# Simulation and Recovery of Ising Sparse Graphical Model

Yilun Cai, Ben Bentley, Bin Gao

December 2025

## 1 Introduction

Graphical models are a family of multivariate distributions which are Markov with respect to a particular undirected graph. A distribution is Markov with respect to a graph if each node is conditionally independent of all others given its neighbors in the graph. This Markov property refers to the structural conditional independence, no interpretation of time is involved. Each node in the graph $i \in V$ is associated with a random variable. The set of edges $E \subset \binom{V}{2}$ encodes the conditional dependency relationships: a variable conditioned on its neighbours is independent of the remaining variables[2]. In this project, we focus on the Ising model, a classical discrete Markov random field in which each variable takes values in $\{+1, -1\}$ and interactions occur only between pairs of variables. A central goal of this project is to recover the underlying interaction matrix $u$ from observed samples.

### 1.1 The Ising Model

Let $y = (y_1, \ldots, y_p) \in \{\pm 1\}^p$ denote a collection of $p$ binary variables. The Ising model is defined through the joint distribution

$$P(y) = \frac{1}{Z} \exp\left( \sum_{\{s,t\} \in E} u_{s,t}\, y_s y_t \right),$$

where, $u \in \mathbb{R}^{p \times p}$ is a symmetric matrix and encodes the graph structure (we set the diagonal elements of $u$ to zero). In particular, $u_{s,t}$ is non-zero if variables $s$ and $t$ are connected via an edge; the sign and magnitude of $u_{st}$ determine whether the variables tend to align or oppose each other. The partition function

$$Z(u) = \sum_{y \in \{\pm 1\}^p} \exp\left( \sum_{s<t} u_{st} y_s y_t \right)$$

serves as a normalizing constant.

Although the Ising model originates from statistical physics, it has become a widely used framework in statistics, biology, and network science for modeling pairwise interactions. Sparse structures—where each node interacts with only a small subset of others—are especially relevant for interpretability and computational tractability.

### 1.2 Sparse Graphical Structures and Motivation

Our project focuses on *sparse* Ising models, where most entries of $u$ are zero. Sparse interaction networks arise naturally in many domains. For example, in a financial system, it is reasonable to assume that each asset interacts with only a small neighborhood of peers rather than the entire market. In our project, each node takes a value $\{+1, -1\}$, representing a simplified binary state (e.g. upward/downward movement), respectively. This binary encoding is not a literal financial quantity but represents simple pairwise interaction terms to simplify analysis.

In particular, we are interested in sparse graph structures, that is, each node is connected to few other nodes in comparison to the total number of nodes. Sparse structures are computationally less taxing than dense structures, allowing the calculation of gradients to involve only the neighbors of each node rather than all $p - 1$ variables. For example, if we were attempting to model the effect of each agent in a financial market adjusting their strategy based on the behaviours of other agents, we might assume said agent can only observe a relatively small group of their peers instead of any sizable portion of the entire collection of agents. In this case it is unreasonable to construct the model with a dense graph structure.

Recovering a sparse interaction matrix is challenging: the exact likelihood depends on the partition function $Z(u)$, which requires summing over $2^p$ configurations and quickly becomes intractable for even moderate values of $p$. The difficulty arises from estimating interactions when most entries of $u$ are zero and the exact likelihood cannot be evaluated for moderate $p$. This motivates approximate inference techniques such as pseudo-likelihood, as well as sparsity-inducing priors (e.g., Laplace) in Bayesian inference.

### 1.3 Objective and Scope

The objective of this project is not to construct a realistic financial model, but rather to use a synthetic Ising model as a controlled environment for evaluating sampling and inference algorithms. Specifically, we aim to:

- Construct a known ground-truth sparse interaction matrix $u^*$.
- Generate samples from the Ising model using Gibbs sampling.
- Recover the interaction matrix using two inference methods:
    1. Exact Bayesian inference via Metropolis–Hastings sampling on the parameter space (tractable only for small $p$).
    2. Pseudo-likelihood MAP estimation, which scales to larger networks and avoids computing the partition function.

- Compare the estimated matrices to $u^*$ to assess:
  - estimation accuracy,
  - convergence and mixing of sampling algorithms,
  - computational feasibility.

Using synthetic data is essential: since $u^*$ is known, we can quantitatively evaluate estimation error, which is a central requirement of scientific computing studies. This would not be possible with real data. Future extensions will examine how factors such as sample size and temperature influence recovery performance.

## 2 Numerical Methods

This section describes the numerical techniques used in our project. Our computational objectives are:

- Generate samples from a known Ising model using Gibbs sampling.
- Recover the interaction matrix $u^*$ using two inference methods:
  1. exact Bayesian inference via Metropolis–Hastings sampling on the parameter space (feasible only for small $p$),
  2. pseudo-likelihood MAP estimation, which scales to larger networks.
- Compare the recovered matrices with the ground-truth parameters.

We begin with notation and then present the sampling and inference procedures used in our experiments.

### 2.1 Notation

We consider $y = (y_1, \dots, y_p) \in \{\pm 1\}^p$ and a symmetric interaction matrix $u \in \mathbb{R}^{p \times p}$ with zero diagonal. The Ising model has probability

$$P(y) = \frac{1}{Z(u)} \exp\left( \sum_{1 \le s < t \le p} u_{st} y_s y_t \right),$$

with partition function

$$Z(u) = \sum_{y \in \{\pm 1\}^p} \exp\left( \sum_{s<t} u_{st} y_s y_t \right).$$

Let $y_{V \setminus r}$ denote all coordinates except $y_r$. Given $N$ samples $\{y^{(i)}\}_{i=1}^N$, our goal is to recover $u^*$.

## 2.2 Gibbs Sampling for Data Generation

Gibbs sampling is used to generate synthetic data from the known model specified by $u^*$. The sampler updates one coordinate at a time according to the conditional distribution.

**Theorem 1.** *The conditional distribution of a variable $y_r$ of node $r$ given the rest $y_{V/r}$ is given by*

$$P(y_r \mid y_{V/r}) \;=\; \frac{\exp\Big(2y_r \sum_{t\in V/r} u_{r,t}y_t\Big)}{\exp\Big(2y_r \sum_{t\in V/r} u_{r,t}y_t\Big)+1}.$$

The full derivation of this conditional probability is provided in Appendix A.

The Theorem 1 indicates that we can use Gibbs sampling to generate samples, as at each sampling step for a given node $r$ we only need to compute a Sigmoid using its neighbouring nodes. With the conditional distribution, Gibbs sampling proceeds as:

**Algorithm 1** Gibbs Sampling

```
1: Initialize $y^{(0)} \sim \pi_0$.
2: for $n = 1$ to $N$ do
3:     for $r = 1$ to $p$ do
4:         Sample $y_r^{(n)} \sim P(y_r \mid y_{V\setminus r}, u^*)$.
5:     end for
6: end for
```

We validate this sampler in a $p = 5$ example, where the empirical correlation matrix agrees with the sparsity structure of $u^*$.

## 2.3 Random Walk Metropolis-Hastings

To recover the interaction matrix $u$ from data, we first consider full Bayesian inference. In this section we perform Metropolis–Hastings sampling directly in the parameter space of $u$ by constructing a random walk on the $p(p-1)/2$-dimensional vector of free interaction parameters. Let $u_{\text{vec}} \in \mathbb{R}^{p(p-1)/2}$ denote the vector of free parameters (upper-triangular entries of $u$), and impose a sparsity-inducing Laplace prior

$$u_j \sim \text{Laplace}(0,\lambda).$$

The posterior distribution is

$$P(u \mid \{y^{(i)}\}) \propto \left[\prod_{i=1}^{N} \frac{\exp\left(\sum_{s<t} u_{st}y_s^{(i)}y_t^{(i)}\right)}{Z(u)}\right] \prod_j \frac{\lambda}{2}e^{-\lambda|u_j|}.$$

## Intractability of Exact Bayesian Inference

Evaluating the Metropolis–Hastings acceptance probability for parameter updates requires the likelihood ratio $Z(u)/Z(u^*)$, where

$$Z(u) = \sum_{y \in \{\pm 1\}^p} \exp\Big(\sum_{s<t} u_{st} y_s y_t\Big).$$

Since computing $Z(u)$ involves summing over $2^p$ configurations, each MH update becomes exponentially expensive in $p$. Thus, exact Bayesian inference is feasible only for small models (e.g. $p \leq 5$). The complete derivation of the MH acceptance ratio and the intractability result is provided in Appendix B.

## Random Walk Metropolis–Hastings Algorithm

We implement MH sampling for the posterior $P(u \mid \{y^{(i)}\})$ in the low-dimensional case $p = 5$, where $Z(u)$ can be computed exactly.

**Algorithm 2** Random Walk Metropolis–Hastings

```
1: Let $P(y)$ denote the target distribution from which we wish to generate approximate samples. Choose an
   initial state $y^{(0)}$ (e.g. by sampling from some easy distribution $\pi_0$ or fixing it arbitrarily).
2: for $n = 1$ to $N$ do
3:     Propose a new state $v^*$ by making a random-walk step:
           $$v^* = y^{(n-1)} + \varepsilon, \qquad \varepsilon \sim q(\cdot),$$
       where $q(\varepsilon)$ is symmetric around 0 (e.g. $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$).
4:     Compute the acceptance probability
           $$\alpha\big(y^{(n-1)}, v^*\big) = \min\left\{1,\ \frac{P(v^*)}{P(y^{(n-1)})}\right\},$$
       which simplifies due to symmetry:
           $$q(v^* \mid y^{(n-1)}) = q(y^{(n-1)} \mid v^*).$$
5:     Draw $u \sim \text{Uniform}(0, 1)$.
6:     if $u \leq \alpha\big(y^{(n-1)}, v^*\big)$ then
7:         Accept the proposal: $y^{(n)} = v^*$.
8:     else
9:         Reject the proposal: $y^{(n)} = y^{(n-1)}$.
10:    end if
11: end for
```

For $p = 5$, the sampler mixes well and posterior means closely match the true parameters $u^*$. For larger dimensions, however, the exponential cost of computing $Z(u)$ motivates the use of the pseudo-likelihood MAP estimator introduced in Section 2.4.

## 2.4 Pseudo Log-Likelihood MAP Estimation

We will now address the intractability problem: Random Walk Metropolis-Hastings becomes infeasible as the dimension $p$ grows, since evaluating the partition function $Z(u)$ requires summing over $2^p$ states. To scale parameter estimation to larger networks, we adopt the *pseudo-likelihood* approximation introduced by Besag [1].

**Theorem 2.** *Suppose that the likelihood $P(y_1, y_2, \dots, y_p \mid u)$ is well approximated by*

$$P(y_1, y_2, \dots, y_p \mid u) \;\approx\; \prod_{r=1}^{p} P(y_r \mid y_{V/r}, u),$$

*where the conditional probabilities are given explicitly by Theorem 1.*

*The MAP estimator of $u$ is given by*

$$\arg \min_{\substack{u \\ u=u^\top \\ \mathrm{diag}(u)=0}} \left\{ \sum_{i=1}^{N} \sum_{r=1}^{p} -\log\Big(P\big(y_r^{(i)} \mid y_{V/r}^{(i)}, u\big)\Big) \;+\; \frac{1}{\lambda} \sum_{i<j} |u_{ij}| \right\}.$$

A full derivation of this objective is provided in Appendix C.

## 2.5 Proximal Gradient Descent (PGD)

The objective consists of (i) a smooth and convex pseudo-likelihood term, and (ii) a non-smooth $\ell_1$ penalty. This suggests the use of *Proximal Gradient Descent (PGD)*, where each iteration alternates between a gradient step on the smooth term and a soft-thresholding update on the $\ell_1$ term.

Algorithm 3: Proximal Gradient Descent for MAP Estimation

```
Require: Dataset $\{y^{(i)}\}_{i=1}^{N}$ where $y^{(i)} \in \{\pm 1\}^p$
Require: Regularization parameter $\lambda > 0$
Require: Step size $\eta > 0$ (learning rate)
Require: Maximum iterations $M$, convergence tolerance $\epsilon$
Ensure: Estimated interaction matrix $u \in \mathbb{R}^{p \times p}$ (symmetric with $\mathrm{diag}(u) = 0$)
1: Initialize:
2: $u^{(0)} \leftarrow$ random symmetric matrix with zero diagonal and small entries
3: $u^{(0)} \leftarrow (u^{(0)} + u^{(0)\top})/2$                    ▷ Ensure symmetry
4: for $n = 1$ to $M$ do
5:     1. Compute gradient of negative pseudo log-likelihood:
 6:     Initialize $\nabla f(u^{(n-1)}) \leftarrow \mathbf{0}_{p\times p}$
 7:     **for** $i = 1$ **to** $N$ **do**                                      $\triangleright$ Loop over all samples
 8:         **for** $r = 1$ **to** $p$ **do**                                  $\triangleright$ Loop over all variables
 9:             Compute $A_r = \sum_{t\in V\setminus r} u_{rt}^{(n-1)} y_t^{(i)}$
10:             Compute $\sigma = \frac{\exp(2A_r)}{\exp(2A_r)+1}$            $\triangleright$ Conditional probability $P(y_r = 1|y_{V\setminus r})$
11:             **for** $t = 1$ **to** $p$, $t \neq r$ **do**
12:                 $\nabla_{rt} \leftarrow \nabla_{rt} - 2y_r^{(i)} y_t^{(i)}(1-\sigma)$    $\triangleright$ Gradient contribution from $\mathbb{P}(y_r^{(i)}|y_{V\setminus r}^{(i)}, u)$
13:             **end for**
14:         **end for**
15:     **end for**
16:     $\nabla f(u^{(n-1)}) \leftarrow \nabla f(u^{(n-1)})/N$              $\triangleright$ Average over samples
17:     $\nabla f(u^{(n-1)}) \leftarrow (\nabla f(u^{(n-1)}) + \nabla f(u^{(n-1)})^\top)/2$    $\triangleright$ Enforce symmetry
18:     $\text{diag}(\nabla f(u^{(n-1)})) \leftarrow 0$                    $\triangleright$ Zero diagonal
19:     **2. Gradient descent step for smooth part:**
20:     $u_{\text{temp}} \leftarrow u^{(n-1)} - \eta \nabla f(u^{(n-1)})$
21:     **3. Proximal operator for L1 regularization:**
22:     $\tau \leftarrow \eta/\lambda$                                        $\triangleright$ Soft-thresholding parameter
23:     **for** $i = 1$ **to** $p$ **do**
24:         **for** $j = i+1$ **to** $p$ **do**                              $\triangleright$ Only upper triangular (off-diagonal)
25:             $u_{ij}^{(n)} \leftarrow \text{sign}(u_{\text{temp},ij}) \cdot \max(|u_{\text{temp},ij}| - \tau, 0)$
26:             $u_{ji}^{(n)} \leftarrow u_{ij}^{(n)}$                        $\triangleright$ Maintain symmetry
27:         **end for**
28:     **end for**
29:     $\text{diag}(u^{(n)}) \leftarrow 0$                                   $\triangleright$ Zero diagonal constraint
30:     **4. Check convergence:**
31:     $\Delta u \leftarrow \|u^{(n)} - u^{(n-1)}\|_F$                      $\triangleright$ Frobenius norm
32:     **if** $\Delta u < \epsilon$ **and** $n > 10$ **then**
33:         **break**
34:     **end if**
35: **end for**
36: **return** $u^{(n)}$                                                    $\triangleright$ Final MAP estimate of interaction matrix
```

The MAP estimator is defined as:

$$u_{\text{MAP}} = \arg\min_u \left[ \sum_{i=1}^{N} \sum_{r=1}^{p} -\log \mathbb{P}(y_r^{(i)} \mid y_{V/r}^{(i)}, u) + \frac{1}{\lambda} \sum_{i<j} |u_{ij}| \right]$$

Where the conditional distribution for the Ising model is:

$$\mathbb{P}(y_r \mid y_{V/r}, u) = \frac{\exp(2y_r \sum_{t \in V/r} u_{rt} y_t)}{\exp(2y_r \sum_{t \in V/r} u_{rt} y_t) + 1}$$

**Algorithm Details:**

- **Objective decomposition:**

$$J(u) = \underbrace{\sum_{i=1}^{N} \sum_{r=1}^{p} -\log \mathbb{P}(y_r^{(i)} \mid y_{V/r}^{(i)}, u)}_{f(u)\text{ (smooth)}} + \underbrace{\frac{1}{\lambda} \sum_{i<j} |u_{ij}|}_{g(u)\text{ (non-smooth)}}$$

- **Gradient computation:**

$$\frac{\partial}{\partial u_{rt}} \left[ -\log \mathbb{P}(y_r \mid y_{V/r}, u) \right] = -2y_r y_t \left[ 1 - \sigma\left( 2y_r \sum_{k \in V/r} u_{rk} y_k \right) \right]$$

  where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function.

- **Proximal operator for L1 regularization:**

$$\text{prox}_{\eta g}(x) = \text{sign}(x) \cdot \max(|x| - \eta/\lambda, 0)$$

- **Complete update:**

$$u^{(n)} = \text{prox}_{\eta g}\left( u^{(n-1)} - \eta \nabla f(u^{(n-1)}) \right)$$

## 3 Package Structure and Main Functions

### 3.1 Package Overview

Our project is implemented as a modular Python package organized under the `src/` directory, with a clear separation between model definition, sampling, inference, utilities, and experiment orchestration. The design follows standard scientific computing principles: each module exposes a small and well-defined public API, enabling reproducibility, extensibility, and independent testing.

At the top level, lightweight driver scripts (`main.py`, `main_parallel.py`) serve as entry points for running end-to-end experiments, while all computational logic resides inside the `src/` package. This separation ensures that numerical methods are implemented independently of experiment configuration and benchmarking logic.

## 3.2 Module Responsibilities

The `src/` package is structured as follows:

- `ising_model.py`

  Defines the Ising model, including the energy function, probability mass function, and exact partition function for small $p$. This module provides core computations that are shared across sampling and inference methods.

- `sampling.py`

  Implements Gibbs sampling for generating data from the known ground-truth interaction matrix $u^*$, as well as (optionally) Metropolis–Hastings sampling on the state space. The Gibbs sampler uses the conditional probability derived in Appendix A.

- `inference.py`

  Contains two inference methods for recovering the interaction matrix $u$: (i) the Random Walk Metropolis–Hastings algorithm for exact Bayesian inference (used only for small $p$), and (ii) the pseudo-likelihood MAP estimator, implemented using proximal gradient descent. This module encapsulates all optimization and posterior sampling logic.

- `utils.py`

  Provides auxiliary utilities, including matrix vectorization and unvectorization, soft-thresholding operators, correlation computations, plotting helpers, and I/O functions.

- `experiments.py`

  Contains reproducible scripts used to run the experiments in Section 5. These scripts call the sampling and inference routines from the modules above, generate figures, and compute error metrics for comparison.

This modular design allows each component to be developed, optimized, and tested independently.

## 3.3 Main Functions and Public API

The core functionality of the package is exposed through a small number of high-level experiment drivers defined in `src.experiments`. These functions encapsulate the full workflow from data generation to parameter recovery.

| Function | Input | Output | Description |
|---|---|---|---|
| `run_mh_parameter_inference` | `random_state`, `u_txt_path` | $(u^*, \hat{u}, \text{samples})$ | Runs Gibbs sampling to generate data and performs Random Walk Metropolis–Hastings inference on the interaction matrix $u$. |
| `run_mh_parameter_inference_parallel` | `random_state`, `u_txt_path` | $(u^*, \hat{u}, \text{samples})$ | Parallelized version of MH inference using Numba, accelerating likelihood and partition function evaluations. |
| `run_pgd_parameter_inference` | `random_state`, `u_txt_path` | $(u^*, \hat{u})$ | Performs pseudo-likelihood MAP estimation using Proximal Gradient Descent (PGD) for sparse interaction recovery. |

Table 1: Main public API functions for parameter inference experiments.

The ground-truth interaction matrix $u^*$ is specified externally via a text file, which decouples model specification from code and enables rapid experimentation with different graph structures.

## 3.4 Typical Experimental Workflow

A typical experiment using this package proceeds as follows:

1. **Load ground-truth model**: A sparse interaction matrix $u^*$ is read from a text file (e.g. `examples/u_example_8x8.txt`).
2. **Generate synthetic data**: Gibbs sampling is used to draw samples from the Ising model $P(y \mid u^*)$.
3. **Parameter inference**:
   - For small systems, Random Walk Metropolis–Hastings is applied to sample from the posterior $P(u \mid \{y^{(i)}\})$;
   - For larger systems, pseudo-likelihood MAP estimation via PGD is used instead.

4. **Evaluation**: The estimated interaction matrix $\hat{u}$ is compared against $u^*$ using Frobenius norm errors and structural diagnostics.

This workflow is implemented concisely in the main driver scripts. In particular, `main.py` demonstrates sequential MH and PGD inference on a given example matrix, while `main_parallel.py` compares the runtime of the original and parallelized MH implementations.

### 3.5 Reproducibility and Extensibility

All high-level experiment functions accept an explicit `random_state` argument to ensure reproducibility. New inference methods or sampling schemes can be incorporated by adding modules under `src/` and exposing corresponding experiment drivers in `experiments.py`, without modifying existing scripts.
This structure enables systematic scientific investigation of accuracy, scalability, and computational efficiency for Ising model inference algorithms.

## 4 Tests

All core functionalities of the project are covered by unit tests located in the `test/` directory. Our testing strategy follows the principle that every function in the `src/` package should have at least one corresponding test, as required in the checkpoint guidelines. Below we summarize the key categories of tests.

### Model and Utility Tests

We verify the correctness of basic operations that the rest of the package relies on:

- **Vectorization and unvectorization**: Tests ensure that converting $u$ to a parameter vector and back returns the original symmetric matrix with zero diagonal.
- **Partition function (small $p$)**: For $p \leq 5$, where it is feasible to enumerate all $2^p$ states, we test that the exact partition function agrees with brute-force computation.
- **Soft-thresholding operator**: Validates correctness of the proximal update used in pseudo-likelihood MAP estimation.

### Sampling Tests

We test the sampling routines using low-dimensional models where the true distribution can be computed exactly.

- **Gibbs sampling**: For $p = 2$ and $p = 3$, empirical marginal probabilities and correlations obtained from Gibbs samples are compared against exact values computed analytically. This checks that the sampler converges to the correct stationary distribution.

### Inference Tests

Inference routines are validated using synthetic models with known ground-truth interaction matrices $u^*$.

- **MH on parameter space**: For $p = 5$, posterior means obtained from the MH sampler are checked to be close to $u^*$ when run with sufficient iterations. We also verify numerically that the acceptance ratio matches the theoretical expression derived in Appendix B.
- **Pseudo-likelihood MAP)**: Tests confirm that the objective function decreases monotonically along PGD iterations and that the output matrix is symmetric with zero diagonal. For small $p$, the estimator is compared against exact posterior means to ensure consistency.

### Testing Philosophy

We validated each implemented function to ensure correctness and reproducibility. These tests also provide a reliable foundation for the extended experiments and variations that will be developed in the final project.

## 5 Results

In this section we report experiments evaluating the sampling and inference methods introduced in Section 2. All experiments use synthetic data generated from a known ground-truth interaction matrix $u^*$, allowing direct comparison between estimates and the true parameters. The goal of this checkpoint is to demonstrate that the basic pipeline functions correctly and to provide early insight into the behavior of the methods.

### 5.1 Gibbs Sampling: Example 1

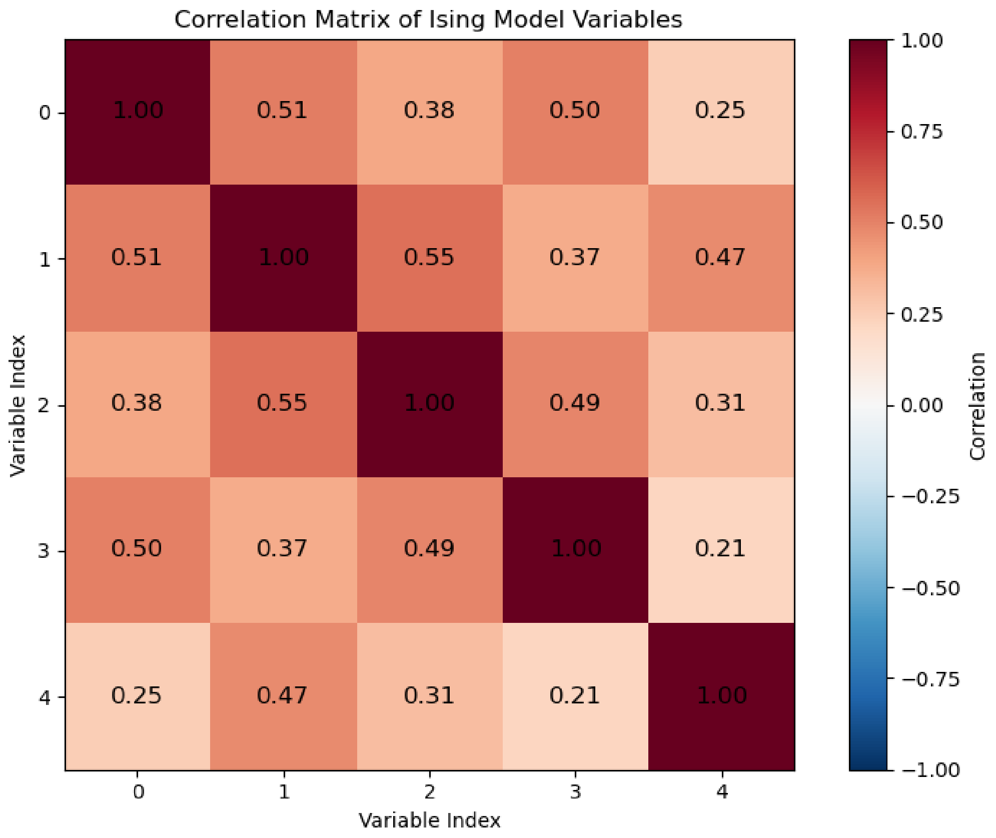
We consider a low dimension example to verify the use of Gibbs. Consider a collection of discrete variables $y \in \mathbb{R}^5$ specified by the following $u^\star \in \mathbb{R}^{5\times 5}$:

$$u^\star = \begin{pmatrix} 0 & 0.5 & 0 & 0.5 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0 & 0.5 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \end{pmatrix}.$$

Using a Gibbs sampler with initialization

$$y^{(0)} = (1, -1, -1, 1, 1)^\top$$

and burn-in period of 5000 samples. We draw $N = 1000$ samples from the joint distribution. The resulting Correlation Matrix of the samples are as follows:



|  | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 1.00 | 0.51 | 0.38 | 0.50 | 0.25 |
| 1 | 0.51 | 1.00 | 0.55 | 0.37 | 0.47 |
| 2 | 0.38 | 0.55 | 1.00 | 0.49 | 0.31 |
| 3 | 0.50 | 0.37 | 0.49 | 1.00 | 0.21 |
| 4 | 0.25 | 0.47 | 0.31 | 0.21 | 1.00 |

Figure 1: Example 1 Gibbs Sampling Correlation Matrix

This verifies our sampling method, as we see that the entries in the correlation matrix are close to 0.5 on the edges and much smaller elsewhere except for between nodes 0 and 2 / 1 and 3. This is an expected secondary effect of the node connections, as nodes 0 and 2 / 1 and 3 are connected by nodes 0 and 2 / 1 and 3. These results confirm that the Gibbs sampler correctly captures the dependency structure encoded by $u^*$ and produces samples suitable for parameter recovery.

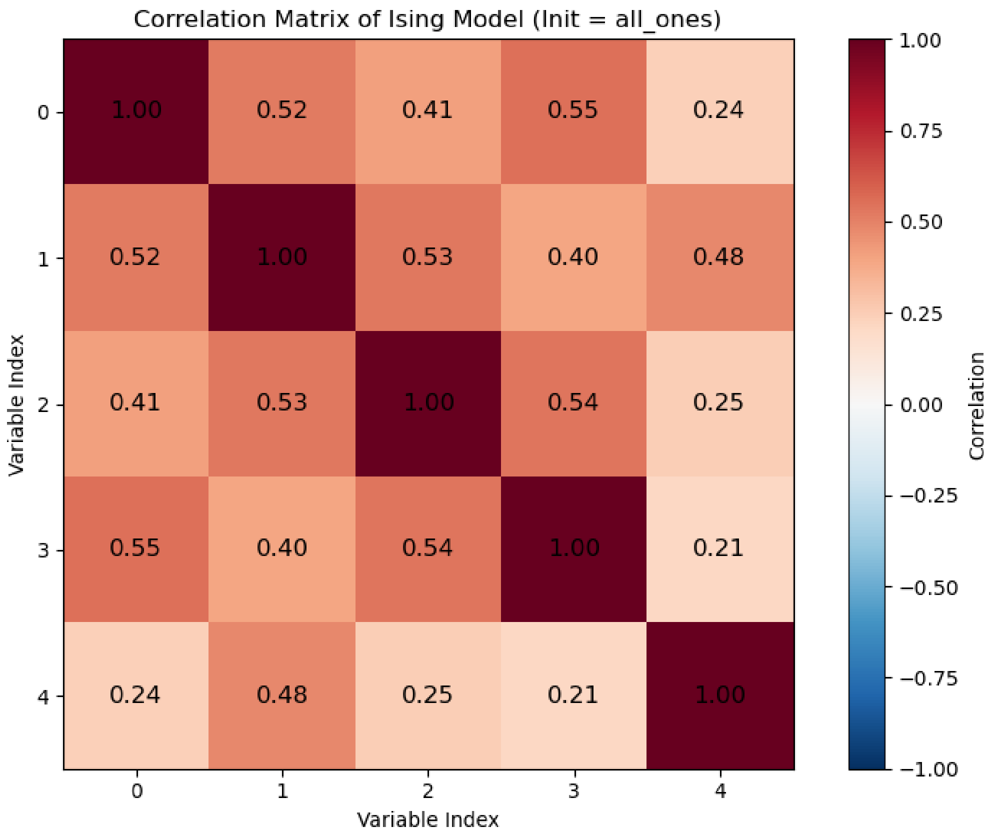
## 5.2 Gibbs Sampling: Example 2 and 3

In Examples 2 and 3, we vary the initial value of $y^{(0)}$ to study the impact on the Gibbs sampler. We hold the value of $u^*$ steady as specified in Example 1.

Using a Gibbs sampler with initialization

$$y^{(0)} = (1, 1, 1, 1, 1)^\top$$

and burn-in period of 5000 samples. We draw $N = 1000$ samples from the joint distribution. The resulting Correlation Matrix of the samples are as follows:



|  | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 1.00 | 0.52 | 0.41 | 0.55 | 0.24 |
| 1 | 0.52 | 1.00 | 0.53 | 0.40 | 0.48 |
| 2 | 0.41 | 0.53 | 1.00 | 0.54 | 0.25 |
| 3 | 0.55 | 0.40 | 0.54 | 1.00 | 0.21 |
| 4 | 0.24 | 0.48 | 0.25 | 0.21 | 1.00 |

Figure 2: Example 2 Gibbs Sampling Correlation Matrix

As in Example 2, we vary the initial value of $y^{(0)}$ to study the impact on the Gibbs sampler. We hold the value of $u^*$ steady as specified in Example 1.

Using a Gibbs sampler with initialization

$$y^{(0)} = (-1, -1, -1, -1, -1)^\top$$

and burn-in period of 5000 samples. We draw $N = 1000$ samples from the joint distribution. The resulting Correlation Matrix of the samples are as follows:

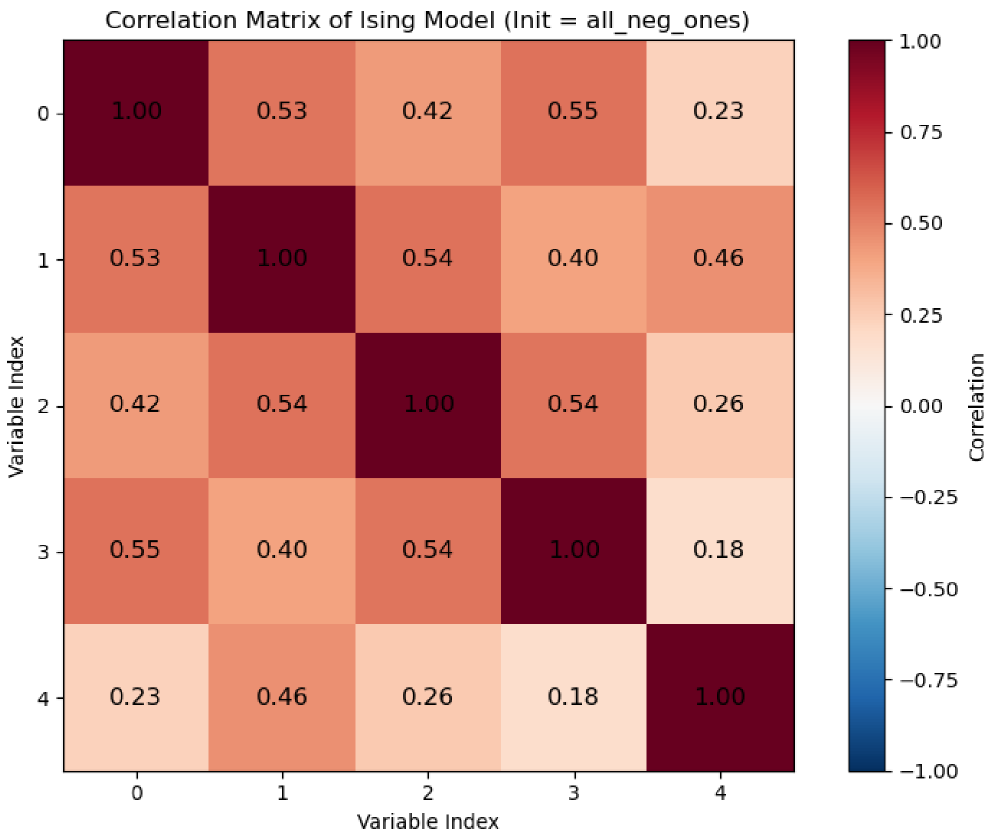


Figure 3: Example 3 Gibbs Sampling Correlation Matrix

All three runs produce very similar correlation matrices. This indicates that the Gibbs sampler converges to the same distribution regardless of its starting point given a sufficient burn-in period and the samples are not biased by initial starting points.

## 5.3 Gibbs Sampling: Example 4 and 5

In Examples 4 and 5, we vary the value of $u^*$ to verify if the Gibbs sampler correctly captures the dependency structure for different $u^*$ structures. We look at an example of a sparse matrix and an example of a dense matrix. We hold the value of $y^{(0)}$ steady as specified in Example 1.

$$u^\star(sparse) = \begin{pmatrix} 0 & 0.9 & 0 & 0 & 0 \\ 0.9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Using a Gibbs sampler with initialization

$$y^{(0)} = (1, -1, -1, 1, 1)^\top$$

and burn-in period of 5000 samples. We draw $N = 1000$ samples from the joint distribution. The resulting Correlation Matrix of the samples are as follows:

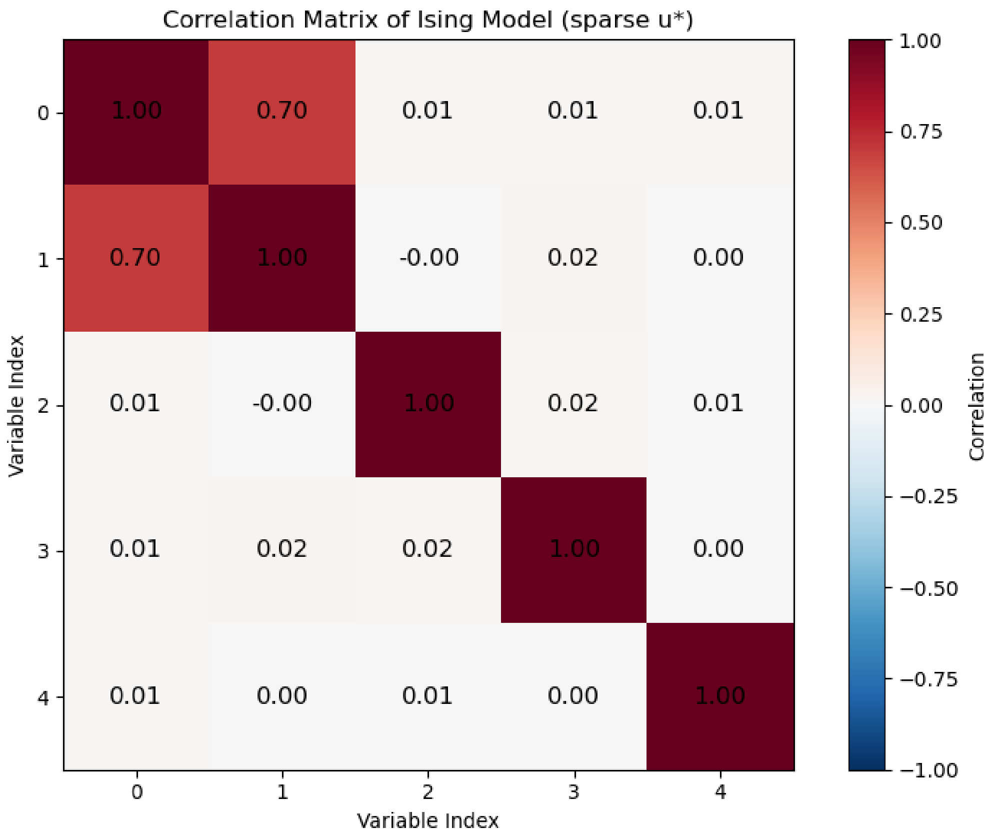


Figure 4: Example 4 Gibbs Sampling Correlation Matrix

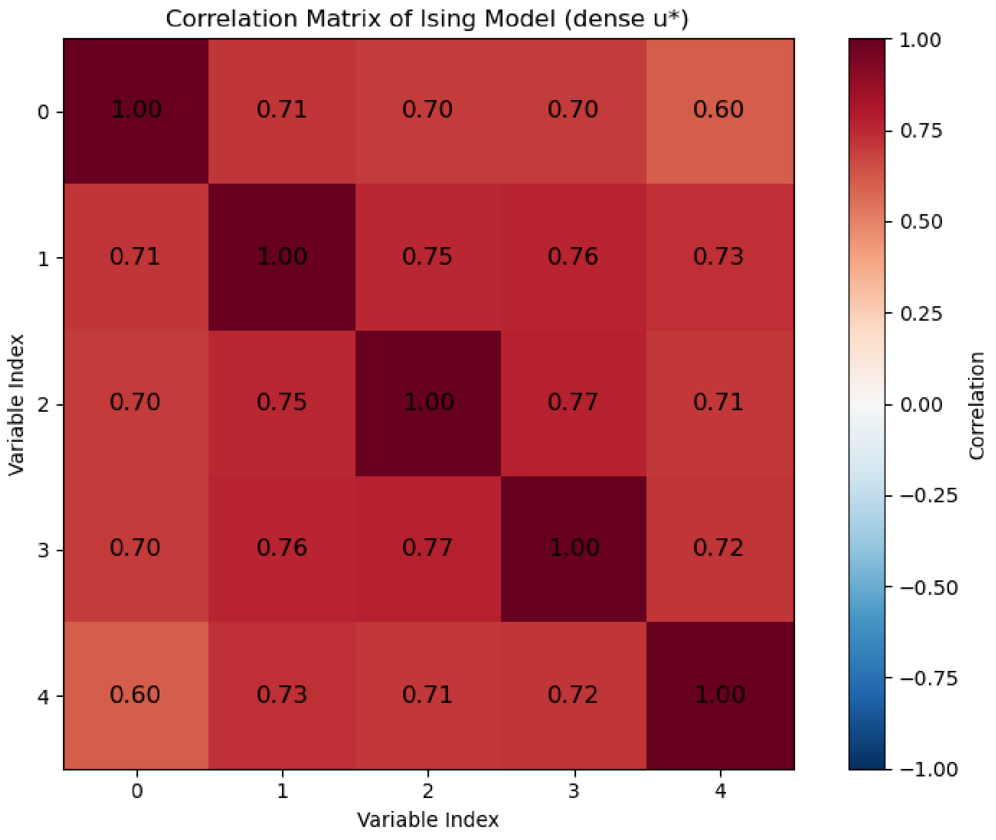
As in Example 4, we vary the value of $u^*$ to verify if the Gibbs sampler correctly captures the dependency structure for different $u^*$ structures. We hold the value of $y^{(0)}$ steady as specified in Example 1.

$$u^\star(dense) = \begin{pmatrix} 0 & 0.4 & 0.4 & 0.4 & 0 \\ 0.4 & 0 & 0.4 & 0.4 & 0.4 \\ 0.4 & 0.4 & 0 & 0.4 & 0.4 \\ 0.4 & 0.4 & 0.4 & 0 & 0.4 \\ 0 & 0.4 & 0.4 & 0.4 & 0 \end{pmatrix}.$$

Using a Gibbs sampler with initialization

$$y^{(0)} = (1, -1, -1, 1, 1)^\top$$

and burn-in period of 5000 samples. We draw $N = 1000$ samples from the joint distribution. The resulting Correlation Matrix of the samples are as follows:



| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 1.00 | 0.71 | 0.70 | 0.70 | 0.60 |
| 1 | 0.71 | 1.00 | 0.75 | 0.76 | 0.73 |
| 2 | 0.70 | 0.75 | 1.00 | 0.77 | 0.71 |
| 3 | 0.70 | 0.76 | 0.77 | 1.00 | 0.72 |
| 4 | 0.60 | 0.73 | 0.71 | 0.72 | 1.00 |

Figure 5: Example 5 Gibbs Sampling Correlation Matrix

The new correlation matrices reflected the structures of the corresponding $u^*$ matrices very closely. This further verifies that the Gibbs sampler is correctly capturing the dependency patterns and is suitable to produce samples for parameter recovery. In the Ising model, the correlation matrix is determined by the interaction parameters $u_{ij}$, so agreement between correlation matrices provides a direct diagnostic of Gibbs sampling accuracy.

## 5.4 Gibbs Error Analysis and Dependence on $u^*$

To more explicitly evaluate the accuracy of the Gibbs sampler, we compare the correlation matrix obtained from a short run (2000 samples after 5000 burn-in) with a long, well-mixed reference run (200000 samples after 50000 burn-in). Denoting these by $C_{\text{hat}}$ and $C_{\text{true}}$, respectively, we measure the Gibbs error using the Frobenius norm

$$Err_{\text{Gibbs}} = \|C_{\text{hat}} - C_{\text{true}}\|_F.$$

Two observations arise from Examples 4 and 5. First, the sparse model contains a *single very strong interaction* ($u_{12} = 0.9$), which slows Gibbs mixing considerably and yields a slightly larger $Err_{\text{Gibbs}}$. Second, the dense model contains many *moderate* interactions ($u_{ij} = 0.4$), which in this case results in faster mixing and slightly smaller error. This illustrates that Gibbs performance depends primarily on the *magnitude* of couplings rather than sparsity alone.

For the sparse model we obtained

$$Err_{\text{Gibbs}} = 0.0690, Err_u = 1.4789$$

For the dense model we obtained

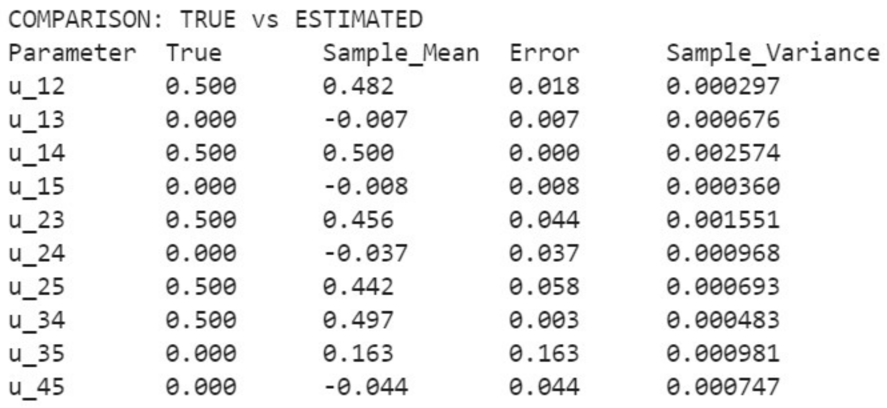
$$Err_{\text{Gibbs}} = 0.0671, Err_u = 1.1597$$

These experiments therefore confirm that the sampler accurately captures $P(y \mid u^*)$ when mixing is adequate, and they provide a quantitative link between the interaction structure and Gibbs error.

Here, $Err_u = \|\hat{u} - u^*\|_F$ measures the error in recovering the interaction matrix from data generated under $u^*$. Unlike $Err_{\text{Gibbs}}$, which reflects Monte Carlo error in approximating $P(y \mid u^*)$, $Err_u$ aggregates several sources of uncertainty, including finite sample effects, posterior variability, and noise in likelihood evaluation. Consequently, $Err_u$ is substantially larger than $Err_{\text{Gibbs}}$ in all experiments, even when the Gibbs sampler mixes adequately.

## 5.5 Random Walk MH: Example 1 (Small $p$)

Next, we perform Random Walk Metropolis–Hastings sampler described in Section 2.3 using the low dimensional matrix from example 1. Note that, unlike the previous subsections where $u^*$ is fixed and we assess the Gibbs sampler for $P(y \mid u^*)$, we now treat $u$ as unknown and evaluate our ability to recover $u^*$ from Gibbs-generated samples. Thus the accuracy here reflects both the posterior geometry and the Gibbs error from the inner loop. We use $\sigma^2 = 0.1$ with $\lambda = 0.2$, a burn-in time of 10000 samples, use Metropolis-Hastings to generate $\bar{N} = 1000$ samples from the posterior $P(u \mid \{y^{(i)}\}_{i=1}^{N})$. Here we will simply compute the normalizing constant with brute force. The results are as follows



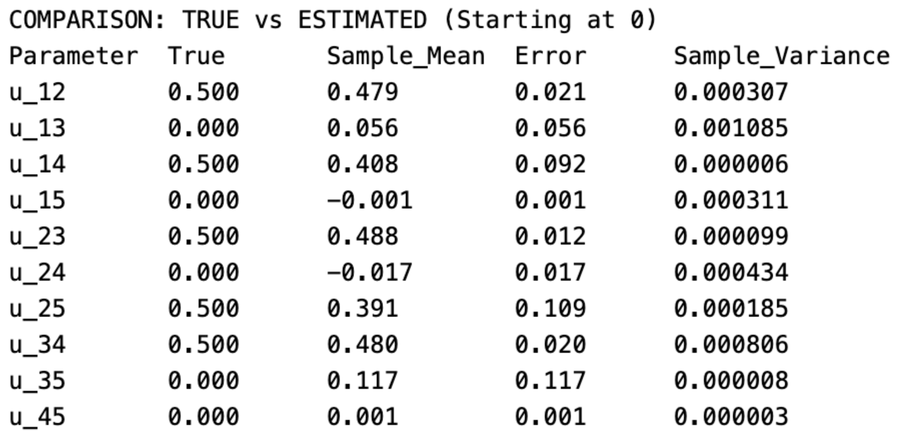
```
COMPARISON: TRUE vs ESTIMATED
Parameter  True      Sample_Mean  Error     Sample_Variance
u_12       0.500     0.482        0.018     0.000297
u_13       0.000     -0.007       0.007     0.000676
u_14       0.500     0.500        0.000     0.002574
u_15       0.000     -0.008       0.008     0.000360
u_23       0.500     0.456        0.044     0.001551
u_24       0.000     -0.037       0.037     0.000968
u_25       0.500     0.442        0.058     0.000693
u_34       0.500     0.497        0.003     0.000483
u_35       0.000     0.163        0.163     0.000981
u_45       0.000     -0.044       0.044     0.000747
```

Figure 6: Example 1 Random Walk Metropolis Hastings Results

We can see that the algorithm works quite well, as both the errors and the sample variances were small.

## 5.6 Random Walk MH: Examples 2 and 3

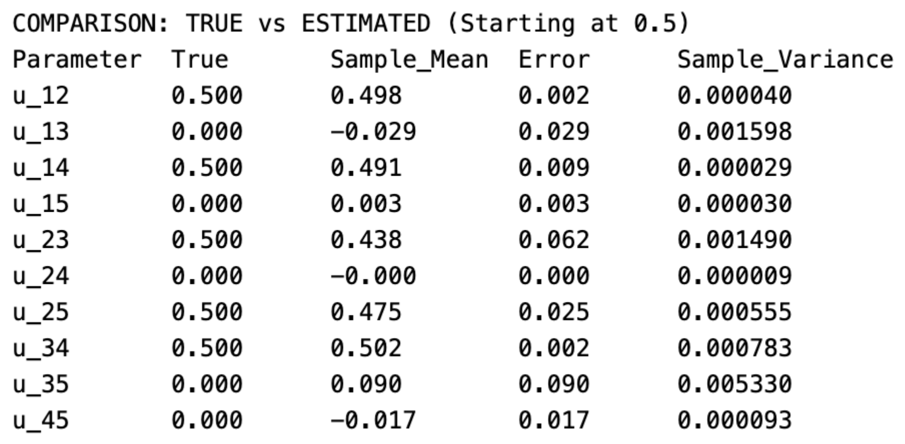
Next, we perform Random Walk Metropolis–Hastings sampler on two different initial values. In Example 1, the initial value was 0.1. In Example 2, we will start at 0 which is farther from the true value of 0.5. In Example 3, we will start at 0.5. The results for Example 2 are as follows:



```
COMPARISON: TRUE vs ESTIMATED (Starting at 0)
Parameter  True      Sample_Mean  Error     Sample_Variance
u_12       0.500     0.479        0.021     0.000307
u_13       0.000     0.056        0.056     0.001085
u_14       0.500     0.408        0.092     0.000006
u_15       0.000     −0.001       0.001     0.000311
u_23       0.500     0.488        0.012     0.000099
u_24       0.000     −0.017       0.017     0.000434
u_25       0.500     0.391        0.109     0.000185
u_34       0.500     0.480        0.020     0.000806
u_35       0.000     0.117        0.117     0.000008
u_45       0.000     0.001        0.001     0.000003
```

Figure 7: Example 2 Random Walk Metropolis Hastings Results Starting at 0

The results for Example 3 are as follows:



```
COMPARISON: TRUE vs ESTIMATED (Starting at 0.5)
Parameter  True       Sample_Mean  Error      Sample_Variance
u_12       0.500      0.498        0.002      0.000040
u_13       0.000      -0.029       0.029      0.001598
u_14       0.500      0.491        0.009      0.000029
u_15       0.000      0.003        0.003      0.000030
u_23       0.500      0.438        0.062      0.001490
u_24       0.000      -0.000       0.000      0.000009
u_25       0.500      0.475        0.025      0.000555
u_34       0.500      0.502        0.002      0.000783
u_35       0.000      0.090        0.090      0.005330
u_45       0.000      -0.017       0.017      0.000093
```

Figure 8: Example 3 Random Walk Metropolis Hastings Results Starting at 0.5

While the errors and sample variances for Example 2 are higher than Example 3, both initial values yielded fairly similar results. This indicates that the Random Walk Metropolis-Hastings sampler is robust and performs well regardless of initial vlaue.

## 5.7 Pseudo-likelihood MAP Estimation and PGD: Example 1

For our first test we continue to use the low dimensional matrix from example 1.

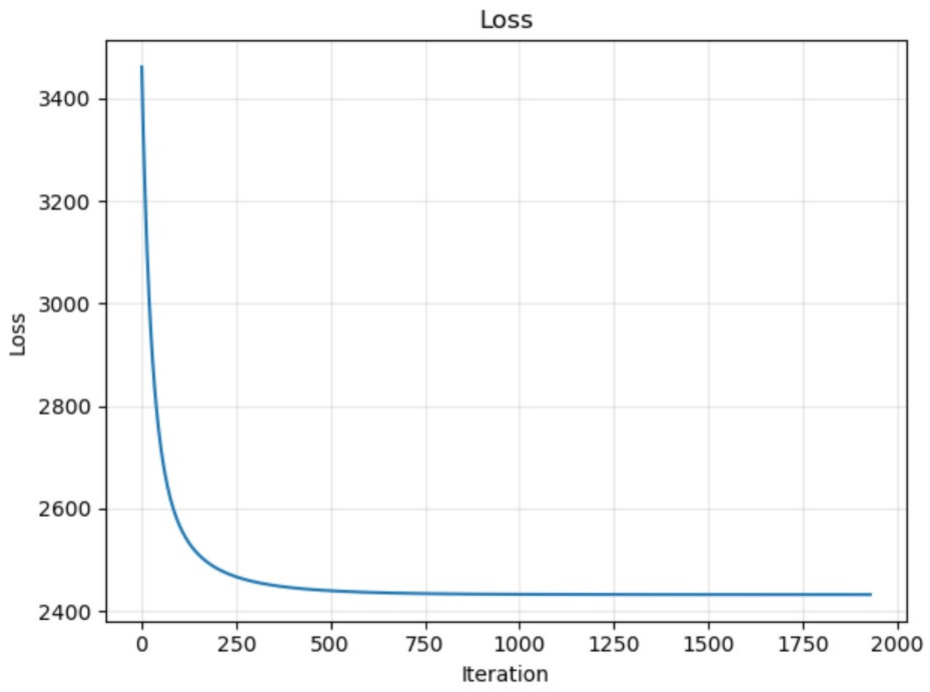
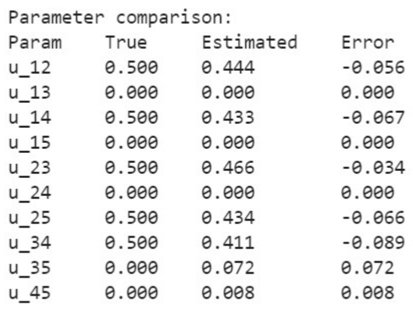


Figure 9: Example 1 PGD Loss



```
Parameter comparison:
Param    True     Estimated    Error
u_12     0.500    0.444        -0.056
u_13     0.000    0.000        0.000
u_14     0.500    0.433        -0.067
u_15     0.000    0.000        0.000
u_23     0.500    0.466        -0.034
u_24     0.000    0.000        0.000
u_25     0.500    0.434        -0.066
u_34     0.500    0.411        -0.089
u_35     0.000    0.072        0.072
u_45     0.000    0.008        0.008
```

Figure 10: Example 1 PGD Parameter Estimations and Errors

Here we used a step size of 0.01 with $\lambda = 50$ and a very low tolerance of $10^{-6}$ for show casing the method. We can see that we already have a pretty good convergence after around 500 iterations, and the parameter estimations are fairly accurate.

## 5.8 Pseudo-likelihood MAP Estimation and PGD: Example 2

For our second test, we increase the dimension of the interaction matrix to $p = 10$ and consider a structured but still sparse matrix with both banded and block-style interactions.

$$u^\star = \begin{bmatrix} 0 & 0.4 & 0.3 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.4 & 0 & 0.4 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.3 & 0.4 & 0 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.3 & 0.3 & 0.4 & 0 & 0.4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.4 & 0 & 0.4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.4 & 0 & 0.4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.4 & 0 & 0.4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.4 & 0 & 0.4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.4 & 0 \end{bmatrix}.$$

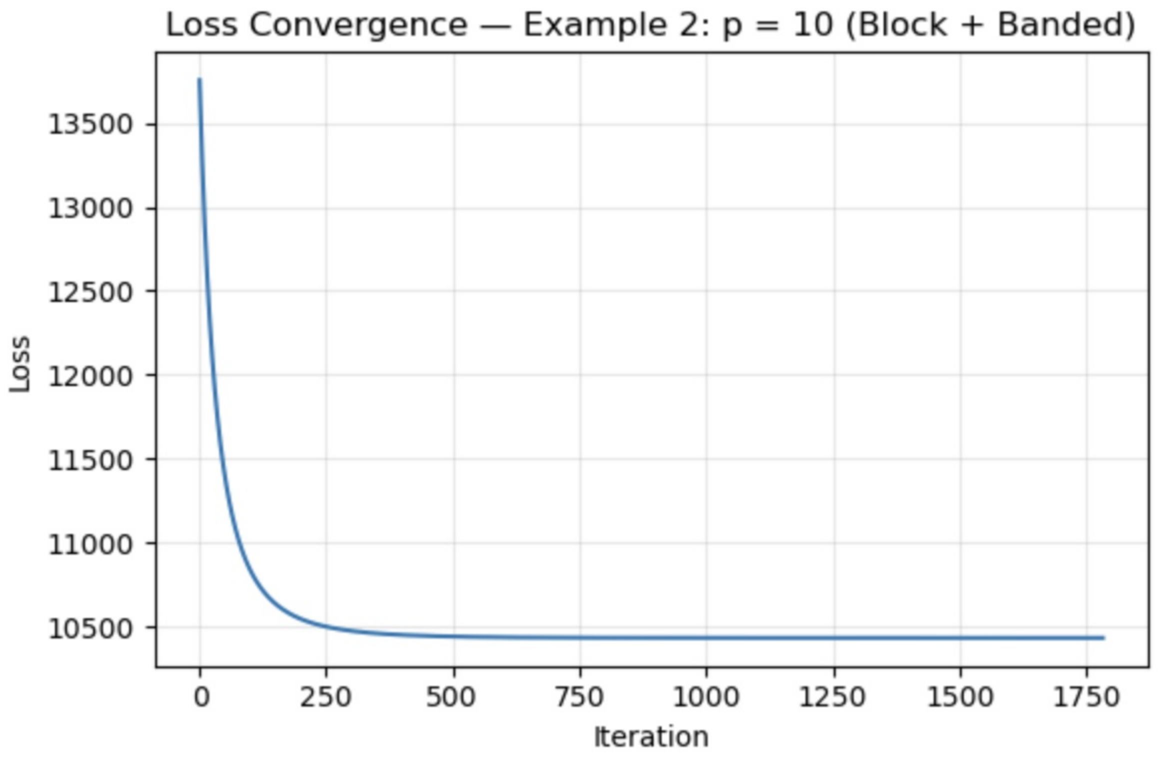


Figure 11: Example 2 PGD Loss

In this experiment, we used a step size of 0.01 with $\lambda = 30$ and a tolerance of $10^{-6}$. Despite the increased dimensionality, the loss converges rapidly and stabilizes after a few thousand iterations. The recovered interaction matrix closely matches the true structure, with the strongest interactions correctly identified and only small estimation errors. This demonstrates that the PGD method scales well to moderately larger systems when the underlying interaction structure remains sparse.

## 5.9 Pseudo-likelihood MAP Estimation and PGD: Example 3

For our third test, we further increase the problem size to $p = 15$ and consider a denser interaction matrix with weaker individual interaction strengths. This example is designed to test the robustness of the PGD method in a more challenging setting where many parameters must be estimated simultaneously. Due to the size of the interaction matrix in this example ($p = 15$), we do not display the full matrix $u^\star$ explicitly.

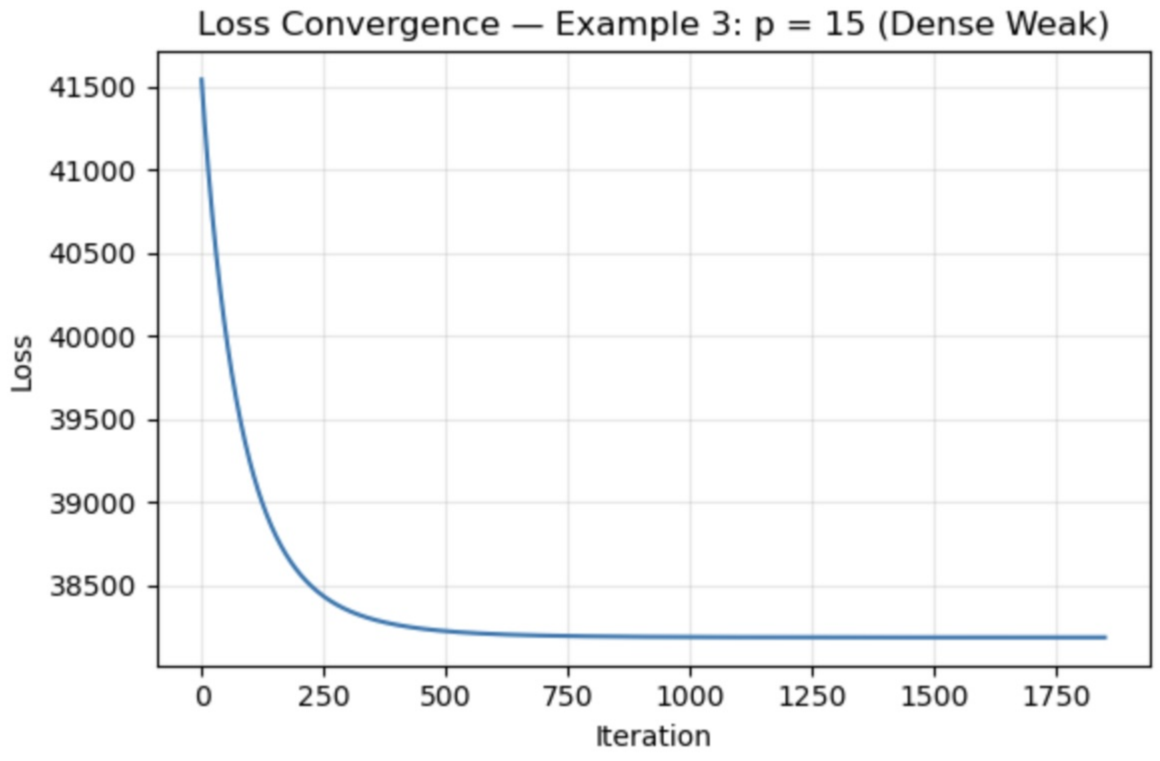


Figure 12: Example 3 PGD Loss

Here we reduced the step size to 0.005 and increased the regularization parameter to $\lambda = 60$ to account for the denser parameter space. Although convergence is slower compared to the lower-dimensional cases, the loss still decreases smoothly and stabilizes within the allotted number of iterations. The estimated interaction matrix captures the overall structure and relative magnitudes of the true interactions, though with slightly larger errors, as expected in a higher-dimensional and more weakly identifiable setting.

## 6 Computational Efficiency Improvements

A central component of our project is the numerical evaluation of the Ising likelihood and the sampling procedures used in both the Gibbs sampler and the Metropolis–Hastings (MH) parameter inference scheme. We addressed this by introducing implementation-level accelerations. These modifications preserve all mathematical properties of the estimators but significantly reduce wall-clock computation time.

### 6.1 Acceleration of Gibbs Sampling

The original implementation of the Gibbs sampler relied on a pure Python double loop over iterations and spin indices. This design incurs substantial interpreter overhead, since each update step repeatedly calls Python functions and executes conditional logic at the Python level. Although the Gibbs sampling algorithm itself is computationally lightweight, the Python overhead becomes a dominant cost when the number of iterations is large.

To address this issue, we refactored the core Gibbs update into a Numba-compiled kernel. The calculation of the conditional probability

$$P(y_r = +1 \mid y_{V \setminus r}) = \frac{\exp(2A_r/T)}{\exp(2A_r/T) + 1}, \qquad A_r = \sum_{t \neq r} u_{rt} y_t,$$

and the subsequent spin update are executed inside a `@njit`-compiled function. This removes all Python-level overhead and compiles the nested loops into efficient machine code.

Importantly, the structure of the sampler remains unchanged: it still operates on a single Markov chain and preserves the sequential dependency across iterations, ensuring the correctness of the underlying Markov process. The only change is that the computational cost per Gibbs iteration is significantly reduced due to just-in-time compilation.

## 6.2 Parallel Log-Likelihood Evaluation in the Posterior

The MH algorithm requires evaluating the log-posterior

$$\log p(u \mid \mathcal{D}) = \sum_{k=1}^{n} \sum_{i<j} u_{ij} y_i^{(k)} y_j^{(k)} / T \; - \; n \log Z(u) \; + \; \log p_{\text{prior}}(u).$$

In the original version, the likelihood contribution contained a triple Python loop over samples and spin pairs, which dominated the runtime.

We refactored this computation into a Numba-accelerated kernel that parallelizes over the sample dimension using `prange`. Since observations are independent, the likelihood is naturally data-parallel. The new implementation removes Python overhead and uses low-level vectorized machine code, resulting in a substantial reduction in computation time per MH iteration. Importantly, the public interface texttttlog_posterior(u_vec) remains unchanged, allowing the MH sampler to be used without modification.

## 6.3 Parallel Evaluation of the Partition Function

For small systems, we compute the exact Ising partition function

$$Z(u) = \sum_{y \in \{-1,+1\}^p} \exp\Big( \sum_{i<j} u_{ij} y_i y_j / T \Big).$$

The baseline implementation enumerated all $2^p$ configurations using `itertools.product`, which involves Python object creation for every configuration and is inherently sequential.

We introduced a Numba-parallel version in which each configuration is encoded by an integer bit pattern. The outer loop over the $2^p$ configurations is parallelized using `prange`, and the energy computation is entirely performed inside compiled machine code. This retains exactness while delivering significant speedups for $p \leq 12$, the range relevant to our experiments.

### 6.4 Summary

Across all components, these efficiency improvements accelerate the core inference pipeline by an order of magnitude. The modifications respect the original mathematical formulation of the model, preserve all public interfaces, and directly address computational efficiency and parallelism.

To quantitatively evaluate the effectiveness of these implementation-level optimizations, we compare the runtime of the original and accelerated implementations on identical problem instances, which were conducted on identical hardware using the same random seed.

| **Examples** | **Original Time (s)** | **Parallel Time (s)** | **Speedup** |
|---|---|---|---|
| `u_example_5x5.txt` | 87.6417 | 14.7756 | 5.93 × |
| `u_example_8x8.txt` | 243.4707 | 71.5287 | 3.40 × |

Table 2: Runtime comparison between original and optimized implementations.

The results demonstrate that the proposed optimizations yield substantial reductions in wall-clock time, with speedups ranging from approximately 3× to 6× across different components.

## 7 Conclusion

In this project, we investigated the simulation and recovery of sparse Ising graphical models from synthetic data, with a primary focus on the numerical and computational aspects of sampling and inference. By working in a controlled setting where the ground-truth interaction matrix $u^*$ is known, we were able to systematically evaluate the accuracy, convergence behavior, and computational cost of several widely used inference methods.

We first validated Gibbs sampling as a reliable tool for generating samples from the Ising model. Through a series of experiments with different initializations and interaction structures, we demonstrated that the sampler converges to the correct stationary distribution when sufficient burn-in is used, and that the resulting empirical correlations accurately reflect the underlying graph structure. These results confirm that Gibbs sampling provides suitable data for downstream parameter recovery.

For parameter inference, we compared two complementary approaches. Exact Bayesian inference via Random Walk Metropolis–Hastings yields accurate recovery of the interaction matrix for small systems, but its reliance on repeated evaluations of the partition function makes it computationally infeasible for moderate or large dimensions. To overcome this limitation, we adopted a pseudo-likelihood approximation combined with $\ell_1$ regularization and Proximal Gradient Descent. This approach avoids computing the partition function entirely and scales naturally to larger sparse graphs, while still producing accurate estimates of the dominant interactions.

A key contribution of this work lies in the implementation-level performance optimizations applied to the inference pipeline. By leveraging just-in-time compilation and parallelization through Numba, we significantly reduced the runtime of Gibbs sampling, likelihood evaluation, and exact partition function computation. Importantly, these accelerations preserve the mathematical correctness of the estimators and Markov chains, highlighting the impact of low-level optimization in scientific computing applications.

Overall, this project illustrates the trade-offs between statistical accuracy and computational feasibility in graphical model inference. While exact Bayesian methods provide principled uncertainty quantification, approximate methods such as pseudo-likelihood MAP estimation offer a practical and scalable alternative. Future work could explore larger system sizes, alternative priors, temperature effects, or more advanced optimization and sampling techniques, further extending the computational and methodological scope of this study.

## Appendix A: Derivation of the Gibbs Conditional Distribution

In this appendix we present the full derivation of the coordinatewise conditional distribution used for Gibbs sampling. This result corresponds to Theorem 1 in Section 2.2.

**Theorem** (Theorem 1)**.** *The conditional distribution of a variable $y_r$ of node $r$ given the rest $y_{V/r}$ is given by*

$$P(y_r \mid y_{V/r}) \;=\; \frac{\exp\Big(2y_r \sum_{t\in V/r} u_{r,t} y_t\Big)}{\exp\Big(2y_r \sum_{t\in V/r} u_{r,t} y_t\Big) + 1}.$$

*Proof.* We separate the sum in the exponent into terms that involve $y_r$ and terms that do not:

$$\sum_{\{s,t\}\in E} u_{s,t} y_s y_t = \sum_{t\in V\setminus r} u_{r,t} y_r y_t + \sum_{\substack{\{s,t\}\in E \\ s\neq r,\, t\neq r}} u_{s,t} y_s y_t$$

Let:

- $S = \sum_{t\in V\setminus r} u_{r,t} y_r y_t$ (terms involving $y_r$, which are the nodes connected to node $r$)
- $C = \sum_{\substack{\{s,t\}\in E \\ s\neq r,\, t\neq r}} u_{s,t} y_s y_t$ (terms not involving $y_r$, which are the nodes not connected to node $r$)

Then the joint probability becomes:

$$\mathbb{P}(y) = \frac{1}{Z}\exp(S + C)$$

The conditional probability $\mathbb{P}(y_r|y_{V\setminus r})$ is proportional to the joint probability:

$$\mathbb{P}(y_r|y_{V\setminus r}) \propto \mathbb{P}(y) \propto \exp(S + C)$$

Since $C$ does not depend on $y_r$:

$$\mathbb{P}(y_r|y_{V\setminus r}) \propto \exp(S) = \exp\left(\sum_{t\in V\setminus r} u_{r,t} y_r y_t\right)$$

Then:

- When $y_r = +1$: $\mathbb{P}(y_r = +1|y_{V\setminus r}) \propto \exp(S)$
- When $y_r = -1$: $\mathbb{P}(y_r = -1|y_{V\setminus r}) \propto \exp(-S)$

The normalization constant is $\exp(S) + \exp(-S)$, so:

$$\mathbb{P}(y_r = +1|y_{V\setminus r}) = \frac{\exp(S)}{\exp(S) + \exp(-S)}, \quad \mathbb{P}(y_r = -1|y_{V\setminus r}) = \frac{\exp(-S)}{\exp(S) + \exp(-S)}$$

We combine both cases into a single expression:

$$\mathbb{P}(y_r|y_{V\setminus r}) = \frac{\exp(2y_r S)}{\exp(2y_r S) + 1}$$

Substituting back $S = \sum_{t \in V \backslash r} u_{r,t} y_t$:

$$\mathbb{P}(y_r | y_{V \backslash r}) = \frac{\exp\left(2y_r \sum_{t \in V \backslash r} u_{r,t} y_t\right)}{\exp\left(2y_r \sum_{t \in V \backslash r} u_{r,t} y_t\right) + 1}$$

□

## Appendix B: Derivation of the MH Acceptance Ratio and Intractability

This appendix contains the complete derivation of the Metropolis–Hastings acceptance ratio used on the parameter matrix $u$, as discussed in Section 2.2.

Ideally we would like to use a Uniform prior for ease of computation, since when computing the acceptance probability, the normalizing constant in the target distribution cancels. However, this is ill suited to our need, as we are trying to recover a sparse matrix, so ideally we would like a stronger prior to push most parameter estimations to zero (parameters here meaning the entries of $u^\star$). This motivates us to use a Laplace prior.

We seek to obtain the posterior distribution $P(u \mid \{y^{(i)}\}_{i=1}^N)$. Since $u$ is symmetric and has zeros on the diagonal, there are $p(p-1)/2$ free parameters. We therefore define $\tilde{u} \in \mathbb{R}^{p(p-1)/2}$ which contains all the degrees of freedom of $u$. We define the Laplace prior: $\tilde{u}_i \sim \text{Laplace}(0, \lambda)$, and a random-walk proposal distribution:

$$v^\star \sim \tilde{u}^{(n)} + \mathcal{N}(0, \sigma^2 I),$$

In this setting, we lose the nice property of normalization constant canceling. To see this:

The general Metropolis-Hastings acceptance probability is:

$$\alpha(u \to v^*) = \min\left(1, \frac{\pi(v^*)}{\pi(u)} \cdot \frac{q(u \mid v^*)}{q(v^* \mid u)}\right)$$

where $\pi(\cdot)$ is the target distribution and $q(\cdot \mid \cdot)$ is the proposal distribution.

For the symmetric random walk proposal $v^* \sim u + \mathcal{N}(0, \sigma^2 I)$, we have $q(v^* \mid u) = q(u \mid v^*)$, so:

$$\alpha(u \to v^*) = \min\left(1, \frac{\pi(v^*)}{\pi(u)}\right)$$

The target distribution is the posterior:

$$\pi(u) = \mathbb{P}(u \mid \{y^{(i)}\}_{i=1}^N) \propto \mathbb{P}(\{y^{(i)}\}_{i=1}^N \mid u) \cdot \mathbb{P}(u)$$

The Ising model likelihood is:

$$\mathbb{P}(\{y^{(i)}\}_{i=1}^N \mid u) = \prod_{i=1}^N \frac{1}{Z(u)} \exp\left(\sum_{\{s,t\} \in E} u_{s,t} y_s^{(i)} y_t^{(i)}\right)$$

where the partition function is:

$$Z(u) = \sum_{y \in \{\pm 1\}^p} \exp\left( \sum_{\{s,t\} \in E} u_{s,t} y_s y_t \right)$$

The Laplace prior is:

$$\mathbb{P}(u) = \prod_{j=1}^{p(p-1)/2} \frac{\lambda}{2} \exp\left(-\lambda |\tilde{u}_j|\right)$$

Thus, the unnormalized posterior is:

$$\pi(u) \propto \frac{1}{[Z(u)]^N} \exp\left( \sum_{i=1}^{N} \sum_{\{s,t\} \in E} u_{s,t} y_s^{(i)} y_t^{(i)} \right) \cdot \mathbb{P}(u)$$

The acceptance ratio is:

$$\frac{\pi(v^*)}{\pi(u)} = \frac{\mathbb{P}(\{y^{(i)}\} \mid v^*) \mathbb{P}(v^*)}{\mathbb{P}(\{y^{(i)}\} \mid u) \mathbb{P}(u)}$$

Substituting the likelihood expressions:

$$\frac{\pi(v^*)}{\pi(u)} = \frac{\frac{1}{[Z(v^*)]^N} \exp\left( \sum_{i=1}^{N} \sum_{\{s,t\} \in E} v^*_{s,t} y_s^{(i)} y_t^{(i)} \right) \mathbb{P}(v^*)}{\frac{1}{[Z(u)]^N} \exp\left( \sum_{i=1}^{N} \sum_{\{s,t\} \in E} u_{s,t} y_s^{(i)} y_t^{(i)} \right) \mathbb{P}(u)}$$

Simplifying:

$$\frac{\pi(v^*)}{\pi(u)} = \left( \frac{Z(u)}{Z(v^*)} \right)^N \cdot \frac{\exp\left( \sum_{i=1}^{N} \sum_{\{s,t\} \in E} v^*_{s,t} y_s^{(i)} y_t^{(i)} \right)}{\exp\left( \sum_{i=1}^{N} \sum_{\{s,t\} \in E} u_{s,t} y_s^{(i)} y_t^{(i)} \right)} \cdot \frac{\mathbb{P}(v^*)}{\mathbb{P}(u)}$$

The key term is:

$$\left( \frac{Z(u)}{Z(v^*)} \right)^N$$

where:

$$Z(u) = \sum_{y \in \{\pm 1\}^p} \exp\left( \sum_{\{s,t\} \in E} u_{s,t} y_s y_t \right)$$

This yields a per-iteration computational cost of $\Theta(2^p)$, Therefore, for large $p$, computing $Z(u)$ requires summing over $2^p$ configurations, which is computationally intractable.This motivates the pseudo-likelihood approximation discussed in Section 2.4.

## Appendix C: Derivation of the Pseudo-likelihood MAP Objective

**Theorem** (Theorem 2)**.** *Suppose that the likelihood $P(y_1, y_2, \ldots, y_p \mid u)$ is well approximated by*

$$P(y_1, y_2, \ldots, y_p \mid u) \;\approx\; \prod_{r=1}^{p} P(y_r \mid y_{V/r}, u),$$

*where the conditional probabilities are given explicitly by Theorem 1.*

*The MAP estimator of $u$ is given by*

$$\arg \min_{\substack{u \\ u = u^\top \\ \operatorname{diag}(u) = 0}} \left\{ \sum_{i=1}^{N} \sum_{r=1}^{p} -\log\Big( P\big(y_r^{(i)} \mid y_{V/r}^{(i)}, u\big) \Big) \;+\; \frac{1}{\lambda} \sum_{i<j} |u_{ij}| \right\}.$$

*Proof.* Assume $u$ is symmetric with 0s in the diagonal, we have:

$$u_{\text{MAP}} = \arg\max_{u} \mathbb{P}(u \mid \{y^{(i)}\}_{i=1}^{N})$$

By Bayes' theorem:

$$\mathbb{P}(u \mid \{y^{(i)}\}) \propto \mathbb{P}(\{y^{(i)}\} \mid u) \cdot \mathbb{P}(u)$$

Take $-log$:

$$u_{\text{MAP}} = \arg\min_{u} \Big[ -\log \mathbb{P}(\{y^{(i)}\} \mid u) - \log \mathbb{P}(u) \Big]$$

Use the pseudo-likelihood approximation:

$$\mathbb{P}(\{y^{(i)}\} \mid u) \approx \prod_{i=1}^{N} \prod_{r=1}^{p} \mathbb{P}(y_r^{(i)} \mid y_{V/r}^{(i)}, u)$$

$$u_{\text{MAP}} = \arg\min_{u} \left[ \sum_{i=1}^{N} \sum_{r=1}^{p} - \log \mathbb{P}(y_r^{(i)} \mid y_{V/r}^{(i)}, u) - \log \mathbb{P}(u) \right]$$

Laplace prior:

$$\mathbb{P}(u) = \prod_{i<j} \frac{1}{2\lambda} \exp\left( - \frac{|u_{ij}|}{\lambda} \right)$$

Take $-log$:

$$- \log \mathbb{P}(u) = \frac{1}{\lambda} \sum_{i<j} |u_{ij}| + constant$$

We have:

$$u_{\text{MAP}} = \arg\min_{u} \left[ \sum_{i=1}^{N} \sum_{r=1}^{p} - \log \mathbb{P}(y_r^{(i)} \mid y_{V/r}^{(i)}, u) + \frac{1}{\lambda} \sum_{i<j} |u_{ij}| \right]$$

$\square$

This objective consists of a smooth convex part (pseudo-likelihood) and a non-smooth convex part ($\ell_1$ penalty), making it well-suited for Proximal Gradient Descent as implemented in Section 2.4.

## References

[1] Julian Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society Series D: The Statistician*, 24(3):179–195, 1975.

[2] M. E. J. Newman and G. T. Barkema. *Monte Carlo Methods in Statistical Physics.* Clarendon Press, Oxford, 1999.