High-Dimensional Gaussian Sampling

Yuejia Zhang

April 7, 2023

SIAM Review 2022, code: `https://github.com/mvono/PyGauss`.

Problem Definition

Sampling from a d-dimensional Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where d may be large.

$$\pi(\boldsymbol{\theta}) = \frac{1}{(2\pi)^{d/2}\det(\boldsymbol{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu})\right).$$

Covariance matrix $\boldsymbol{\Sigma}$ positive definite. Precision matrix $\mathbf{Q} = \boldsymbol{\Sigma}^{-1}$ exists and also positive definite.

Special Cases

- $d = 1$

Algorithm 1 Box–Muller sampler

1: Draw $u_1,\ u_2 \overset{\text{i.i.d.}}{\sim} \mathcal{U}((0,1])$.
2: Set $\tilde{u}_1 = \sqrt{-2\log(u_1)}$.
3: Set $\tilde{u}_2 = 2\pi u_2$.
4: **return** $(\theta_1, \theta_2) = \left(\mu + \frac{\tilde{u}_1}{\sqrt{q}} \sin(\tilde{u}_2), \mu + \frac{\tilde{u}_1}{\sqrt{q}} \cos(\tilde{u}_2)\right)$.

Special Cases

Algorithm 2 Sampler when $\mathbf{Q}$ is a diagonal matrix

for $i \in [d]$ **do** ▷ In some programming languages, this loop can be vectorized.
  Draw $\theta_i \sim \mathcal{N}(\mu_i, 1/q_i)$.
end for
return $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_d)^\top$.

General Cases

Algorithm 3 Cholesky sampler

1: Set $\mathbf{C} = \text{chol}(\mathbf{Q})$. ▷ $\mathbf{Q} = \mathbf{C}\mathbf{C}^\top$
2: Draw $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$.
3: Solve $\mathbf{C}^\top \mathbf{w} = \mathbf{z}$ w.r.t. $\mathbf{w}$.
4: **return** $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{w}$.

Problem:

- Computational cost $\mathcal{O}(d^3 + d^2 T)$ (T is the number of samples), only when $\mathbf{Q}$ is unchanged.
- Storage requirement $\Theta(d^2)$.

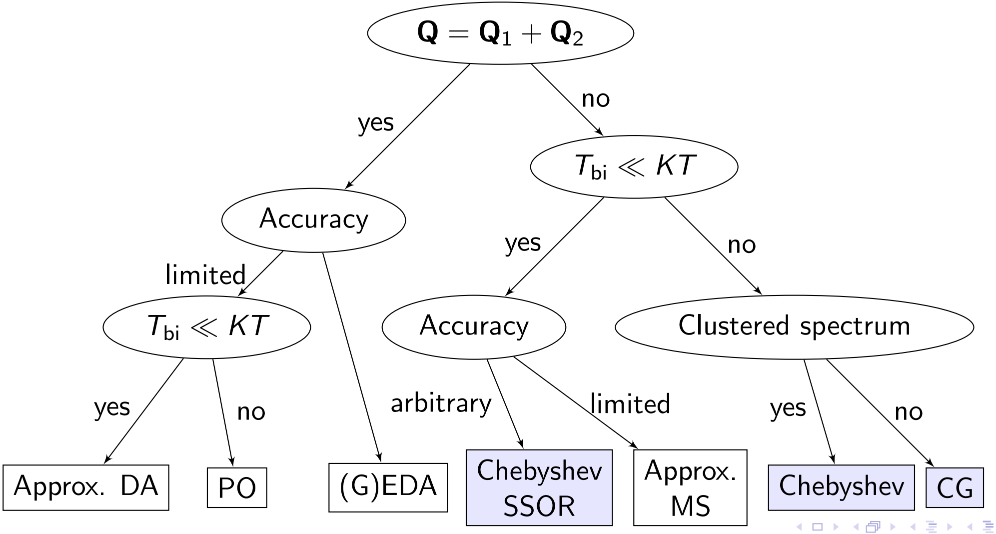
More Efficient Solutions

- Square Root approximation: Approximate $\mathbf{Q}^{1/2}$.
- Conjugate Gradient: Solve a linear system w.r.t. $\mathbf{Q}$.
- Matrix Splitting: A generalization of Gibbs Sampler.
- Data Augmentation: Make use of structure $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$, introduce auxiliary variable to facilitate sampling.

Improvement:

- Computational cost $\mathcal{O}(Kd^2T)$ (K is the number of iterations), or $\mathcal{O}(d^2(T + T_{\text{bi}}))$ (T_{bi} is the number of burn-in samples).
- Storage requirement $\Theta(d)$.

How to Choose the Sampler



Bayesian Ridge Regression

Conditional prior for $\boldsymbol{\theta}$: Gaussian i.i.d.,

$$p(\boldsymbol{\theta} \mid \tau) \propto \exp\left(-\frac{1}{2\tau}||\boldsymbol{\theta}||^2\right),$$

$$p(\tau) \propto \frac{1}{\tau}\mathbf{1}_{\mathbb{R}_+\setminus\{0\}}(\tau).$$

Posterior:

$$p(\boldsymbol{\theta}, \tau \mid \mathbf{y}) \propto \frac{1}{\tau}\mathbf{1}_{\mathbb{R}_+\setminus\{0\}}(\tau) \ \exp\left(-\frac{1}{2\tau}||\boldsymbol{\theta}||^2 - \frac{1}{2\sigma^2}||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||^2\right).$$

Bayesian Ridge Regression (Cont.d)

Conditional posterior distribution associated to $\boldsymbol{\theta}$: Gaussian with precision matrix and mean vector

$$\mathbf{Q} = \frac{1}{\sigma^2}\mathbf{X}^\top\mathbf{X} + \tau^{-1}\mathbf{I}_d,$$
$$\boldsymbol{\mu} = \frac{1}{\sigma^2}\mathbf{Q}^{-1}\mathbf{X}^\top\mathbf{y}.$$

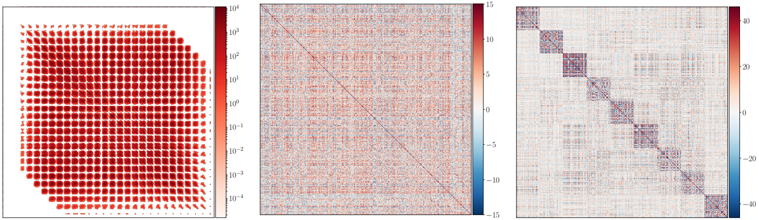


Figure: Examples of precision matrices $\mathbf{X}^\top\mathbf{X}$ for the MNIST, leukemia abd CoEPrA datasets.

Square Root Factorization

Extension of Cholesky sampler:

1. $\mathbf{Q} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$.
2. $\mathbf{Q} = \mathbf{B}^2$ with $\mathbf{B} = \mathbf{U}\boldsymbol{\Lambda}^{1/2}\mathbf{U}^\top$.
3. $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$.
4. Solve $\mathbf{B}\mathbf{w} = \mathbf{z}$ w.r.t. $\mathbf{w}$ and compute $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{w}$.

We have $f(\mathbf{Q}) = \mathbf{U}f(\boldsymbol{\Lambda})\mathbf{U}^\top$ for real continuous f.
Approximate $f(\lambda_i) \approx 1/\sqrt{\lambda_i}, \quad \forall i \in [d]$ with Chebyshev polynomials.

Chebyshev Sampler

The change of interval:

$$g_j = \left[\cos\left(\pi \frac{2j+1}{2K_{\text{cheby}}}\right) \frac{(\lambda_u - \lambda_l)}{2} + \frac{\lambda_u + \lambda_l}{2}\right]^{-1/2}, \quad j \in [0, K_{\text{cheby}}].$$

The Chebyshev coefficients:

$$c_k = \frac{2}{K_{\text{cheby}}} \sum_{j=0}^{K_{\text{cheby}}} g_j \cos\left(\pi k \frac{2j+1}{2K_{\text{cheby}}}\right), \quad k \in [0, K_{\text{cheby}}].$$

Chebyshev Sampler

Algorithm 4 Approx. square root sampler using Chebyshev polynomials

Draw $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$.
Set $\alpha = \dfrac{2}{\lambda_u - \lambda_l}$ and $\beta = \dfrac{\lambda_u + \lambda_l}{\lambda_u - \lambda_l}$.
Initialize $\mathbf{u}_1 = \alpha \mathbf{Q} \mathbf{z} - \beta \mathbf{z}$ and $\mathbf{u}_0 = \mathbf{z}$.
Set $\mathbf{u} = \frac{1}{2} c_0 \mathbf{u}_0 + c_1 \mathbf{u}_1$ and $k = 2$.
while $k \leq K_{\text{cheby}}$ **do** ▷ Compute the K_{cheby}-truncated Chebyshev series.
    Compute $\mathbf{u}' = 2(\alpha \mathbf{Q} \mathbf{u}_1 - \beta \mathbf{u}_1) - \mathbf{u}_0$.
    Set $\mathbf{u} = \mathbf{u} + c_k \mathbf{u}'$.
    Set $\mathbf{u}_0 = \mathbf{u}_1$ and $\mathbf{u}_1 = \mathbf{u}'$.
    $k = k + 1$.
end while

Perturbation before Optimization

Rewrite in *information form*:

$$\pi(\boldsymbol{\theta}) \propto \exp\left(-\frac{1}{2}\boldsymbol{\theta}^\top \mathbf{Q}\boldsymbol{\theta} + \mathbf{b}^\top \boldsymbol{\theta}\right),$$

where $\mathbf{b} = \mathbf{Q}\boldsymbol{\mu}$.

1. Draw a Gaussian vector $\mathbf{z}' \sim \mathcal{N}(\mathbf{0}_d, \mathbf{Q})$.
2. Solve a linear system $\mathbf{Q}\boldsymbol{\theta} = \mathbf{Q}\boldsymbol{\mu} + \mathbf{z}'$ using conjugate gradient methods. (If $\mathbf{u} \sim \mathcal{N}(\mathbf{Q}\boldsymbol{\mu}, \mathbf{Q})$, then $\mathbf{Q}^{-1}\mathbf{u} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$.)

Optimization with Perturbation

The linear system we solved

$$\mathbf{Q}\theta = \mathbf{b} + \mathbf{z}'$$

can also be seen as a perturbed version of the linear system

$$\mathbf{Q}\theta = \mathbf{b},$$

where $\mathbf{b} = \mathbf{Q}\mu$.

Add a perturbation step (a univariate Gaussian sampling step) to turn the classical CG solver into a CG sampler.

Sequentially builds a Gaussian vector with a covariance matrix being the best k-rank approximation of $\mathbf{Q}^{-1}$ in the Krylov subspace $\mathcal{K}_k(\mathbf{Q}, \mathbf{r}^{(0)})$.

CG Sampler

Set $k = 1$, $\mathbf{r}^{(0)} = \mathbf{c} - \mathbf{Q}\boldsymbol{\omega}^{(0)}$, $\mathbf{h}^{(0)} = \mathbf{r}^{(0)}$, $d^{(0)} = \mathbf{h}^{(0)\top}\mathbf{Q}\mathbf{h}^{(0)}$ and $\mathbf{y}^{(0)} = \boldsymbol{\omega}^{(0)}$.

while $\left\|\mathbf{r}^{(k)}\right\| \geq \epsilon$ **do**

  Set $\gamma^{(k-1)} = \dfrac{\mathbf{r}^{(k-1)\top}\mathbf{r}^{(k-1)}}{d^{(k-1)}}$.

  Set $\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - \gamma^{(k-1)}\mathbf{Q}\mathbf{h}^{(k-1)}$.

  Set $\eta^{(k)} = -\dfrac{\mathbf{r}^{(k)\top}\mathbf{r}^{(k)}}{\mathbf{r}^{(k-1)T}\mathbf{r}^{(k-1)}}$.

  Set $\mathbf{h}^{(k)} = \mathbf{r}^{(k)} - \eta^{(k)}\mathbf{h}^{(k-1)}$.

  Set $d^{(k)} = \mathbf{h}^{(k)\top}\mathbf{Q}\mathbf{h}^{(k)}$.

  Set $\mathbf{y}^{(k)} = \mathbf{y}^{(k-1)} + \dfrac{z}{\sqrt{d^{(k-1)}}}\mathbf{h}^{(k-1)}$ where $z \sim \mathcal{N}(0,1)$. ▷ Perturbation

  $k = k + 1$.

end while

Set $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{y}^{(K_{\mathrm{CG}})}$ where K_{CG} is the number of CG iterations.

return $\boldsymbol{\theta}$.

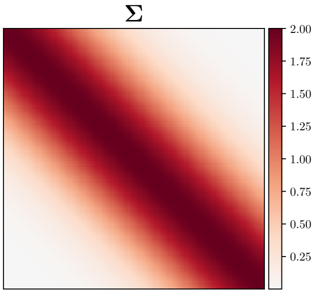
Iterative Approaches or Factorization Approaches?

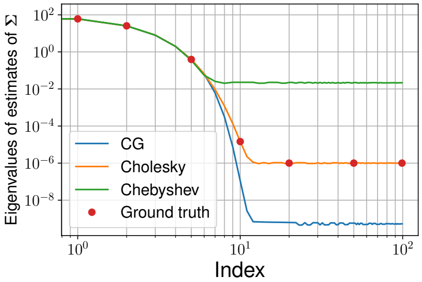
- Memory needs of order $\Theta(d^2)$ prohibitive.
- If storage not an issue, $K \ll (d + T - 1)/T$?
- Gaussian sampling step embedded within a Gibbs sampler, with a varying covariance or precision matrix: $K \ll d$?

Settings

$$\mathbf{\Sigma}_{ij} = 2\exp\left(-\frac{(s_i - s_j)^2}{2a^2}\right) + \epsilon\delta_{ij}, \quad \forall i, j \in [d].$$

where $\{s_i\}_{i\in[d]}$ are evenly spaced scalars on $[-3, 3]$, $\epsilon > 0$.
$a = 1.5$ and $\epsilon = 10^{-6}$, small eigenvalues of $\mathbf{\Sigma}$ clustered together near 10^{-6}.





Results for Accuracy

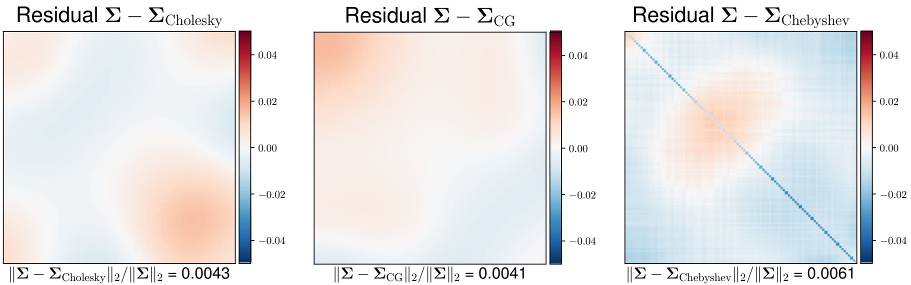
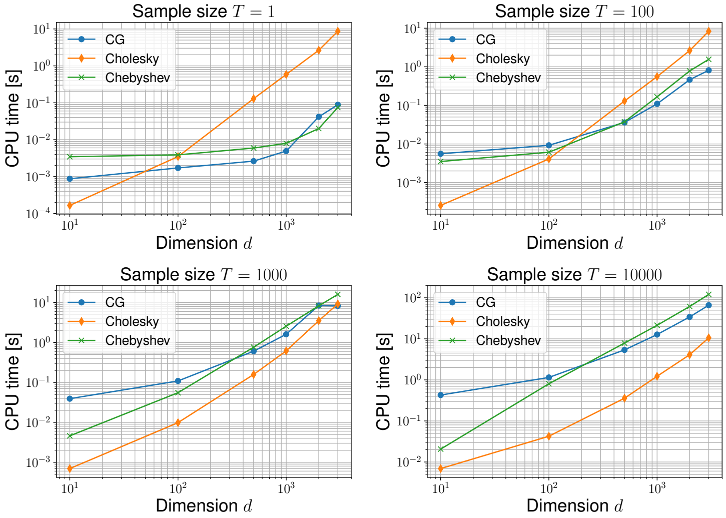
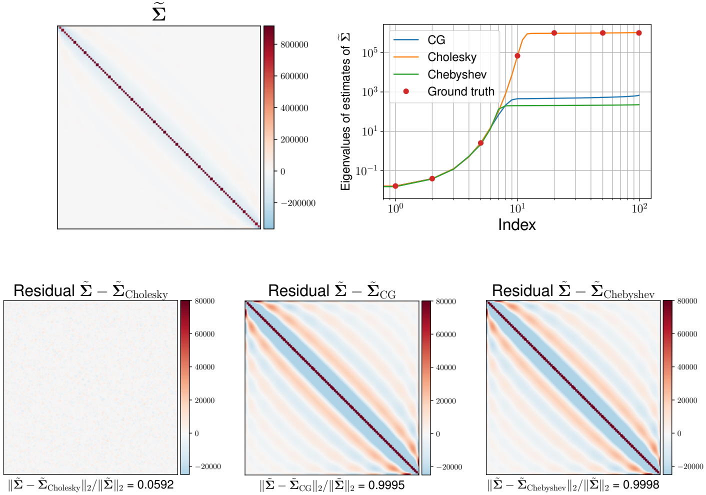


Figure: Results of the three considered samplers for the sampling from $\mathcal{N}(\mathbf{0}_d, \mathbf{\Sigma})$ in dimension $d = 100$.

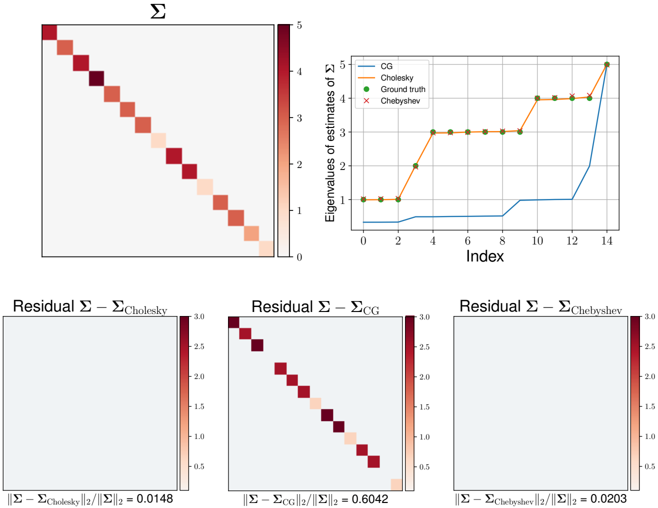
Results for CPU Time



Results When Large Eigenvalues Are Clustered



CG or Chebyshev?



Conditional Gaussian Distribution

If $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$, then

$$\mathrm{E}(\boldsymbol{\theta}_i \mid \boldsymbol{\theta}_{-i}) = \boldsymbol{\mu} - \frac{1}{\mathbf{Q}_{ii}} \sum_{j \neq i} \mathbf{Q}_{ij}(\boldsymbol{\theta}_j - \boldsymbol{\mu}_j),$$

$$\mathrm{Prec}(\boldsymbol{\theta}_i \mid \boldsymbol{\theta}_{-i}) = \mathbf{Q}_{ii},$$

$$\mathrm{Corr}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j \mid \boldsymbol{\theta}_{-ij}) = -\frac{\mathbf{Q}_{ij}}{\sqrt{\mathbf{Q}_{ii}\mathbf{Q}_{jj}}}.$$

Compare the above results with

$$\mathrm{Var}(\boldsymbol{\theta}_i) = \boldsymbol{\Sigma}_{ii},$$

$$\mathrm{Corr}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \frac{\boldsymbol{\Sigma}_{ij}}{\sqrt{\boldsymbol{\Sigma}_{ii}\boldsymbol{\Sigma}_{jj}}}.$$

Gibbs Sampler

Algorithm 5 Component-wise Gibbs sampler

Input: Number T of iterations and initialization $\boldsymbol{\theta}^{(0)}$.

1: Set $t = 1$.
2: **while** $t \leq T$ **do**
3:     **for** $i \in [d]$ **do**
4:         Draw $z \sim \mathcal{N}(0, 1)$.
5:         Set $\theta_i^{(t)} = \dfrac{[\mathbf{Q}\boldsymbol{\mu}]_i}{Q_{ii}} + \dfrac{z}{\sqrt{Q_{ii}}} - \dfrac{1}{Q_{ii}}\left(\sum_{j>i} Q_{ij}\theta_j^{(t-1)} + \sum_{j<i} Q_{ij}\theta_j^{(t)}\right)$.
6:     **end for**
7:     Set $t = t + 1$.
8: **end while**
9: **return** $\boldsymbol{\theta}^{(T)}$.

Rewrite into Gauss–Seidel Linear Systems

Each iteration solves the linear system

$$(\mathbf{L}+\mathbf{D})\theta^{(t)} = \mathbf{Q}\mu + \mathbf{D}^{1/2}\mathbf{z} - \mathbf{L}^\top\theta^{(t-1)},$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$.
By setting $\mathbf{M} = \mathbf{L} + \mathbf{D}$ and $\mathbf{N} = -\mathbf{L}^\top$ so that $\mathbf{Q} = \mathbf{M} - \mathbf{N}$,

$$\mathbf{M}\theta^{(t)} = \mathbf{Q}\mu + \tilde{\mathbf{z}} + \mathbf{N}\theta^{(t-1)},$$

where $\mathbf{N} = -\mathbf{L}^\top$ is strictly upper triangular and $\tilde{\mathbf{z}} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{D})$ is easy to sample.

Matrix Splitting Sampler

Algorithm 6 MCMC sampler based on exact matrix splitting

Input: Number T of iterations, initialization $\theta^{(0)}$ and splitting $\mathbf{Q} = \mathbf{M} - \mathbf{N}$.

1: Set $t = 1$.
2: **while** $t \leq T$ **do**
3: Draw $\tilde{\mathbf{z}} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{M}^\top + \mathbf{N})$.
4: Solve $\mathbf{M}\theta^{(t)} = \mathbf{Q}\mu + \tilde{\mathbf{z}} + \mathbf{N}\theta^{(t-1)}$ w.r.t. $\theta^{(t)}$.
5: Set $t = t + 1$.
6: **end while**
7: **return** $\theta^{(T)}$.

Other Matrix Splitting Schemes

Table: The matrices $\mathbf{D}$ and $\mathbf{L}$ denote the diagonal and strictly lower triangular parts of $\mathbf{Q}$, respectively, and ω is a positive scalar.

Sampler	$\mathbf{M}$	$\mathbf{N}$	$\mathrm{cov}(\tilde{\mathbf{z}}) = \mathbf{M}^\top + \mathbf{N}$	convergence
Richardson	$\mathbf{I}_d/\omega$	$\mathbf{I}_d/\omega - \mathbf{Q}$	$2\mathbf{I}_d/\omega - \mathbf{Q}$	$0 < \omega < 2/\left\Vert\mathbf{Q}\right\Vert$
Jacobi	$\mathbf{D}$	$\mathbf{D} - \mathbf{Q}$	$2\mathbf{D} - \mathbf{Q}$	$\lvert Q_{ii}\rvert > \sum_{j\neq i}\lvert Q_{ij}\rvert \ \forall i \in [d]$
Gauss–Seidel	$\mathbf{D} + \mathbf{L}$	$-\mathbf{L}^\top$	$\mathbf{D}$	always
SOR	$\mathbf{D}/\omega + \mathbf{L}$	$\frac{1-\omega}{\omega}\mathbf{D} - \mathbf{L}^\top$	$\frac{2-\omega}{\omega}\mathbf{D}$	$0 < \omega < 2$

Error of t-th Order Polynomial

Given a linear system $\mathbf{Q}\boldsymbol{\theta} = \mathbf{v}$ and linear solvers based on the matrix splitting $\mathbf{Q} = \mathbf{M} - \mathbf{N}$, consider the recursion,

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \mathbf{M}^{-1}(\mathbf{v} - \mathbf{Q}\boldsymbol{\theta}^{(t)}).$$

The error at iteration t,

$$\mathbf{e}^{(t+1)} = \boldsymbol{\theta}^{(t+1)} - \mathbf{Q}^{-1}\mathbf{v},$$

is equal to

$$\mathbf{e}^{(t+1)} = (\mathbf{I}_d - \mathbf{M}^{-1}\mathbf{Q})^t\mathbf{e}^{(0)}.$$

Can we find another t-th order polynomial P_t that achieves a lower error?

$$\rho(\mathrm{P}_t(\mathbf{M}^{-1}\mathbf{Q})) < \rho((\mathbf{I}_d - \mathbf{M}^{-1}\mathbf{Q})^t).$$

Polynomial Accelerated Solver

Consider the second-order iterative scheme, for any $t \in \mathbb{N}$,

$$\boldsymbol{\theta}^{(t+1)} = \alpha_t \boldsymbol{\theta}^{(t)} + (1 - \alpha_t)\boldsymbol{\theta}^{(t-1)} + \beta_t \mathbf{M}^{-1}(\mathbf{v} - \mathbf{Q}\boldsymbol{\theta}^{(t)}),$$

where $(\alpha_t, \beta_t)_{t\in\mathbb{N}}$ are a set of acceleration parameters.
This iterative method yields an error at step t given by

$$\mathbf{e}^{(t+1)} = \mathrm{P}_t(\mathbf{M}^{-1}\mathbf{Q})\mathbf{e}^{(0)},$$

where P_t stands for a scaled Chebyshev polynomial.
Optimal values for $(\alpha_t, \beta_t)_{t\in\mathbb{N}}$ are given by

$$\alpha_t = \tau_1 \beta_t \quad \text{and} \quad \beta_t = \left(\tau_1 - \tau_2^2 \beta_{t-1}\right)^{-1},$$

$\tau_1 = [\lambda_{\min}(\mathbf{M}^{-1}\mathbf{Q}) + \lambda_{\max}(\mathbf{M}^{-1}\mathbf{Q})]/2$ and $\tau_2 = [\lambda_{\max}(\mathbf{M}^{-1}\mathbf{Q}) - \lambda_{\min}(\mathbf{M}^{-1}\mathbf{Q})]/4$.

Symmetric Splitting Scheme

Denote by $\mathbf{M}_{\text{SOR}}$ and $\mathbf{N}_{\text{SOR}}$ the matrices involved in the SOR splitting such that $\mathbf{Q} = \mathbf{M}_{\text{SOR}} - \mathbf{N}_{\text{SOR}}$.

Then for any $0 < \omega < 2$, the SSOR (symmetric SOR) splitting is defined by $\mathbf{Q} = \mathbf{M}_{\text{SSOR}} - \mathbf{N}_{\text{SSOR}}$ with

$$\mathbf{M}_{\text{SSOR}} = \frac{\omega}{2-\omega}\mathbf{M}_{\text{SOR}}\mathbf{D}^{-1}\mathbf{M}_{\text{SOR}}^{\top} \quad \text{and} \quad \mathbf{N}_{\text{SSOR}} = \frac{\omega}{2-\omega}\mathbf{N}_{\text{SOR}}\mathbf{D}^{-1}\mathbf{N}_{\text{SOR}}^{\top} \; .$$

Approximate Matrix Splitting

Algorithm 7 MCMC sampler based on approximate matrix splitting

Input: Number T of iterations, initialization $\theta^{(0)}$ and splitting $\mathbf{Q} = \mathbf{M} - \mathbf{N}$.

1: Set $t = 1$.
2: **while** $t \leq T$ **do**
3: Draw $\tilde{\mathbf{z}}' \sim \mathcal{N}(\mathbf{0}_d, \tilde{\mathbf{M}})$. $\triangleright$ $\tilde{\mathbf{M}} = \mathbf{D}$ or $2\left(\mathbf{D} + 2\omega \mathbf{I}_d\right)$.
4: Solve $\mathbf{M}\theta^{(t)} = \mathbf{Q}\mu + \tilde{\mathbf{z}}' + \mathbf{N}\theta^{(t-1)}$.
5: Set $t = t + 1$.
6: **end while**
7: **return** $\theta^{(T)}$.

Approximate Matrix Splitting Samplers

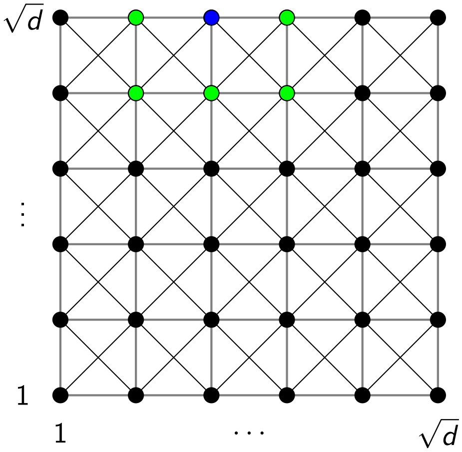
Sampler	$\mathbf{M}$	$\mathbf{N}$	$\text{cov}(\tilde{\mathbf{z}}') = \tilde{\mathbf{M}}$
Hogwild with blocks of size 1	$\mathbf{D}$	$-\mathbf{L} - \mathbf{L}^\top$	$\mathbf{D}$
Clone MCMC	$\mathbf{D} + 2\omega \mathbf{I}_d$	$2\omega \mathbf{I}_d - \mathbf{L} - \mathbf{L}^\top$	$2\left(\mathbf{D} + 2\omega \mathbf{I}_d\right)$

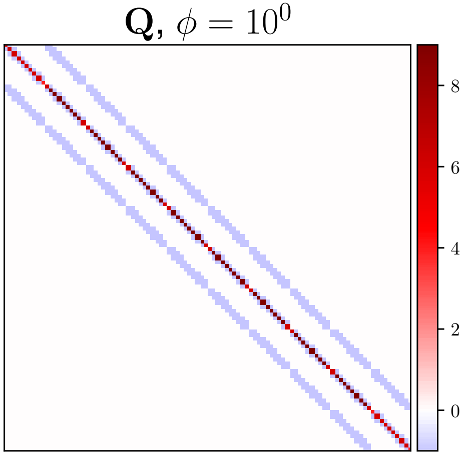
$$\widetilde{\mathbf{Q}}_{\text{MS}} = \begin{cases} \mathbf{Q}\left(\mathbf{I}_d - \mathbf{D}^{-1}(\mathbf{L} + \mathbf{L}^\top)\right) & \text{for the Hogwild sampler} \\ \mathbf{Q}\left(\mathbf{I}_d - \frac{1}{2}(\mathbf{D} + 2\omega^{-1}\mathbf{I}_d)^{-1}\mathbf{Q}\right) & \text{for clone MCMC.} \end{cases}$$

Iterative Sampler or MCMC Sampler?

- Iterative Sampler: K iterations to generate one sample.
- MCMC Sampler: burn-in period of length T_{bi}.
- $T + T_{\text{bi}} \ll KT$?
- MCMC methods when a large number $T \gtrsim T_{\text{bi}}$ of samples is desired. Iterative methods when a small number $T \lesssim T_{\text{bi}}/K$ of samples is desired.

Settings





Impact of ϕ

$$\mathbf{Q} = \mathbf{I}_d + \phi \mathbf{L}.$$

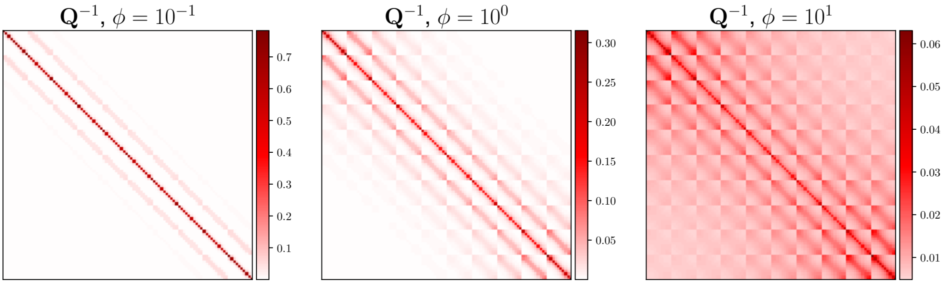


Figure: $\mathbf{Q}^{-1} = \mathbf{\Sigma}$ for $\phi \in \{0.1, 1, 10\}$. $d = 100$.

Accuracy Results

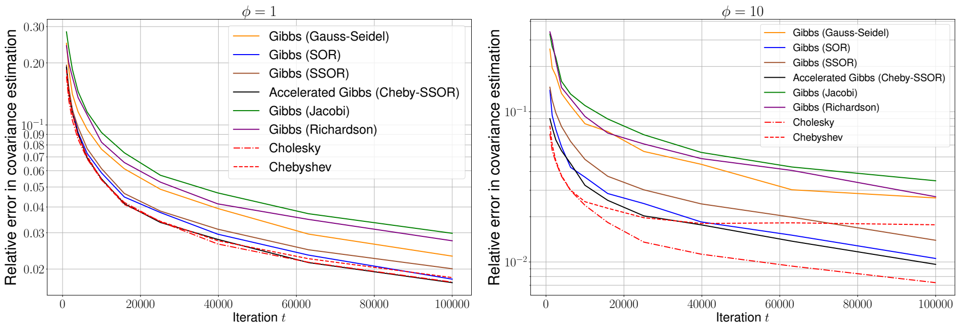


Figure: Relative error associated to the estimation of the covariance matrix $\mathbf{Q}^{-1}$ defined by $\|\mathbf{Q}^{-1} - \mathrm{var}(\boldsymbol{\theta}^{(1:t)})\|_2 / \|\mathbf{Q}^{-1}\|_2$ w.r.t. the number of iterations t, with $d = 100$ (left: $\phi = 1$, right: $\phi = 10$).

Idea

- $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$ directly from the statistical model under construction.
- Introduce one (or several) auxiliary variable $\mathbf{u} \in \mathbb{R}^k$. $(\boldsymbol{\theta}, \mathbf{u})$ simple conditional distributions, then use Gibbs sampler.
- Marginalization of $\mathbf{u}$, retrieves the target distribution $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$:

$$\int_{\mathbb{R}^k} \pi(\boldsymbol{\theta}, \mathbf{u}) = \pi(\boldsymbol{\theta}).$$

EDA Model

Assume that $\mathbf{Q}_2$ presents a particular and simpler structure (e.g., diagonal or circulant) than $\mathbf{Q}_1$.
Introduces the joint distribution with p.d.f.

$$\pi(\boldsymbol{\theta}, \mathbf{u}_1) \propto \exp\left(-\frac{1}{2}\left[(\boldsymbol{\theta}-\boldsymbol{\mu})^\top \mathbf{Q}(\boldsymbol{\theta}-\boldsymbol{\mu}) + (\mathbf{u}_1-\boldsymbol{\theta})^\top \mathbf{R}(\mathbf{u}_1-\boldsymbol{\theta})\right]\right),$$

with $\mathbf{R} = \omega^{-1}\mathbf{I}_d - \mathbf{Q}_1$ and $0 < \omega < \|\mathbf{Q}_1\|^{-1}$, where $\|\cdot\|$ is the spectral norm.
$\mathbf{Q}_1$ and $\mathbf{Q}_2$ decoupled.

GEDA Model

When $\mathbf{Q}$ from a hierarchical Bayesian model, $\mathbf{Q}_1 = \mathbf{G}_1^\top \boldsymbol{\Lambda}_1 \mathbf{G}_1$, where $\boldsymbol{\Lambda}_1$ is a positive definite (and very often diagonal) matrix.
GEDA introduces an additional auxiliary variable $\mathbf{u}_2$ such that the augmented p.d.f. writes

$$\pi(\boldsymbol{\theta}, \mathbf{u}_1, \mathbf{u}_2) \propto \exp\left(-\frac{1}{2}\left[(\boldsymbol{\theta} - \boldsymbol{\mu})^\top \mathbf{Q}(\boldsymbol{\theta} - \boldsymbol{\mu}) + (\mathbf{u}_1 - \boldsymbol{\theta})^\top \mathbf{R}(\mathbf{u}_1 - \boldsymbol{\theta})\right]\right),$$
$$\times \exp\left(-\frac{1}{2}(\mathbf{u}_2 - \mathbf{G}_1\mathbf{u}_1)^\top \boldsymbol{\Lambda}_1(\mathbf{u}_2 - \mathbf{G}_1\mathbf{u}_1)\right).$$

Algorithm of (G)EDA

Algorithm 8 Gibbs sampler based on exact data augmentation (G)EDA

Input: Number T of iterations and initialization $\boldsymbol{\theta}^{(0)}$, $\mathbf{u}_1^{(0)}$.

Set $t = 1$.
while $t \leq T$ **do**
    Draw $\mathbf{u}_2^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{u}_2}, \mathbf{Q}_{\mathbf{u}_2}^{-1})$. ▷ Only if GEDA is considered.
    Draw $\mathbf{u}_1^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{u}_1}, \mathbf{Q}_{\mathbf{u}_1}^{-1})$.
    Draw $\boldsymbol{\theta}^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \mathbf{Q}_{\boldsymbol{\theta}}^{-1})$.
    Set $t = t + 1$.
end while
return $\boldsymbol{\theta}^{(T)}$.

Comparison of EDA and GEDA

Sampler	$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \mathbf{Q}_{\boldsymbol{\theta}}^{-1})$	$\mathbf{u}_1 \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{u}_1}, \mathbf{Q}_{\mathbf{u}_1}^{-1})$	$\mathbf{u}_2 \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{u}_2}, \mathbf{Q}_{\mathbf{u}_2}^{-1})$
EDA	$\mathbf{Q}_{\boldsymbol{\theta}} = \omega^{-1}\mathbf{I}_d + \mathbf{Q}_2$ $\boldsymbol{\mu}_{\boldsymbol{\theta}} = \mathbf{Q}_{\boldsymbol{\theta}}^{-1}(\mathbf{R}\mathbf{u}_1 + \mathbf{Q}\boldsymbol{\mu})$	$\mathbf{Q}_{\mathbf{u}_1} = \mathbf{R}$ $\boldsymbol{\mu}_{\mathbf{u}_1} = \boldsymbol{\theta}$	- -
GEDA	$\mathbf{Q}_{\boldsymbol{\theta}} = \omega^{-1}\mathbf{I}_d + \mathbf{Q}_2$ $\boldsymbol{\mu}_{\boldsymbol{\theta}} = \mathbf{Q}_{\boldsymbol{\theta}}^{-1}(\mathbf{R}\mathbf{u}_1 + \mathbf{Q}\boldsymbol{\mu})$	$\mathbf{Q}_{\mathbf{u}_1} = \omega^{-1}\mathbf{I}_d$ $\boldsymbol{\mu}_{\mathbf{u}_1} = \boldsymbol{\theta} - \omega(\mathbf{Q}_1\boldsymbol{\theta} - \mathbf{G}_1^\top \boldsymbol{\Lambda}_1^{-1}\mathbf{u}_2)$	$\mathbf{Q}_{\mathbf{u}_2} = \boldsymbol{\Lambda}_1$ $\boldsymbol{\mu}_{\mathbf{u}_2} = \mathbf{G}_1\mathbf{u}_1$

Example

Consider Gaussian sampling problems in high dimensions $d \in [10^4, 10^6]$: Cholesky factorization both computationally and memory prohibitive.
Common sampling problem in image processing and linear inverse problem, usually called *deconvolution* or *deblurring* in image processing:

$$\mathbf{y} = \mathbf{S}\boldsymbol{\theta} + \boldsymbol{\varepsilon},$$

where $\mathbf{y} \in \mathbb{R}^d$ blurred and noisy observation, $\boldsymbol{\theta} \in \mathbb{R}^d$ is the unknown original image, $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{\Gamma})$ with $\mathbf{\Gamma} = \mathrm{diag}(\gamma_1, \ldots, \gamma_d)$ a synthetic structured noise such that $\gamma_i \sim 0.7\delta_{\kappa_1} + 0.3\delta_{\kappa_2}$, for all $i \in [d]$. $\mathbf{S} \in \mathbb{R}^{d \times d}$ circulant convolution matrix associated to the space-invariant box blurring kernel $\frac{1}{9}\mathbf{1}_{3\times 3}$.

Prior and Posterior

Smoothing conjugate prior:

$$\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}_d, (\frac{\xi_0}{d}\mathbf{1}_{d\times d} + \xi_1 \boldsymbol{\Delta}^\top \boldsymbol{\Delta})^{-1}),$$

where $\boldsymbol{\Delta}$ is the discrete two-dimensional Laplacian operator; $\xi_0 = 1$ ensures that this prior is non-intrinsic while $\xi_1 = 1$ controls the smoothing.

Prior and Posterior

Gaussian posterior:

$$\boldsymbol{\theta} \mid \mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\mu}, \mathbf{Q}^{-1}\right),$$

where

$$\mathbf{Q} = \mathbf{S}^\top \boldsymbol{\Gamma}^{-1} \mathbf{S} + \frac{\xi_0}{d} \mathbf{1}_{d \times d} + \xi_1 \boldsymbol{\Delta}^\top \boldsymbol{\Delta},$$
$$\boldsymbol{\mu} = \mathbf{Q}^{-1} \mathbf{S}^\top \boldsymbol{\Delta}^{-1} \mathbf{y}.$$

Decompose $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$ with $\mathbf{Q}_1 = \mathbf{S}^\top \boldsymbol{\Gamma}^{-1} \mathbf{S}$ and $\mathbf{Q}_2 = \frac{\xi_0}{d} \mathbf{1}_{d \times d} + \xi_1 \boldsymbol{\Delta}^\top \boldsymbol{\Delta}$.

Circulant (Toeplitz) Matrix

$\mathbf{Q}$ is a block circulant matrix with circulant blocks,

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 & \dots & \mathbf{Q}_M \\ \mathbf{Q}_M & \mathbf{Q}_1 & \dots & \mathbf{Q}_{M-1} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{Q}_2 & \mathbf{Q}_3 & \dots & \mathbf{Q}_1 \end{pmatrix}, \tag{1}$$

where $\{\mathbf{Q}_i\}_{i \in [M]}$ are M circulant matrices.

Circulant (Toeplitz) Matrix

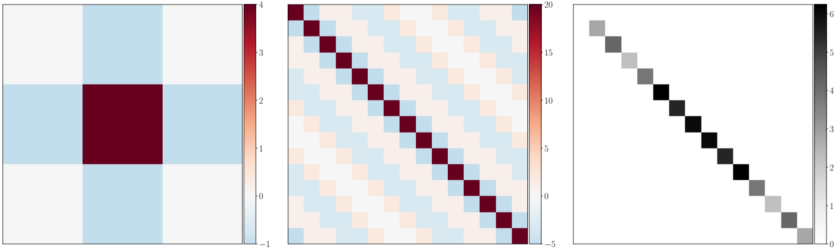


Figure: From left to right: example of a 3×3 Laplacian filter, the associated circulant precision matrix $\mathbf{Q} = \boldsymbol{\Delta}^\top \boldsymbol{\Delta}$ for convolution with periodic boundary conditions and its counterpart diagonal matrix $\mathbf{F}\mathbf{Q}\mathbf{F}^{\mathsf{H}}$ in the Fourier domain, where $\mathbf{F}$ and its Hermitian conjugate $\mathbf{F}^{\mathsf{H}}$ are unitary matrices associated with the Fourier and inverse Fourier transforms.

Efficient Sampling of Block Circulant Matrix with Circulant Blocks

Algorithm 9 Sampler when $\mathbf{Q}$ is a block circulant matrix with circulant blocks

Input: M and N, the number of blocks and the size of each block, respectively.

1: Compute $\mathbf{F} = \mathbf{F}_M \otimes \mathbf{F}_N$. ▷ $\mathbf{F}_M$ is the $M \times M$ unitary matrix associated to the Fourier transform and $\otimes$ denotes the tensor product.
2: Draw $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$.
3: Set $\mathbf{\Lambda}_\mathbf{q} = \mathrm{diag}(\mathbf{q})$. ▷ $\mathbf{q}$ is the d-dimensional vector built by stacking the first columns of each circulant block of $\mathbf{Q}$.
4: Set $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{F}^{\mathrm{H}} \mathbf{\Lambda}_\mathbf{q}^{-1/2} \mathbf{F}\mathbf{z}$.
5: **return** $\boldsymbol{\theta}$.

Computational complexity: $\mathcal{O}(d \log(d))$. Memory requirement: $\Theta(d)$.

Compare Efficiency using ESSR

Effective sample size ratio per second (ESSR): For a MCMC sampler, the ESSR gives an estimate of the equivalent number of i.i.d. samples that can be drawn in one second.

$$\mathrm{ESSR}(\vartheta) = \frac{1}{T_1}\frac{\mathrm{ESS}(\vartheta)}{T} = \frac{1}{T_1\left(1 + 2\sum_{t=1}^{\infty}\rho_t(\vartheta)\right)},$$

where T_1 is the CPU time in seconds required to draw one sample and $\rho_t(\vartheta)$ is the lag-t autocorrelation of a scalar parameter ϑ.

Results

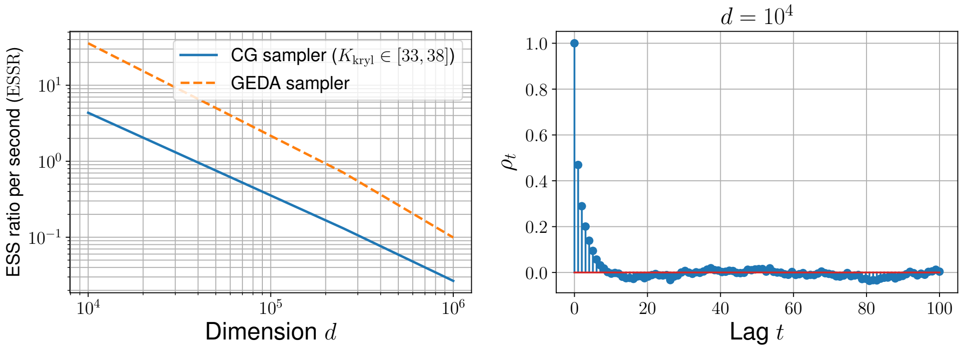


Figure: (left) ESS ratio per second (ESSR); (right) autocorrelation function ρ_t shown for $d = 10^4$. For both figures, we used the slowest component of $\boldsymbol{\theta}$ as the scalar summary ϑ.

Surrogate Probability Distribution

Common form of surrogate p.d.f.:

$$\kappa(\boldsymbol{\theta}, \mathbf{u}) \propto \pi(\boldsymbol{\theta}) \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \mathbf{u})^\top \mathbf{R}(\boldsymbol{\theta} - \mathbf{u})\right)$$

where $\mathbf{u} \in \mathbb{R}^d$ auxiliary variable, $\mathbf{R} \in \mathbb{R}^{d \times d}$ is a symmetric preconditioner.
Restrict $\mathbf{R}$ to be positive definite,

$$\int_{\mathbb{R}^d} \pi(\boldsymbol{\theta}, \mathbf{u}) d\mathbf{u} = Z^{-1} \pi(\boldsymbol{\theta}) \int_{\mathbb{R}^d} \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \mathbf{u})^\top \mathbf{R}(\boldsymbol{\theta} - \mathbf{u})\right) d\mathbf{u} = \pi(\boldsymbol{\theta})$$

holds almost surely with $Z = \det(\mathbf{R})^{-1/2}(2\pi)^{d/2}$.

From Exact Data Augmentation to Exact Matrix Splitting

Change of variable $\mathbf{v} = \mathbf{R}\mathbf{u}$,

$$\mathbf{v} \mid \boldsymbol{\theta} \sim \mathcal{N}\left(\mathbf{R}\boldsymbol{\theta}, \mathbf{R}\right),$$
$$\boldsymbol{\theta} \mid \mathbf{v} \sim \mathcal{N}\left((\mathbf{Q}+\mathbf{R})^{-1}(\mathbf{v}+\mathbf{Q}\boldsymbol{\mu}), (\mathbf{Q}+\mathbf{R})^{-1}\right).$$

Rewrite the Gibbs sampling steps as an auto-regressive process of order 1 w.r.t. $\boldsymbol{\theta}$, the sampling strategy is equivalently

$$\tilde{\mathbf{z}} \sim \mathcal{N}\left(\mathbf{Q}\boldsymbol{\mu}, 2\mathbf{R}+\mathbf{Q}\right),$$
$$\boldsymbol{\theta}^{(t)} = (\mathbf{Q}+\mathbf{R})^{-1}\left(\tilde{\mathbf{z}} + \mathbf{R}\boldsymbol{\theta}^{(t-1)}\right).$$

From Exact Data Augmentation to Exact Matrix Splitting (Cont.d)

Define $\mathbf{M} = \mathbf{Q} + \mathbf{R}$ and $\mathbf{N} = \mathbf{R}$,

$$\tilde{\mathbf{z}} \sim \mathcal{N}\left(\mathbf{Q}\mu, \mathbf{M}^\top + \mathbf{N}\right), \theta^{(t)} \quad = \mathbf{M}^{-1}\left(\tilde{\mathbf{z}} + \mathbf{N}\theta^{(t-1)}\right),$$

which boils down to the Gibbs sampler based on exact MS.

Table: Equivalence relations between exact DA and exact MS approaches.

$\mathbf{R} = \mathrm{cov}(\mathbf{v}\vert\theta)$	$(\mathbf{Q}+\mathbf{R})^{-1} = \mathrm{cov}(\theta\vert\mathbf{v})$	$\mathbf{M}^\top + \mathbf{N} = \mathrm{cov}(\tilde{\mathbf{z}})$	DA sampler	MS sampler
$\frac{\mathbf{I}_d}{\omega} - \mathbf{Q}_1$	$\left(\frac{\mathbf{I}_d}{\omega} + \mathbf{Q}_2\right)^{-1}$	$\frac{2\mathbf{I}_d}{\omega} + \mathbf{Q}_2 - \mathbf{Q}_1$	EDA	Richardson
$\frac{\mathbf{D}_1}{\omega} - \mathbf{Q}_1$	$\left(\frac{\mathbf{D}_1}{\omega} + \mathbf{Q}_2\right)^{-1}$	$\frac{2\mathbf{D}_1}{\omega} + \mathbf{Q}_2 - \mathbf{Q}_1$	**EDAJ**	Jacobi

Proximal point algorithm (PPA)

Algorithm 10 PPA

Choose an initial value $\boldsymbol{\theta}^{(0)}$, a positive semi-definite matrix $\mathbf{R}$ and a maximal number of iterations T.
Set $t = 1$.
while $t \leq T$ **do**
$$\boldsymbol{\theta}^{(t)} = \operatorname*{arg\,min}_{\boldsymbol{\theta} \in \mathbb{R}^d} f(\boldsymbol{\theta}) + \frac{1}{2} \left\| \boldsymbol{\theta} - \boldsymbol{\theta}^{(t-1)} \right\|_{\mathbf{R}}^2.$$
end while
return $\boldsymbol{\theta}^{(T)}$.

Here $\|\boldsymbol{\theta}\|_{\mathbf{R}}^2 \triangleq \boldsymbol{\theta}^\top \mathbf{R} \boldsymbol{\theta}$ defines the *weighted* norm w.r.t. $\mathbf{R}$ for all $\boldsymbol{\theta} \in \mathbb{R}^d$.

Revisit Unifying Proposal Distribution

Recap:

$$\kappa(\boldsymbol{\theta}, \mathbf{u}) \propto \pi(\boldsymbol{\theta}) \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \mathbf{u})^\top \mathbf{R}(\boldsymbol{\theta} - \mathbf{u})\right).$$

Define $\mathbf{u} = \boldsymbol{\theta}^{(t-1)}$, then

$$\kappa(\boldsymbol{\theta}, \mathbf{u}) \triangleq p\left(\boldsymbol{\theta} | \mathbf{u} = \boldsymbol{\theta}^{(t-1)}\right) \propto \pi(\boldsymbol{\theta}) \exp\left(-\frac{1}{2}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(t-1)}\right)^\top \mathbf{R}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(t-1)}\right)\right).$$

PPA is its deterministic version!

Connection between Optimization and Simulation

In fact, searching for the maximum a posteriori estimator under the proposal distribution $P(\cdot \mid \boldsymbol{\theta}^{(t-1)})$ with density $p(\cdot \mid \boldsymbol{\theta}^{(t-1)})$ boils down to solving

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\arg\min}\ \underbrace{-\log\pi(\boldsymbol{\theta})}_{f(\boldsymbol{\theta})}+\frac{1}{2}\left\|\boldsymbol{\theta}-\boldsymbol{\theta}^{(t-1)}\right\|_{\mathbf{R}}^{2}.$$