tinyDA

Multilevel Delayed Acceptance MCMC for Human Beings.

Mikkel Bue Lykkegaard

December 12, 2023

Data-Centric Engineering Group, University of Exeter, UK
digiLab, The Gallery, Kings Wharf, Exeter, UK

digiLab

- **Consultancy:** First-of-a-kind solutions.
- **Software:** User-friendly Uncertainty Quantification.
- **Academy:** Closing the skills gap.

MLDA: The Core Algorithm

"Monte Carlo is an extremely bad method; it should only be used when all alternative methods are worse."

– **Alan Sokal**

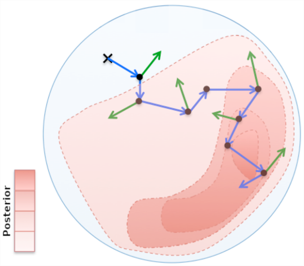
Metropolis-Hastings

Algorithm 1. Metropolis-Hastings (MH): for $j = 0$ to $N - 1$:

- Given θ^j, generate a proposal ψ distributed as $q(\psi|\theta^j)$,
- Accept proposal ψ as the next state, i.e. set $\theta^{j+1} = \psi$, with probability

$$\alpha(\psi|\theta^j) = \min\left\{1, \frac{\pi_t(\psi)q(\theta^j|\psi)}{\pi_t(\theta^j)q(\psi|\theta^j)}\right\} \tag{1}$$

otherwise reject ψ and set $\theta^{j+1} = \theta^j$.



Delayed Acceptance MCMC

Algorithm 2. Delayed Acceptance (DA): for $j = 0$ to $N-1$:

- Given θ^j, generate proposal ψ by invoking one step of **MH** Alg. 1 for coarse target π_{C}:

$$\psi = \mathbf{MH}\left(\pi_{\mathrm{C}}(\cdot), q(\cdot|\cdot), \theta^j, 1\right). \tag{2}$$

- Accept proposal ψ as the next state, i.e. set $\theta^{j+1} = \psi$, with probability

$$\alpha(\psi|\theta^j) = \min\left\{1, \frac{\pi_{\mathrm{F}}(\psi) q_{\mathrm{C}}(\theta^j|\psi)}{\pi_{\mathrm{F}}(\theta^j) q_{\mathrm{C}}(\psi|\theta^j)}\right\} \tag{3}$$

otherwise reject proposal ψ and set $\theta^{j+1} = \theta^j$.

Here, $q_{\mathrm{C}}(\cdot|\cdot)$ is the coarse level transition kernel

$$q_{\mathrm{C}}(\psi|\theta^j) = q(\psi|\theta^j)\alpha_{\mathrm{MH}}(\psi|\theta^j) + (1 - r(\theta^j))\delta_{\theta^j}(\psi) \tag{4}$$

Detailed Balance

Detailed Balance

For target density π_t, detailed balance for the transition kernel $K(\cdot|\cdot)$ may be written as

$$\pi_t(x)K(y|x) = \pi_t(y)K(x|y).$$

Lemma 1: If the proposal transition kernel $q(\cdot|\cdot)$ in Alg. 1 is in detailed balance with some distribution π^*, then the acceptance probability may be written as

$$\alpha(\psi|\theta^j) = \min\left\{1, \frac{\pi_t(\psi)\pi^*(\theta^j))}{\pi_t(\theta^j)\pi^*(\psi)}\right\}. \tag{5}$$

NB: For a state-independent coarse model, $q_C(\cdot|\cdot)$ **is** in detailed balance with $\pi_C(\cdot)$!

Surrogate Transition MCMC

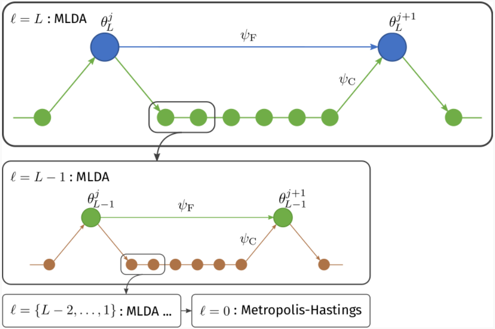
Lemma 2: Let $K_1(x|y)$ and $K_2(x|y)$ be two transition kernels that are in detailed balance with a density π and that commute. Then their composition $(K_1 \circ K_2)$ is also in detailed balance with π.

By induction $q_{\mathrm{C}}^n(\cdot|\cdot)$, is in detailed balance with $\pi_{\mathrm{C}}(\cdot)$ for any n.

Multilevel Delayed Acceptance

Theorem 1

Multilevel Delayed Acceptance (MLDA) generates a Markov chain that is in detailed balance with π_{F}.

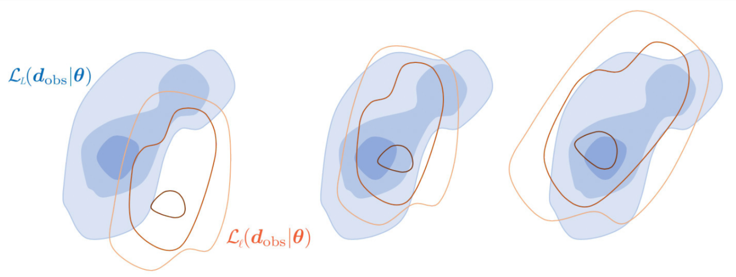


See **Lykkegaard, M. B**., Dodwell, T. J., Fox, C., Mingas, G., & Scheichl, R. (2023). *Multilevel Delayed Acceptance MCMC*. SIAM/ASA Journal on Uncertainty Quantification for more details.

Adaptive Error Models

- When the **coarse** approximation is **poor**, the fine level **acceptance rate** can be **low**.
- Every time we **promote a proposal**, we can evaluate $\mathcal{F}_F - \mathcal{F}_C$.
- We can then apply a **multilevel trick** to our statistical model.

$$\mathbf{d} = \mathcal{F}_C + \underbrace{\mathcal{F}_F - \mathcal{F}_C}_{\mathcal{B} \sim \mathcal{N}(\mu_\mathcal{B}, \Sigma_\mathcal{B})} + \underbrace{\mathbf{e}}_{\mathcal{N}(0, \Sigma_e)}$$



A Multilevel Adaptive Error Model

Add these terms to the **likelihood** on $\ell \in \{0, \dots, L-1\}$:

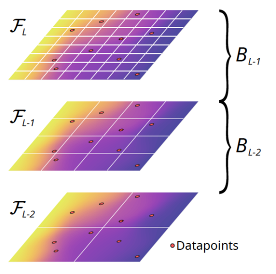
$$\mathcal{L}_\ell \propto \exp\left(-\frac{1}{2}(\mathcal{F}_\ell(\theta) + \boldsymbol{\mu}_{\mathcal{B}} - \mathbf{d})^T(\Sigma_{\mathcal{B}} + \Sigma_{\mathbf{e}})^{-1}(\mathcal{F}_\ell(\theta) + \boldsymbol{\mu}_{\mathcal{B}} - \mathbf{d})\right)$$

- Apply **recursively** across all levels.
- The parameters of the bias distributions can be built **sequentially** with very **little overhead**:

$$\boldsymbol{\mu}_{\mathcal{B},i+1} = \frac{1}{i+1}\left(i\boldsymbol{\mu}_{\mathcal{B},i} + \mathcal{B}(\theta_{i+1})\right)$$

and

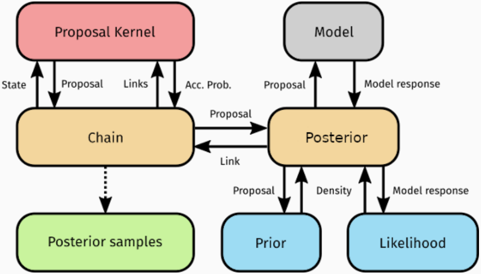
$$\Sigma_{\mathcal{B},i+1} = \frac{i-1}{i}\Sigma_{\mathcal{B},i} + \frac{1}{i}\left(i\boldsymbol{\mu}_{\mathcal{B},i}\boldsymbol{\mu}_{\mathcal{B},i}^T - (i+1)\boldsymbol{\mu}_{\mathcal{B},i+1}\boldsymbol{\mu}_{\mathcal{B},i+1}^T + \mathcal{B}(\theta_{i+1})\mathcal{B}(\theta_{i+1})^T\right)$$



tinyDA: Overview

tinyDA: Overview

- **Sampling algorithms:**
  - Metropolis–Hastings
  - Delayed Acceptance
  - MLDA
- **Proposals:**
  - RWM
  - pCN
  - Adaptive Metropolis
  - Operator-weighted pCN
  - DREAM
  - MALA and Kernel MALA
- **Error models:**
  - State-independent
  - State-dependent



`tinyDA`: Technical Details

- Pure **Python**
- **Free** and open-source
- Fully imperative
- Minimal **dependencies**:
  - the SciPy stack
  - ArviZ (for diagnostics)
  - Ray (for parallelisation)
- **UM-Bridge** support
- Modular and extensible

tinyDA: A minimal example

```
# set up the statistical model
prior = multivariate_normal(mean=np.zeros(2), cov=np.eye(2))
likelihood = tda.GaussianLogLike(y_data, sigma**2*np.eye(n_data))

# define a model (here it's a linear regression)
model = lambda theta: theta[0] + theta[1]*x_data

# initialise the tinyda.Posterior object
posterior = tda.Posterior(prior, likelihood, model)

# create a proposal and sample from the posterior
proposal = tda.AdaptiveMetropolis(1e-2*np.eye(2), adaptive=True)
samples = tda.sample(my_posterior, proposal, iterations=1200)

# convert the output to an arviz.InferenceData object
idata = tda.to_inference_data(samples, burnin=200)
```

tinyDA: UM-Bridge integration

```
# import tinyDA and UM-Bridge
import tinyDA as tda
import umbridge

# connect to the UM-Bridge model.
umbridge_model = umbridge.HTTPModel("http://0.0.0.0:4242",
                                    "forward")

# the model parameters are a log-Gaussian process,
# so we have to transform the GP input parameters
E = lambda x: 1e5*np.exp(x)

# wrap the UM-Bridge model in the tinyDA UM-Bridge interface
model = tda.UmBridgeModel(umbridge_model, pre=E)

# initialise the Posterior
posterior = tda.Posterior(prior, likelihood, model)

# initialise MALA proposal
proposal = tda.MALA()
...
```

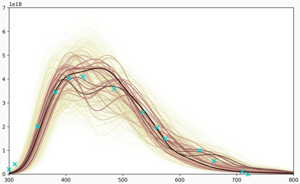
tinyDA: Applications

FuseUQ: Tritium Desorption

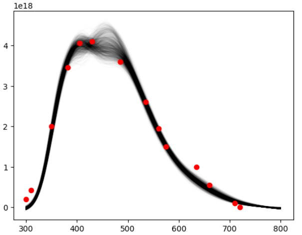
Joint work with Stephen Dixon (UKAEA)

MOOSE
Multiphysics Object-Oriented Simulation Environment

- Construct **functional GP** emulator.
  - Draw **QMC samples** from the prior.
  - Output **dimension reduction** using SVD.
  - Predict **modal strengths** using **GP emulator**.
  - Propagate the **uncertainty** to the function.



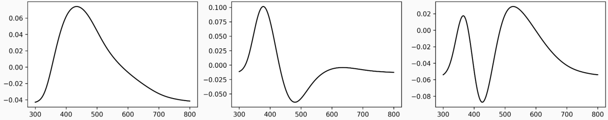
- Run **MLDA** using `tinyDA`.
  - **Functional GP** is used as **coarse model**.
  - **(FEM) model** is used as **fine model** and served by **UM-Bridge**.
  - Sample **posterior** using **Adaptive Metropolis**.



FuseUQ: Functional GP

Joint work with Stephen Dixon (UKAEA)

- **POD modes** of the model response:



- GP predicts the **modal strengths**:

$$\xi_i(\theta) \sim \mathrm{GP}_i(\mu(\theta), k(\theta, \theta')); \quad i = \{i, \ldots, 6\} \tag{6}$$

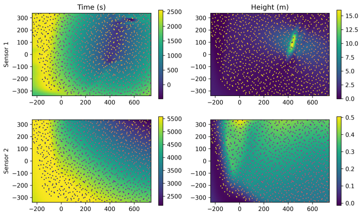
- ***Predictive uncertainty*-aware** likelihood functional:

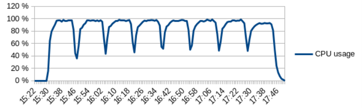
$$\hat{\mathcal{L}}(d|\theta) \propto -\frac{1}{2}(\mu_{\mathrm{GP}}(\theta) - d)^T \, \Sigma_{\mathrm{GP}}(\theta)^{-1} \, (\mu_{\mathrm{GP}}(\theta) - d) \tag{7}$$

UM-Bridge integration: Tsunami model

Joint work with Anne Reinarz and Linus Seelinger

- Create **GP emulator**.
  - Draw **QMC samples** from the **prior**.
  - Train **GP emulator**.
- Set up **3-level** MLDA with `tinyDA`.
  - Use GP emulator as the **coarsest model**.
  - Both **intermediate** and **finest** model are served by **UM-Bridge**.
  - Use **prior samples** to approximate **proposal covariance**.
- Run **100 samplers** in **parallel**.
  - `tinyDA` and the GP surrogate were deployed on small **local cluster**.
  - The UM-Bridge service was deployed on a **cloud computing platform**.





High-level priors: Poisson Point Process

- Forward model is the **integral equation**

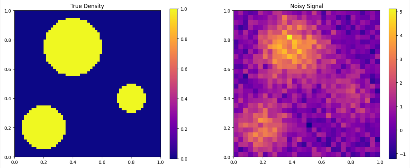
$$g(f, \mathbf{s}) = \iint_T \frac{h}{\|\mathbf{s} - \mathbf{t}\|_2^3} f(\mathbf{t})\, \mathrm{d}\mathbf{t}$$

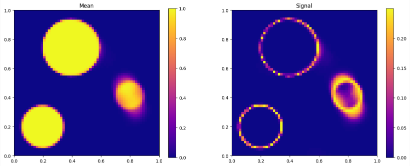
- The **objective** is to **recover** $p(f|d)$, with

$$d = g(f) + \varepsilon$$

- The **prior** $p(f)$ is a **Poisson Point Process** over the **number of (discrete) objects** with uniform location and radius.

- The **MLDA sampler** is driven by a `PoissonPointProposal`.





High-level priors: Markov Random Field Priors

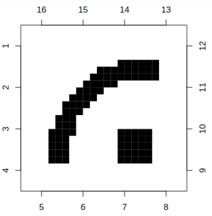
Joint work with Dave Higdon and Colin Fox

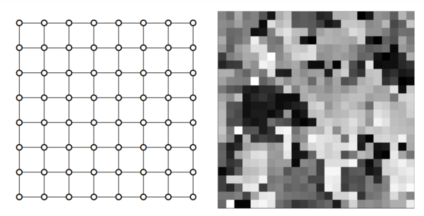
- Markov Random Field Prior:

$$\pi(x) \propto \exp\left\{\beta \sum_{i \sim j} u(x_i - x_j)\right\}, \; x \in [2.5, 4.5]^m$$

with

$$u(d) = \begin{cases} \frac{1}{s}(1 - |d/s|^3)^3 & \text{if } -s < d < s \\ 0 & \text{if } |d| \geq s\,. \end{cases}$$





MLDA for Electrical Impedance Tomography

Joint work with Dave Higdon and Colin Fox

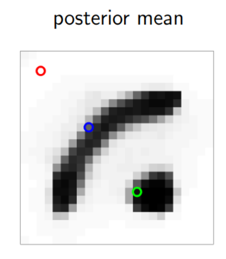
- Forward model is a **PDE**:

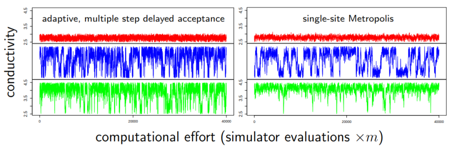
$$\begin{aligned} -\nabla \cdot x(s)\,\nabla v(s) &= 0 \\ x(s)\,\frac{\partial v(s)}{\partial n(s)} &= j(s) \end{aligned}$$

- The **fine model** solves the problem on a 24×24 grid, and the **coarse** on an 8×8 grid.

- An **adaptive error model** is used to correct the coarse model.

- The **MLDA** sampler significantly **improves** the MCMC sampling.





Future Directions

tinyDA

- ~~**Proposal distributions** that use **gradients** on the coarsest level: MALA, Kernel MALA.~~
- Higher–order **adaptive error modelling**. $\Rightarrow$ Gaussian Processes?
- Automatic accounting for **variance reduction**.
- **Embedded spaces** for hierarchical models.
- Wrapper for framework-agnostic **adaptive coarse models**.

tinyDA

`pip install tinyda`

https://github.com/mikkelbue/tinyDA

Questions?