance in high dimension
    😢 : usually slow, probability-based error estimation

Deterministic algorithm

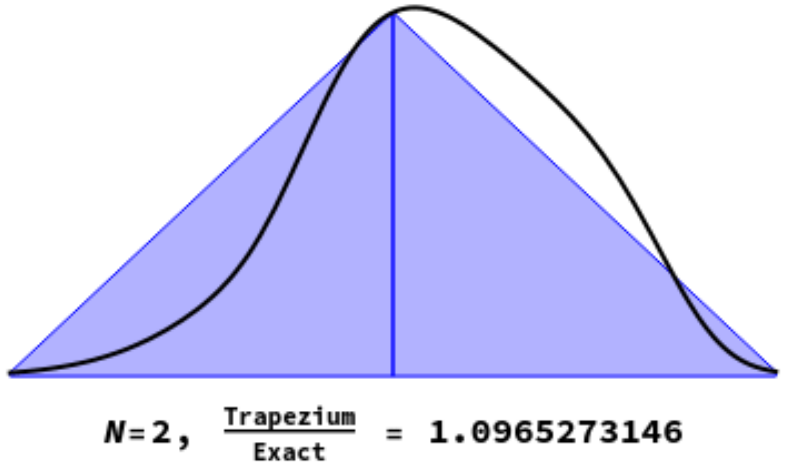
- 1-d Example integrand: $f(x) = \exp(-x^2) + \exp[-\left(x - \frac{1}{2}\right)^4]$
  - Rectangle approximation (middle-point rule)

$$\int_a^b dx\, f(x) = \lim_{N\to\infty} \sum_{i=1}^{N} \delta x f\left(a + \left(i + \frac{1}{2}\right)\delta x\right), \quad \delta x = \frac{b-a}{N}$$



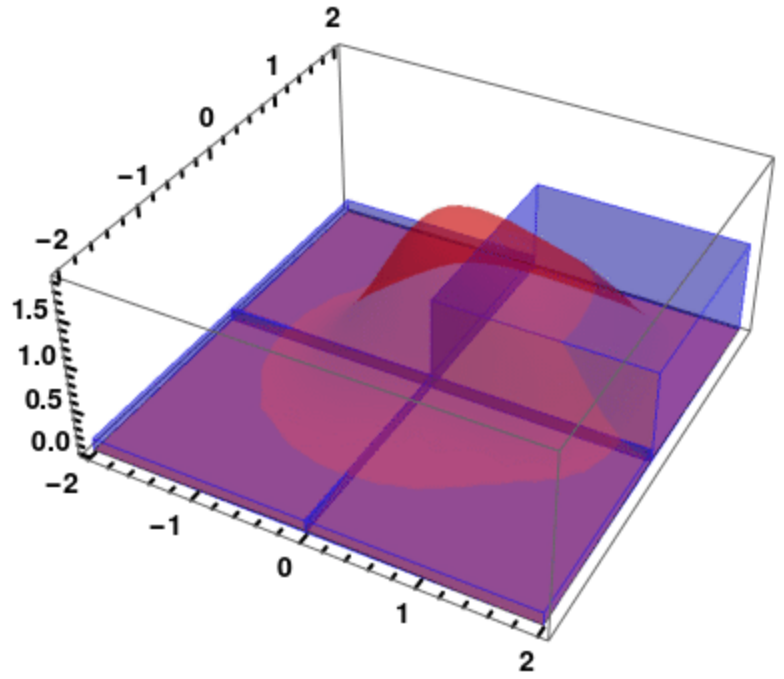
- 1-d Example integrand: $f(x) = \exp(-x^2) + \exp[-\left(x - \frac{1}{2}\right)^4]$
  - Trapezium approximation

$$\int_a^b dx\, f(x) = \lim_{N\to\infty} \sum_{i=1}^{N} \frac{\delta x}{2} \left[f\left(a + i\delta x\right) + f\left(a + (i+1)\,\delta x\right)\right], \quad \delta x = \frac{b-a}{N}$$



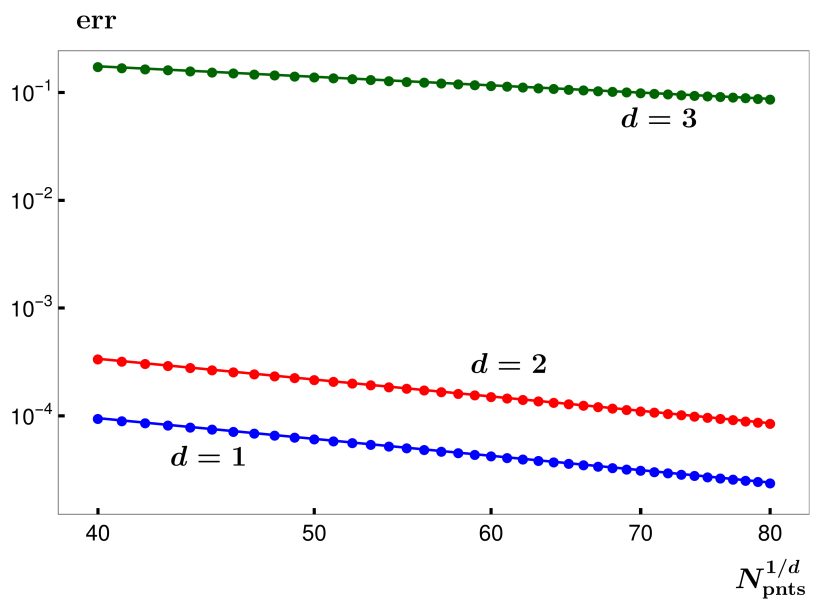
- 2-d Example integrand: $f(x) = \exp(-x^2 - y^2) + \exp[-\left(x - \frac{1}{2}\right)^4 - \left(y - \frac{1}{2}\right)^4]$
  - Cuboid approximation

$$\int_a^b dx \int_c^d dy\, f(x,y) = \lim_{N_1,N_2\to\infty} \sum_{i,j=1}^{N_1,N_2} \delta x \delta y f(a + i\delta x, c + j\delta y), \quad \delta x = \frac{b-a}{N_1}, \delta y = \frac{d-c}{N_2}$$



$N = 2^2$, $\frac{\text{Cuboid}}{\text{Exact}} = 0.8977738358$

- Error estimation
  - 1-dimension
    - rectangle and trapezium approximation: error $\sim \mathcal{O}\left(N_{\text{pnts}}^{-2}\right)$
    - > Simposon rule: error $\sim \mathcal{O}\left(N_{\text{pnts}}^{-4}\right)$
  - d-dimension
  - error $\sim \mathcal{O}\left(N_{\text{pnts}}^{-\lambda_1/d}\right)$, λ_1: scaling power in 1-d case



$$\ln \text{err} = \lambda_1 \ln N_{\text{pnts}}^{-\lambda_1/d} + C$$
$$\implies \text{err} = e^C N_{\text{pnts}}^{-\lambda_1/d}$$

Fit $d = 1$: $\lambda_1/d = 1.985$ 😄

Fit $d = 2$: $\lambda_1/d = 0.992$ 😒

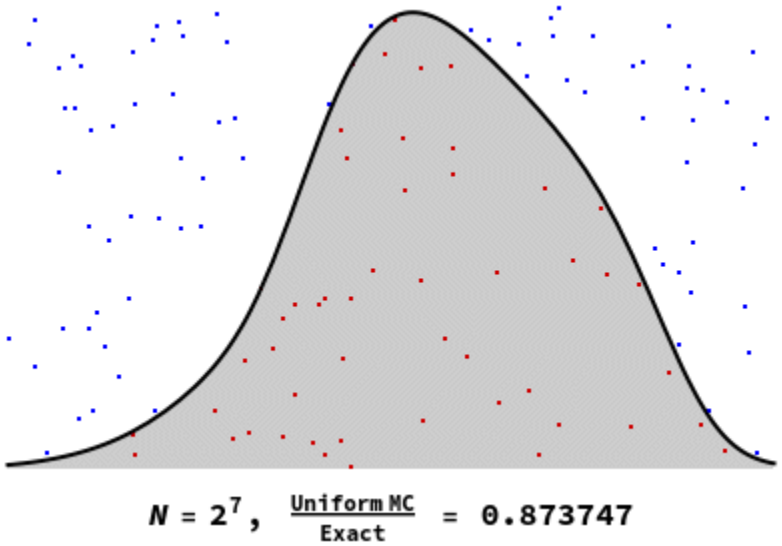
Fit $d = 3$: $\lambda_1/d = 0.668$ 😢

Monte-Carlo Integration

- Hit-or-Miss method

$$\int_a^b dx\, f(x) = \lim_{N\to\infty} \frac{N_{0<y<f(x)}}{N} \times A = P(0 < y < f(x)) \times A$$

A denotes the area of rectangular sampling region



- Hit-or-Miss method

$$\lim_{\Delta x \to 0} N_{x_i < x < x_i + \Delta x_i \wedge 0 < y < f(x_i)} = N \lim_{\Delta x \to 0} \frac{f(x_i)\Delta x_i}{M(b-a)}$$

M: denotes the hight of rectangular sampling region

Taking $\Delta x_i = \frac{b-a}{N}$

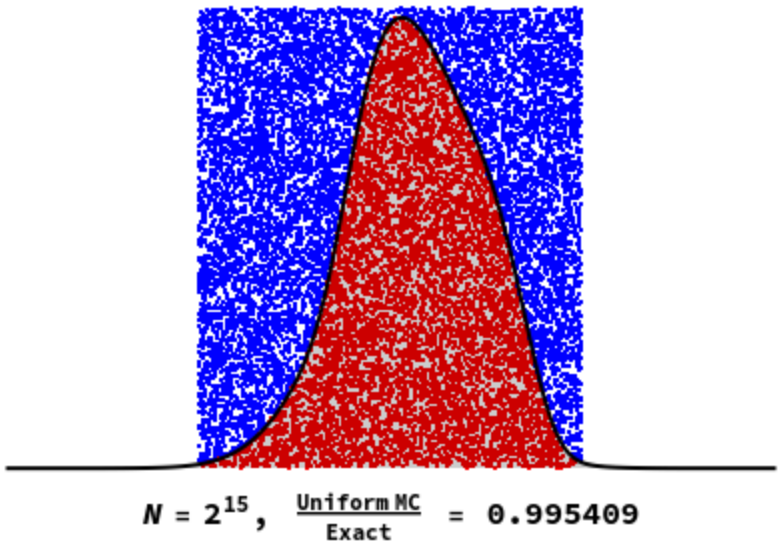
$$\lim_{\Delta x \to 0} N_{0 < y < f(x)} = N \sum_i \frac{f(x_i)}{NM} = \frac{N}{M} \left(\frac{1}{N} \sum_i f_i \right) = \frac{N}{M} \langle f \rangle$$

$$\implies \int_a^b dx\, f(x) = \lim_{N \to \infty} \frac{N_{0 < y < f(x)}}{N} \times M(b-a) = \langle f(x) \rangle (b-a) = \left\langle \frac{f(x)}{(b-a)^{-1}} \right\rangle$$

Probability interpretation of integration

$$\int_a^b dx\, f(x) = \left\langle \frac{f(x)}{p(x)} \right\rangle$$

- Hit-or-Miss method
  - Defects:
    - lots of sampling points are wasted in non-essential regions and become worse as dimension grows
    - Requires $M \geq \max(f)$, prrior information of $f(x)$



- Hint from

$$\int_a^b dx\, f(x) = \left\langle \frac{f(x)}{p(x)} \right\rangle = \int_a^b dx\, \frac{f(x)}{p(x)} p(x)$$

  If we chose $p(x)$ closely assemble the profile of $f(x)$, or equivalently $f(x)/p(x)$ is raughly a constant, then the wasted sampling points are largely reduced.

- Impotance sampling
  Sampling x_i according to the importance of integrand to integration

$$\text{larger/smaller } f(x) \;\leftrightarrow\; \text{larger/smaller } p(x)$$

- Impotance sampling
  - Approximate the profile of integrand $f(x)$ by some probability distribution function $p(x)$
  - Generate sample points $\{x_1, x_2, \cdots, x_N\}$ according to $p(x)$
  - Integral estimator

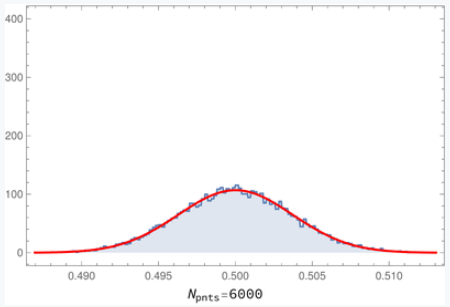
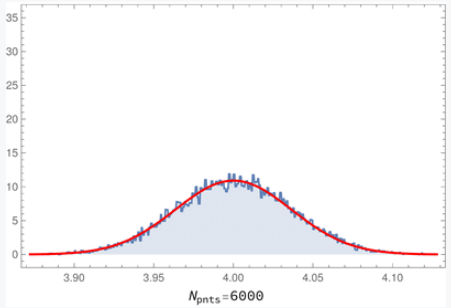
$$\int_a^b dx\, f(x) = \int_a^b dx\, \left[\frac{f(x)}{p(x)}\right] p(x) = \left\langle \frac{f(x)}{p(x)} \right\rangle = \boxed{\frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}}$$

> $p(x_i) = (a-b)^{-1}$reduce to uniform sampling case and $\int_a^b dx\, f(x) = \frac{b-a}{N}\sum_i f(x_i)$

  - $p(x)$ does not necessarily resemble the profile of $f(x)$ exactly.

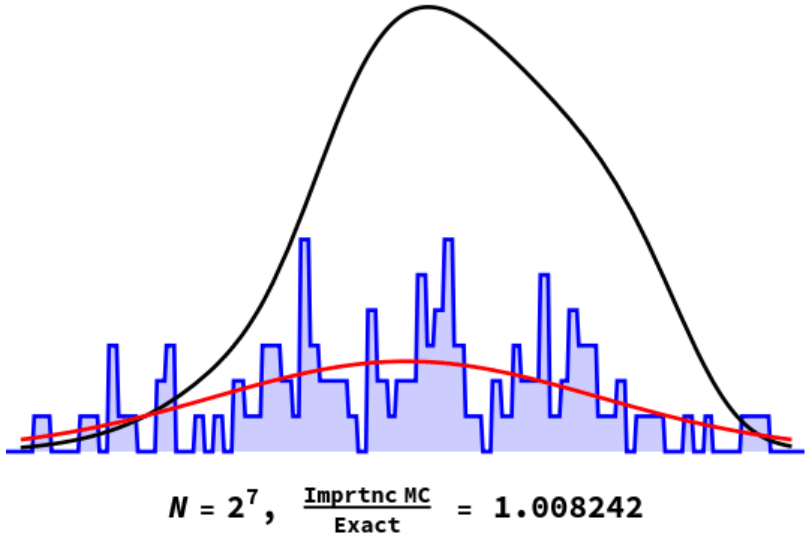
> "exactly" means $p(x) = f(x)/\int dx\, f(x)$, but impossible before the integration is known

- Error estimation of Monte-Carlo algorithm
  - Theoretical base: Central Limit Theorem
    - Suppose $\xi \sim p(\xi)$, $\mu = \int d\xi\, \xi p(\xi)$, $\sigma^2 = \int dx\, (\xi - \mu)^2 p(x)$
      1. Sampling N samples: $\xi_1, \xi_2, \cdots, \xi_N \implies \tilde{\mu} = \sum_{i=1}^{N} \xi_i / N$
      2. Repeat 1 for M times $\implies \tilde{\mu}_1, \tilde{\mu}_2, \cdots, \tilde{\mu}_M$
    - The distribution of $\tilde{\mu}$ approaches $N\left(\mu, \sigma/\sqrt{N}\right)$ if N is large enough
  - Demonstation: $M = 2 \times 10^4$, $N = 6 \times 10^3 \sim 6 \times 10^4$
    - Red: $\frac{1}{\sqrt{2\pi}\sigma_N} \exp[-x^2/2\sigma_N^2]$, $\sigma_N = \sigma/\sqrt{N}$, Blue: Histrogram distribution of $\tilde{\mu}$

$\xi \sim U(0,1)$	$\xi \sim \chi(4)$
$\mu = 1/2$, $\sigma = 1/2\sqrt{3}$	$\mu = 4$, $\sigma = 2\sqrt{2}$
Distribution of $\tilde{\mu}$ 	Distribution of $\tilde{\mu}$ 

- Error estimation of Monte-Carlo algorithm
  - Error estimation is in probability sense: the true value has 68.3% probability lying in $-\frac{\sigma}{\sqrt{N}} + \mu \leq \tilde{\mu} \leq \frac{\sigma}{\sqrt{N}} + \mu$
  - **Error scaling ~ $\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$ and DOES NOT dependt on the dimension** 😄
  - Increase accuracy by: $N \uparrow$: error/2 requires $4N$, $\sigma \downarrow$: variance reduction

- Importance sampling
  - 1-D example: $x \sim p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$



    - red: taget $p(x)$
    - light-blue: histogram distribution of sample

when the N_{pnts} growing *larger*, the histogram distribution $\rightarrow p(x)$.

- Importance sampling
  - 😢
    - prior information of integrand is required for choosing a proper $p(x)$.
    - difficult to choose a proper $p(x)$ in high dimensional integration case.
    - Can not adopt to different integrands with distinct profile.
  - ***Solution***

    An adaptive sampling algorithm that adjust itself to mimic different integrands

Vegas Algorithm

- Vegas Algorithm

> G.P. Lepage, *A New Algorithm for **Adaptive Multidimensional Integration***, *Journal of Computational Physics 27, 192–203, (1978)*

- 😄
  - No prior information about the profile of $f(x)$ (integrand) is required, automatically generate $p(x)$ approximating $f(x)$'s profile.
  - Converges faster than naive hit-or-miss sampling.

- Vegas Algorithm
  - ❶ Uniformly generate $\boldsymbol{x} = x_1, \cdots, x_{N-1}$ for $0 < x_i < 1$ and $\Delta x_i = x_i - x_{i-1}$
  - ❷ Subdividing Δx_i into $m_i + 1$ intervals, where

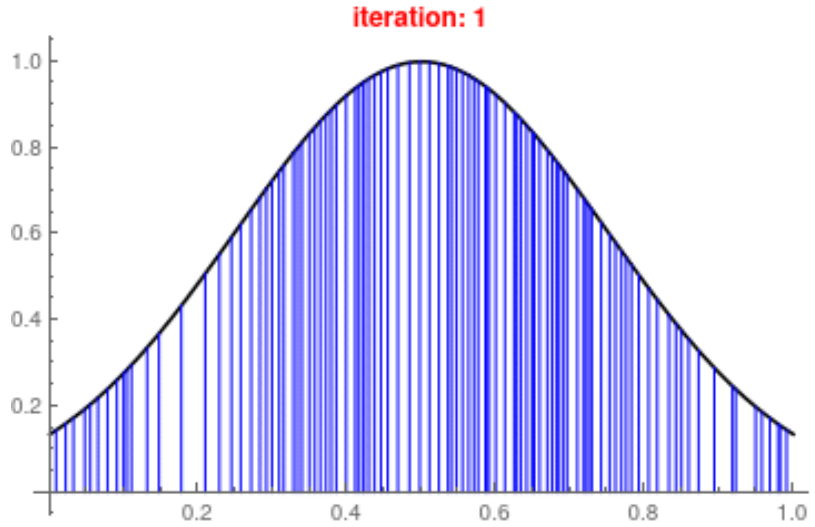
$$m_i = K \left[\left(\frac{\bar{f}_i \Delta x_i}{\sum_j \bar{f}_j \Delta x_j} - 1 \right) \left(\log \frac{\bar{f}_i \Delta x_i}{\sum_j \bar{f}_j \Delta x_j} \right)^{-1} \right]^{\alpha},$$

$$\bar{f}_i = \sum_{x \in [x_{i-1}, x_i]} |f(x)| \approx \frac{1}{\Delta x_i} \int_{x_i}^{x_{i+1}} dx \; |f(x)|$$

Empirically, $1 < \alpha < 2$, $K \sim \mathcal{O}\left(10^3\right)$

  - > There exists other form of suggesting m_i, e.g.
    >
    > $$m_i = K \left(\frac{\bar{f}_i \Delta x_i}{\sum_j \bar{f}_j \Delta x_j} \right)$$
    >
    > Practically, m_i is used to damp the subdividing procedure and avoid rapid change of intervals' size (introducing Unstable effects)

- Vegas Algorithm
  - ❸ Merge all subdivisions back into N intervals $\boldsymbol{x} = x'_1, x'_2, \cdots, x'_{N-1}$
  - ❹ Repeat ❶ ❷ ❸ untill optimal grid is reached (m_i become constant with respect to iteration number)
  - In ❸, $N_{\mathrm{dvsns}} \approx \mathrm{const} \implies$ adjusting $\Delta x_i \downarrow\uparrow$ when $|f(x)| \uparrow\downarrow$.



As shown in the animation, as iteration proceed, the region has larger (smaller) contribution to the integral gets denser subdivision and smaller intervals.

- Vegas Algorithm
  - Multi-dimension case (e.g. $2-d$)
    - Assume the probability distribution is factorizable along axes

$$p(x,y) = p_x(x)p_y(y)$$

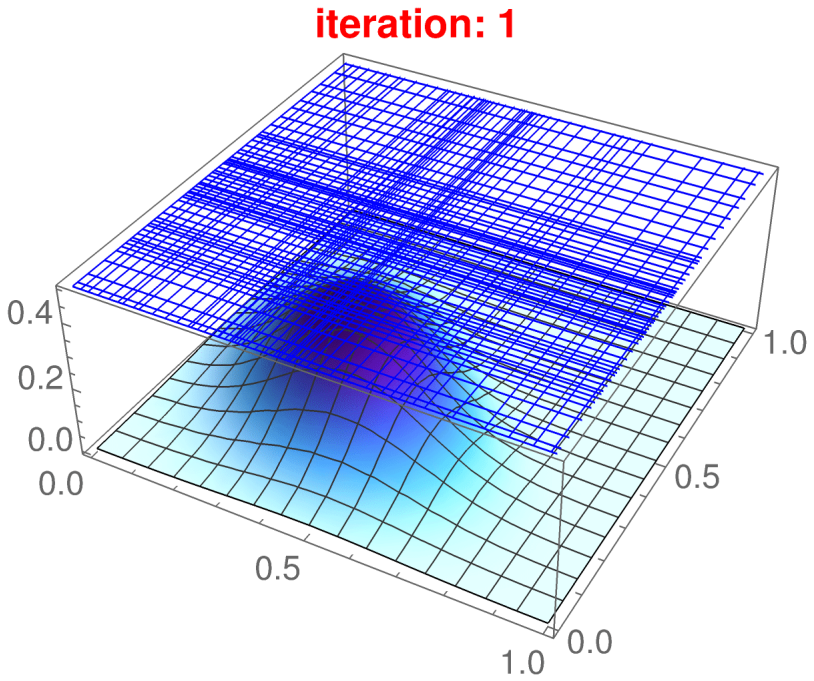
    - Replacing $\bar{f}_i$ in step 2 of 1 dimensional case by

$$\bar{f}^2_{x,i} \equiv \sum_{x\in[x_{i-1},x_i]} \sum_{y\in[0,1]} \frac{f(x,y)^2}{p_y(y)^2} \approx \frac{1}{\Delta x_i} \int_{x_i}^{x_{i+1}} dx \int_0^1 dy\, f(x,y)^2$$

$$\bar{f}^2_{y,i} \equiv \sum_{x\in[y_{i-1},y_i]} \sum_{x\in[0,1]} \frac{f(x,y)^2}{p_x(x)^2} \approx \frac{1}{\Delta y_i} \int_{y_i}^{y_{i+1}} dy \int_0^1 dx\, f(x,y)^2$$

    - Then calculate the corresponding $m_{x,i}$, $m_{y,i}$ and flow step ❸, ❹ in x, y-direction separately

- Vegas Algorithm
  - Multi-dimension case (e.g. 2-d)
    - Demo animation on adaptive griding



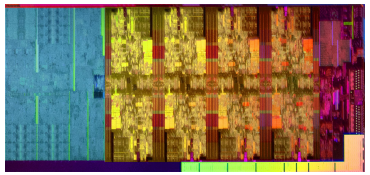
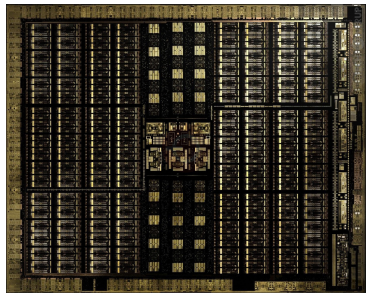
- Vegas Algorithm
  - Numerical libraries
    - CPU: ***G***NU ***S***cientific ***L***ibrary (GSL)
      CUBA, a library for multidimensional numerical integration
    - GPU: ***gVegas*** by J. Kanzaki
  - Why GPU?

$$m_i = K \left[\left(\frac{\bar{f}_i \Delta x_i}{\sum_j \bar{f}_j \Delta x_j} - 1 \right) \left(\log \frac{\bar{f}_i \Delta x_i}{\sum_j \bar{f}_j \Delta x_j} \right)^{-1} \right]^{\alpha},$$

$$\bar{f}_i = \sum_{x \in [x_{i-1}, x_i]} |f(x)| \approx \frac{1}{\Delta x_i} \int_{x_i}^{x_{i+1}} dx \; |f(x)|$$

performs evaluation of integrand on $\mathcal{O}(10^4 \sim 10^6)$ points x_i, suitable for parallel execution.

- CPU vs GPU

	CPU (i9-9900K)	**GPU (TU-102)**
die	 360mm^2	 770mm^2
cores	8	4352
performance (single/double precesion)	1.22/0.61 TFLOPS	26.9/13.45 TFLOPS
capability	complicate operations (Ph. D students)	simple operations (middle school students)
task: $\int dxe^{-x}, y''(t)+ky(t)=0$	efficient and precise	unable to process
task: square of $\pi+(1,2,\cdots,4000)$	slow	very fast

- Importance sampling vs Stratified sampling
  - Importance sampling
    - Finer sampling points where the ***integrand value*** is larger
  - Stratified sampling (variance reduction)
    - Finer sampling points where the ***potential error*** is larger
  - Routine using stratified sampling
    - GSL MISER
    - Divonne in CUBA Library
    - GSL VEGAS also apply stratified sampling after adaptive importance sampling for error reduction

- Monte-Carlo integration - Integral and error estimator

$$I = \int dx\, f(x) = \int dx\, \frac{f(x)}{p(x)} p(x) = \left\langle \frac{f}{p} \right\rangle \approx \frac{1}{N_{\text{pnts}}} \sum_{i=1}^{N_{\text{pnts}}} f(x_i)$$

  - In each iteration

$$\mathcal{I} = \frac{1}{N_{\text{pnts}}} \sum_{i=1}^{N_{\text{pnts}}} f(x_i)\,, \quad \sigma^2 = E\left(\frac{f^2}{p^2}\right) - E\left(\frac{f}{p}\right)^2 = \frac{1}{N_{\text{pnts}} - 1} \left[\sum_{i=0}^{N_{\text{pnts}}-1} \frac{f(x_i)^2}{p(x_i)^2} - \mathcal{I}^2 \right]$$

  - For accumulated all iterations

- Monte-Carlo integration - error scaling
  - $\sigma_I \sim \mathcal{O}\left(N_{\text{pnts}}^{-1/2}\right)$, which is independent of dimension d. 😄
  - For $d > 4$, Monte-Carlo algorithm beats deterministic method (rectangle, trapezium approximation)
  - Probability theory utilities are available
- **Minimize error**

  $p_{\text{prfct}}(x)$ precisely matches $f(x)$'s profile $\iff \frac{f(x)}{p_{\text{prfct}}(x)} = C, \left\langle \frac{f}{p_{\text{prfct}}} \right\rangle = C, \sigma = 0$

  HOWEVER, $C = \int_a^b dx\, f(x)$

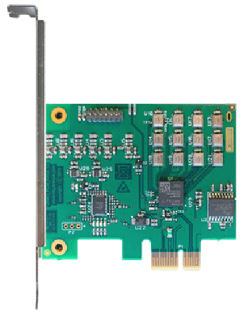
Integrating $f(x)$ is closely related to sampling distribution $p_{\text{prfct}}(x)$

Part II Application: Sampling a distribution

- Random number generation
  - True Random Number Generator (TRNG)
    - Need hardware support

    - Human inputs: mouse movements, network trafic..., The BIG Bell Test
- Pseudo Random Number Generator (PRNG)
  - Deterministic
  - Hard to predict if one do not know which algorithm is used
  - Randomness test
- > Quasi random numbers, low discrepancy sequence ...

- Random number generation
  - True Random Number Generator (TRNG)
    - Need hardware support (physical quantum effect on board)



    - Pseudo Random Number Generator (PRNG)
      - Deterministic
      - Hard to predict if one do not know which algorithm is used
      - Randomness test
    - > Quasi random numbers, low discrepancy sequence ...

- Sampling uniform distribution $U(0, 1)$
  - Linear Congruential Generator (LCG)
  - Mersenne twister
  - Xorshift and it's variants (as used in `gVegas`)
  - ...

- Sampling an arbitary distribution $p(x)$
  - Metropolis-Hastings algorithm

    ❶ Initialize $\boldsymbol{x} = \boldsymbol{x}^{(0)}$, $\boldsymbol{X} = \emptyset$ N_{acptd}=0

    ❷ `for i=0, i<N, i++`

    `for i=0, i<n, i++`

    Propose candidate (update) $\tilde{x}_i = x_i + \delta_i$

    Calculate $\eta = p(\tilde{\boldsymbol{x}})/p(\boldsymbol{x})$, generate $\xi \sim U(0,1)$

    `if` $\xi < \eta$, accept $\tilde{x}_i$, $\boldsymbol{x} = \tilde{\boldsymbol{x}}$, N_{acptd}++

    `else` reject $\tilde{x}_i$: $\boldsymbol{x} = \boldsymbol{x}$

    Append $\boldsymbol{x}$ to $\boldsymbol{X}$

    ❸ Calculate acceptance $r = \frac{N_{\text{acptd}}}{N \times n}$, tune amplitude of δ_i keeping $r \sim 0.6$

> - Thermalization
>   - Drop $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, x_{N_{\text{thrml}}}$ (building detailed balanced state)
> - Autocorrelation
>   - Take every N_{crr} generated $\boldsymbol{x}$ as final output, or use bin

Part III Application:
Path integral of 1-D quantum harmonic oscillator

- Action of Harmonic oscillator

$$S = \int dt\, L(x, \dot{x}) = \int dt\, \frac{m\dot{x}^2}{2} - \frac{k}{2}x^2$$

Wick rotation $\tau = it$, $\frac{d}{dt} = i\frac{d}{d\tau}$

$$S_E = -\int d\tau\, \frac{m\dot{x}^2}{2} + \frac{k}{2}x^2 = -\int d\tau H$$

- > Natural Unit $\hbar = c = 1$
- Propagator

$$\langle x_f, \tau_f | x_i, \tau_i \rangle = \int \mathcal{D}x(\tau)\, e^{-S_E[x(t)]}$$

- Position space eigen function

$$\begin{aligned}\langle x,\tau_f|x,\tau_i\rangle =& \langle x|e^{-H(\tau_f-\tau_i)}|x\rangle = \sum_{n=0}^{\infty}\langle x|e^{-H(\tau_f-\tau_i)}|n\rangle\langle n|x\rangle \\ &= \sum_{n=0}^{\infty}|\langle x|n\rangle|^2 e^{-E_n(\tau_f-\tau_i)} = \sum_{n=0}^{\infty}|\psi_n(x)|^2 e^{-E_n(\tau_f-\tau_i)}\end{aligned}$$

  - **For large enough $\tau_f-\tau_i$**, the ground state survives

$$\langle x,\tau_f|x,\tau_i\rangle = \int \mathcal{D}x(\tau)e^{-S_E[x(\tau)]} \approx |\psi_0(x)|^2 e^{-E_0(\tau_f-\tau_i)}$$

    and the ground state energy

$$\int dx\,\langle x,\tau_f|x,\tau_i\rangle = \int dx|\psi_0(x)|^2 e^{-E_0(\tau_f-\tau_i)} = e^{-E_0(\tau_f-\tau_i)}$$

- Discretized form of path integral

$$\int \mathcal{D}x(\tau) e^{-S_E[x(\tau)]} = \int \prod_{k=1}^{N-1} dx_k \exp\left[-\sum_{i=0}^{N-1} \frac{1}{2} \left(\frac{x_{k+1} - x_k}{\delta\tau} \right)^2 + \frac{1}{2} x_k^2 \right]$$

Discretized timespace

$$\int \mathcal{D}x(\tau) = \lim_{N\to\infty} \int \prod_{k=1}^{N-1} dx_k$$

Discretized action (Euclidean)

$$\tilde{S}_E = \sum_{k=0}^{N-1} \frac{1}{2m} \left(\frac{x_{k+1} - x_k}{\delta\tau} \right)^2 + \frac{k}{2} x_k^2$$

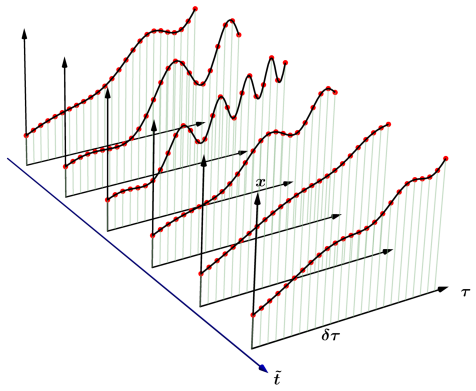
$x_k = x(\tau_k)$, $x_0 = x_i = x_f = x_N = x$

$\delta\tau = (\tau_f - \tau_i)/N$

- Functional path integral $\leftrightarrow$ $N-1$ dimensional numerical integration

- Functional path integral $\leftrightarrow$ $N-1$ dimensional numerical integration
  - One set of $x_1 = x(\tau_1), x_2 = x(\tau_2), \cdots, x_N(\tau_N)$ defines one path
  - In lattice QFT terminology: path $\Longrightarrow$ **field configuration**

HO $x(\tau)$	**Field** $\phi(x,\tau)$
position distributed over time	field strength distributed over spacetime
$x(\tau_1), x(\tau_2), \cdots, x(\tau_N)$	$\phi(x_1,\tau_1), \phi(x_1,\tau_2), \cdots, \phi(x_{N'},\tau_N)$



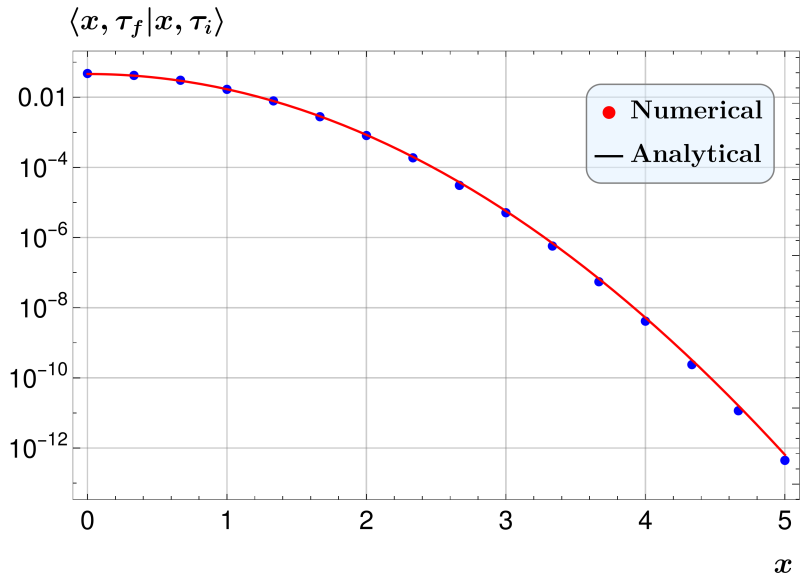
- Numerical Integration
  - Integrand is a (very) high dimensional function

$$\mathcal{F}\left(x_1, x_2, \cdots, x_{N-1}\right) = \exp\left[-\sum_{i=0}^{N-1} \frac{1}{2}\left(\frac{x_{k+1} - x_k}{\delta\tau}\right)^2 + \frac{1}{2}x_k^2\right]$$

  - Suppose we want to extract $\psi_0(x)$ and E_0
    - Requires large $\tau_f - \tau_i$
    - Smaller discretization error (δt) $\iff$ large $N = \frac{\tau_f - \tau_i}{\delta\tau}$ $\iff$ large d

Need Monte-Carlo integration !

- Path integral - Vegas integration
  - e.g. $N = 9$ intermediate x, $T = \tau_f - \tau_i = 4$, $\delta\tau = 0.5$



    - **Red**: Vegas integration $\langle x, \tau_f | x, \tau_i \rangle$
    - **Black**: canonical quantization $|\psi_0(x)|^2 e^{-E_0 T} = (2\pi)^{-1/2} e^{-x^2 - E_0(\tau_f - \tau_i)}$

- Since Monte-Carlo integration is closely related to sampling

$$x(\tau) \sim p\left[x\left(\tau\right)\right] = e^{-S[x(\tau)]} / \int \mathcal{D}x(t)\, e^{-S[x(\tau)]}$$

  The expectation value of observable $\mathcal{O}\left[x(t)\right]$

$$\langle \mathcal{O} \rangle = \frac{\int \mathcal{D}x(\tau)\, e^{-S[x(\tau)]} \mathcal{O}[x(\tau)]}{\int \mathcal{D}x(\tau)\, e^{-S[x(\tau)]}}$$

- If we can generate $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}\} \sim p[x(\tau)]$ (each $\boldsymbol{x}^{(k)}$ denotes one set of $x(\tau_1), \cdots, x(\tau_N)$)
- Observable $\langle \mathcal{O} \rangle$'s estimator

$$\langle \mathcal{O} \rangle \approx \frac{1}{N} \sum_{k=1}^{N} \mathcal{O}\left(\boldsymbol{x}^{(k)}\right)$$

- How to generate desired configurations $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}\} \sim p[x(\tau)]$
  Metropolis-Hasting algorithm and it's variants e.g.
  Hybrid Monte-Carlo (HMC), https://arxiv.org/abs/1701.02434v2

Exercise

- Try GSL Vegas (CPU) and gVeags (GPU) to calculate the path integral of 1-d quantum oscillator with finxed initial and final position, compare with analytical results

  You need to fill in the correct integrand for path integral
  - `CPU` in `Exercise/CPU/hrmnc_oscltr.cpp`
  - `GPU` in `Exercise/GPU/gVegasFunc.cu`

> - How to extract the ground state wave function $\psi_0(x)$ and ground state energy E_0 ?, try your solutions

- Instructions
  - Compile:
    - Change compile file `compile.sh` access permission to executable

      ```
      chmod +x ./compile.sh
      ```

    - Compile

      ```
      ./compile.sh
      ```

  - Launch calculation:

    ```
    hrmnc_oscltr.o Nt dt xmax Nx Npnts
    ```

    `Nt` , `dt` : number of discretization $x(\tau_k)$ and time step size $\mathrm{dt} = \tau_{k+1} - \tau_k$
    `xmax` , `Nx` : maxium x-value and number of x between $x = 0$ and $x = \mathrm{xmax}$
    `Npnts` : number of points used in Vegas integration
  - Output data file `CPU_hrmnc_oscltr.dat`

- A example `python` script for compile, run and analysis
  - Change `hrmnc_oscltr.py` access permission to executable

    ```
    chmod +x ./hrmnc_oscltr.py
    ```

  - Usage

    ```
      ./hrmnc_oscltr.py DEVICE MODE
    ```

    - `DEVICE` : source directory `CPU` or `GPU`
    - `MODE` : `compile` , `run` , `analysis` or `clean`
  - Example: `hrmnc_oscltr.py CPU run`
  - Parameters in `hrmnc_oscltr.py` , check previous slide
  - Output data file `CPU_hrmnc_oscltr.dat`

- Metropolis-Hastings example
  - Generating

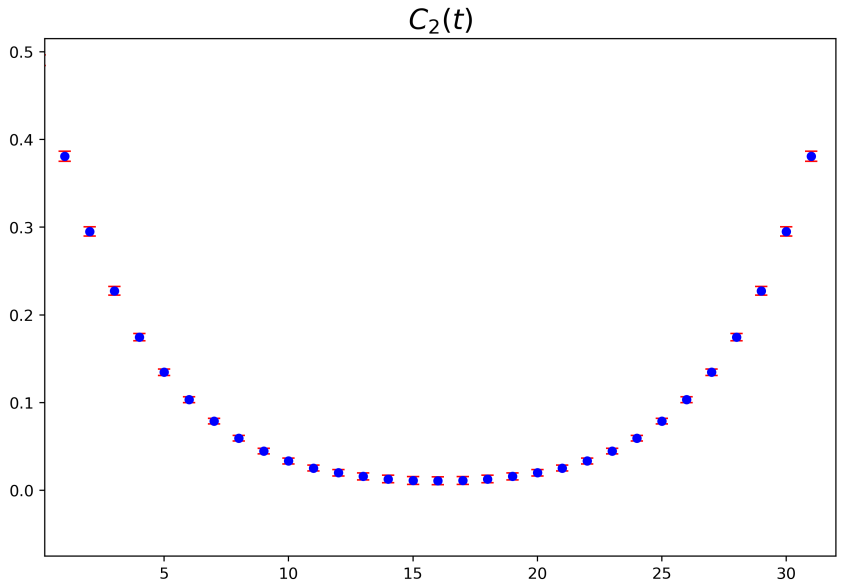
$$\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}\} \sim p[x(\tau)] = \frac{e^{-S[x(\tau)]}}{\int \mathcal{D}x(\tau) e^{-S[x(\tau)]}}$$

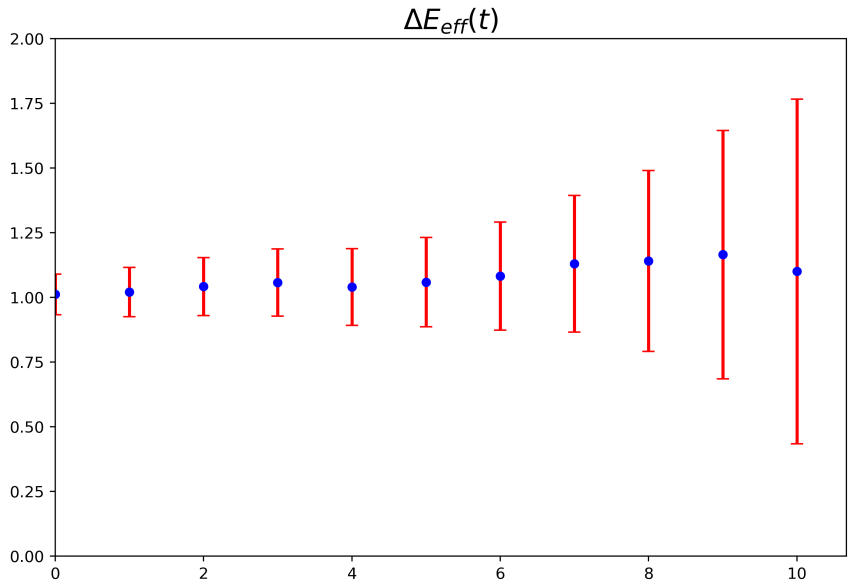
- Calculate the 2-point correlation function

$$C_2(n) = \frac{1}{N_\tau} \sum_j \left\langle x_{(j+n) \mathrm{mod} N_\tau} x_j \right\rangle$$

- Extract the energy level $\Delta E = E_1 - E_0$

$$\Delta E_n = \ln \frac{G_n}{G_{n+1}}$$


$C_2(t)$
0.5
0.4
0.3
0.2
0.1
0.0
5
10
15
20
25
30


$\Delta E_{eff}(t)$
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2
4
6
8
10