Information Content of Partial Computations

division with remainder

\[
\begin{array}{cccc}
2 & 3 & 7 & 3 \\
\div & 7 & 3 & 6 \\
\end{array}
= \quad X & X & . & X & X
\]
Information Content of Partial Computations

division with remainder

\[
\begin{array}{cccc}
2 & 3 & 7 & 3 & 6 \\
\end{array}
\div
\begin{array}{cccc}
7 & 3 & 6 \\
\end{array}
=
\begin{array}{cccc}
3 & X & . & X & X
\end{array}
\]

\[
\begin{array}{cccc}
2 & 2 & 0 & 8 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 6 & 5 & 6
\end{array}
\]
Information Content of Partial Computations

division with remainder

\[
\begin{array}{cccccc}
2 & 3 & 7 & 3 & 6 & \div & 7 & 3 & 6 & = & 3 & 2 & \cdot & X & X \\
2 & 2 & 0 & 8 & 0 \\
1 & 6 & 5 & 6 \\
1 & 4 & 7 & 2 \\
1 & 8 & 4 \\
\end{array}
\]
Information Content of Partial Computations

division with remainder

\[
\begin{array}{ccccc}
2 & 3 & 7 & 3 & 6 \\
\div & 7 & 3 & 6 & =
\end{array}
\]

\[
\begin{array}{cccc}
2 & 2 & 0 & 8 & 0 \\
1 & 6 & 5 & 6 \\
1 & 4 & 7 & 2 \\
1 & 8 & 4 \\
1 & 4 & 7 & . & 2 \\
3 & 6 & . & 8
\end{array}
\]
Information Content of Partial Computations

division with remainder

\[
\begin{align*}
2 & 3 7 3 6 \div 7 3 6 = 3 2 . 2 5 \\
2 & 2 0 8 0 \\
1 & 6 5 6 \\
1 & 4 7 2 \\
1 & 8 4 \\
1 & 4 7 . 2 \\
3 & 6 . 8 \\
3 & 6 . 8 \\
0 & 
\end{align*}
\]
What about ML computations?

Contemporary computational tasks are more challenging

What happens with

- a neural net if we stop the “training” of a neural network after four steps of sgd?
- . . . on only 1% of the data set?
- a GP regressor if we stop the Gauss-Jordan elimination after three steps?
- a DP mixture model if we only run MCMC for ten samples?
- a robotic controller built using all these methods?
What about ML computations?

Contemporary computational tasks are more challenging

What happens with

- a neural net if we stop the “training” of a neural network after four steps of sgd?
- . . . on only 1% of the data set?
- a GP regressor if we stop the Gauss-Jordan elimination after three steps?
- a DP mixture model if we only run MCMC for ten samples?
- a robotic controller built using all these methods?

As data-sets becomes infinite, ML models increasingly complex, and their applications permeate our lives, we need to model effects of approximations more explicitly to achieve fast, reliable AI.
Machine learning methods are chains of numerical computations

- linear algebra (least-squares)
- optimization (training & fitting)
- integration (MCMC, marginalization)
- solving differential equations (RL, control)

Are these methods just black boxes on your shelf?
A numerical method estimates a function’s latent property given the result of computations.

- integration estimates \( \int_a^b f(x) \, dx \)
given \( \{f(x_i)\} \)
- linear algebra estimates \( x \) s.t. \( Ax = b \)
given \( \{As = y\} \)
- optimization estimates \( x \) s.t. \( \nabla f(x) = 0 \)
given \( \{\nabla f(x_i)\} \)
- analysis estimates \( x(t) \) s.t. \( x' = f(x, t) \),
given \( \{f(x_i, t_i)\} \)

- Computations yield “data” / “observations”
- Non-analytic quantities are “latent”
- Even deterministic quantities can be uncertain.
If computation is inference, it should be possible to build probabilistic numerical methods that take in probability measures over inputs, and return probability measures over outputs, which **quantify uncertainty** arising from the uncertain input and the finite information content of the computation.
Classic methods identified as maximum a-posteriori
probabilistic numerics is anchored in established theory

quadrature
Gaussian quadrature \rightarrow Gaussian process regression

linear algebra
conjugate gradients \rightarrow Gaussian conditioning

nonlinear optimization
BFGS \rightarrow autoregressive filtering

ordinary differential equations
Runge-Kutta \rightarrow Gauss-Markov extrapolation

[Diaconis 1988]
[Hennig 2014]
[Hennig & Kiefel 2013]
[Schober, Duvenaud & Hennig 2014]
Integration

\[ F = \int_{a}^{b} f(x) \, dx \]
Integration
a toy problem

\[ f(x) = \exp\left(-\sin^2(3x) - x^2\right) \quad \text{and} \quad F = \int_{-3}^{3} f(x) \, dx = ? \]
Integration

a toy problem

\[ f(x) = \exp\left(-\sin^2(3x) - x^2\right) \]

\[ \leq \exp(-x^2) \]

\[ F = \int_{-3}^{3} f(x) \, dx = ? \]

\[ \int \exp(-x^2) = \sqrt{\pi} \]
Monte Carlo
(almost) no assumptions, stochastic convergence

\[ F = \int \exp\left(-\sin^2(3x) - x^2\right) \, dx \]

\[ = Z \int \frac{f(x)}{g(x)} \frac{g(x)}{Z} \, dx \]

\[ \approx Z \frac{1}{N} \sum_{i} \frac{f(x_i)}{g(x_i)} = \hat{F} \]

\[ Z = \int g(x) \, dx \]

\[ x_i \sim \frac{g(x)}{Z} \]

\[ \text{var} \hat{F} = \frac{\text{var}_g(f/g)}{N} \]
Monte Carlo
(almost) no assumptions, stochastic convergence

\[ F = \int \exp \left( -\sin^2(3x) - x^2 \right) \, dx \]

\[ = Z \int \frac{f(x)}{g(x)} \frac{g(x)}{Z} \, dx \]

\[ \approx Z \frac{1}{N} \sum_{i} \frac{f(x_i)}{g(x_i)} = \hat{F} \]

\[ Z = \int g(x) \, dx \]

\[ x_i \sim \frac{g(x)}{Z} \]

\[ \text{var} \, \hat{F} = \frac{\text{var}_{g}(f/g)}{N} \]

\[ \Rightarrow \text{adding randomness enforces stochastic convergence} \]
The probabilistic approach
integration as nonparametric inference

\[ p(f) = \mathcal{GP}(f; 0, k) \quad \text{with} \quad k(x, x') = \min(x, x') + b \]
\[ p(z) = \mathcal{N}(z; \mu, \Sigma) \implies p(Az) = \mathcal{N}(Az; A\mu, A\Sigma A^\top) \]
\[ p\left( \int_a^b f(x) \, dx \right) = \mathcal{N}\left[ \int_a^b f(x) \, dx; \int_a^b m(x) \, dx, \int\int_a^b k(x, x') \, dx \, dx' \right] \]
\[ = \mathcal{N}(F; 0, -1/6(b^3 - a^3) + 1/2[b^3 - 2a^2b + a^3] - (b - a)^2c) \]
Active Collection of Information
choise of evaluation nodes

\[ x_t = \arg \min \left[ \text{var}_p(F | x_1, \ldots, x_{t-1}) (F) \right] \]
Active Collection of Information

Choice of evaluation nodes

\[ x_t = \arg \min \left[ \text{var}_p(F | x_1, \ldots, x_{t-1})(F) \right] \]
Active Collection of Information

choise of evaluation nodes

\[ x_t = \arg \min \left[ \text{var}_p(F | x_1, \ldots, x_{t-1})(F) \right] \]

active node placement for maximum expected error reduction gives regular grid.
Active Collection of Information

choise of evaluation nodes

\[ x_t = \arg \min \left[ \text{var}_P(F | x_1, \ldots, x_{t-1}) (F) \right] \]

active node placement for maximum expected error reduction gives regular grid.
Active Collection of Information

choise of evaluation nodes

\[ x_t = \arg \min \left[ \text{var}_{p(F| x_1, \ldots, x_{t-1})}(F) \right] \]

active node placement for maximum expected error reduction gives regular grid.
Active Collection of Information

choise of evaluation nodes

[T. Minka, 2000]

\[ x_t = \arg \min_p \left[ \text{var}_p(F | x_1, \ldots, x_{t-1}) (F) \right] \]

active node placement for maximum expected error reduction gives regular grid.
Active Collection of Information
choise of evaluation nodes

\[ x_t = \arg\min \left[ \text{var}_p(F \mid x_1, \ldots, x_{t-1})(F) \right] \]

active node placement for maximum expected error reduction gives regular grid.
$x_t = \arg \min \left[ \text{var}_p(F \mid x_1, \ldots, x_{t-1}) (F) \right]$

active node placement for maximum expected error reduction gives regular grid.
Active Collection of Information
choise of evaluation nodes

\[ x_t = \arg \min \left[ \text{var}_{p(F|x_1, \ldots, x_{t-1})}(F) \right] \]

active node placement for maximum expected error reduction gives regular grid.
Active Collection of Information

choise of evaluation nodes

[T. Minka, 2000]

\[ x_t = \arg \min \left[ \text{var}_{p(F \mid x_1, \ldots, x_{t-1})}(F) \right] \]

active node placement for maximum expected error reduction gives regular grid.
Active Collection of Information
choise of evaluation nodes

\[ x_t = \arg \min \left[ \text{var}_p(F \mid x_1, \ldots, x_{t-1})(F) \right] \]

active node placement for maximum expected error reduction
gives regular grid.
Posterior Mean is a Linear Spline...


\[ p(f \mid y, x) = \mathcal{GP}(f; m_y, k_y) \]
\[ m_y(x) = \sum_i k(x, x_i)\alpha_i \]
\[ k(x, x_i) = \min(x, x_i) + b \]
\[ \alpha_i = k_X X^{-1} y \]
We just re-discovered the Trapezoid Rule!
A classic numerical method, derived as a learning machine

\[ p(f \mid y, x) = \mathcal{GP}(f; m_y, k_y) \]

\[ m_y(x) = \sum_i k(x, x_i) \alpha_i \]

\[ k(x, x_i) = \min(x, x_i) + b \]

\[ \alpha_i = k_x X^{-1} y \]

\[ \mathbb{E}_{p(f \mid y)}[F] = \int m_y(x) \, dx = \sum_{i=1}^{N-1} (x_{i+1} - x_i) \frac{1}{2} (f(x_{i+1}) + f(x_i)) \]
The trapezoid rule is the maximum a-posteriori estimate for $F$ under a Wiener process prior on $f$.

Node placement by information maximization under this prior.
introducing random numbers into a deterministic problem may not be a good strategy

recipe for a probabilistic numerical method estimating $x$ from $y(t)$:
  - choose model $p(x, y(t))$
  - choose action rule / policy / strategy
  $[t_1, \ldots, t_{i-1}, y(t_1), \ldots, y(t_{i-1})] \rightarrow t_i$

some classic numerical methods can be derived entirely from an inference perspective, using classic statistical methods
  - the trapezoid rule is a MAP estimate under a Wiener process prior on the integrand; regular node placement arises from information-greediness

the probabilistic interpretation as such does not ensure the posterior distribution is well calibrated
Customized Numerics

machine learning providing new numerics

[Hennig, Osborne, Girolami, RSPA 2015]

\[ k(x, x') = \exp \left( -\frac{(x - x')^2}{2\lambda^2} \right) \]

Encodes
- smooth (infinitely differentiable) \( f \)
- exponentially decaying Fourier power spectrum

But ignores
- non-stationarity
- positivity
Customized Numerics

machine learning providing new numerics

[Hennig, Osborne, Girolami, RSPA 2015]

\[ k(x, x') = \exp\left(-\frac{(x - x')^2}{2\lambda^2}\right) \]

Encodes
- smooth (infinitely differentiable) \( f \)
- exponentially decaying Fourier power spectrum

But ignores
- non-stationarity
- positivity
No Such Thing as a Free Lunch!

incorrect assumptions give arbitrarily bad performance

[Hennig, Osborne, Girolami, RSPA 2015]

Computations collect information about a latent quantity. The more valid prior information is available, the cheaper the computation—if the algorithm uses the prior information!
Computations collect information about a latent quantity. The more valid prior information is available, the cheaper the computation—if the algorithm uses the prior information!

No Such Thing as a Free Lunch!

incorrect assumptions give arbitrarily bad performance

[Hennig, Osborne, Girolami, RSPA 2015]
No Such Thing as a Free Lunch!
incorrect assumptions give arbitrarily bad performance

[Hennig, Osborne, Girolami, RSPA 2015]

Computations collect information about a latent quantity. The more valid prior information is available, the cheaper the computation—if the algorithm uses the prior information!
Model Mismatch can be Detected at Runtime

"a numerical conscience"

\[ r = \log \frac{E_{\tilde{f}}[p(\tilde{f}(x))]}{p(f(x))} = (f(x) - \mu(x))^\top K^{-1}(f(x) - \mu(x)) - N \]
including tangible prior information in the prior can give tailored numerics that drastically improve computational efficiency

in contrast to physical data sources, in numerical problems, prior assumptions can be rigorously verified, because the problem is stated in a formal (programming) language!

incorrect prior assumptions can catastrophically affect performance

model assumptions can be adapted at runtime, using established statistical techniques, e.g. “type-II maximum likelihood”
So why is everyone (in ML) using MCMC?
\[
\sqrt{f(x)} \cdot \exp(-x^2/2) \sim \mathcal{GP}[0, k = \exp(-1/2(x - x')^\top \Lambda^{-1}(x - x'))]
\]

- encodes positivity
- encodes nonstationarity
select evaluation nodes at $\arg\max \text{var}[f(x) \cdot \exp(-x^2)]$

- this scales to higher input dimensionality

more formal analysis in arXiv 1410.2392 [Oates et al.] & 1506.02681 [Briol et al.]
select evaluation nodes at \( \arg \max \text{var}[f(x) \cdot \exp(-x^2)] \)

this scales to higher input dimensionality

more formal analysis in arXiv 1410.2392 [Oates et al.] & 1506.02681 [Briol et al.]
Warped Sequential Active Bayesian Integration (WSABI)

[Gunther, Osborne, Garnett, Hennig, Roberts, NIPS 2014]

- select evaluation nodes at $\text{arg max } \text{var}[f(x) \cdot \exp(-x^2)]$
- this scales to higher input dimensionality

more formal analysis in arXiv 1410.2392 [Oates et al.] & 1506.02681 [Briol et al.]
select evaluation nodes at \( \arg \max \text{var}[f(x) \cdot \exp(-x^2)] \)

- this scales to higher input dimensionality

more formal analysis in arXiv 1410.2392 [Oates et al.] & 1506.02681 [Briol et al.]
select evaluation nodes at \( \arg \max \text{var}[f(x) \cdot \exp(-x^2)] \)

this scales to higher input dimensionality

more formal analysis in arXiv 1410.2392 [Oates et al.] & 1506.02681 [Briol et al.]
select evaluation nodes at \( \arg \max \text{var}[f(x) \cdot \exp(-x^2)] \)

this scales to higher input dimensionality

more formal analysis in arXiv 1410.2392 [Oates et al.] & 1506.02681 [Briol et al.]
Warped Sequential Active Bayesian Integration (WSABI)

[Gunther, Osborne, Garnett, Hennig, Roberts, NIPS 2014]

- select evaluation nodes at \( \text{arg max} \text{ var}[f(x) \cdot \exp(-x^2)] \)
- this scales to higher input dimensionality

more formal analysis in arXiv 1410.2392 [Oates et al.] & 1506.02681 [Briol et al.]
Probabilistic Numerics Need Not Be Expensive

WSABI is time-competitive with MCMC [Gunther, Osborne, Garnett, Hennig, Roberts, NIPS 2014]
end of Integration part

- computation is inference
- there is a deep formal connection between basic numerical methods and basic statistical models
- ignoring salient prior information causes drastic computational cost increase. Black boxes may be convenient, but they are not efficient
- machine learning can help numerics, and vice versa
Ordinary Differential Equations

\[ x'(t) = f(x(t), t) \quad x(t_0) = x_0 \quad x : \mathbb{R} \to \mathbb{R}^N \]
Runge-Kutta Methods

iterative linear extrapolation

\[ x(t) \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>( t_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( c_1 )</td>
<td>1</td>
<td>( t_0 + c_1 )</td>
</tr>
<tr>
<td></td>
<td>( c_2 )</td>
<td></td>
<td>( t_0 + c_2 )</td>
</tr>
<tr>
<td></td>
<td>( h )</td>
<td></td>
<td>( t_0 + h )</td>
</tr>
</tbody>
</table>

\[
y_1 = f(1x_0, t_0)
\]

\[
y_2 = f \left( 1x_0 + w_{11}y_1, t_0 + c_1 \right)
\]
Runge-Kutta Methods

iterative linear extrapolation

\[\hat{x}(t_0 + h) = x_0 + \sum_i b_i y_i\]

\[
\begin{array}{c|ccc}
0 & 1 & & \\
c_1 & 1 & w_{11} & \\
c_2 & 1 & w_{21} & w_{22} \\
\hline
h & & & \\
\end{array}
\]

\[
y_1 = f(x_0, t_0)
\]
\[
y_2 = f(x_0 + w_{11}y_1, t_0 + c_1)
\]
\[
y_{s+1} = f(x_0 + \sum_i^s w_{si}y_i, t_0 + c_s)
\]

\[\parallel \hat{x}(t_0 + h) - x(t_0 + h) \parallel = O(h^p), p = 23,\]
Runge-Kutta Methods

iterative linear extrapolation

\[ x(t) \]

\[ \hat{x}(t_0 + h) = x_0 + \sum_i b_i y_i \]

- RK choose \((c, w, b)\) such that \( \| \hat{x}(t_0 + h) - x(t_0 + h) \| = \mathcal{O}(h^p) \)
Gauss-Markov inference on ODEs

a probabilistic model matching Runge-Kutta

[Schöber, Duvenaud, Hennig, NIPS 2014]
Gauss-Markov inference on ODEs

a probabilistic model matching Runge-Kutta

[Schober, Duvenaud, Hennig, NIPS 2014]

- Linear extrapolation suggests Gaussian process model
- Polynomial form suggests Wiener state-space model

\[ dz = F \, dt + L \, d\omega \]

\[
d \begin{bmatrix}
  x(t) \\
x'(t) \\
\vdots \\
h^k/k! x^{(k)}(t)
\end{bmatrix} = \frac{1}{h} \begin{bmatrix}
  0 & 1 & \cdots \\
  0 & 2 & \ddots \\
  \vdots & \ddots & k \\
  0 & \cdots & 0
\end{bmatrix} dt + \sigma \begin{bmatrix}
  0 \\
  \vdots \\
  k \\
  1
\end{bmatrix} d\omega
\]

- Inference through filtering
Calibrating Uncertainty
within the parametrized class

- posterior mean $\mu | y = kK^{-1}y$ invariant under $k \rightarrow \theta^2 k$
- posterior covariance $k | y = k - kK^{-1}k$ scaled by $\theta^2$
- connection to local error estimation in existing methods
as in integration, classic families of ODE solvers can be interpreted as MAP inference

for each classic solver, whole family of probabilistic solvers with same point estimate, different uncertainties

probabilistic solvers need not be expensive. Can even have exact same cost as classic methods
Propagating Uncertainty

geodesics on an uncertain Riemannian metric [Hauberg, Schober, Liptrot, P.H., Feragen, MICCAI 2015]

- Shortest paths (geodesics) on Riemannian manifold of metric $M$ obey

$$x''(t) = -1/2M^{-1}(x(t)) \left[ \frac{\partial \hat{M}(x(t))}{\partial x(t)} \right]^T (x'(t) \otimes x'(t)) = f(x(t), x'(t), t)$$
Propagating Uncertainty

geodesics on an uncertain Riemannian metric [Hauberg, Schober, Liptrot, P.H., Feragen, MICCAI 2015]

Fig. 2. Top row: The density of a single GP geodesic under the random metric. The density heatmap is projected into axis-aligned slices; the background image is the expected metric trace; and the outline is where at least one expert annotated the CST.

Fig. 3. Example shortest paths in the CST and ILF using both Dijkstra’s algorithm on a deterministic metric, and a GP solver with a random Riemannian metric.

Fig. 4. Agreement with at least one expert in the Catani atlas as estimated by Dijkstra’s algorithm and the GP solver.

Shortest paths (geodesics) on Riemannian manifold of metric $M$ obey

$$x''(t) = -\frac{1}{2}M^{-1}(x(t)) \left[ \frac{\partial M(x(t))}{\partial x(t)} \right]^T (x'(t) \otimes x'(t)) = f(x(t), x'(t), t)$$

what if $M \sim \mathcal{N}(m, V)$ is uncertain (inferred from data)?
numerical methods able to deal with uncertain (probabilistic) inputs, and returning uncertain (probabilistic) outputs, allow control of computational effort

(this is not the same as probabilistic programming!)
The probabilistic view of computation

- computation is (active) inference
- several classic methods can be interpreted precisely as MAP inference
  - Gaussian Quadrature—Gaussian process regression
  - Runge-Kutta Methods—Autoregressive Filtering
- [more to come on Monday]
- correct prior information can reduce runtime
- prior assumptions can be tested at runtime
- probabilistic uncertainty can be propagated between different numerical computations, allowing control of cost

“[PN is] a re-emerging area of very active research”

http://probabilistic-numerics.org
Monte Carlo is not all that Different

MC as a ‘cautious’ limit

\[ p(f) = \mathcal{GP}(f; 0; k_{\lambda}^{OU} + c) \]

\[ \lambda \to \infty \quad \Rightarrow \quad k(x, x') \triangleq |x - x'| \]

\[ \lambda \to 0 \quad \Rightarrow \quad k(x, x') \to \delta(x - x') \]

\[ k_{\lambda}^{OU}(x, x') = \exp \left( -\frac{|x - x'|}{\lambda} \right) \]

\[ \to \text{trapezoid rule} \]

\[ \to \text{averaging} \]
Monte Carlo is not all that Different
MC as a ‘cautious’ limit

\[ p(f) = \mathcal{GP}(f; 0; k_{\lambda}^{OU} + c) \]

\[ \lambda \rightarrow \infty \quad \Rightarrow \quad k(x, x') \triangleright |x - x'| \]

\[ \lambda \rightarrow 0 \quad \Rightarrow \quad k(x, x') \rightarrow \delta(x - x') \]

\[ k_{\lambda}^{OU}(x, x') = \exp \left( -\frac{|x - x'|}{\lambda} \right) \quad \rightarrow \text{trapezoid rule} \]

\[ k_{\lambda}^{OU}(x, x') = \exp \left( -\frac{|x - x'|}{\lambda} \right) \quad \rightarrow \text{averaging} \]
Monte Carlo is not all that Different

**MC as a ‘cautious’ limit**

\[
p(f) = \mathcal{G}\mathcal{P}(f; 0; k_{\lambda}^{OU} + c)
\]

\[
\lambda \to \infty \quad \Rightarrow \quad k(x, x') \overset{\Delta}{=} |x - x'|
\]

\[
\lambda \to 0 \quad \Rightarrow \quad k(x, x') \to \delta(x - x')
\]

\[
k_{\lambda}^{OU}(x, x') = \exp\left(-\frac{|x - x'|}{\lambda}\right)
\]

\[\rightarrow \text{trapezoid rule}\]

\[\rightarrow \text{averaging}\]
Monte Carlo is not all that Different
MC as a ‘cautious’ limit

\[ p(f) = \mathcal{GP}(f; 0; k_{\lambda}^{OU} + c) \]

\[ \lambda \to \infty \quad \Rightarrow \quad k(x, x') \overset{\Delta}{=} |x - x'| \]

\[ \lambda \to 0 \quad \Rightarrow \quad k(x, x') \to \delta(x - x') \]

\[ k_{\lambda}^{OU}(x, x') = \exp \left( -\frac{|x - x'|}{\lambda} \right) \]

\[ \rightarrow \text{trapezoid rule} \]

\[ \rightarrow \text{averaging} \]
Monte Carlo is not all that Different
MC as a ‘cautious’ limit

\[ p(f) = \mathcal{GP}(f; 0; k^{OU}_\lambda + c) \]
\[ k^{OU}_\lambda(x, x') = \exp\left(-\frac{|x - x'|}{\lambda}\right) \]
\[ \lambda \to \infty \implies k(x, x') \overset{\Delta}{=} |x - x'| \]
\[ \lambda \to 0 \implies k(x, x') \to \delta(x - x') \]

\( \rightarrow \) trapezoid rule
\( \rightarrow \) averaging
Monte Carlo is not all that Different

MC as a ‘cautious’ limit

MC is optimal for totally unstructured (but integrable) $f$. But

- random node placement is not necessary! Introducing randomness to a deterministic problem is generally a bad idea.

- no integrand is totally unstructured. Prior knowledge is available! Computations should use as much available prior information as possible.
Monte Carlo is not all that Different
MC as a ‘cautious’ limit

MC is optimal for totally unstructured (but integrable) $f$. But
- random node placement is not necessary!
  Introducing randomness to a deterministic problem is generally a bad idea.
- no integrand is totally unstructured. Prior knowledge is available!
  Computations should use as much available prior information as possible.
What is a Random Number?

What is a sequence of random numbers?

- 662244111144334455553366666666
- 169399375105820974944592307816
- 712904263472610590208336044895
- 1000111111011111111100101000001
- 011100000011100100110111101100011
What is a Random Number?

What is a sequence of random numbers?

- 662244111144334455553366666666 dice, doubled
- 169399375105820974944592307816
- 712904263472610590208336044895
- 100011111101111111100101000001
- 0111000001110010011011101100011 deterministic sequence, corrupted by horizontal coin drop from unknown height

For use in Monte Carlo, the important property is freedom from patterns (because it implies anytime unbiasedness) in fact, use for MC only really requires the right density, disorder is just helpful for the argument.

For use in cryptography, the important property is unpredictability.

Randomness is a philosophically dodgy concept uncertainty is a much clearer idea.

http://www.stat.fsu.edu/pub/diehard/
What is a Random Number?

What is a sequence of random numbers?

- 66224411114433445553366666666  
  dice, doubled
- 169399375105820974944592307816  
  41-70th digits of \( \pi \)
- 712904263472610590208336044895
- 1000111111011111111100101000001
- 011100000011100100110111101100011

For use in Monte Carlo, the important property is freedom from patterns (because it implies anytime unbiasedness).

For use in cryptography, the important property is unpredictability.

Randomness is a philosophically dodgy concept. Uncertainty is a much clearer idea.

http://www.stat.fsu.edu/pub/diehard/
What is a Random Number?

What is a sequence of random numbers?

- 662244111144334455553366666666 dice, doubled
- 169399375105820974944592307816 41-70th digits of $\pi$
- 712904263472610590208336044895 von Neumann method, seed 908344
- 1000111111011111111100101000001
- 01110000011100100110111101100011
What is a Random Number?

What is a sequence of random numbers?

- 662244111144334455553366666666 dice, doubled
- 169399375105820974944592307816 41-70th digits of $\pi$
- 712904263472610590208336044895 von Neumann method, seed 908344
- 1000111111011111111100101000001 bits from G. Marsaglia’s ‘diehard CD’
- 01110000011100100110111101100011
What is a Random Number?

What is a sequence of random numbers?

- 662244111144334455553366666666 dice, doubled
- 169399375105820974944592307816 41-70th digits of $\pi$
- 712904263472610590208336044895 von Neumann method, seed 908344
- 1000111111011111111100101000001 bits from G. Marsaglia’s ‘diehard CD’
- 011100000011100100110111101100011 deterministic sequence, corrupted by horizontal coin drop from unknown height

http://www.stat.fsu.edu/pub/diehard/
What is a Random Number?

What is a sequence of random numbers?

- 662244111144334455553366666666 dice, doubled
- 169399375105820974944592307816 41-70th digits of π
- 712904263472610590208336044895 von Neumann method, seed 908344
- 100011111101111111110101000001 bits from G. Marsaglia’s ‘diehard CD’
- 01110000011100100110111101100011 deterministic sequence, corrupted by horizontal coin drop from unknown height

- for use in Monte Carlo, the important property is freedom from patterns (because it implies anytime unbiasedness)
- in fact, use for MC only really requires the right density, disorder is just helpful for the argument
- for use in cryptography, the important property is unpredictability
- randomness is a philosophically dodgy concept
- uncertainty is a much clearer idea

http://www.stat.fsu.edu/pub/diehard/
How Expensive is a Computation?

Ex: Riemannian Statistics

\[ x''(t) = -\frac{1}{2} M^{-1}(x(t)) \left[ \frac{\partial M(x(t))}{\partial x(t)} \right]^\top (x'(t) \otimes x'(t)) = f(x(t), x'(t), t) \]

- Shortest paths (geodesics) on Riemannian manifold of metric \( M \) obey

- Karcher mean \( \mu \) of data set \( \{x_i\}_{i=1,...,N} \) is point minimizing
  \[ \sum_{i}^{N} \text{Distance}(\mu, x_i) \]
to find Karcher mean, do gradient descent from initial guess $\mu_0$

$$\mu_{k+1} = \mu_k - \alpha \nabla_\mu \sum_i^N \text{Distance}(\mu, x_i)$$

- requires solving $N$ initial value problems, over and over.
How Much Information is Needed?

Ex: Riemannian Statistics

[Schober, Hauberg, Lawrence, Hennig; in prep.]
How Much Information is Needed?

Ex: Riemannian Statistics

[Schober, Hauberg, Lawrence, Hennig; in prep.]
How Much Information is Needed?
Ex: Riemannian Statistics

[Schober, Hauberg, Lawrence, Hennig; in prep.]