Sampling beyond MCMC

- Simple MCMC is a good general tool, but
  - curse of dimensionality
  - requires tuning — e.g., proposal distributions
  - inefficient

- Other sampling techniques exist
  - usually for cases when you have more information about the distributions
  - **Gibbs sampling** — need to have the conditional probabilities for different parameters, $P(\theta_1|\theta_2,d)$
  - **Hamiltonian Monte Carlo** — need derivatives $\partial P(\theta)/\partial \theta$
Gibbs Sampling

- Metropolis-Hastings with Proposal = conditional dist’n
  - all samples accepted
  - satisfies detailed balance
  - no adjustable parameters in the algorithm
- suited to hierarchical models (often written in terms of the conditionals)
- Algorithm:
  - \( x_1^{(n+1)} \sim P(x_1|x_2^{(n)}, x_3^{(n)}, \ldots) \)
  - \( x_2^{(n+1)} \sim P(x_2|x_1^{(n+1)}, x_3^{(n)}, \ldots) \)
  - \( x_3^{(n+1)} \sim P(x_3|x_1^{(n+1)}, x_2^{(n+1)}, \ldots) \)
- Should change (reverse/randomize) the order 1, 2, 3, … in successive steps
- Caveats: can fail badly if the distribution isn’t aligned with the axes and/or highly curved
- *Otherwise often use “metropolis-within-Gibbs”

Especially good if these can be “analytically” sampled

McKay, Information Theory...
Gibbs Sampling

- Algorithm:
  - \( x_1^{(n+1)} \sim P(x_1 | x_2^{(n)}, x_3^{(n)}, \ldots) \)
  - \( x_2^{(n+1)} \sim P(x_2 | x_1^{(n+1)}, x_3^{(n)}, \ldots) \)
  - \( x_3^{(n+1)} \sim P(x_3 | x_1^{(n+1)}, x_2^{(n+1)}, \ldots) \)

- Note that conditional distributions are just the full distribution with the other parameters held fixed (up to normalization).

\[
P(x | y) = \frac{P(x, y)}{P(y)} \propto P(x, y)
\]

- In a hierarchical model, get the full posterior by multiplying out all the distributions that appear

- See Alan Heavens’ talk later…

McKay, Information Theory…
Hamiltonian Monte Carlo (HMC)

- (aka Hybrid Monte Carlo; Duane et al 1987)
- Analogy with dynamical systems, which explore (position, momentum) phase space over time
  - Potential \( U(\theta_i) = -\ln P(\theta_i) \) w/ “positions” \( \theta_i \)
  - KE \( K(u_i) = \frac{1}{2} u \cdot u \) w/ “momenta” \( u_i \sim N(0, \sigma^2) \)
  - Hamiltonian \( H(\theta_i, u_i) = U(\theta_i) + K(u_i) \)
  - Density \( P(\theta_i, u_i) = e^{-H(\theta, u)} \)
  - 2N parameters!
- Evolve as dynamical system
  - ignore (marginalize over) momenta

\[
\begin{align*}
\dot{\theta}_i &= \frac{\partial H}{\partial u_i} = u_i \\
\dot{u}_i &= -\frac{\partial H}{\partial \theta_i} = \frac{\partial \ln P}{\partial \theta_i}
\end{align*}
\]
- Need to discretize the system (time derivatives)
- Values of $(\theta_i, u_i)$ at different times: proposed MC samples
- If exact dynamics, $H$ conserved, $\implies$ all samples accepted
  - in practice, approximate evolution (and, e.g., numerical derivatives)
  - so, accept $(\theta_i, u_i)^*$ as step $n+1$ with probability
    \[
    \min \left[ 1, \exp \left( -H^* + H^{(n)} \right) \right]
    \]
**HMC Algorithm (1)**

- **Algorithm** (Hajian *PRD*75 083525, 2007)

1: initialize \(x_0\)
2: for \(i = 1\) to \(N_{\text{samples}}\)
3: \(u \sim \mathcal{N}(0, 1)\)
4: \((x_0^*, u_0^*) = (x_{i-1}, u)\)
5: for \(j = 1\) to \(N\)
6: make a leapfrog move: \((x_{j-1}^*, u_{j-1}^*) \rightarrow (x_j^*, u_j^*)\)
7: end for
8: \((x^*, u^*) = (x_N, u_N)\)
9: draw \(\alpha \sim \text{Uniform}(0, 1)\)
10: if \(\alpha < \min\{1, e^{-(H(x^*, u^*) - H(x, u))}\}\)
11: \(x_i = x^*\)
12: else
13: \(x_i = x_{i-1}\)
14: end for

Only propose every \(N\) timesteps

Discretisation step! (see problem sheet)
HMC Algorithm (2)

R version (Neal, in Handbook of MCMC)

```r
HMC = function (U, grad_U, epsilon, L, current_q)
{
  q = current_q
  p = rnorm(length(q),0,1) # independent standard normal variates
  current_p = p

  # Make a half step for momentum at the beginning
  p = p - epsilon * grad_U(q) / 2

  # Alternate full steps for position and momentum
  for (i in 1:L)
  {
    # Make a full step for the position
    q = q + epsilon * p
    # Make a full step for the momentum, except at end of trajectory
    if (i!=L) p = p - epsilon * grad_U(q)
  }

  # Make a half step for momentum at the end.
  p = p - epsilon * grad_U(q) / 2

  # Negate momentum at end of trajectory to make the proposal symmetric
  p = -p

  # Evaluate potential and kinetic energies at start and end of trajectory
  current_U = U(current_q)
  current_K = sum(current_p^2) / 2
  proposed_U = U(q)
  proposed_K = sum(p^2) / 2

  # Accept or reject the state at end of trajectory, returning either
  # the position at the end of the trajectory or the initial position
  if (runif(1) < exp(current_U-proposed_U+current_K-proposed_K))
  {
    return (q)  # accept
  }
  else
  {
    return (current_q)  # reject
  }
}
```

Single \(L\)-step trajectory

Leapfrog method
HMC vs Metropolis-Hastings

Figure 30.2. (a,b) Hamiltonian Monte Carlo used to generate samples from a bivariate Gaussian with correlation \( \rho = 0.998 \). (c,d) For comparison, a simple random-walk Metropolis method, given equal computer time.

MacKay, *Information Theory…*

Neil, *Handbook of MCMC*

Figure 6: Values for the variable with largest standard deviation for the 100-dimensional example, from a random-walk Metropolis run and an HMC run with \( L = 150 \). To match computation time, 150 updates were counted as one iteration for random-walk Metropolis.
HMC with millions of parameters

- From large-scale structure observations to the *primordial* density field
- Forward physics model from primordial density to observed galaxy distribution
- Related work from Jasche, Lavaux, Kitaura, F. Leclercq & B. Wandelt
HMC as a generic tool

- Gelman et al, STAN (http://mc-stan.org/)
- Uses *automatic differentiation* to get derivatives for ~anything that can be built up from elementary functions
- e.g., SED fitting
# Stan Code

```stan
data {
  int<lower=1> N_comp;       // # of greybody components
  // (fixed model parameter)
  int<lower=1> N_band;       // number of photometric bands
  vector[N_band] nu_obs;     // observed frequency
  vector[N_band] flux;       // observed flux
  vector[N_band] sigma;      // error
  real z;                    // redshift
}
transformed data {
  vector[N_band] nu;         // rest frame frequency
  nu = (1+z)*nu_obs;
}
functions {
  real greybody(real beta, real T, real nu) {
    // greybody, normalized to unit flux at nu=nu_0
    real h_over_k;
    real x;
    real nu_bar;
    real x_bar;
    nu_bar = 1000;
    h_over_k = 0.04799237;     // K/Ghz
    x = h_over_k * nu / T;
    x_bar = h_over_k * nu_bar / T;
    return (pow(nu/nu_bar, 3+beta) * expm1(x_bar) / expm1(x));
  }
}
parameters {
  // nb. N_comp, N_band are data
  vector<lower=0>[N_comp] amplitude;
  positive_ordered[N_comp] T;
  // greybody factor
  vector<lower=0, upper=3>[N_comp] beta;
}
model {
  real fluxes[N_band, N_comp];
  vector[N_band] totalflux;
  for (band in 1:N_band) {
    for (comp in 1:N_comp) {
      // vectorize over this?
      fluxes[band, comp] = amplitude[comp] *
      greybody(beta[comp], T[comp], nu[band]);
    }
    totalflux[band] = sum(fluxes[band]);
  }
  // try a proper prior on temperature;
  // needed since ordered vectors don't have limits
  T ~ uniform(3,100);
  flux ~ normal(totalflux, sigma);
}
```
Inference from a Gaussian: Averaging

- The simplest “linear model”
- Consider $data = signal + noise$,
- $d_i = \mu + n_i$ for data points $i=1\ldots N$

  - Noise, $n_i$, has zero mean, known variance $\sigma^2$
    - Assign a Gaussian to $(d_i - \mu)$
      - Alternately: keep $n_i$ as a parameter and marginalize over it with $p(d_i|n_i, \mu I) = \delta(d_i-n_i-\mu)$

  - Prior for $s$ (i.e., $a$ and $b$)?
    - To be careful of limits, could use Gaussian with width $\Sigma$, take $\Sigma \to \infty$ at end of calculation
      - Same answer with uniform dist’n in $(-\Sigma_1, \Sigma_2) \to (-\infty, \infty)$
Inference from a Gaussian: Averaging

- **Posterior:**
  
  \[ P(\mu | d) = \frac{1}{\sqrt{2\pi \sigma_b^2}} \exp \left[ -\frac{1}{2} \frac{(\mu - \bar{d})^2}{\sigma_b^2} \right] \]

  - best estimate of signal is average ± stdev:
    \[ \mu = \bar{d} \pm \sigma_b = \bar{d} \pm \sigma/\sqrt{N} \]

- What if we don’t know \( \sigma \)? try Jefferys \( P(\sigma | I) \propto 1/\sigma \)
  
  - marginalize over \( \mu \): \( P(\mu | d) \propto \left[ \mu^2 - 2\mu \bar{d} + \bar{d}^2 \right]^{-1/2} \)
    - Student t or Cauchy distribution
      - (very broad distribution!)
A toy model: estimating the mean and variance

- Back to our averaging problem, \( d_i = s + n_i \)
- \( P(n_i|I) = \text{Gaussian w/} \)
  \( \langle n_i \rangle = 0, \langle n^2 \rangle = \sigma^2 \)
- \( P(s|I) = \text{Uniform} \)
- Toy version of measuring cosmological maps and power spectra (see Alan Heaven’s talk)
- Take \( \sigma^2 \text{unknown w/ prior} \)
  \( P(\sigma) \propto 1/\sigma \) (improper…)

Hierarchical model (see Alan's talk):

\[ \begin{align*}
\mu & \quad \sigma \\
\sigma & \quad n_i \\
& \quad d_i \\
& \text{Measured}
\end{align*} \]
A toy model: estimating the mean and variance

Back to our averaging problem, \( d_i = \mu + n_i \)

\[
P(\mu, \sigma \mid d) = \frac{1}{\sigma} \left( \frac{1}{(2\pi\sigma^2)^{n/2}} \right) \exp \left[ -\frac{n}{2\sigma^2} \left( d^2 - 2\mu \bar{d} + \mu^2 \right) \right]
\]

\[\propto \frac{1}{\sigma^{n+1}} \exp \left[ -\frac{1}{2} \frac{(\mu - \bar{d})^2}{\sigma^2/n} \right] \exp \left[ -\frac{n}{2\sigma^2} \left( d^2 - \bar{d}^2 \right) \right]\]

- Unknown noise variance \( \sigma^2 \), Uniform prior on \( \mu \)
- Posterior is Gaussian in \( \mu \), Gamma in \( 1/\sigma^2 \)
- Conditionals are known for Gibbs.

Algorithm:

\[
\mu \mid (\sigma^2, d) \leftarrow \text{Normal} \left( \bar{d}, \sigma^2/n \right)
\]

\[
\sigma^2 \mid (\mu, d) \leftarrow \text{InvGamma} \left( \frac{n-1}{2}, \frac{n}{2} \left[ d^2 - 2\mu \bar{d} + \mu^2 \right] \right)
\]
Case study

- Estimating a mean and variance.