Parameter fitting for damped harmonic oscillator using Stan

Konsta Parkkali, Valtteri Turkki

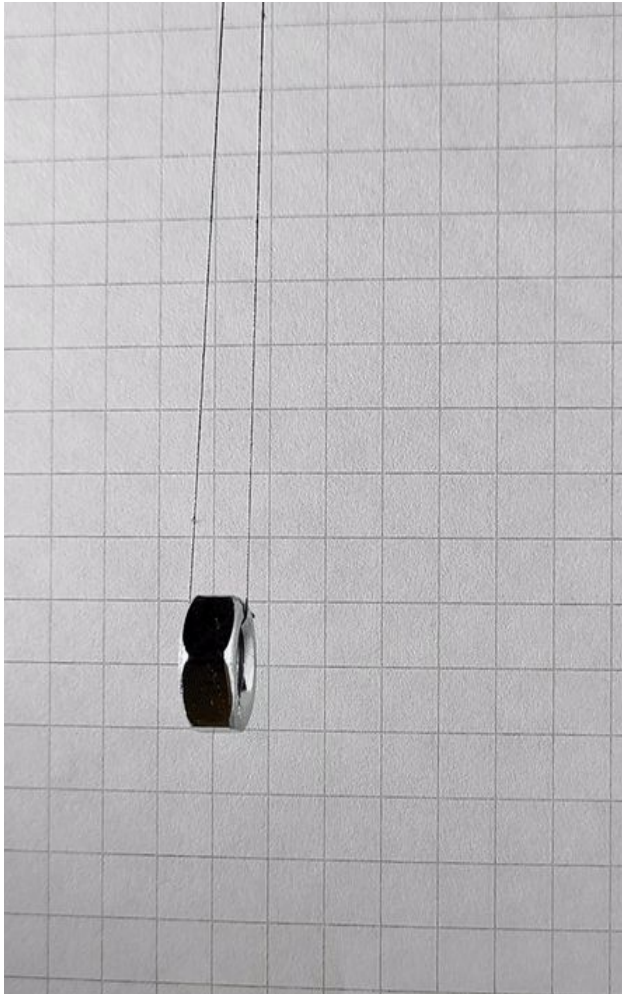
CS-E5710 Bayesian Data Analysis, fall 2022

Aalto University

“In this project we shall go nuts with nuts using HMC-NUTS.”

Introduction

- Inspiration from Stan case study Lotka-Volterra*
  - Motivation: Estimating the parameters of a dynamic system using bayesian inference
- Data was hard to find → measure own data
- Studied system was simple harmonic oscillator
  - We used two different pendulums
  - Interests in estimating the damping
- Two differential equation based Bayesian models
  - Pooled and hierarchical
  - Separate model was left out due to project length



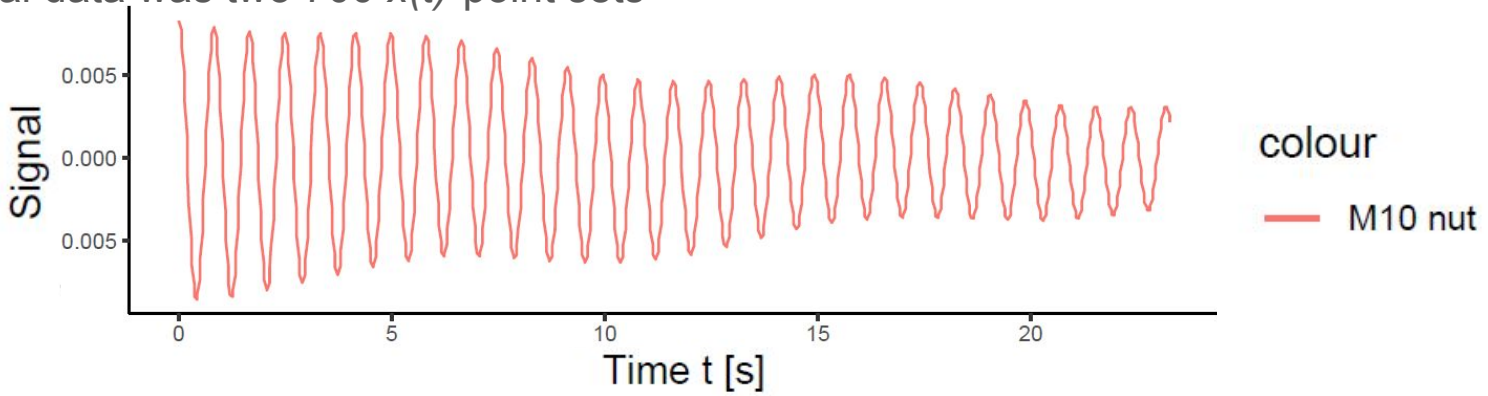
Aalto University

*https://mc-stan.org/users/documentation/case-studies/lotka-volterra-predator-prey.html

Data

$$\ddot{x} = -\frac{\mu}{ml}\dot{x} - \frac{g}{l}x$$

- We needed damped oscillations for the ODE
  - First we tried glass, but a classic pendulum was better
- Measurement setup:
  - A nut hang on the end of a string → nearly a point mass
  - Oscillation observed by taking a video using phone camera
  - The position *x(t)* was tracked using Tracker software (https://physlets.org/tracker/)
  - Final data was two 700 *x(t)*-point sets



Aalto University

Models - Priors

$$\ddot{x} = -\frac{\mu}{ml}\dot{x} - \frac{g}{l}x$$

- Ill-posedness is evident → informative priors were needed
- The mass *m*, string length *l* and gravity *g* are well known
  - This allows regularizing the solution
- All parameters positive by construction
- Normal priors used with best estimate mean and scale given confidence
- The 90% confidence levels:
  - Mass: 12 ± 8 grams
  - String length: 17.5 ± 5 cm
  - Gravitational acceleration: 9.81 ± 0.05 m/s^2
  - Initial state: x_0 = 8 ± 4 mm, $\dot{x}_0$ = 0 ± 0.05 m/s
- Air-resistance:
  - No prior information
  - ODE characteristic equation gives upper limit $\mu < \sqrt{4m^2lg}$
  - → μ < 0.1 kg*m/s

Aalto University

Models - Likelihood model

- Given the parameters $\phi = (m, g, \mu, l)$ and the initial condition $(x_0, \dot{x}_0)$ for the pendulum, we can simulate the oscillation by solving the ODE
- Measured data is normally distributed from the simulated data
  - Measurement error, model approximations, imperfect oscillation
  - $x_t \sim N(sim(x_0, \dot{x}_0, \phi, t), \sigma)$, where $\sigma \sim N_+(0,1)$
- Pooled model: pendulums have same underlying parameters, different initial conditions
- Hierarchical model: pendulums can have also different masses and air-resistances.
  - Hyperparameters μ_0 and m_0

Models - Pooled model

$$
\begin{aligned}
x_{t,i} &\sim N(sim(x_{0,i}, \dot{x}_{0,i}, \phi, t), \sigma) \text{ for } i = 1, 2, t = 1, \ldots, 700, \text{ and where } \phi = (m, g, \mu, l). \\
\sigma &\sim N_+(0, 1) \\
m &\sim N_+(0.012, 0.0049) \\
g &\sim N_+(9.81, 0.03) \\
\mu &\sim U(0, 0.1) \\
l &\sim N_+(0.175, 0.03) \\
x_{0,i} &\sim N(0.008, 0.002) \text{ for } i = 1, 2 \\
\dot{x}_{0,i} &\sim N(0.0, 0.07) \text{ for } i = 1, 2
\end{aligned}
$$

Models - Hierarchical model

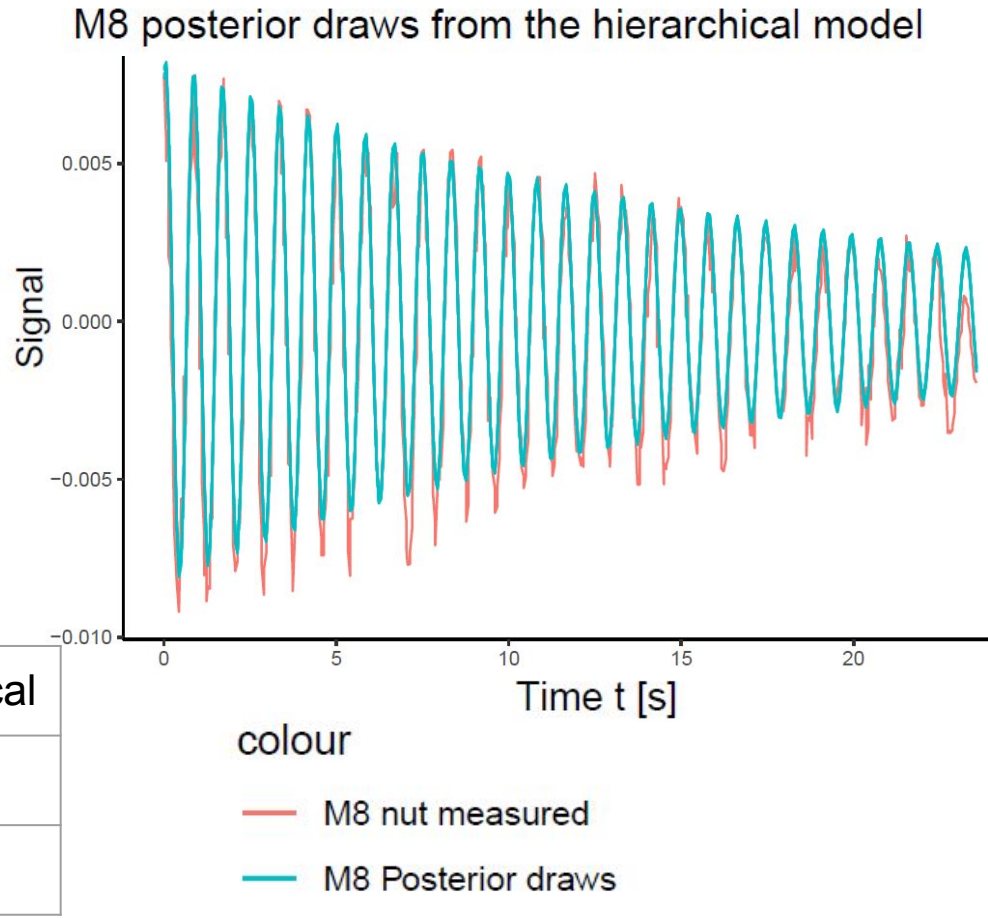
$$
\begin{aligned}
x_{t,i} &\sim N(sim(x_{0,i}, \dot{x}_{0,i}, \phi_i), \sigma) \text{ for } i = 1, 2, t = 1, \ldots, 700, \text{ and where } \phi_i = (m_i, g, \mu_i, l). \\
\sigma &\sim N_+(0, 1) \\
m_0 &\sim N_+(0.012, 0.0005) \\
m_i &\sim N_+(m_0, 0.0049) \text{ for } i = 1, 2 \\
g &\sim N_+(9.81, 0.03) \\
\mu_0 &\sim U(0, 0.1) \\
\mu_i &\sim N_+(\mu_0, 2\mu_0) \text{ for } i = 1, 2 \\
l &\sim N_+(0.175, 0.03) \\
x_{0,i} &\sim N(0.008, 0.002) \text{ for } i = 1, 2 \\
\dot{x}_{0,i} &\sim N(0.0, 0.07) \text{ for } i = 1, 2.
\end{aligned}
$$

Implementation and running

- Models computationally heavy
  - Laptop and JupyterHub could do only 200 samples in 15 minutes + memory problems
  - The final sampling was done on a cluster with > 20 cores and 1TB of RAM
  - 6 chains, 300 + 300 samples, all chains had fixed initial points
- Convergence properties
  - $\hat{R}$-values were good (≤1.015)
  - Posterior chains had no trends
  - Divergences and maximum treedepths hit ≈ 0
  - Effective sample size estimates ESS_{bulk} and ESS_{tail} were ≥ 320

Results - Model checking

- Pooled and hierarchical models were compared using PSIS-LOO cross-validation
  - Results favoured the hierarchical model slightly
  - All pareto-k estimates were well under 0.5



	Pooled	Hierarchical
ELPD-LOO	7750	7761
p_{eff}	8.8	10.3

Aalto University

Results - Numerical values

- First nut-pendulum seems to have larger air-resistance
  - Uncertainty is quite large, differences might be due to imperfect oscillation

Pooled model		
Parameter	Mean	90%-quantile
Mass m [g]	13	[6, 21]
Air-resistance μ $[\frac{kg \times m}{s}]$	2.2×10^{-4}	$[1.1 \times 10^{-4},\ 3.5 \times 10^{-4}]$
String length l [m]	0.171	[0.170, 0.172]
Gravity g $[m/s^2]$	9.81	[9.78, 9.86]
Hierarchical model		
Air-resistance M8 μ_1 $[\frac{kg \times m}{s}]$	2.4×10^{-4}	$[1.3 \times 10^{-4},\ 3.7 \times 10^{-4}]$
Air-resistance M10 μ_2 $[\frac{kg \times m}{s}]$	2.0×10^{-4}	$[0.9 \times 10^{-4},\ 3.3 \times 10^{-4}]$

Summary

- The aim of the project was to fit ODE to simple nut pendulum data using Stan
  - Ill-posed problem with 4 system parameters and 7 parameters in total
- Two models were used: Pooled and hierarchical
  - Models showed only minimal differences
  - Model comparison with LOO-CV supports this
  - Both models were computationally heavy
- The damping was different between the nuts
  - Though with so small difference that the root cause cannot be certainly assigned
  - Either real difference in the nut properties or difference between the two measurements

Aalto University

Contact: konsta.parkkali@aalto.fi, valtteri.turkki@aalto.fi

Additional material

Additional material - ODE

- System can be described with one coordinate θ
  - Equivalently with x-coordinate using small angle sine:

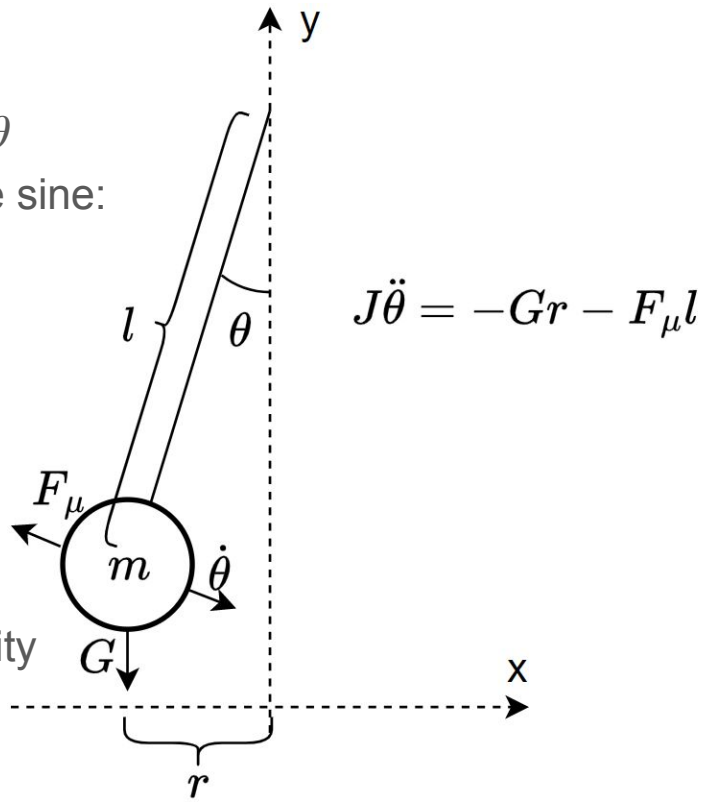
$$\sin(\theta) = \frac{x}{l} \quad \Rightarrow \quad \theta \approx \frac{x}{l}$$

- The ODE is then a 2nd order linear ODE

$$\ddot{x} = -\frac{\mu}{ml}\dot{x} - \frac{g}{l}x$$

- Assumptions:
  - θ is small, air-resistance is proportional to velocity

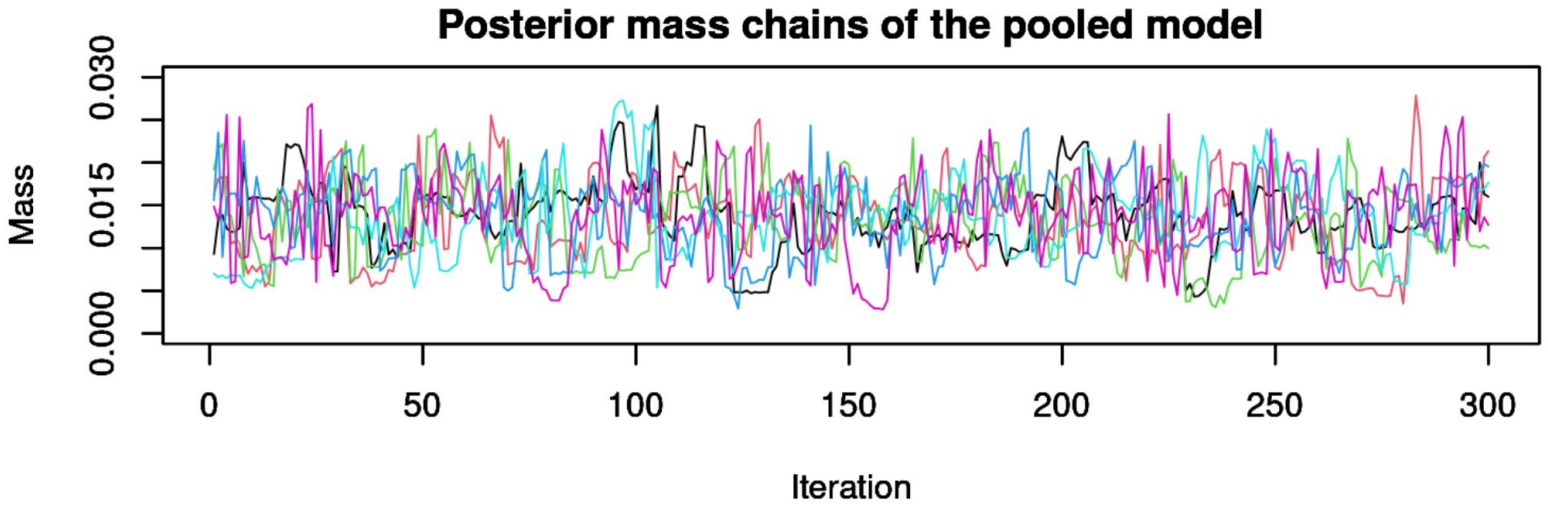
$$F_\mu = \mu\dot{x}$$



Aalto University

Additional material - Convergence

- There were no problems with convergence
  - Posterior chains had no trends



Aalto University

Additional material - Stan code

```
functions {

  // Function that encapsulates the dynamics of the system
  vector dx_dt(real t, vector x, real m, real l, real mu, real g) {
    vector[2] dxdt;
    dxdt[1] = x[2];
    dxdt[2] = (-mu/(m*l))*x[2] -  (g/l)*x[1];
    return dxdt;
  }

  // Function that solves the ODE and returns the position array
  real[] simulate_x(int N, real[] t, vector s0, real m,
                    real l, real mu, real g){
    vector[2] z[N-1] = ode_rk45(dx_dt, s0, 0.0, t[2:N], m, l, mu, g);
    array[1] real x0;
    x0[1] = s0[1];
    real x_sim[N] = append_array(x0, z[,1]);
    return x_sim;
  }
}
```

Aalto University

Additional material - Stan code

```
// The problem data
data {
  int<lower=1> N;
  matrix[N,2] x;
  array[N,2] real ts;
  matrix[8, 2] pr_params; // Matrix for prior parameters
}
```

Additional material - Stan code

```
// The problem parameters
parameters {
  vector[2] state01;
  vector[2] state02;
  real<lower=0> sigma;
  real<lower=0.000001> m; // Prevent division by zero issues
  real<lower=0.0001> l; // Prevent division by zero issues
  real<lower=0> mu;
  real<lower=0> g;
}
```

Additional material - Stan code

```
// The ODE-transformation from params to simulated data
transformed parameters {
  real x_sim1[N] = simulate_x(N, ts[,1], state01, m, l, mu, g);
  real x_sim2[N] = simulate_x(N, ts[,2], state02, m, l, mu, g);
}
```

Additional material - Stan code

```
// The actual model
model {
  // The priors
  m ~ normal(pr_params[1,1], pr_params[1,2]);
  l ~ normal(pr_params[2,1], pr_params[2,2]);
  mu ~ uniform(pr_params[3,1], pr_params[3,2]);
  g ~ normal(pr_params[4,1], pr_params[4,2]);
  state01[1] ~ normal(pr_params[5,1], pr_params[5,2]); // Position of nut 1
  state01[2] ~ normal(pr_params[7,1], pr_params[7,2]); // Velocity of nut 1
  state02[1] ~ normal(pr_params[6,1], pr_params[6,2]); // Position of nut 2
  state02[2] ~ normal(pr_params[7,1], pr_params[7,2]); // Velocity of nut 2
  sigma ~ normal(pr_params[8,1], pr_params[8,2]);

  // The likelihood model
  for (t in 1:N) {
    x[t,1] ~ normal(x_sim1[t], sigma);
    x[t,2] ~ normal(x_sim2[t], sigma);
  }
}
```

Aalto University

Additional material - Stan code

```
// Computing the needed quantities for LOO-CV and other statistics
generated quantities {
  // Posterior draws
  real x_draws1[N] = simulate_x(N, ts[,1], state01, m, l, mu, g);
  real x_draws2[N] = simulate_x(N, ts[,2], state02, m, l, mu, g);

  // Log-likelihoods
  matrix[N,2] log_lik;
  for (t in 1:N){
    log_lik[t,1] = normal_lpdf(x[t,1] | x_sim1[t], sigma);
    log_lik[t,2] = normal_lpdf(x[t,2] | x_sim2[t], sigma);
  }
}
```