Data-Driven, Bayesian Uncertainty Quantification for Large-Scale Simulations

Lina Kulakova

WITH: P. Angelikopoulos, P. Chatzidoukas, C. Papadimitriou, P. Koumoutsakos
Motivation

‣ Simulations in Engineering and Life Sciences usually involve computationally intensive models (e.g. Molecular Dynamics)

‣ Bayesian inference => large numbers of model evaluations

‣ High Performance Computing (HPC) is a must!

‣ How to exploit HPC architectures for Bayesian UQ+P?
Examples of Models
Example 1: Red Blood Cell model (1/3)

\[
\mathbf{F}_{\text{cell}} = \sum_{n=1}^{N} \mathbf{F}_{0,n-1,n,n+1}^{\text{dihedral,1}} + \mathbf{F}_{0,n,N+n,n+1}^{\text{dihedral,2}} + \mathbf{F}_{0,n,n+1}^{\text{triangle}} + \mathbf{F}_{0,n}^{\text{bond}}
\]

Calibrate for:
- maximum spring extension \( x_0 \)
- persistence length \( p \)

\[
x_n = \frac{b_n}{l_0/x_0}
\]

[spring-like force]

\[
[\text{dissipative force}]
\]

\[
\overline{dW}_{ij}^S = dW_{ij}^S - tr[dW_{ij}^S]1/3
\]
**Example 1: Red Blood Cell model (2/3)**

**Data:** stretching experiment (Suresh et al., 2005)

![Image](https://example.com/image1)

*Credit: Suresh et al., 2005*

**Simulation:** uDeviceX  
[https://github.com/uDeviceX/uDeviceX](https://github.com/uDeviceX/uDeviceX)

**Stochastic forward model:**

\[
D = M(\theta) + \epsilon, \epsilon \sim N(0, \Sigma),
\]

\[
\Sigma = \text{diag}(\sigma_1^2 + \tau_1^2 + \omega_1^2, \ldots, \sigma_2^2 + \tau_1^2 + \omega_1^2, \sigma_2^2 + \tau_2^2 + \omega_2^2, \ldots, \sigma_2^2 + \tau_2^2 + \omega_2^2)
\]
\[ \eta \dot{y}_i = F_i + F_i^R \]

**Cell with N subcellular elements**

**Parameters of \( F_i \):**

**Shape of the Morse-like potential:**

\[
V(r, \varphi) = u_0 \left( e^{2\rho (1-r^2/(\varphi^2 r_0^2))} - \alpha e^{\rho (1-r^2/(\varphi^2 r_0^2))} \right) \varphi^3
\]

**Stiffness**

\[
\kappa = \kappa_0 N^{-1/3} \left( 1 - \lambda N^{-1/3} \right)
\]

**Viscosity**

\[
\eta = \eta_0 / N
\]

\( r \)

\( v \)
Data: strain vs time (Desprat et al., 2005)

Stochastic forward model:

\[ D = M(\theta) \]

discrepancy:  

\[ SSE = \sum_{i=1}^{N_D} (\varepsilon_S(t_i) - \varepsilon_D(t_i))^2 \]
Example 3: Lennard-Jones for Helium (1/3)

Lennard-Jones potential:

\[ V_{LJ}(r; \sigma_{LJ}, \epsilon_{LJ}) = 4\epsilon_{LJ} \left[ \left( \frac{\sigma_{LJ}}{r} \right)^{12} - \left( \frac{\sigma_{LJ}}{r} \right)^{6} \right] \]

Data:

Boltzmann factor:

\[ f_B = \exp \left( -\frac{H}{T k_B} \right) \]

relative probability of a particular arrangement with a given energy

\[ \sigma_{LJ} = 0.2556 [nm] \]

\[ \epsilon_{LJ} = 0.141 \left[ \frac{ag \cdot nm^2}{ns^2 \cdot K} \right] \]

non-Gaussian!

PDF of the Boltzmann factor for a system with 1000 atoms
**Example 3: Lennard-Jones for Helium (2/3)**

**Discrepancies:**

1. **Gaussian Setting**
   
   \[ \rho(x, y) = \sqrt{\left( \frac{\mu_x - \mu_y}{\mu_x} \right)^2 + \left( \frac{\sigma_x - \sigma_y}{\sigma_x} \right)^2} \]

   mean and standard deviation

2. **Quantile Setting**
   
   \[ \rho(x, y) = \left( \sum_{k=1}^{4} \left( \frac{q_k(x) - q_k(y)}{q_k(x)} \right)^2 \right)^{1/2} \]

   \[ q = (0.2, 0.4, 0.6, 0.8) \]

   quantiles

3. **Kullback-Leibler Setting**

   \[ \rho(x, y) = D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} \, dx \]

   shows how much information was lost when approximating \( P \) with \( Q \)
Bayesian Inference Algorithms
**TMCMC**


- **Initialise**
  - Sample from the prior PDF $f(\theta | MD_i)$
  - Set annealing parameter: $p = 0$

- **Update $p$**
  - $p = p + dp$, s.t.
  - $\text{COV}[p(D|\theta^{(n)})^{dp}] = \beta$

- **Select**
  - Sample with weights proportional to the scaled likelihood $p(D|\theta^{(n)})^{dp}$

- **Rejuvenate**
  - Run MH from every selected sample: $\text{MH}(\mathcal{N}(\theta_i, \Sigma_g))$

- **Stop**

- **Yes**
  - $p >= 1$?

- **No**
  - $p >= 1$?

ABC-SubSim

- **Initialise**
  - Sample $\theta$ from the prior PDF $f(\theta|MD_i)$
  - Sample $y$ from $f(y|\theta, MD_i)$

- **Update $\delta$**
  - Set $\delta$ s.t. 20% of samples have discrepancy $< \delta$

- **Select**
  - Select sample if its discrepancy $< \delta$

- **Stop**

- **$\delta \leq \delta^*$?**
  - **Yes**
    - **Rejuvenate**
      - Run MH from every selected sample: $\text{MH}(\mathcal{N}(\theta_i, \Sigma_g))$
  - **No**
    - **Rejuvenate**
      - Run MH from every selected sample: $\text{MH}(\mathcal{N}(\theta_i, \Sigma_g))$
HPC Tools
HPC Approach: TORC

P. Hadjidoukas et al., 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2012

- Runs on various architectures ranging from laptops to clusters
- Automatic load-balancing
- Integrable with external software used for model evaluation (e.g. LAMMPS)
- Task-based parallelism
- Easy to write parallel code
TORC: code example

serial code

```c
void task(double *x, double *y)
{
    *y = x[0] + x[1];
}
int main(int argc, char *argv[])
{
    double result[100];
    for (int i=0; i<100; i++)
    {
        double d[2] = {drand48(), drand48()};
        task(d, &result[i]);
    }
    return 0;
}
```

TORC code

```c
void task(double *x, double *y)
{
    *y = x[0] + x[1];
}
int main(int argc, char *argv[])
{
    double result[100];
    torc_register_task(task);
    torc_init(argc, argv, MODE_MW);
    for (int i=0; i<100; i++)
    {
        double d[2] = {drand48(), drand48()};
        torc_task(-1, task, 2,
                  2, MPI_DOUBLE, CALL_BY_COP,
                  1, MPI_DOUBLE, CALL_BY_RES,
                  &d, &result[i]);
    }
    torc_waitall();
    return 0;
}
```
Pi4U: UQ Library on top of TORC

- Open-source library distributed under LGPL licence
- Available at http://www.cse-lab.ethz.ch/software/Pi4U

- Algorithms:
  - **TMCMC** (for exact Bayesian inference)
  - **ABC-SubSim** (for approximate Bayesian inference)
  - **CMA-ES** (for optimisation)
  - **Subset Simulation** (for rare events sampling)
  - **A-PNDL** (for adaptive parallel numerical differentiation)
Results
Example 1: Red Blood Cell model (3/3)

Algorithm: TMCMC

Done in collaboration with S. Litvinov, D. Alexeev

1024 samples per stage, 128 GPU nodes on Piz Daint (CSCS), 5 hours of wall-clock time per stage. CUDA + TORC workers.

MPV: \((0.49, 4.86)\) (our calibration) vs \((0.45, 1.99)\) (Fedosov, PhD thesis, 2010)

robust prediction

posterior distribution of the parameters

the prediction is within the error bars, but the trend is wrong
In Fig. 2A, we observe that final parameter range of prescribe uniform priors in the given range.

The model parameters to calibrate are not the same for the two models. In the SEM++ without HI model, we value of the acceptance rate.

is set to 0.5, and the minimum acceptance rate is set to 0.05. The stopping criterion of the method is based on the framework. In both cases, we generate

We perform simulations for the two different cell stretching models using the ABC-SubSim through the SSE

Desprat et al. [6]). The SEM++ simulations with and without HI are run as described previously. The discrepancy

N

we wish to carefully assess a range of possible model parameters. We extract experimental data values of

creep response [6]. In the previous section, we had not considered a large range of parameter variations, while here

We use ABC-SubSim to find the most likely SEM++ model for cells with respect to the experimentally measured

In silico

Algorithm: ABC-SubSim

2000 samples per stage, 384 CPUs on Brutus cluster (ETHZ), 2.5 hours of wall-clock time per stage. TORC workers.

posterior distribution of the parameters

robust prediction

Credit: A. Economides, G. Tauriello, 2015

	

	

	

	

	

	

	

	

	

	

	

	

	

	

	

	

	

	

	

	

	

	

	

	

	

	

	


model fails to fit the data!
Example 3: Lennard-Jones for Helium (3/3)

**Gaussian Setting**

unidentifiable

---

**Quantile Setting**

---

**Kullback-Leibler Setting**

---

“true” parameters:

\[
\sigma = 0.2556 \\
\epsilon = 0.141
\]

<table>
<thead>
<tr>
<th>Model</th>
<th>([\sigma_l, \sigma_r])</th>
<th>(\bar{\sigma})</th>
<th>(u_\sigma)</th>
<th>([\epsilon_l, \epsilon_r])</th>
<th>(\bar{\epsilon})</th>
<th>(u_\epsilon)</th>
<th>(N_{\text{gen}})</th>
<th>(\delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(M_G)</td>
<td>[0.1, 0.8]</td>
<td>0.2437</td>
<td>12.7%</td>
<td>[0.01, 1.0]</td>
<td>0.443</td>
<td>60.5%</td>
<td>4</td>
<td>0.02</td>
</tr>
<tr>
<td>(M_Q)</td>
<td>[0.1, 0.8]</td>
<td>0.2591</td>
<td>2.1%</td>
<td>[0.01, 1.0]</td>
<td>0.136</td>
<td>6.3%</td>
<td>7</td>
<td>2e-5</td>
</tr>
<tr>
<td>(M_{KL})</td>
<td>[0.1, 0.8]</td>
<td>0.2737</td>
<td>6.2%</td>
<td>[0.01, 1.0]</td>
<td>0.128</td>
<td>18.4%</td>
<td>6</td>
<td>0.09</td>
</tr>
</tbody>
</table>

15360 samples per stage, 4096 CPUs on Piz Daint (CSCS), 0.4 hours of wall-clock time per stage. MPI (LAMMPS)+TORC workers.
Pi4U allows to exploit efficiently HPC architectures for Bayesian inference in large-scale models

We must re-examine the validation of many classical simulation models in Science and Engineering

Bayesian inference offers a way to do this systematically
Thank you!