① Table of Contents Lecture Notes UQ Probability Refresher (p. 3-25): Basics (p. 3), Marginals / Bayes Rule (p. 4-6), Parameter Estimation, Likelihood function, Prior / Posterior Robust Prediction (p. 7-15), Laplace Approximation (p. 16-18, 22-25), Binomial / Geometric Distribution (p. 20/21), Monte Carlo Methods (p. 26-68): Monte Carlo Integration (p. 26-28), „Hit or miss"-algorithm (p. 28/29), Approximation Error / Uncertainty Propagation (p. 29-35), Inverse Transform Sampling (p. 41-50), Importance Sampling (p. 58-62), Rejection Sampling (p. 63/64), Metropolis-Hastings Method (p. 65-68) Appendix 1 (p. 70-78): Rejection Sampling proof (p. 70/71), Acceptance-Rejection Technique (p. 73-76) Appendix 2: MCMC proof (p. 80-91) Script UQ Bayesian Framework (p. 3-7): Bayes Theorem Robust Prediction (p. 3) Laplace Approximation (p. 8-12): High-Dimensional Approximation (p. 10), Monte Carlo Methods (p. 13-15): Importance Sampling (p. 14/15) Sampling Methods (p. 16-25): Inverse Transform Sampling (p. 16-19), Rejection Sampling (p. 19-23), Metropolis-Hastings Method (p. 23-25) Lecture 04: HPC / Advanced MPI High-Throughput / High-Performance (p. 17/56), MPI Topologies (p. 19-29), MPI Vector Datatypes (p. 31-38), MPI Struct Datatypes (p. 39-56) Lecture 05: Communication Tolerant Programming Intra-Node Data Motion (p. 6-20): KLTs / ULTs (p. 11), MPI+KLT (p. 12), MPI+MPI (p. 13) Network Overhead (p. 21-37): Communication Avoiding / Hiding (p. 27), Rank Prioritization (p. 31/32), Overdecomposition / Granularity / MPI+ULT (p. 33-37) Hierarchical Decomposition (p. 39-47) Lecture 07: High Throughput Computing HPC / HTC (p. 4-8), Korali (p. 10-19), Sample Scheduling (p. 21-32), Sample Distribution (p. 34-42): Divide-and-Conquer (p. 35), Load Imbalance (p. 36), Producer / Consumer (p. 37) MPI and Sample Distribution (p. 44-50): Lecture 08: Particle Methods Leap Frog Scheme (p. 37), Explicit / Implicit Euler & Implicit Midpoint (p. 43), Symplectic Euler (p. 44), Verlet Algorithm (p. 53), Runge Kutta Methods (p. 57), Cell Lists (p. 62-68), Remeshing (p. 72-81) Lecture 09: GPU Programming Basics Hardware Acceleration: History (p. 3-15), General-Purpose Computing on GPU (p. 25-48): Streaming Multiprocessor / Warp (p. 20/21), Function Qualifiers (p. 26), Memory Management (p. 29/30), Threads / Blocks / Grid (p. 31-34), Identification Variables (p. 38-41), Warp Coherency / Divergence (p. 42/43), Synchronization / Errors (p. 46/47) Lecture 10: GPU Memory Hierarchy and Optimizations History of Memory Hierarchies (p. 3-12), Memory Hierarchy on GPUs (p. 13-25): Variable Qualifiers (p. 16), Shared Memory (p. 20-23), Banked Access (p. 24) Lecture 11: Parallel Reduction Reduction Pattern (p. 3-5), Optimizing Parallel Reductions (p. 6-46): Complexity (p. 8), Kernel Decomposition (p. 10/11), Interleaved Addressing (p. 13-16), Non-divergent branch (p. 17-19), Sequential Addressing (p. 20/21), First Add During Load (p. 23-25), Warp Unrolling (p. 26-28), Complete Unrolling (p. 30-34), Cost / Brent's Theorem (p. 35-37), Algorithm Cascading (p. 37-41) Lecture 12: Streams / Warp-Level Primitives Warp-Level Primitives (p. 2-17): Warp-Level Synchronization (p. 8), Shuffle Collectives (p. 9-12), Ballot Collectives (p. 12), Match Collectives (p. 13), Butterfly Shuffle (p. 16) GPU Stream Pipelining (p. 18-42): Streams (p. 20/21), Default Stream (p. 25), Pinned Memory (p. 26), Explicit Synchronization (p. 31/32), Implicit Synchronization (p. 33), Stream Scheduling (p. 34-40) Exercises Exercise 06: Sampling / Monte Carlo Inversion Sampling (p. 9), Resection Sampling (p. 10), Importance Sampling (p. 11), Metropolis Hastings (p. 12-16) Exercise 06 Supplementary: TMCMC Sampling Importance Resampling (p. 1-8), TMCMC (p. 9-20) Exercise 07 Supplementary: CMA-ES Exercise 08: PGAS Model RPCs (p. 5), Futures (p. 6), Composing Futures (p. 9), Exercise 08: UPC++ Tutorial Hello World (p. 13), Barriers (p. 14), Broadcast (p. 15/16), Global Address Space (p. 17/18), RPCs (p. 19), Conjoining / Composing Futures (p. 20/21), Return Values / Fire-and-Forget (p. 22/23), Nested RPCs (p. 24) Exercise 09: CUDA Basics Memory Management (p. 14), Error Handling (p. 15), Compilation (p. 17) Exercise 10: Warp-Level functions List of functions (p. 13) Exercise 11: Atomics / Cell Lists Atomics (p. 6), Cell Lists (p. 8-12)

① Programming C++ Arrays: Static Creation: double data[10];, Dynamic: double* data = new double[10];, Initialization: int x[] = {1,2,3};, Iteration: for(int i=0; i<len; i++){ arr[i]; } Vectors: Required header: #include <vector>, Creation: std::vector<int> vec; or with length n, initialized to 0: std::vector(n, 0);, Appending: vec.push_back(i);, Length: vec.size();, Iterating: for(int i=0; i<vec.size(); i++){ vec[i]; }, Deleting by index: vec.erase(vec.begin()+i);, Accessing as array: int* a = &vec[0];, Copying: When initializing new vector: std::vector<int> vec2(vec);, otherwise: vec2=vec;, Swapping (constant complexity): vec.swap(vec2);, Swapping for arrays: std::swap(a,b); printf: Required header: #include <stdio.h>, Syntax: printf("format string", arg1, arg2, ...);, Format specifiers: %d: Signed int, %u: Unsigned int, %c: Character, %s: String, %p: Pointer address cout: Required header: #include <iostream>, Syntax: std::cout << "Output" << std::endl; Structs: E.g. struct Student { char name[20]; int id; int age; }, Creating an instance: Student s, Accessing fields: s.id=4; Makefiles: E.g. exec: <Tab> g++ -O3 -std=c++11 -Wall main.cpp -o $@ MPI Includes/Compilation: #include <mpi.h> Compilation with mpic++ file.cpp -o file Math: #include <cmath> Defines abs(float arg);, exp(float arg);, log(float arg);, log10(float arg);, pow(float base, float exp);, sqrt(float arg);, sin, cos, ceil, floor. For pi: Include <cmath> (or <math.h>) and use M_PI Uniform Random Variable: #include <random> std::default_random_engine generator; std::uniform_real_distribution<double> distribution(a,b); double number = distribution(generator); Normal Distribution: std::normal_distribution<double> distribution(mean, stddev);

Exercises Normal Distribution Mean / Variance: $\mathbb{E}[X] = \int_{-\infty}^{+\infty} x f_X(x)\,dx = \int_{-\infty}^{+\infty} (x-\mu) f_X(x)\,dx + \mu \int_{-\infty}^{+\infty} f_X(x)\,dx = \frac{1}{\sqrt{2\pi}\sigma^2} \int_{-\infty}^{+\infty} y e^{-\frac{y^2}{2\sigma^2}}\,dy + \mu = \mu$ [integral is 0 because of symmetry]. $\mathbb{E}[(X-\mu)^2] = \int_{-\infty}^{+\infty} (x-\mu)^2 f_X(x)\,dx \xrightarrow{y=x-\mu,\ a=\frac{1}{2\sigma^2}} \sqrt{\frac{a}{\pi}} \int_{-\infty}^{+\infty} y^2 e^{-ay^2}\,dy = -\sqrt{\frac{a}{\pi}} \frac{\partial}{\partial a} \int_{-\infty}^{+\infty} e^{-ay^2}\,dy = -\sqrt{\frac{a}{\pi}} \frac{\partial}{\partial a} \sqrt{\frac{\pi}{a}} = \frac{1}{2a} = \sigma^2$ Cauchy Quotient: PDF of quotient $Q = X/Y$ is given by: $f_Q(q) = \int_{-\infty}^{+\infty} |x| f_{X,Y}(qx, x)\,dx$. Given two RVs with $f_X(x) = \mathcal{N}(x|0,\sigma_x^2)$ and $f_Y(y) = \mathcal{N}(y|0,\sigma_y^2)$, show that $Q=X/Y$ follows a Cauchy dist. with loc. parameter $x_0=0$ and scale $\gamma = \sigma_x/\sigma_y$, where the PDF of a Cauchy dist. is: $f(x) = \frac{1}{\pi} \frac{\gamma}{(x-x_0)^2+\gamma^2}$. We have: $f_Q(q) = \frac{1}{2\pi\sigma_x\sigma_y} \int_{-\infty}^{+\infty} |x| \exp\left(-\frac{x^2}{2\sigma_x^2} - \frac{q^2x^2}{2\sigma_y^2}\right) dx = \frac{1}{2\pi\sigma_x\sigma_y} \int_{-\infty}^{+\infty} |x| \exp\left(-x^2\left(\frac{1}{2\sigma_x^2} + \frac{q^2}{2\sigma_y^2}\right)\right) dx \overset{\text{symm.}}{=} \frac{1}{\pi\sigma_x\sigma_y} \int_0^{+\infty} x \exp\left(-x^2\left(\frac{1}{2\sigma_x^2} + \frac{q^2}{2\sigma_y^2}\right)\right) dx$. With $\int_0^\infty x \exp(-kx^2)\,dx = \frac{1}{2k}$, we get for $k = \frac{\sigma_y^2 + q^2\sigma_x^2}{2\sigma_x^2\sigma_y^2}$: $f_Q(q) = \frac{1}{\pi} \frac{\sigma_x/\sigma_y}{q^2 + (\sigma_x/\sigma_y)^2}$ Bayesian Inference: Given N i.i.d. RVs with $X_i \sim \mathcal{N}(\mu, \gamma)$. $d = \{d_i\}_{i=1}^N$. Likelihood of μ: $L(\mu) = p(d|\mu) = \prod_{i=1}^N p(d_i|\mu) = (2\pi)^{-N/2} \exp\left(-\frac{1}{2}\sum_{i=1}^N (d_i-\mu)^2\right)$. $\log p(d|\mu) = -\frac{N}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^N (d_i-\mu)^2$. MLE of μ: $\hat{\mu} = \arg\max_\mu -\frac{1}{2}\sum_{i=1}^N (d_i-\mu)^2$. Diff. w.r.t. μ gives $\hat{\mu} = \frac{1}{N}\sum_{i=1}^N d_i$ Linear Model: Given $y = \beta x + \epsilon$ with $\epsilon \sim \mathcal{N}(0,\sigma^2)$, one point $D = \{x_0, y_0\}$ and an uninformative prior for β, calculate MAP/std. dev. of $\beta|D$: $p(\beta|D) = \frac{p(y_0|\beta)\,p(\beta)}{p(y=y_0)} \propto \mathcal{N}(y_0|\beta x_0, \sigma^2) \propto \exp\left(-\frac{1}{2\sigma^2}(y_0 - \beta x_0)^2\right) = \exp\left(\frac{x_0^2}{2\sigma^2}\left(\beta - \frac{y_0}{x_0}\right)^2\right) = \mathcal{N}\left(\frac{y_0}{x_0}, \frac{\sigma^2}{x_0^2}\right)$. Therefore $\hat{\beta}_{MAP} = \frac{y_0}{x_0}$ with std. dev. $\frac{\sigma}{x_0}$. Using a Gaussian prior $\beta \sim \mathcal{N}(0, \tau^2)$, we get $p(\beta|D) = \mathcal{N}\left(\frac{x_0 y_0}{x_0^2 + \frac{\sigma^2}{\tau^2}}, \frac{\sigma^2}{x_0^2 + \frac{\sigma^2}{\tau^2}}\right)$ Inverse Transformation: • Exponential dist. with rate γ, $p(x) = \gamma e^{-\gamma x}$: $P_X(x) = \int_0^x \gamma e^{-\gamma s}\,ds = 1 - e^{-\gamma x}$, if $x \geq 0$. $y = 1 - e^{-\gamma x} \to e^{-\gamma x} = 1-y \to x = -\frac{\ln(1-y)}{\gamma}$. $P_X^{-1}(y) = -\frac{\ln(1-y)}{\gamma}$ for $y \in [0,1)$. • Multinoulli dist. with $Pr(X=r) = \frac{a_r}{a_0}$. $P_X^{-1}(u) = 1$, if $0 \leq u < \frac{a_1}{a_0}$; 2, if $\frac{a_1}{a_0} \leq u < \frac{a_1+a_2}{a_0}$, ...

③ **Probability Basics:** $P[\bigcup_{i=1}^{\infty} A_i] = \sum_{i=1}^{\infty} P[A_i]$ for pairwise disjoint A_i, i.e. $A_i \cap A_k = \emptyset$

- For arbitrary A, B: $P[A \cup B] = P[A] + P[B] - P[A \cap B]$, $P[A \cap B] = P[B|A]P[A]$

Conditional Probability: $P[B|A] = \frac{P[B \cap A]}{P[A]}$ **Law of Total Probability:** $\bigcup_{i=1}^{n} A_i = \Omega$, then: $P[B] = \sum_{i=1}^{n} P[B|A_i]P[A_i]$ **Bayes Theorem:** $P[A|B] = \frac{P[B|A]P[A]}{P[B]}$ with normalizing constant: $P[B] = \int P[B|A]P[A]\,dA$ **Independence:** Two events A, B are independent if $P[A \cap B] = P[A]P[B] \iff P[A|B] = P[A] \wedge P[B|A] = P[B]$. For n events: $P[\bigcap_{i=1}^{m} A_{k_i}] = \prod_{i=1}^{m} P[A_{k_i}]$ for every finite subset, i.e. $\{k_1, \dots, k_m\} \subseteq \{1, \dots, n\}$ **Random Variables:** A random variable X is a function $X: \Omega \to \mathbb{R}$ with cumulative distribution function $F_X(x) := P[X \le x] = P[\{\omega | X(\omega) \le x\}]$. The probability density function (if it exists) of X is $f_X(x) = \frac{dF_X(x)}{dx} \iff F_X(x) = \int_{-\infty}^{x} f_X(t)\,dt$ with $f_X \ge 0$ and $\int_{-\infty}^{\infty} f_X(t)\,dt = 1$. We have $P[a < X \le b] = F_X(b) - F_X(a) = \int_a^b f_X(t)\,dt$ **Expectation Value:** $E[X] = \int_{-\infty}^{\infty} x f_X(x)\,dx$. For $Y = g(X)$, we have: $E[Y] = \int_{-\infty}^{\infty} g(x) f_X(x)\,dx$. **Linearity:** $E[aX + b] = aE[X] + b$ **Variance:** $Var[X] := E[(X - E[X])^2] = E[X^2] - (E[X])^2 = \int_{-\infty}^{+\infty} (x-\mu)^2 f(x)\,dx = \int_{-\infty}^{\infty} x^2 f(x)\,dx - \mu^2$. $Var[aX+b] = a^2 Var[X]$. $Var[X_1 + X_2] = Var[X_1] + Var[X_2] + 2Cov[X_1, X_2]$ with $Cov[X_1, X_2] = E[(X_1 - E[X_1])(X_2 - E[X_2])] = E[X_1 X_2] - E[X_1]E[X_2]$. If $Cov[X, Y] = 0$, X and Y are uncorrelated (and independent $\Rightarrow$ uncorrelated). **Sample Variance:** $s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (Y_i - \bar{Y})^2$ **Marginals:** $f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)\,dy$, $f_Y(y) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)\,dx$. In the discrete case: $p(x) = \sum_y p(x,y)$ **Law of Large Numbers:** For X_i i.i.d., $\bar{X}_n = \frac{1}{n}(X_1 + \dots + X_n)$ converges in probability (weak form) / almost surely (strong form) to μ. **Central Limit Theorem:** For X_i i.i.d. with $E[X_i] = \mu$ and $Var[X_i] = \sigma^2$, let $S_n = \sum_{i=1}^{n} X_i$, $S_n^* = \frac{S_n - E[S_n]}{\sqrt{Var[S_n]}} = \frac{S_n - n\mu}{\sigma\sqrt{n}}$ and $\bar{X}_n = \frac{1}{n} S_n$. Then $S_n \sim \mathcal{N}(n\mu, n\sigma^2)$, $S_n^* \sim \mathcal{N}(0,1)$, $\bar{X}_n \sim \mathcal{N}(\mu, \frac{1}{n}\sigma^2)$ **Random Variable Transform:** Consider RV X and let $Y = g(X)$. Then $F_Y(y) = F_X(g^{-1}(y))$ and $f_Y(y) = f_X(x)\left|\frac{dx}{dy}\right| = f_X(g^{-1}(y))\left|\frac{d}{dy} g^{-1}(y)\right|$ **Discrete Distributions:** 1.) Uniform Distribution: $p_X(x_k) = \frac{1}{n}$, $E[X] = \frac{1}{n}\sum_{i=1}^{n} x_i$ 2.) Binomial Distribution: $W(X) = \{0, 1, 2, \dots, n\}$ and $p_X(k) = \binom{n}{k} p^k (1-p)^{n-k}$. $E[X] = np$, $Var[X] = np(1-p)$ 3.) Geometric Distribution: $W(X) = \mathbb{N}$, $p_X(k) = p(1-p)^{k-1}$. $E[X] = \frac{1}{p}$, $Var[X] = \frac{1-p}{p^2}$ 4.) Poisson Distribution: $W(X) = \mathbb{N}$, $p_X(k) = e^{-\lambda}\frac{\lambda^k}{k!}$. $E[X] = Var[X] = \lambda$

Continuous Distributions: 1.) Uniform Distribution: $f_X(t) = \frac{1}{b-a}$ for $a \le t \le b$. $F_X(t) = \frac{t-a}{b-a}$. $E[X] = \frac{a+b}{2}$, $Var[X] = \frac{1}{12}(b-a)^2$ 2.) Exponential Distribution: $f_X(t) = \lambda e^{-\lambda t}$ for $t \ge 0$. $F_X(t) = 1 - e^{-\lambda t}$ for $t \ge 0$. $E[X] = \frac{1}{\lambda}$, $Var[X] = \frac{1}{\lambda^2}$. 3.) Normal Distribution: $f_X(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}$. $E[X] = \mu$, $Var[X] = \sigma^2$. When $X \sim \mathcal{N}(\mu, \sigma^2)$, $\frac{X-\mu}{\sigma} \sim \mathcal{N}(0,1)$ 4.) Laplace Distribution: $f_X(t) = \frac{1}{2b}\exp\left(-\frac{|t-\mu|}{b}\right)$ $F_X(t) = \frac{1}{2} + \frac{1}{2}\operatorname{sgn}(t-\mu)\left(1 - \exp\left(-\frac{|t-\mu|}{b}\right)\right)$. $E[X] = \mu$, $Var[X] = 2\sigma^2$ 5.) Cauchy Distribution: $f_X(t) = \frac{1}{\pi}\frac{1}{1+t^2}$, $F_X(t) = \frac{1}{2} + \frac{1}{\pi}\operatorname{arctan}(t)$ **Bayesian Framework:** We have $p_{X|Y}(x|y) = \frac{p_{Y|X}(y|x)p_X(x)}{p_Y(y)}$. $p_{X|Y}$ is the posterior probability, $p_{Y|X}$ is the likelihood function, p_X is the prior distribution. If we make a prediction and use the prior uncertainty, this is called prior robust prediction: $p(y) = \int p(y|x)p(x)\,dx$. If the posterior distribution is used (where d denotes data), posterior robust prediction: $p(y|d) = \int p(y|x)p(x|d)\,dx$. **Laplace Approximation:** Approximates distribution $p(x)$ by normal distribution. Let $C(x) = \log(p(x))$. We find $\hat{x}$ s.t. $\frac{\partial p}{\partial x}|_{\hat{x}} = 0$ and $\frac{\partial^2 p}{\partial x^2}|_{\hat{x}} < 0$ or equiv. $\frac{\partial C}{\partial x}|_{\hat{x}} = 0$ and $\frac{\partial^2 C}{\partial x^2}|_{\hat{x}} < 0$. Then approximate p with $\mathcal{N}(\hat{x}, \sigma^2)$ with $\sigma^2 = -\left(\frac{\partial^2 C}{\partial x^2}|_{\hat{x}}\right)^{-1}$ **Monte Carlo Integration:** Want to compute for $h: \mathbb{R}^k \to \mathbb{R}$: $\mathbb{E}_f[h(X)] = \int h(x) f(x)\,dx$. By Law of large numbers, for $X_1, \dots, X_n \overset{i.i.d}{\sim} f$, $\frac{1}{n}\sum_{i=1}^{n} h(x_i) \to \mathbb{E}_f[h(X)]$ with $Var[\frac{1}{n}\sum_{i=1}^{n} h(x_i)] = \frac{1}{n} Var[h(X_1)]$, which doesn't depend on dimension. **Inverse CDF Transformation:** 1.) Draw $U \sim Unif(0,1)$ 2.) Let $X = F^{-1}(U)$, then X is distributed according to f. E.g. exponential distribution with $F(x) = 1 - e^{-x/\lambda}$: $u = 1 - e^{-x/\lambda} \iff e^{-x/\lambda} = 1 - u \iff -\frac{x}{\lambda} = \log(1-u) \iff x = F^{-1}(u) = -\lambda\log(1-u)$ **Importance Sampling:** Instead of evaluating $\int f(x)\,dx$, we evaluate $\int \frac{f(x)}{p(x)} p(x)\,dx = E_p[\frac{f(x)}{p(x)}]$ by sampling from p, i.e. $\frac{1}{N}\sum_{k=1}^{N} \frac{f(x^{(k)})}{p(x^{(k)})}$ with $\{x^{(k)}\}_{k=1}^{N}$ i.i.d from p **Rejection Sampling:** 1.) Find easy-to-sample density q 2.) Scale q by M s.t. graph of Mq is

④ always above graph of p, i.e. $M \geq \max_x \frac{p(x)}{q(x)}$ 3.) Sample $x \sim q$ and $u \sim \mathcal{U}(0, Mq(x))$ 4.) If $u < p(x)$, accept/keep **Metropolis-Hastings**: Given current state y, proposal density $p_{x|y}$, target density p_{eq}: 1.) Generate $x \sim p(\cdot|y)$ 2.) Generate $u \sim \mathcal{U}(0,1)$ 3.) If $u < \frac{p(y|x)p_{eq}(x)}{p(x|y)p_{eq}(y)}$, accept x (i.e. next round has $y=x$ and x is sample output), else reject (i.e. next round y stays and is sample output again). Metropolis algorithm: $p_{x|y}$ is symmetric, therefore only $\frac{p_{eq}(x)}{p_{eq}(y)}$ considered.

Math **Chain Rule**: $\frac{d}{dx}[f(g(x))] = f'(g(x))g'(x)$ **Quotient Rule**: $\frac{d}{dx}\left[\frac{f(x)}{g(x)}\right] = \frac{g(x)f'(x) - f(x)g'(x)}{[g(x)]^2}$ **Derivatives**: $\frac{d}{dx}\ln(x) = \frac{1}{x}$, $\frac{d}{dx}n^x = n^x \ln(n)$, $\frac{d}{dx}\sin(x) = \cos(x)$, $\frac{d}{dx}\cos(x) = -\sin(x)$, $\frac{d}{dx}\sqrt{x} = \frac{1}{2\sqrt{x}}$ **Multivariate Gaussian**: Shape of the ellipse is determined by eigenvectors of Σ, vector with largest eigenvalue is largest axis of ellipse. $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ -> Round distribution around mean

Snippets Own coords in MPI cart. topology: `MPI_Comm_rank(cart_comm, &rank); MPI_Cart_coords(cart_comm, rank, 3, &coords[0]);` CUDA Grid-Stride Loops: `for(int i=blockIdx.x*blockDim.x+threadIdx.x; i<n; i+=)` CUDA no. of threads with true condition in warp: `int warpSum=__popc(__ballot_sync(0xFFFFFFFF, condition));`

Theory **Advantages/Disadvantages Particle Methods** Advantages: 1.) Conservative 2.) Can solve some advection problems exactly (e.g. linear advection) 3.) Lower/No numerical diffusion 4.) Easier Refinement Disadvantages: 1.) More complex algorithms for finding neighbor particles 2.) Fewer Choices of methods to solve linear systems 3.) Harder to impose boundary conditions **PDE Notation** $\nabla = \left(\frac{\partial}{\partial x_1}, \dots, \frac{\partial}{\partial x_n}\right)^T$, $u_t = \frac{\partial^2}{\partial t^2}u(t,\dots)$, $u_{xy} = \frac{\partial^2}{\partial x \partial y}u(x,y,\dots)$, $\Delta = \nabla^T \cdot \nabla = \sum_{k=1}^{d} \frac{\partial^2}{\partial x_k^2}$ **Amdahl's Law** Speedup $S_n = \frac{1}{1-p+\frac{p}{n}}$, where p = parallel fraction. For $n \to \infty$, $S_\infty = \frac{1}{1-p}$ **Brent's Law** Instead of N processors (that can exploit maximum concurrency), we consider $p < N$. Then: $T_p \leq T_N + \frac{T_1 - T_N}{p}$ Or less precisely: $T_p = O\left(T_N + \frac{T_1}{p}\right)$ **Strong Scaling Analysis** We keep problem size fixed and report $S_p = \frac{t_1}{t_p}$. Number of cores/threads (x-axis) are plotted against speedup. Strong scaling efficiency is $\eta_s(p) = \frac{S_p}{p}$, ideally 1. **Weak Scaling Analysis** Number of nodes/problem size is increased s.t. work per node is constant. If amount of work e.g. linear with problem size, multiplying nodes by c requires multiplying problem size by c. For quadratic work, multiplication by $\sqrt{c}$ required. Weak Scaling Efficiency: $\eta_w(p) = \frac{T(1)}{T(p)}$. Number of cores/threads is plotted against efficiency. **Roofline Model** Operational Intensity: $\frac{FLOPs}{Byte}$. Given nominal peak bandwidth B [GB/s] / peak performance $\mathcal{F}$ [GFlop/s], performance of code with intensity I is $P = \min(BI, \mathcal{F})$. Ridge point is $I_b = \frac{\mathcal{F}}{B}$, for $I < I_b$ memory-bound and for $I > I_b$ compute bound. Graph: Horizontal line at $\mathcal{F}$ until ridge point, then line that goes through B at 1. **Data Types**: Single precision/float = 4 bytes, Double precision/double = 8 bytes, int/long int = 4 bytes, long long int = 8 bytes **1D Diffusion: Finite Differences** Explicit: $u_i^{n+1} = u_i^n + \gamma \frac{\delta t}{\delta x^2}(u_{i+1}^n + u_{i-1}^n - 2u_i^n)$ Implicit: $u_i^{n+1} = u_i^n + \gamma \frac{\delta t}{\delta x^2}(u_{i+1}^{n+1} + u_{i-1}^{n+1} - 2u_i^{n+1})$ **2D Diffusion: Finite Differences** $p_{i,j}^{n+1} = p_{i,j}^n + \alpha t D\left(\frac{p_{i,j-1}^n - 2p_{i,j}^n + p_{i,j+1}^n}{\Delta x^2} + \frac{p_{i-1,j}^n - 2p_{i,j}^n + p_{i+1,j}^n}{\Delta y^2}\right)$. With $\Delta x = \Delta y = \Delta r$: $p_{i,j}^{n+1} = p_{i,j}^n + \frac{\Delta t D}{\Delta r^2}(p_{i,j-1}^n - 2p_{i,j}^n + p_{i,j+1}^n + p_{i-1,j}^n - 2p_{i,j}^n + p_{i+1,j}^n)$ **Binomial Coefficient** $\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}$

⑤ MPI Topologies: Creation: MPI_Cart_create(MPI_Comm comm_old, int ndims, int* dims, int* periods, int reorder, MPI_Comm* comm_cart); For grid with fair node split: MPI_Dims_create(int nnodes, int ndims, int *dims); Deletion: MPI_Comm_free(MPI_Comm* comm_cart); Example: int size; MPI_Comm_size(&size); int nums[3]={0,0,0}; int periodic[3]={0,0,0}; MPI_Dims_create(size,3,nums); MPI_Comm cart_comm; MPI_Cart_create(MPI_COMM_WORLD,3,nums,periodic,true,&cart_comm); MPI_Comm_free(&cart_comm); Finding neighbors: MPI_Cart_shift(MPI_Comm comm, int dir, int disp, int *source, int *dest); e.g.: MPI_Cart_shift(cart_comm,0,1,&left,&right); Vector Datatype: MPI_Type_vector(int count, int blocklength, int stride, MPI_Datatype oldtype, MPI_Datatype *newtype); → blocklength contiguous entries, spaced at stride. Multidimensional Problems: MPI_Type_create_subarray(int ndims, int sizes[], int subsizes[], int starts[], int order, MPI_Datatype oldtype, MPI_Datatype *newtype); order is MPI_ORDER_C (row-major) or MPI_ORDER_FORTRAN (col-major). sizes = size of the full array. Struct Datatypes: MPI_Type_create_struct(int count, int blocklengths[], MPI_Aint offsets[], MPI_Datatype types[], MPI_Datatype* newtype); Committing: MPI_Type_commit(MPI_Datatype* type); Freeing: MPI_Type_free(MPI_Datatype* type); For portability: MPI_Get_address(void* location, MPI_Aint *address); Example: MPI_Aint p_lb, p_a, p_nsteps, p_ub; MPI_Get_address(&p, &p_lb); M_G_a(&p.a, &p_a); M_G_a(&p.nsteps, &p_nsteps); M_G_a(&p+1, &p_ub); int blocklens[]={0,2,1,0}; MPI_Datatype types[]={MPI_LB, MPI_DOUBLE, MPI_INT, MPI_UB}; MPI_Aint offsets[]={0, p_a-p_lb, p_nsteps-p_lb, p_ub-p_lb}; Alternative for portability: Including <stddef.h> and using offsetof(p, a); for offsets. And for padding (after struct), MPI_Type_create_resized(oldtype, lb, extent, *newtype); E.g.: MPI_Type_create_struct(1, blocklens, offsets, types, &MPI_P_NOPAD); MPI_Type_create_resized(MPI_P_NOPAD, offsets[0], (MPI_Aint) sizeof(struct particle), &MPI_P); MPI_Type_commit(&MPI_P);

CUDA Data Transfer: Allocation with cudaMalloc(void** devPtr, size) and cudaMallocHost(void** ptr, size) → for page-locked/pinned memory. Freeing with cudaFree(void* ptr) / cudaFreeHost(void* ptr). Transfer with cudaMemcpy(void* dst, void* src, size_t num_bytes, enum cudaMemcpyKind dir); with dir=cudaMemcpyHostToHost/cudaMemcpyHostToDevice/cudaMemcpyDeviceToHost/cudaMemcpyDeviceToDevice. Kernel Launch: Kernel<<<gridDim, blockDim, shmSize, stream>>>(); where shmSize/stream is optional. gridDim/blockDim can be 1D/2D/3D, e.g for 2D: dim3 blocks(N/4, N/4); dim3 threads(4,4); Identification Variables: Always with members .x/.y/.z: gridDim: Size/dimensions of grid of blocks. - blockIdx: Index of block - blockDim: Size/dimensions of each block. - threadIdx: Index of thread. In 1D case, therefore often: int index = blockIdx.x * blockDim.x + threadIdx.x; if(index < N)...; In 2D: int idx = blockIdx.x * blockDim.x + threadIdx.x; int idy = blockIdx.y * blockDim.y + threadIdx.y; Usually round up numBlocks: int threadsPerBlock = 32; int numBlocks = (N + threadsPerBlock - 1) / threadsPerBlock; Shared Memory: If shmSize specified on launch: extern __shared__ double s; (and e.g size_t shmSize = 1000*sizeof(double);) If statically specified: __shared__ double s[BLOCK_SIZE][BLOCK_SIZE]; After saving, usually need to use __syncthreads(); s.t. every thread in the block has saved value. Warp-Level Primitives: int shfl(int var, int srcRank); → Returns value of srcRank, e.g. value = shfl(value, 0); shfl_down/shfl_up(int var, int delta); → Same as shfl, but value from tid+delta is returned. __syncwarp(unsigned mask); → Wait until all warp lanes in mask have executed stmt. E.g. __syncwarp(0xffffffff); (shuffle also available with sync, should be used) Streams: cudaStream_t stream; cudaStreamCreate(&stream); cudaStreamSynchronize(stream); cudaStreamDestroy(stream); Error Checking: cudaError_t err = cudaGetLastError(); if(err != cudaSuccess) { std::cout << cudaGetErrorString(err); } Includes/Compilation: #include <cuda_runtime.h>

⑥ Compilation with: nvcc -O3 -std=c++14 --compiler-options "-Wall -Wextra" file.cu -o file (needs to have .cu extension!) Atomics: atomicAdd(double* address, double val); → Add val to address. atomicInc(int* address, int val); → Increment address if lower than val („rollover value"), otherwise set to 0.

Exercises (cont. 2): Show that $\alpha_M(x|y)$ is a special case, $\alpha_M(x|y) = \min\{1, \frac{q(y|x)p(x)}{q(x|y)p(y)}\}$. What's $s(x|y)$? Let $r(x|y) = \frac{q(y|x)p(x)}{q(x|y)p(y)}$. Then $\alpha_M(x|y) = \min\{1, r(x|y)\}$, $\alpha(x|y) = \frac{s(x|y)}{1+\frac{1}{r(x|y)}}$. If $r(x|y) < 1$: $\alpha_M(x|y) = r(x|y) = r(x|y)\frac{1+\frac{1}{r(x|y)}}{1+\frac{1}{r(x|y)}} = \frac{1+r(x|y)}{1+\frac{1}{r(x|y)}} \Rightarrow s(x|y) = 1+r(x|y)$. If $r(x|y) \geq 1$: $\alpha_M(x|y) = 1 = \frac{1+\frac{1}{r(x|y)}}{1+\frac{1}{r(x|y)}} \Rightarrow s(x|y) = 1+\frac{1}{r(x|y)}$. Need to show that $s(x|y) = s(y|x)$: $s(y|x) = \{1+r(y|x), \text{if } r(y|x) < 1;\ 1+\frac{1}{r(y|x)}, \text{oth.}\} = \{1+\frac{1}{r(x|y)}, \text{if } \frac{1}{r(x|y)} < 1;\ 1+r(x|y), \text{oth}\} = \{1+\frac{1}{r(x|y)}, \text{if } r(x|y) \geq 1;\ 1+r(x|y), \text{oth}\} = s(x|y)$.

MH acceptance crit. in log scale? $\log(\alpha_M(x|y)) = \log\min\{1, \frac{q(y|x)p(x)}{q(x|y)p(y)}\} = \min\log\{1, \frac{q(y|x)p(x)}{q(x|y)p(y)}\} = \min\{0, \log(\frac{q(y|x)p(x)}{q(x|y)p(y)})\} = \min\{0, \log q(y|x) + \log p(x) - \log q(x|y) - \log p(y)\}$

If q is symmetric: $\log(\alpha_M(x|y)) = \min\{0, \log p(x) - \log p(y)\}$

Exercises (cont.) Sum of Two Normal Distribution: $X \sim \mathcal{N}(\mu_x, \sigma_x)$ and $Y \sim \mathcal{N}(\mu_y, \sigma_y)$. Show $Z = X+Y \sim \mathcal{N}(\mu_z, \sigma_z)$ with $\mu_z = \mu_x + \mu_y$, $\sigma_z^2 = \sigma_x^2 + \sigma_y^2$.

$f_Z(y) = f_{X+Y}(y) = \int_{-\infty}^{+\infty} f_Y(y-x) f_X(x) dx \propto \int_{-\infty}^{+\infty} \exp\{-\frac{(y-x-\mu_y)^2}{2\sigma_y^2}\} \exp\{-\frac{(x-\mu_x)^2}{2\sigma_x^2}\} dx$.

$a = y-\mu_y$, $b = \mu_x$, $c = 2\sigma_x^2$, $d = 2\sigma_y^2$ → $f_Z(y) \propto \int_{-\infty}^{+\infty} \exp\{-[\frac{(x-a)^2}{c} + \frac{(x-b)^2}{d}]\} dx$. Term in exp can be rewritten as $\alpha(x-\beta)^2 + \frac{(a+b)^2}{c+d}$ with $\alpha = \frac{c+d}{cd}$, $\beta = \frac{da+bc}{d+c}$. Leads to:

$f_Z(y) \propto \exp\{-\frac{(y-\mu_x-\mu_y)^2}{2(\sigma_x^2+\sigma_y^2)}\} \int_{-\infty}^{+\infty} e^{-\alpha(x-\beta)^2} dx \sim \mathcal{N}(\mu_x+\mu_y, \sqrt{\sigma_x^2+\sigma_y^2})$ [integral disappears]

Monte Carlo Integration: Suppose $X \sim \mathcal{U}_{ni} f(a,b)$. Then $\mu_f = \mathbb{E}[f] = \frac{1}{b-a}\int_a^b f(x)dx \rightarrow \int_a^b f(x)dx = (b-a)\mu_f$. An unbiased estimator for μ_f is: $\hat{\mu}_f = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \rightarrow \int_a^b f(x)dx \approx (b-a)\hat{\mu}_f$. Is estimator for Integral unbiased? Yes, $I = \int_a^b f(x)dx = (b-a)\int_a^b \frac{f(x)}{b-a}dx = (b-a)\mathbb{E}[f(x)]$ if $x \sim U(a,b)$. $\hat{I} = \frac{b-a}{N}\sum_{i=1}^{N} f(x_i) \Rightarrow \mathbb{E}[\hat{I}] = \frac{b-a}{N}\mathbb{E}[\sum_{i=1}^{N} f(x_i)] = (b-a)\mathbb{E}[f(x)] = I$. What is the variance of the int. estimator? $\mathrm{Var}[\hat{I}] = \mathrm{Var}[\frac{b-a}{N}\sum_{i=1}^{N} f(x_i)] = \frac{(b-a)^2}{N}\mathrm{Var}[f(x)]$

Inversion Method for Gaussian Sampling: $X = \sqrt{2}\,\mathrm{erf}^{-1}(2U-1)$ with $U \sim \mathcal{U}(0,1)$ and $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x \exp(-t^2)dt$, show $X \sim \mathcal{N}(0,1)$: $F_X(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} \exp(\frac{-t^2}{2})dt = \frac{1}{2} + \frac{1}{\sqrt{2\pi}}\int_0^x \exp(\frac{-t^2}{2})dt = \frac{1}{2} + \frac{1}{\sqrt{\pi}}\int_0^{x/\sqrt{2}} \exp(-t^2)dt = \frac{1}{2}(1+\mathrm{erf}(\frac{x}{\sqrt{2}}))$. With $x = \sqrt{2}\,\mathrm{erf}^{-1}(2u-1)$, we get $u = F_X(x)$.

Importance Sampling: Re-writing $\mathbb{E}_f[h(x)] = \int h(x)f(x)dx$ to $\mathbb{E}_g[h(x)\frac{f(x)}{g(x)}] = \int h(x)\frac{f(x)}{g(x)}g(x)dx$. Show the ideal g (min. variance) is given by $g^*(x) = \frac{|h(x)|f(x)}{\int|h(x)|f(x)dx}$. $\mathrm{Var}_g^{\hat{I}}(g) = \mathbb{E}_g[\frac{h^2(x)f^2(x)}{g^2(x)}] - (\mathbb{E}_g[\frac{h(x)f(x)}{g(x)}])^2$. Second term doesn't depend on x, for the first because Jensens inequality: $\mathbb{E}_g[\frac{h^2(x)f^2(x)}{g^2(x)}] \geq (\mathbb{E}_g[\frac{h(x)f(x)}{g(x)}])^2 = (\int h(x)f(x)dx)^2$. Lower bound is achieved by g^*.

MCMC, detailed balance: Given proposal distribution q, $\alpha(x|y) = \frac{s(x|y)}{1+\frac{q(x|y)p(y)}{q(y|x)p(x)}}$, s any symmetric function, show that α transition prob. $t(x|y) = \alpha(x|y)q(x|y)$ satisfies detailed balance.

Detailed balance condition: $t(x|y)p(y) = t(y|x)p(x)$. We have $t(x|y)p(y) = \alpha(x|y)q(x|y)p(y) = \frac{s(x|y)q(x|y)p(y)}{1+\frac{q(x|y)p(x)}{q(y|x)p(x)}} \cdot \frac{q(y|x)p(x)}{q(y|x)p(x)} = \frac{s(x|y)q(x|y)p(y)q(y|x)p(x)}{q(y|x)p(x) + q(x|y)p(y)} = \frac{s(x|y)q(y|x)p(x)}{\frac{q(y|x)p(x)}{q(x|y)p(y)} + \frac{q(x|y)p(y)}{q(x|y)p(y)}} = \frac{s(x|y)q(y|x)p(x)}{1+\frac{q(y|x)p(x)}{q(x|y)p(y)}} = \frac{s(y|x)q(y|x)p(x)}{1+\frac{q(y|x)p(x)}{q(x|y)p(y)}} = \alpha(y|x)q(y|x)p(x) = t(y|x)p(x)$