Orders of Magnitude:

Stan Algorithms and Engineering

Bob Carpenter

Center for Computational Mathematics

Flatiron Institute

June 2023

Flatiron Institute

The **mission** of the **Flatiron Institute** is to **advance scientific research** through **computational methods**, including data analysis, theory, modeling and simulation.

- **Center for Computationa**: Math (stats/ML), Biology, Neuroscience, Astrophysics, Quantum Physics
- part of **Simons Foundation**
  - US$300M+ science and education grants per year
  - US$5B endowment

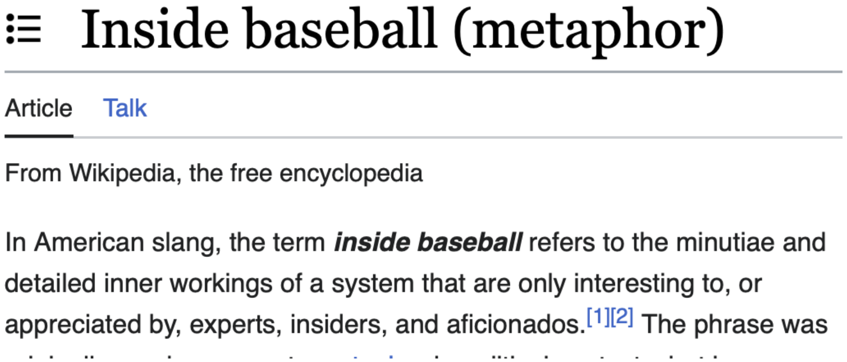
Pop Quiz

- Who’s the **most famous Stan** in **St. Louis**?

Stan the Man (active 1941–1963)

- Not Ulam. Not a stalker fan.
- **Stan “The Man” Musial**. Outfielder, ***St. Louis Cardinals.***

Wikipedia: One of the **greatest and most consistent** hitters in baseball



☰ Inside baseball (metaphor)

Article Talk

From Wikipedia, the free encyclopedia

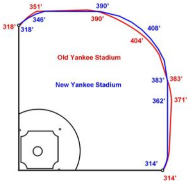
In American slang, the term ***inside baseball*** refers to the minutiae and detailed inner workings of a system that are only interesting to, or appreciated by, experts, insiders, and aficionados.[1][2] The phrase was

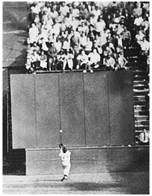
Swing for the fences

- 2011: I move **from industry to Columbia** to work with Gelman.
- I was **getting scooped** on crowdsourcing.
- Michael Collins (CS) suggests I **work on harder problems**.
- I **listened** and we started the **Stan** project.

Where are the fences?

- **Sometimes**: an **unknown unknown**.
- **Usually**: about an **order of magnitude** away
- I'm a **computer scientist**, so that's only $\times 2$ (≈ Gelman & Hill)





Part I

MCMC for Bayes

Bayesian Inference (Bayes $\approx$ 1750, Laplace $\approx$ 1800)

- **unknown** (parameters) $\theta \in \mathbb{R}^D$; **observed** (data) $y \in \mathbb{R}^N$
- **Estimation**:

$$\hat{\theta} = \mathbb{E}[\theta \mid y] = \int_{\mathbb{R}^D} \theta \cdot p(\theta \mid y) \, \mathrm{d}\theta.$$

- **Event probabilities**

$$\Pr[A \mid y] = \mathbb{E}[\mathrm{I}_A(\theta) \mid y] = \int_{\mathbb{R}^D} \mathrm{I}_A(\theta) \cdot p(\theta \mid y) \, \mathrm{d}\theta.$$

- **Posterior prediction**

$$p(\tilde{y} \mid y) = \mathbb{E}[p(\tilde{y} \mid \theta) \mid y] = \int_{\mathbb{R}^D} p(\tilde{y} \mid \theta) \cdot p(\theta \mid y) \, \mathrm{d}\theta.$$

Monte Carlo (Fermi, von Neumann, Ulam ≈ 1930s–1940s)

- Go-to approach to **high-dimensional integration**
- **Randomized algorithm** for deterministic integrals

$$\mathbb{E}[f(\theta)] \mid y] \approx \frac{1}{M} \sum_{m=1}^{M} f\left(\theta^{(m)}\right)$$

for **independent and identically distributed** (i.i.d.) draws

$$\theta^{(m)} \sim p(\theta \mid y)$$

- Proof: **central limit theorem** (CLT) + law of **unconscious statistician**.

Markov chain Monte Carlo (Ulam, late 1940s)

- When **too hard to draw i.i.d.** from posterior.
- Draw $\theta^{(0)}, \ldots, \theta^{(M)}$ from a homogeneous **Markov chain**

$$\theta^{(0)} \sim q_0(\theta) \tag{1}$$

$$\theta_{m+1} \sim q(\theta_{m+1} \mid \theta_m). \tag{2}$$

- **Ergodic theorem** says we can use MCMC just like MC, when chain
  - is **irreducible** (doesn't get stuck),
  - is **aperiodic** (doesn't visit partition cyclically), and
  - has **stationary distribution** s.t. $\theta^{(m+1)} \sim p(\theta \mid y)$ if $\theta^{(m)} \sim p(\theta \mid y)$.
- Proof: equilibrium convergence + law of unconscious statistician

Accept/reject balance (Metropolis et al. 1950)

- **propose** $\theta^* \sim q(\theta \mid \theta_m)$,
  - where q is **symmetric**, i.e., $q(\theta' \mid \theta) = q(\theta \mid \theta')$
- **accept** with probability $\min\left(1,\ \frac{p(\theta^* \mid y)}{p(\theta_m \mid y)}\right)$

```
theta[0] = q0_rng()
for m in range(M):
   theta_star = q_rng(theta[m])
   u = uniform_rng(0, 1)
   accept = log(u) < log p(theta_star) - log p(theta[m])
   theta[m + 1] = theta_star if accept else theta[m]
```

Non-reversible proposals (Hastings 1970)

- **propose** $\theta^* \sim q(\theta \mid \theta_m)$,
  - where q is **not necessarily symmetric**
- **accept** with probability $\min\left(1, \frac{p(\theta^* \mid y)}{p(\theta_m \mid y)} \cdot \frac{q(\theta_m \mid \theta^*)}{q(\theta^* \mid \theta_m)}\right) t$
- If q is symmetric, second term drops out, reduces to Metropolis

Detailed Balance

- Let $p(\theta_{m+1} \mid \theta_m)$ be the Markov chain **transition kernel**
  - reject probability makes it mixed discrete/continuous
- Metropolis-Hastings **accept** step ensures MCMC kernel satisfies **detailed balance**,

$$p(\theta \mid y) \cdot p(\theta' \mid \theta) = p(\theta' \mid y) \cdot p(\theta \mid \theta').$$

  - ensures chain has stationary distribution $p(\theta \mid y)$
  - given irreducibility and aperiodicity

HMC

(Duane et al. 1987)

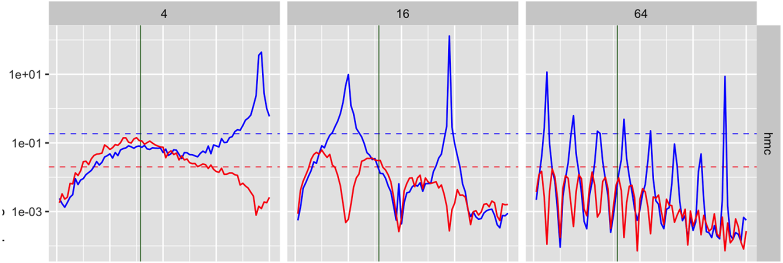
- **couples momentum** $\rho \in \mathbb{R}^D$ to sample over **phase space** $\mathbb{R}^D \times \mathbb{R}^D$
- **Hamiltonian** $H(\theta, \rho) = -\log p(\theta, \rho) = -\log p(\theta) - \log p(\rho)$,
  - **Kinetic energy**: $-\log p(\rho) = -\log \mathrm{normal}(0, \mathrm{I}_D) = \frac{1}{2} \cdot \theta^\top \cdot \theta$.
  - **Potential energy**: $-\log p(\theta) = -\log p(\theta \mid y)$
- **Hamiltonian Monte Carlo** (HMC) couples two stationary-preserving transition kernels:
  - **Exact momentum** refresh: $\rho_{m+1} \sim \mathrm{normal}(0, 1)$.
  - **Metropolis proposal**: $(\theta^*, -\rho^*)$, where (θ^*, ρ^*) solves Hamiltonian dynamics from initial (θ_m, ρ_{m+1}) to proposal (θ^*, ρ^*) at time t.
  - Metropolis **corrects numerical integration error** solving ODE
  - Momentum **flip for reversibility** (required, but erased)

Why is HMC so good?

- **Long distance** proposals with **high acceptance** rates
- **Effective sample size** (and hence **mixing**) is proportional to expected squared jump distance
- An **exact solution** preserves initial Hamiltonian, so 100% accept
- The **leapfrog integrator** used to solve dynamics is **symplectic**
  - steps preserve volume (hence no Hastings correction for Jacobian)
  - symplectic integrators very good at **preserving Hamiltonian**
  - not so great at **solving dynamics**, but doesn't matter—we only need **long jumps with high acceptance**

HMC is hard to tune

- ***y*-axis**: ESS; ***x*-axis**: step size; **facets**: steps
- **blue**: Y; **red**: Y^2
- **dashed lines**: NUTS



HMC + Euclidean metric

- add symmetric, positive-definite **metric** M to define distance

$$||\theta|| = \theta^{\top} \cdot M \cdot \theta \tag{3}$$

$$d(\theta, \theta') = ||\theta - \theta'|| \tag{4}$$

- Stan estimates M as inverse posterior covariance,

$$\widehat{M} \approx \mathrm{cov}[\theta \mid y].$$

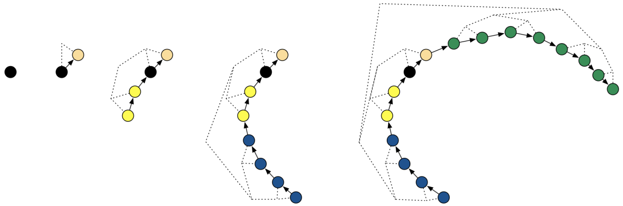
- **kinetic energy** now $-\log p(\rho) = \frac{1}{2} \cdot \rho^{\top} \cdot M \cdot \rho$,
- so **momentum refresh** $\rho \sim \mathrm{normal}(0, M^{-1})$.

No-U-turn sampler (NUTS) (Hoffman and Gelman 2013)

- **Tuning HMC** dynamics (step size, number of steps) is **very hard**
- No-U-turn sampler (NUTS) automatically tunes
  - **stepsize** during warmup iterations
  - **metric** during warmup iterations
  - **number of steps** dynamically
- Betancourt (2017) added several improvements to NUTS for Stan

NUTS (cont.)

- Sampling algorithm
  - randomly simulate **forward vs. backward** time, **doubling steps**
  - **until U-turn** (momentum brings ends closer)
  - **select a step** along path, **biased** toward **last doubling**
  - **slice sampling**, revised to more efficient **multinomial**



Generalized HMC (Horowitz 1991)

- Uses **partial momentum refresh**
  - preserves momentum for directed movement across posterior
- Fix $\lambda \in (0, 1)$ (lower preserves more momentum)
- **G-HMC refresh**:

$$\rho_{m+1} = \sqrt{\lambda} \cdot z_m + \sqrt{1 - \lambda} \cdot \rho_m,$$

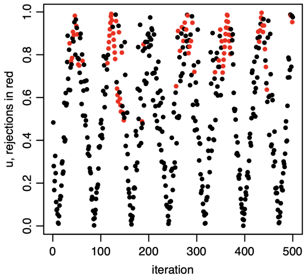
  where $z_m \sim \mathrm{normal}(0, M^{-1})$.
- This is also an **exact update**
  - i.e., if $\rho_m \sim \mathrm{normal}(0, M^{-1})$, then $\rho_{m+1} \sim \mathrm{normal}(0, M^{-1})$.

Uh oh! What about the Flip?

- **Why not** use G-HMC with a single step and tune λ?
- Remember that **momentum flip**?
  - it's required for **reversibility** of Metropolis
  - **thrown away** in basic HMC by composing momentum update
  - but **preserved** in G-HMC
- **Without high acceptance**, momentum flip produces **random walk**.
- High acceptance means **small step size** in Hamiltonian dynamics.

Non-reversible accept (Neal 2020)

- Instead of uniform u, use **sawtooth** pattern, **jittered** for ergodicity
  - **not reversible**, but **preserves stationary**
- Iteration vs. accept probability u (red reject) **groups acceptance**:



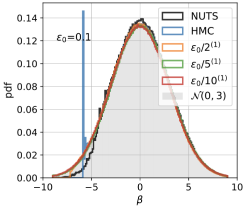
Delayed Rejection (Mira 2001)

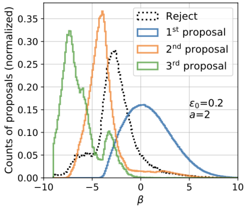
- K-step delayed rejection involves K distinct proposals
  - Step 1. Propose and accept/reject as usual with Metropolis.
  - Step 2. **If rejected, propose again** with a new proposal and accept/reject with **Metropolis-Hastings**

    $\vdots$
  - Step K. If rejected, propose one last time.
- Need **Hastings correction** to ensure detailed balance.
- Proposals may depend on previous proposal(s).
- **Innovation:** Only retry if acceptance probability was low

Delayed Rejection HMC (Modi et al. 2022)

- **Multiple scales** (varying curvature) along an **entire path**
- Each **retry cuts step size** by constant c and **multiplies steps** by c (e.g., $c = 2$ or $c = 5$)
  - earlier attempts kept step size and extended path
  - ours better if gradient-based **Hamiltonian diverged**





(funnel draws/accept)

MEADS (Hoffman and Sountsov 2022)

- **Massively parallel** version of Neal's non-reversible accept **G-HMC**
- Uses **complementary chains for adaptation**
  - novel, efficient **principal eigenvalue** estimation of step size
  - covariance in complementary chains used to **estimate metric**
  - **accelerates adaptation**, robust to single chains getting stuck (Bales 2019)
  - much easier to **parallelize**
  - **little waste** and **easy restart**
- conveniently a Markov chain **w.o. warmup phase**
  - still need **burn-in**

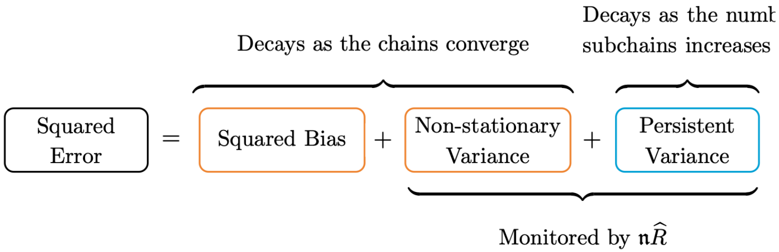
DR-G-HMC

(Modi, Roualdes, . . . in progress)

- Apply **delayed rejection** to **generalized HMC**
- **Retries** use **one step** instead of constant time
- **Multiple scales** adapted **within a trajectory**
- solves **non-centered funnel** (like DR-HMC)
- Use MEADS-like **parallel tuning**

Convergence: nested $\widehat{R}$ (Margossian et al. 2022)

- **GPUs** run **1000+** parallel chains
- **One draw**/chain **minizes wall time**
- **Nest blocks** with **same init**; monitor **transient bias** + **variance**



Part II
Stan Language

Tuples (next release)

- Required **design doc** choices
  - mainly on syntax and including **named structs**
- **Massive refactoring** required in transpiler & I/O (thanks Brian and Steve)
  - **my bad** in original code assuming **dense rectangular**

```
tuple(int, real) x = (42, 3.14);  // type & construct

int a = x.1;    real b = x.2;     // rvalue (access)

x.1 += 1;       x.2 = sqrt(x.2);  // lvalue (assign)
```

Closures and lambdas (design accepted)

- **Closures bind** non-local variables in function bodies
- **Lambdas** for anonymous inline function definitions (C++-like syntax)
- Full **type** support in the object language
  - variables with function types and assignment
  - function arguments with function types (i.e., higher-order functions)
- Support **comprehensions** with **partial evaluation**
  - e.g., scalable GP kernels

Ragged arrays (design pending)

- Not much more to say about this
- Challenging to **statically size**
- Need to **generalize math** library

- **Bigger challenge: sparse matrices**
  - Eigen C++ support is poor
  - Have to roll our own algorithms (e.g., log determinant)

More types and constraints

- **Orthonormal matrices** (generalizes unit vector)
  - e.g., hyperspherical statistics and rotations
  - tricky $SO(N)$ geometry (rotation vs. reflection)
- **Composable** transforms
  - e.g., affine compositions for hierarchical rates or probabilities
- **Pluggable** transforms
  - e.g., specialized simplex transforms for scale

Part III

Automatic Differentiation

Mat-var to Var-mat **(underway)**

- **Reverse-mode** autodiff uses **autodiff variables**
  - one per value evaluated in the constraints and log density
  - C++ class instance store a **value** and **adjoint**
  - virtual function to apply chain rule
- Stan matrices currently store a **matrix of variables**
- Moving to more efficient **variable of matrixes**
  - more **memory locality**
  - less **copying**
- **huge refactor** of math lib and transpiler

Expression templates (well underway)

- original Stan **evaluated** expression templates
- goal: **lazy evaluation** plus **static compilation**
- improved Stan tries to **propagate** expression templates
  - **reduces copying**
  - accelerate **composed code compilation**avoids eager, unnecessary copies
  - Fortrain-like speed; how C++ is faster than C
- very **tricky C++**

GPU kernel **(starting)**

- Very **expensive to move** data between GPU and CPU
- now use where order of computation $/gg$ order of data
  - e.g., matrix-matrix multiply, Cholesky factorization, **not** matrix-vector multiply, dot prducts, etc.
- Write **GPU code** for entire **math library**
- **Order of magnitude (or two!)** speedup (cf. JAX)
- **Enormous** undertaking

Part IV
Approximate Inference

(cf. **MCMC**, which is asymptotically **exact**)

Laplace Approximation (released)

- Given a **posterior mode**

$$\theta^* = \arg\max_\theta \, p(\theta \mid y)$$

- **Second-order Taylor expansion** is called **Laplace approximation**

$$\theta \sim \text{multi\_normal}(\theta^*, (-H)^{-1}(\theta^*)),$$

  where $(-H)^{-1}$ is the inverse negative Hessian

- pairs with our built-in gradient-based **quasi-Newton optimization**
  - uses **L-BFGS** for local curvature adjustment (i.e., **precondition**)

Autodiff Variational Inference (ADVI)

- Given posterior $p(\theta \mid y)$ on the **unconstrained scale**
  - i.e., $p(\theta \mid y) > 0$ for all $\theta \in \mathbb{R}^D$
- **Variational approximation** is $\text{multi\_normal}\left(\theta \mid \hat{\mu}, \hat{\Sigma}\right)$, where

$$\hat{\mu}, \hat{\Sigma} = \text{argmin}_{\mu,\Sigma} \, \text{KL}\big[\text{multi\_normal}(\theta \mid \mu, \Sigma) \,\big|\big|\, p(\theta \mid y)\big]$$

- **Sample** $\theta^{(1)}, \ldots \theta^{(M)} \sim \text{multi\_normal}\left(\theta \mid \hat{\mu}, \hat{\Sigma}\right)$,
- **Inverse transform** draws back to constrained scale
- **Impovements** by (1) Welandawe, Andersen, Vehtari, and Huggins (2022); (2) Domke and Agrawal (2022).

ADVI objective evaluation

- KL-divergence is integral, w. $q(\theta) = \mathrm{multi\_normal}(\mu, \Sigma)$.

$$\mathrm{KL}[\, q \mid\mid p \,] = \underbrace{\int_{\mathbb{R}^D} q(\theta) \cdot \log q(\theta) \, \mathrm{d}\theta}_{\text{entropy of } q} - \underbrace{\int_{\mathbb{R}^D} q(\theta) \cdot \log p(\theta \mid y) \, \mathrm{d}\theta}_{\text{cross entropy } q \text{ to } p} \quad (5)$$

$$\approx -\mathrm{H}[\mathrm{multi\_normal}(\mu, \Sigma)] + \frac{1}{M} \sum_{m=1}^{M} \log p(\theta^{(m)} \mid y), \quad (6)$$

where $\theta^{(m)} \sim \mathrm{multi\_normal}(\mu, \Sigma)$.

ADVI SGD

- Need **gradient** $\nabla_\theta \, \mathrm{KL}[\, q \;||\; p \,]$ to minimize

$$\arg\min_{\mu,\Sigma} \mathrm{KL}\big[\, \text{multi\_normal}(\theta \mid \mu, \Sigma) \;\big|\big|\; p(\theta \mid y) \,\big]$$

- **Entropy** term can be handled analytically or by Monte Carlo
- **Cross-entropy** requires **stochastic gradient**,

$$\nabla_\theta \sum_{m=1}^{M} \log p(\theta^{(m)} \mid y) = \sum_{m=1}^{M} \nabla_\theta \log p(\theta^{(m)} \mid y),$$

  with nested derivative by **automatic differentiation**

ADVI improvements (in progress)

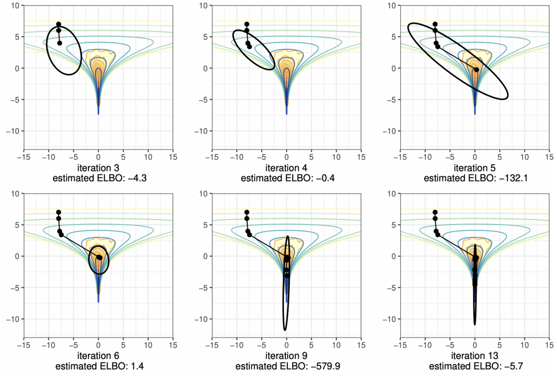
- Need to select a **step size**
  - current algorithm too weak; parallel grid search way better
- Need to select an **SGD algorithm**
  - ADVI is vanilla SGD; **ADAM**'s persistent momentum better
- Need to select **gradient estimator**
  - Stan uses vanilla reparameterization; **stick the landing** is better
- Stan uses standard normal **initialization**
  - **Multiple inits** better (Laplace, prior, standard normal, etc.)
- Stan just **returns constrained draws**
  - **importance resampling** is better

Pathfinder (next release!)

- **Quasi-Newton** variational inference with **L-BFGS**
- At each point on optimization trajectory:
  - lay down a multivariate normal approximation
  - covariance is approximate **negative inverse Hessian**
  - estimated with finite differences of autodiff gradient
  - low-rank (number of gradients $J \approx 5$) plus diagonal: $\mathcal{O}(D \cdot J^2)$
- Choose approximation with **lowest KL-divergence** (Monte Carlo)
- Importance **resample**
- More robust: **multiple paths**, combined importance resampling

Pathfinder illustrated

- Ellipses = Taylor approximation; **lower left** is best (but **overconcentrated**)



The future: normalizing flows (Domke and Agrawal)

- Like ADVI with **normalizing flows** as approximating family
  - Generate **inital** $X \sim q_X(x)$ (cf. Domke and Agrawal)
  - **Transform** with **deep neural net** $f_\beta . \mathbb{R}^D \rightarrow \mathbb{R}^D$, $\Theta = f_\beta(X)$,

$$q_\Theta(\theta \mid \beta) = q_X\left(f_\beta^{-1}(\theta)\right) \cdot \left| J_{f_\beta^{-1}}(\theta) \right|$$

  - **Optimize** as in ADVI: $\beta^* = \arg\min_\beta \mathrm{KL}[\, q_\Theta(\theta \mid \beta) \,||\, p(\theta \mid y) \,]$
- For VI, neural net must have **efficient Jacobian and sampling**
- Real non-volume preserving (**RealNVP**) flows work with **JAX**
  - complementary affine layers, $\tanh()$ non-linearity ($\approx$ 10 deep, 12 wide)
  - fits centered hieararchical IRT-2PL (additive + multiplicative + funnel)

More work in progress

- score-function based **control variates**
- **amortized variational inference** (BRMS)
- **black-box nested Laplace** approximation
- low-dimensionsional (1–2), massively **scalable Gaussian processes**