

Hierarchical-block conditioning approximations for high-dimensional multivariate normal probabilities

Hyunseok Yang, Kyeongwon Lee, and Songhyun Kim

December 9, 2019

Department of Statistics, SNU

Table of contents

1. Introduction
2. Multidimensional Conditioning Approximations
3. Hierarchical-Block Approximation
4. Block Reordering
5. Numerical Examples
 - Cholesky Factorization
 - Multidimensional Conditioning Approximations
 - Hierarchical-Block Approximations
6. Conclusion

Introduction

Introduction

- The computation of the multivariate normal (MVN) probability

$$\Phi_n(\mathbf{a}, \mathbf{b}; 0, \mathbf{\Sigma}) = \int_a^b \frac{1}{\sqrt{(2\pi)^n |\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2} \mathbf{x}^T \mathbf{\Sigma}^{-1} \mathbf{x}\right) d\mathbf{x}, \quad (1)$$

where \mathbf{a} and \mathbf{b} are integration limits, the mean vector μ is assumed to be 0, $\mathbf{\Sigma}$ is a positive-definite covariance matrix, is required for a variety of applications.

- Various methods to compute MVN probability are suggested such as Richtmyer Quasi-Monte Carlo (QMC) (Genz and Bretz, 2009)
- However, In high-dimensional settings (large n), it is hard to compute (1) directly.
- We review new approaches proposed by Cao et al. (2019) to approximate high-dimensional multivariate normal probability (1) using the hierarchical matrix \mathcal{H} (Hackbusch, 2015) for the covariance matrix $\mathbf{\Sigma}$.

The methods are based on the concepts of

1. The bivariate conditioning method (Trinh and Genz, 2015)
 - LDL decomposition and truncated expectations
 - Generalized to d -dimensioned-case
2. The hierarchical QMC method (Genton et al., 2018)
 - Hierarchical cholseky decomposition: efficient and fast computation
 - QMC: beneficial on computation complexity and memory management
3. Reordering (Trinh and Genz, 2015)
 - Univariate reordering (Trinh and Genz, 2015)
 - Generalized the upper case to blockwise reordering

Multidimensional Conditioning Approximations

Quasi Monte Carlo Method

- Monte Carlo Error bound : $O(N^{-1/2})$ for monte carlo(MC) method
- Genz and Bretz (2009) claimed independent sample points is the reason of slow convergence.
- Via employing low discrepancy sets for sequence, QMC is asymptotically efficient than MC.
- With $\Delta \sim U[0, 1]^n$,

$$L_N = \{\mathbf{z} + \Delta \bmod 1 : \mathbf{z} \in K_N\}$$

$$K_N = \{i\mathbf{q} \bmod 1, i = 1, \dots, N\}$$

where $\mathbf{q} = \sqrt{\mathbf{p}}$ and \mathbf{p} is set of prime numbers.

- Since square root of prime numbers is irrational and linear independent over the rational numbers,

Quasi Monte Carlo Method

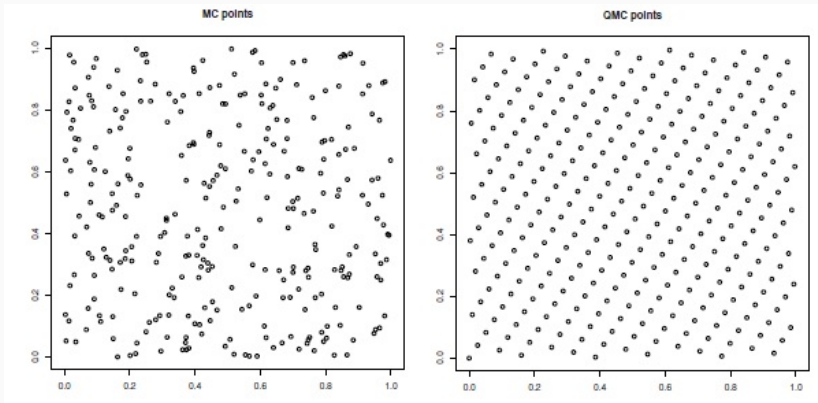


Figure 1: Comparison of MC and QMC sample points(Genz and Bretz, 2009)

Quasi Monte Carlo Method

$$\Phi_n(\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}; \Sigma) = \Phi_n(\mathbf{a} \leq \mathbf{L}\mathbf{y} \leq \mathbf{b}; I_n)$$

$$\begin{aligned} &= \int_{a_1 \leq l_{11}y_1 \leq b_1} \phi(y_1) \cdots \int_{a_n \leq l_{n1}^t y_1 \leq b_n} \phi(y_n) d\mathbf{y} \\ &= \int_{\tilde{a}_1}^{\tilde{b}_1} \phi(y_1) \int_{\tilde{a}_2(y_1)}^{\tilde{b}_2(y_1)} \phi(y_2) \cdots \int_{\tilde{a}_n(y_1, \dots, y_{n-1})}^{\tilde{b}_n(y_1, \dots, y_{n-1})} \phi(y_n) d\mathbf{y} \end{aligned}$$

$$\text{with } \tilde{a}_i(y_1, \dots, y_{i-1}) = \frac{a_i - \sum_{j=1}^{i-1} l_{ij}y_j}{l_{ii}}$$

$$\text{and } (\tilde{b}_i(y_1, \dots, y_{i-1})) = \frac{b_i - \sum_{j=1}^{i-1} l_{ij}y_j}{l_{ii}}$$

$$= \int_{\Phi(\tilde{a}_1)}^{\Phi(\tilde{b}_1)} \int_{\Phi(\tilde{a}_2(\Phi^{-1}(z_1)))}^{\Phi(\tilde{b}_2(\Phi^{-1}(z_1)))} \cdots \int_{\Phi(\tilde{a}_n(\Phi^{-1}(z_1), \dots, \Phi^{-1}(z_{n-1})))}^{\Phi(\tilde{b}_n(\Phi^{-1}(z_1), \dots, \Phi^{-1}(z_{n-1})))} dz(y_i = \Phi^{-1}(z_i))$$

$$= (e_1 - d_1) \int_0^1 (e_2(w_1) - d_2(w_1)) \cdots$$

$$\int_0^1 (e_n(w_1, \dots, w_{n-1}) - d_n(w_1, \dots, w_{n-1})) \int_0^1 dw$$

$$\text{with } z_i = d_i + (e_i - d_i)w_i$$

(2)

Quasi Monte Carlo Method

```
1: procedure MVN( $\mu$ ,  $\Sigma$ ,  $\mathbf{a}$ ,  $\mathbf{b}$ ,  $ns$ ,  $N$ )
2:    $\mathbf{L}$  = cholesky( $\Sigma$ )
3:    $\mathbf{a} = \mathbf{a} - \mu$ ;  $\mathbf{b} = \mathbf{b} - \mu$ 
4:    $T = 0$ ,  $N = 0$ ,  $V = 0$ 
5:    $\mathbf{p}$  = vector of primes less than  $\frac{5n \log n + 1}{4}$ ;  $\mathbf{q} = \sqrt{\mathbf{p}}$ 
6:    $\mathbf{P} = \mathbf{1}_{ns}$ 
7:    $ans = 0$ 
8:   for  $i = 1, \dots, ns$  do
9:      $l_i = 0$ ,  $\Delta \sim U(0, 1)^n$ 
10:    for  $j = 1, \dots, N$  do
11:       $\mathbf{X}[1 : n, j] = (j + 1)\mathbf{q} + \Delta$ 
12:       $\mathbf{X}[1 : n, j] = 2|\mathbf{X}[1 : n, j] - \text{floor}(\mathbf{X}[1 : n, j])| - 1$ 
13:    end for
14:    sample =  $\mathbf{O}_{n, N}$ 
15:     $\mathbf{s}, \mathbf{c}, \mathbf{d}, \mathbf{dc}, \mathbf{P} = \mathbf{0}_N$ 
16:    for  $j = 1, \dots, n$  do
17:      if  $j > 1$  then
18:         $\mathbf{c} = \min(1, \mathbf{c} + \mathbf{X}[j - 1, :] \odot \mathbf{dc})$ 
19:        sample[ $i - 1, 1 : N$ ] =  $\Phi^{-1}(\mathbf{c})$ 
20:         $\mathbf{s} = \text{sample}[1 : i - 1, 1 : N]^T \mathbf{L}[1 : i - 1, j]$ 
21:      end if
22:       $\mathbf{P}^* = \Phi(\frac{\mathbf{b} - \mathbf{s}}{\mathbf{L}[i, j]}) - \Phi(\frac{\mathbf{a} - \mathbf{s}}{\mathbf{L}[i, j]})$ 
23:    end for
24:     $ans += \text{mean}(\mathbf{P})$ 
25:  end for
26:  return  $ans / ns$ 
27: end procedure
```

Algorithm 1: Multivariate Normal Probability with Quasi Monte Carlo Method

Conditioning Approximation

- Mendell and Elston (1974), Kamakura (1989), and Trinh and Genz (2015) exploit Cholesky factors from LDL decomposition rather than dealing with original covariance matrix.
- Bivariate example is follow:

$$\Sigma = \begin{pmatrix} \Sigma_{1,1} & \mathbf{R}^T \\ \mathbf{R} & \hat{\Sigma} \end{pmatrix}, \text{ with } \mathbf{L} = \begin{pmatrix} \mathbf{I}_2 & \mathbf{O} \\ \mathbf{1} : \mathbf{M} & \mathbf{L} \end{pmatrix} \text{ and } \mathbf{D} = \begin{pmatrix} \mathbf{D}_1 & \mathbf{O} \\ \mathbf{O} & \hat{\mathbf{D}}, \end{pmatrix}$$

where $\Sigma_{1,1}$, \mathbf{D}_1 is a 2×2 matrix.

- From $\mathbf{D}_1 = \Sigma_{1,1}$, $\mathbf{M} = \mathbf{R}\mathbf{D}_1^{-1}$, $\hat{\mathbf{D}} = \hat{\Sigma} - \mathbf{M}\mathbf{D}_1\mathbf{M}^T$

$$\begin{aligned} \Phi_n(\mathbf{a}, \mathbf{b}; \mathbf{0}, \Sigma) &= \frac{1}{\sqrt{|\mathbf{D}|}(2\pi)^n} \int_{\alpha_1}^{\beta_1} \int_{\alpha_2}^{\beta_2} e^{-\frac{1}{2}\mathbf{x}_2^T \mathbf{D}_1^{-1} \mathbf{x}_2} \\ &\quad \dots \int_{\alpha_{2k-1}}^{\beta_{2k-1}} \int_{\alpha_{2k}}^{\beta_{2k}} e^{-\frac{1}{2}\mathbf{x}_{2k}^T \mathbf{D}_1^{-1} \mathbf{x}_{2k}} \end{aligned} \quad (3)$$

Conditioning Approximation

Cao et al. (2019) generalizes bivariate method of Trinh and Genz (2015) to d -dimensional. The details are Algorithms 2

```
1: procedure LDL( $\Sigma$ )
2:    $\mathbf{L} \leftarrow \mathbf{I}_m, \mathbf{D} \leftarrow \mathbf{O}_m$ 
3:   for  $i = 1 : d : m - d + 1$  do
4:      $\mathbf{D}[i : i + d - 1, i : i + d - 1] \leftarrow \Sigma[i : i + d - 1, i : i + d - 1]$ 
5:      $\mathbf{L}[i + d : m, i : i + d - 1] \leftarrow \Sigma[i + d : m, i : i + d - 1] \mathbf{D}^{-1}[i : i + d - 1, i : i + d - 1]$ 
6:      $\Sigma[i + d : m, i + d : m] \leftarrow \Sigma[i + d : m, i + d : m] - \mathbf{L}[i + d : m, i : i + d - 1] \mathbf{D}^{-1}[i : i + d - 1, i : i + d - 1] \mathbf{L}[i : i + d - 1, i + d : m]$ 
7:     if  $i + d < m$  then
8:        $\mathbf{D}[i + d : m, i + d : m] \leftarrow \Sigma[i + d : m, i + d : m]$ 
9:     end if
10:  end for
11:  return  $\mathbf{L}$  and  $\mathbf{D}$ 
12: end procedure
```

Algorithm 2: LDL decomposition

Conditioning Approximation

When $s = \frac{m}{d}$ is integer, results of Algorithm 2, \mathbf{L}, \mathbf{D} can be written as

$$\mathbf{L} = \begin{pmatrix} \mathbf{I}_d & \mathbf{O}_d & \cdots & \mathbf{O}_d \\ \mathbf{L}_{2,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \mathbf{I}_d & \mathbf{O}_d \\ \mathbf{L}_{s,1} & \cdots & \mathbf{L}_{s,s-1} & \mathbf{I}_d \end{pmatrix}, \mathbf{D} = \begin{pmatrix} \mathbf{D}_1 & \mathbf{O}_d & \cdots & \mathbf{O}_d \\ \mathbf{O}_d & \ddots & \ddots & \vdots \\ \vdots & \ddots & \mathbf{D}_{s-1} & \mathbf{O}_d \\ \mathbf{O}_d & \cdots & \mathbf{O}_d & \mathbf{D}_s \end{pmatrix}$$

with d -dimensional identity matrix \mathbf{I}_d and d -dimensional zero matrix \mathbf{O}_d and d -dimensional positive-definite matrix $\mathbf{D}_1, \dots, \mathbf{D}_s$.

As in (3), transformation, $Y = LX$ provides m -dimensional multivariate normal probability as the product of s d -dimensional multivariate normal probabilities as

$$\Phi_m(\mathbf{a}, \mathbf{b}; \mathbf{0}, \Sigma) = \int_{\alpha_1}^{\beta_1} \phi_d(\mathbf{y}_1; \mathbf{D}_1) \int_{\alpha_2}^{\beta_2} \phi_d(\mathbf{y}_2; \mathbf{D}_2) \quad (4)$$

$$\cdots \int_{\alpha_s}^{\beta_s} \phi_d(\mathbf{y}_s; \mathbf{D}_s) d\mathbf{y}_s \cdots d\mathbf{y}_2 d\mathbf{y}_1, \quad (5)$$

where $\alpha_i = \mathbf{a}_i - \sum_{j=1}^{i-1} \mathbf{L}_{ij} \mathbf{y}_j$, $\beta_i = \mathbf{b}_i - \sum_{j=1}^{i-1} \mathbf{L}_{ij} \mathbf{y}_j$

Conditioning Approximation

```
1: procedure CMVN( $\Sigma$ ,  $\mathbf{a}$ ,  $\mathbf{b}$ ,  $d$ )
2:    $\mathbf{y} \leftarrow \mathbf{0}$ ,  $P \leftarrow 1$ 
3:   for  $i = 1 : s$  do
4:      $j \leftarrow (i - 1)d$ 
5:      $\mathbf{g} \leftarrow \mathbf{L}[j + 1 : j + d, 1 : j]\mathbf{y}[1 : j]$ 
6:      $\alpha \leftarrow \mathbf{a}[j + 1 : j + d] - \mathbf{g}$ 
7:      $\beta \leftarrow \mathbf{b}[j + 1 : j + d] - \mathbf{g}$ 
8:      $\mathbf{D}' \leftarrow \mathbf{D}[j + 1 : j + d, j + 1 : j + d]$ 
9:      $P \leftarrow P \cdot \Phi_d(\alpha, \beta; \mathbf{0}, \mathbf{D}')$ 
10:     $\mathbf{y}[j + 1 : j + d] \leftarrow E[\mathbf{Y}']$ 
11:   end for
12:   return  $P$  and  $\mathbf{y}$ 
13: end procedure
```

Algorithm 3: d-dimensional conditioning algorithm

Multidimensional Truncated Expectations

- The truncated expectation is expressed as

$$\begin{aligned} E(X^{e_j}) &= \frac{1}{\Phi(\mathbf{a}, \mathbf{b}; \boldsymbol{\mu}, \boldsymbol{\Sigma})} \int_{\mathbf{a}}^{\mathbf{b}} x_j \phi_d(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{x} \\ &= \frac{1}{\Phi(\mathbf{a}, \mathbf{b}; \boldsymbol{\mu}, \boldsymbol{\Sigma})} F_j^d(\mathbf{a}, \mathbf{b}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \end{aligned}$$

- Theore 1 has same form with bivariate version of Trinh and Genz (2015) with $d = 2$
- It allows us to calculate $E[Y']$ in Algorithm 3 with Φ which can be obtained with quasi monte calro method proposed by Genz (1992)

Multidimensional Truncated Expectations

Theorem

Kan and Robotti (2017)

$$F_j^d(\mathbf{a}, \mathbf{b}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mu_j \Phi_d(\mathbf{a}, \mathbf{b}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \mathbf{e}_j^T \boldsymbol{\Sigma} \mathbf{c},$$

where \mathbf{c} is a vector with l th component defined as

$$\begin{aligned} c_l &= \phi_1(a_l; \mu_l, \sigma_l^2) \Phi_{d-1}(\mathbf{a}_{-l}, \mathbf{b}_{-l}; \hat{\boldsymbol{\mu}}^1, \hat{\boldsymbol{\Sigma}}_l) \\ &\quad - \phi_1(b_l; \mu_l, \sigma_l^2) \Phi_{d-1}(\mathbf{a}_{-l}, \mathbf{b}_{-l}; \hat{\boldsymbol{\mu}}^2, \hat{\boldsymbol{\Sigma}}_l) \\ \hat{\boldsymbol{\mu}}_l^1 &= \mu_{-l} + \boldsymbol{\Sigma}_{-l,l} \frac{a_l - \mu_l}{\sigma_l^2}, \hat{\boldsymbol{\mu}}_l^2 = \mu_{-l} + \boldsymbol{\Sigma}_{-l,l} \frac{b_l - \mu_l}{\sigma_l^2}, \\ \hat{\boldsymbol{\Sigma}}_l &= \boldsymbol{\Sigma}_{-l,-l} - \frac{1}{\sigma_l^2} \boldsymbol{\Sigma}_{-l,l} \boldsymbol{\Sigma}_{l,-l} \end{aligned}$$

Multidimensional Conditioning Approximation with Univariate Reordering

Appropriate integration order on conditioning algorithm possibly improves estimation accuracy

- Schervish (1984) : integral with shortest integration interval widths be the outermost integration variables
- Gibson et al. (1994) : variables which have smallest expected values be the outermost integration variables.
Since innermost integrals which have smaller variation have the most influence with this order, overall variance reduces.
- Trinh and Genz (2015) also employs this ordering, and Cao et al. (2019) generalized it to d -dimensional problem.

Multidimensional Conditioning Approximation with Univariate Reordering

```
1: procedure RCMVN( $\Sigma$ , a, b, d)
2:    $y \leftarrow 0$ ,  $C \leftarrow \Sigma$ 
3:   for  $i = 1 : m$  do
4:     if  $i > 1$  then
5:        $y[i-1] \leftarrow \frac{\phi(a') - \phi(b')}{\Phi(b') - \Phi(a')}$ 
6:       end if
7:        $j \leftarrow \operatorname{argmin}_{1 \leq j \leq m} \{ \Phi(\frac{b[j] - C[j, 1:i-1]y[1:i-1]}{\sqrt{\Sigma[j, j] - C[j, 1:i-1]C^T[j, 1:i-1]}}) - \Phi(\frac{a[j] - C[j, 1:i-1]y[1:i-1]}{\sqrt{\Sigma[j, j] - C[j, 1:i-1]C^T[j, 1:i-1]}}) \}$ 
8:        $\Sigma[:, (i, j)] \leftarrow \Sigma[:, (j, i)]; \Sigma[(i, j), :] \leftarrow \Sigma[(j, i), :]$ 
9:        $C[:, (i, j)] \leftarrow C[:, (j, i)]; C[(i, j), :] \leftarrow C[(j, i), :]$ 
10:       $a[(i, j)] = a[(j, i)]$ 
11:       $b[(i, j)] = b[(j, i)]$ 
12:       $C[i, i] \leftarrow \sqrt{\Sigma[i, i] - C[i, 1:i-1]C^T[i, 1:i-1]}$ 
13:       $C[j, i] \leftarrow \frac{\Sigma[j, i] - C[i, 1:i-1]C^T[j, 1:i-1]}{C[i, i]}$ , for  $j = i+1, \dots, m$ 
14:       $a' = \frac{a[i] - C[i, 1:i-1]y[1:i-1]}{C[i, i]}$ 
15:       $b' = \frac{b[i] - C[i, 1:i-1]y[1:i-1]}{C[i, i]}$ 
16:    end for
17:    return CMVN( $\Sigma$ , a, b, d) as in Algorithm 3
18: end procedure
```

Algorithm 4: d-dimensional conditioning algorithm with univariate reordering

Hierarchical-Block Approximation

Hierarchical Cholesky Decomposition

- Hackbusch (2015) proposed hierarchical matrix and its cholesky decomposition method.
- $A = LU$ have the structure

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & O \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{12}^T \\ O & L_{22}^T \end{pmatrix}$$

with lower triangular matrix L_{11}, L_{22} .

- It leads to four tasks:
 - (a) compute L_{11} via Cholesky decomposition of A_{11}
 - (b) compute L_{12} from $L_{21}L_{11}^T = A_{21}$
 - (c) low rank approximation of $L_{12} = UV^T$
 - (d) compute L_{22} via Cholesky decomposition of $A_{22} - L_{21}L_{21}^T$
- We have applied low rank approximation with svd to (c) each block of its decomposition to make implementation efficiently and save storage while accuracy is preserved, i.e.

$$A = UDV^T = \sum_{i=1}^n d_i u_i v_i^T \approx \sum_{i=1}^k d_i u_i v_i^T.$$

Hierarchical Cholesky Decomposition

Hierarchical cholesky decomposition of $n \times n$ matrix into $m \times m$ blocks is implemented like below.

```
1: procedure HCHOL( $A, n, m, rank$ )
2:   for  $i = 1 : \log_2(\frac{n}{m})$  do
3:      $nb = n/2^i$ 
4:      $x = 0, y = nb$ 
5:     for  $j = 1 : 2^{i-1}$  do
6:        $\mathbf{U}, \mathbf{D}, \mathbf{V} = \text{lowrankSVD}(A[xbegin + 1 : xbegin + nb, ybegin + 1 : ybegin + nb], rank)$ 
7:        $\mathbf{A}[x + 1 : x + nb, y + 1 : y + rank] = \mathbf{U}\mathbf{D}$ 
8:        $\mathbf{A}[x + 1 : x + nb, y + rank + 1 : y + nb] = \mathbf{O}$ 
9:        $\mathbf{A}[y + 1 : y + nb, x + 1 : x + rank] = \mathbf{V}\mathbf{D}$ 
10:       $\mathbf{A}[y + 1 : y + nb, x + rank + 1 : x + nb] = \mathbf{O}$ 
11:       $x+ = 2nb, y+ = 2nb$ 
12:    end for
13:  end for
14: end procedure
```

Algorithm 5: Hierarchical cholesky decomposition

The Hierarchical-Block Conditioning Method

- Let $\phi_m(\mathbf{x}; \Sigma)$ be a pdf of the m -dimensional normal distribution $N(\mathbf{0}, \Sigma)$ and $(\mathbf{B}, \mathbf{UV}^T)$ be the hierarchical Cholesky decomposition of the covariance matrix Σ .
- Then,

$$\begin{aligned} \Phi_n(\mathbf{a}, \mathbf{b}; \mathbf{0}, \Sigma) &= \int_{\mathbf{a}'_1}^{\mathbf{b}'_1} \phi_m(\mathbf{x}_1; \mathbf{B}_1 \mathbf{B}_1^T) \\ &\quad \cdots \int_{\mathbf{a}'_r}^{\mathbf{b}'_r} \phi_r(\mathbf{x}_r; \mathbf{B}_r \mathbf{B}_r^T) d\mathbf{x}_r \cdots d\mathbf{x}_1, \end{aligned} \quad (6)$$

where \mathbf{a}' , \mathbf{b}' , $i = 1, \dots, r$, are the corresponding segments of the updated \mathbf{a} and \mathbf{b} .

- Note the probabilities $\Phi_m(\mathbf{a}_i, \mathbf{b}_i; \mathbf{0}, \mathbf{B}_i \mathbf{B}_i^T)$ can be computed using
 1. Quasi-Monte Carlo method (HMVN)
 2. d -dimensional conditioning algorithm (HCMVN)
 3. d -dimensional conditioning algorithm with univariate reordering (HRCMVN)

The Hierarchical-Block Conditioning Method

```
1: procedure HMVN( $a$ ,  $b$ ,  $\Sigma$ ,  $d$ )
2:    $\mathbf{x} \leftarrow \mathbf{0}$  and  $P \leftarrow 1$ 
3:    $[\mathbf{B}, \mathbf{UV}] \leftarrow \text{choldecomp\_hmatrix}(\Sigma)$ 
4:   for  $i = 1 : r$  do
5:      $j \leftarrow (i - 1)m$ 
6:     if  $i > 1$  then
7:        $o_r \leftarrow$  row offset of  $\mathbf{U}_{i-1}\mathbf{V}_{i-1}^T$ 
8:        $o_c \leftarrow$  column offset of  $\mathbf{U}_{i-1}\mathbf{V}_{i-1}^T$ 
9:        $\ell \leftarrow \dim(\mathbf{U}_{i-1}\mathbf{V}_{i-1}^T)$ 
10:       $\mathbf{g} \leftarrow \mathbf{U}_{i-1}\mathbf{V}_{i-1}^T\mathbf{x}[o_c + 1 : o_c + \ell]$ 
11:       $\mathbf{a}[o_r + 1 : o_r + \ell] = \mathbf{a}[o_r + 1 : o_r + \ell] - \mathbf{g}$ 
12:       $\mathbf{b}[o_r + 1 : o_r + \ell] = \mathbf{a}[o_r + 1 : o_r + \ell] - \mathbf{g}$ 
13:    end if
14:     $\mathbf{a}_i \leftarrow \mathbf{a}[j + 1 : j + m]$ 
15:     $\mathbf{b}_i \leftarrow \mathbf{b}[j + 1 : j + m]$ 
16:     $P = P * \Phi_m(\mathbf{a}_i, \mathbf{b}_i; \mathbf{0}, \mathbf{B}_i\mathbf{B}_i^T)$ 
17:     $\mathbf{x}[j + 1 : j + m] \leftarrow \mathbf{B}_i^{-1}E(\mathbf{X}_i)$ 
18:  end for
19: end procedure
```

Algorithm 6: Hierarchical-block conditioning algorithm

Computational Complexity

- Let $M(\cdot)$ be the complexity of the QMC simulation in the given dimension.
- Cao et al. (2019) summarize the time efficiency of the d -dimensional conditioning algorithm as Table 1.

	MVN prob	Trunc exp	Upd limits
HMVN	$\frac{n}{m}M(m)$	$2nM(m) + O(nm^2)$	$O(mn + kn\log(n/m))$
HCMVN	$\frac{n}{d}M(d) + O(m^2n)$	$2nM(d) + O(nd^2)$	$O(mn + kn\log(n/m))$
HRCMVN	$\frac{n}{d}M(d) + O(m^2n)$	$2nM(d) + O(nd^2)$	$O(mn + kn\log(n/m))$

Table 1: Complexity decomposition of the HMVN, HCMVN, and HRCMVN

Block Reordering

Block Reordering

- The value of probability based on n -dimensioned multivariate normal random variable comprises of m multiplications of d -dimensional integrals.
- Recall the RCMVN algorithm(3) : as computing each d -dimensional integral values, integration variables were arranged in order of increasing order of CMVN probability values, from outer to inner
- Permutes the block of LDL-decomposed covariance matrix, in order of RCMVN probability values of each blocks
- Result accuracy and time cost is compared among HMCVN, HCMVN, HRCMVN with/without block reordering.

Block Reordering

```
procedure BLOCKREORDER( $G, \rho, a, b, m, ind$ )  
   $G, \rho, a, b, m, ind$  given,  $P \leftarrow 0$   
  for  $i = 1 : m : n - m + 1$  do  
     $s \leftarrow ind[i : i + m - 1]$   
     $A \leftarrow \rho(G, s)$   
     $a' \leftarrow a[s]$   
     $b' \leftarrow b[s]$   
     $P \leftarrow [P, RCMVN(A, a', b', 1).P]$   
  end for  
  sort( $ind, P, m$ )  
  return  $ind$   
end procedure
```

Algorithm 7: Blockwise reordering

Block Reordering

```
procedure HCMVN_BRO( $a, b, \Sigma, d$ )  
   $\mathbf{x} \leftarrow \mathbf{0}, P \leftarrow 1, \text{ind} \leftarrow [1, \dots, n]$   
   $[\mathbf{B}, \mathbf{UV}] \leftarrow \text{choldecomp\_hmatrix}(\Sigma)$   
   $\mathbf{B} \leftarrow \text{Blockreorder}(G, \rho, a, b, m, \text{ind})$   
  for  $i = 1 : r$  do  
     $j \leftarrow (i - 1)m$   
    if  $i > 1$  then  
       $o_r \leftarrow \text{row offset of } \mathbf{U}_{i-1} \mathbf{V}_{i-1}^T$   
       $o_c \leftarrow \text{column offset of } \mathbf{U}_{i-1} \mathbf{V}_{i-1}^T$   
       $l \leftarrow \text{dim}(\mathbf{U}_{i-1} \mathbf{V}_{i-1}^T)$   
       $\mathbf{g} \leftarrow \mathbf{U}_{i-1} \mathbf{V}_{i-1}^T \mathbf{x}[o_c + 1 : o_c + l]$   
       $\mathbf{a}[o_r + 1 : o_r + l] = \mathbf{a}[o_r + 1 : o_r + l] - \mathbf{g}$   
       $\mathbf{b}[o_r + 1 : o_r + l] = \mathbf{a}[o_r + 1 : o_r + l] - \mathbf{g}$   
    end if  
     $\mathbf{a}_i \leftarrow \mathbf{a}[j + 1 : j + m]$   
     $\mathbf{b}_i \leftarrow \mathbf{b}[j + 1 : j + m]$   
     $P = P * \Phi_m(\mathbf{a}_i, \mathbf{b}_i; \mathbf{0}, \mathbf{B}_i \mathbf{B}_i^T)$   
     $\mathbf{x}[j + 1 : j + m] \leftarrow \mathbf{B}_i^{-1} E[\mathbf{X}_i]$   
  end for  
end procedure
```

Algorithm 8: Hierarchical-block conditioning algorithm with Block Reordering

Numerical Examples

Cholesky Factorization

Data

Exponential covariance matrix, $\Sigma_{ij} = \exp(-\|\mathbf{s}_i - \mathbf{s}_j\|/\beta)$ is set with $\beta = 0.3$. n points, $\mathbf{s}_1, \dots, \mathbf{s}_n$ is evenly distributed over unique square with Morton's order which defined recursively as described in figure 2.

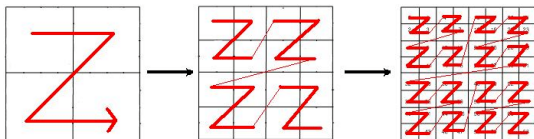


Figure 2: Morton's order (Salem and Arab, 2016)

Methods

1. The *chol* function from **LinearAlgebra** package
 2. The *dpotrf* from **LAPACK** package
 3. Hierarchical cholesky decomposition which suggested by Hackbusch (2015) are implemented.
- With various n , three Cholesky methods are applied and results are below table 2.
 - In low rank approximation at algorithm 5, rank is about $n^{1/4}$.
 - Hierarchical cholesky decomposition provides $\Sigma \approx L_H L_H^T$.
 - Its relative error is defined as $\frac{\|\Sigma - L_H L_H^T\|_2}{\|\Sigma\|_2}$

Cholesky Factorization

n	256	1024	4096	16384
chol	0.001s	0.0097s	0.414s	156.3s
dpotrf	0.0007s	0.0132s	0.431s	154.1s
hierarchical cholesky	0.153s	0.076s	0.916s	37.3s
Error of hierarchical cholesky	1.06e-7	9.97e-7	1.11e-3	1.87e-3

Table 2: Execution times for Cholesky factorization

Results

- Hierarchical cholesky decomposition is more efficient than other classical cholesky method with large dimension.
- Table 2 ensures accuracy of hierarchical cholesky decomposition proposed by Hackbusch (2015).

Methods

1. Classical Monte Carlo(MC) is estimate probabilities from acceptance ratio,
2. Richtmyer Quasi-Monte Carlo(QMC) introduced in the section 2.

Settings

- Varying sample size N and dimension d , two Monte Carlo methods are compared and results are below table 3 and 4.
- Set $\Sigma = I_d$, $\mathbf{a}_i = -\infty$ and $\mathbf{b}_i = 0$, i.e. true probabilities are $1/2^d$ s
- Repeat 20 times.

Multivariate Normal Probabilities

(n, d)	4	8	12	16	20
500	12.2%	56.8%	161.9%	100.0%	100.0%
	0.294ms	0.016ms	0.018ms	0.019ms	0.019ms
1000	9.5%	50.6%	193.8%	100.0%	100.0%
	0.046ms	0.041ms	0.028ms	0.028ms	0.034ms
1500	8.9%	38.7%	150.2%	100.0%	100.0%
	0.055ms	0.046ms	0.042ms	0.048ms	0.045ms
2000	5.4%	26.5%	102.4%	100.0%	100.0%
	0.070ms	0.065ms	0.072ms	0.058ms	0.055ms
2500	5.1%	32.0%	100.1%	100.0%	100.0%
	0.073ms	0.092ms	0.081ms	0.083ms	0.076ms

Table 3: Results for the classical Monte Carlo

Multivariate Normal Probabilities

(n, d)	4	8	12	16	20
500	0.0%	0.0%	0.0%	0.0%	0.0%
	0.058ms	0.003ms	0.006ms	0.006ms	0.011ms
1000	0.0%	0.0%	0.0%	0.0%	0.0%
	0.003ms	0.009ms	0.013ms	0.017ms	0.020ms
1500	0.0%	0.0%	0.0%	0.0%	0.0%
	0.006ms	0.011ms	0.016ms	0.019ms	0.030ms
2000	0.0%	0.0%	0.0%	0.0%	0.0%
	0.011ms	0.012ms	0.013ms	0.025ms	0.036ms
2500	0.0%	0.0%	0.0%	0.0%	0.0%
	0.009ms	0.022ms	0.033ms	0.038ms	0.056ms

Table 4: Results for the Richtmyer Quasi-Monte Carlo

Results

- MC fails even d is not large enough.
- QMC is numerically stable and faster than MC
- All the multivariate normal distribution probabilities required in the next experiments are calculated using QMC.

d-dimensional Conditioning Algorithm without/with Reordering

Settings

- 250 MVN problems with various values of m and d
- $\Sigma = \mathbf{Q}\mathbf{J}\mathbf{Q}^T$ is simulated with $\mathbf{Q} \sim \text{Haar distribution}$ and $\mathbf{J} = \text{diag}(j_i)$ where $j_1, \dots, j_m \sim U(0, 1)$
- Integration limits $a_i = -\infty$ and $b_i \sim (U, m)$ for $i = 1 \dots, m$

Theorem

Stewart (1980) Let the independent vectors x_1, \dots, x_n be distributed $N(0, \sigma^2 \mathbf{I})$. For $j = 1, 2, \dots, n - 1$, let \mathbf{H}_{x_j} be the Householder transformation that reduces x_j to $r_{jj}e_1$, where r_{ij} is obtained in QR decomposition of $[x_1, \dots, x_n]$ Let $\mathbf{H}_j = \text{diag}(\mathbf{I}_{j-1}, \bar{\mathbf{H}}_j)$. Let $\mathbf{D} = \text{diag}(\text{sign}(r_{11}), \dots, \text{sign}(r_{nn}))$. Then the product $\mathbf{Q} = \mathbf{D}\mathbf{H}_1 \dots \mathbf{H}_{n-1}$ follows Haar Distribution.

d-dimensional Conditioning Algorithm without/with Reordering

(m, d)	1	2	4	8	16
Without univariate reordering					
16	3.7%	3.5%	3.6%	3.8%	2.9%
	0.029ms	0.201ms	0.431ms	0.676ms	1.372ms
32	2.4%	2.9%	2.9%	3.3%	2.7%
	0.001ms	0.390ms	0.833ms	1.283ms	2.545ms
64	1.9%	2.1%	2.1%	1.8%	1.9%
	0.004ms	0.762ms	1.686ms	2.545ms	5.004ms
128	1.3%	1.5%	1.3%	1.2%	1.4%
	0.024ms	1.505ms	3.333ms	5.146ms	10.548ms
With univariate reordering					
16	3.3%	3.1%	3.3%	3.6%	2.7%
	0.007ms	0.203ms	0.439ms	0.680ms	1.363ms
32	2.3%	2.6%	2.6%	3.2%	2.6%
	0.004ms	0.393ms	0.841ms	1.289ms	2.544ms
64	2.0%	2.1%	2.1%	1.9%	1.9%
	0.014ms	0.773ms	1.695ms	2.552ms	5.022ms
128	1.2%	1.5%	1.4%	1.2%	1.4%
	0.097ms	1.593ms	3.462ms	5.268ms	10.7861ms

Table 5: Errors and execution times of the d-dimensional conditioning method

Results

- Estimated value is compared with approximated value obtained via quasi monte carlo method with a sample size of 10^4 , which ensures error below 10^{-4}
- Estimation error tended to decrease as d increases with each m since larger d implies less discarded correlation information.
- Spent time grows to a linear fashion with m while it grows exponentially with d .

Hierarchical-Block Approximations

Methods

- M1, HMVN(): Calculate multivariate normal probabilities using hierarchical-block approximation
- M2, HCMVN(): Calculate multivariate normal probabilities using hierarchical-block conditioning approximation
- M3, HRCMVN(): Calculate multivariate normal probabilities using hierarchical-block conditioning approximation with univariate reordering

Data

1. Constant covariance matrix: $k(x_i, x_j) = \theta + (1 - \theta)\delta_{ij}$ for some $|\theta| < 1$.
2. 1D exponential covariance matrix: $k(x_i, x_j) = \exp(-d_{ij}/\beta)$ for some $\beta > 0$, where d_{ij} is the distance between x_i and x_j .

Settings

- Simulation size = 20
- Integration limits $a_i = -\infty$ and $b_i \sim (U, n)$ for $i = 1 \dots, n$
- $\theta = 0.7$, $d_{ij} = 1$, and $\beta = 10$ as in Cao et al. (2019)
- Fix $d = 4$ for HCMVN and HRCMVN.

Table 6, Figure 3 and 4 are errors and execution times under the constant covariance structure and 1D exponential covariance structure respectively.

Hierarchical-Block Approximations

m	16			32			64		
n	256	512	1024	256	512	1024	256	512	1024
Constant covariance structure									
M1	8.22%	7.11%	8.66%	8.94%	7.88%	6.68%	10.58%	8.05%	9.78%
M2	8.37%	7.08%	8.60%	8.91%	7.77%	6.61%	10.58%	8.26%	9.91%
M3	8.51%	7.10%	8.70%	9.51%	7.92%	7.00%	10.68%	7.94%	9.63%
1D exponential covariance matrix									
M1	2.87%	0.00%	0.01%	0.07%	1.31%	0.00%	2.65%	0.27%	0.57%
M2	3.28%	0.01%	0.90%	0.07%	1.31%	0.01%	2.65%	0.28%	0.57%
M3	4.73%	0.09%	2.11%	2.17%	1.90%	0.16%	3.72%	1.25%	0.66%

Table 6: Relative errors under the constant covariance structure and 1D exponential covariance structure

Hierarchical-Block Approximations- Execution Time i

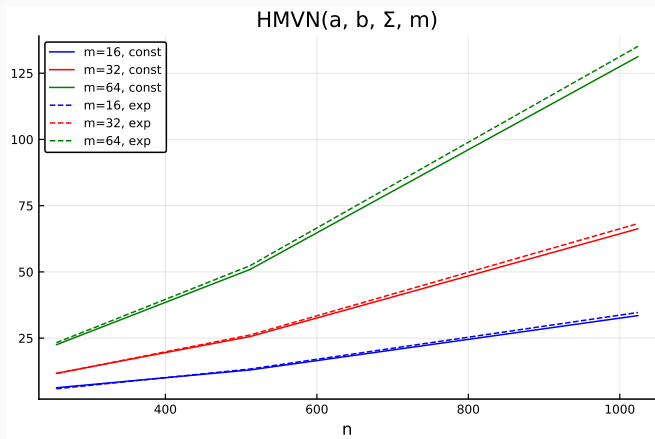
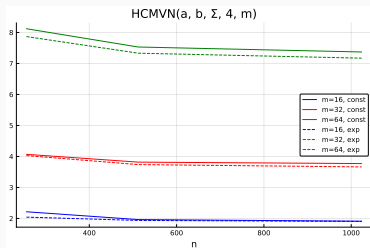
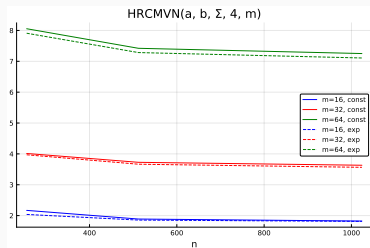


Figure 3: Execution time (seconds) for the hierarchical-block approximation

Hierarchical-Block Approximations- Execution Time ii



(a) HCMVN



(b) HRCMVN

Figure 4: Execution time (seconds) for the hierarchical-block conditioning approximation

Conclusion

Conclusion

- d -dimensional conditioning and block reordering scheme improved accuracy under the exponential covariance structure, with negligible amount of computing time and cost.
- The hierarchical decomposition method is effective approach for the case of smooth covariance function, such as exponential covariance structure.
- Numerical results were not perfectly homogenous with these of Cao et al. (2019) mainly due to distinct computing environment.
- Discussions:
 1. Deal with $m \nmid n$ case
 2. Efficient ways of estimating truncated expectations of MVN random variables.

References

- Cao, J., Genton, M. G., Keyes, D. E., and Turkiyyah, G. M. (2019). Hierarchical-block conditioning approximations for high-dimensional multivariate normal probabilities. *Statistics and Computing*, 29(3):585–598.
- Genton, M. G., Keyes, D. E., and Turkiyyah, G. (2018). Hierarchical decompositions for the computation of high-dimensional multivariate normal probabilities. *Journal of Computational and Graphical Statistics*, 27(2):268–277.
- Genz, A. (1992). Numerical computation of multivariate normal probabilities. *Journal of computational and graphical statistics*, 1(2):141–149.

- Genz, A. and Bretz, F. (2009). *Computation of multivariate normal and t probabilities*, volume 195. Springer Science & Business Media.
- Gibson, G., Glasbey, C., and Elston, D. (1994). Monte carlo evaluation of multivariate normal integrals and sensitivity to variate ordering. *Advances in Numerical Methods and Applications*, pages 120–126.
- Hackbusch, W. (2015). *Hierarchical matrices: algorithms and analysis*, volume 49. Springer.
- Kamakura, W. A. (1989). The estimation of multinomial probit models: A new calibration algorithm. *Transportation Science*, 23(4):253–265.
- Kan, R. and Robotti, C. (2017). On moments of folded and truncated multivariate normal distributions. *Journal of Computational and Graphical Statistics*, 26(4):930–934.

- Mendell, N. R. and Elston, R. (1974). Multifactorial qualitative traits: genetic analysis and prediction of recurrence risks. *Biometrics*, pages 41–57.
- Salem, F. K. A. and Arab, M. A. (2016). Comparative study of space filling curves for cache oblivious tu decomposition. *arXiv preprint arXiv:1612.06069*.
- Schervish, M. J. (1984). Algorithm as 195: Multivariate normal probabilities with error bound. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 33(1):81–94.
- Stewart, G. W. (1980). The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM Journal on Numerical Analysis*, 17(3):403–409.
- Trinh, G. and Genz, A. (2015). Bivariate conditioning approximations for multivariate normal probabilities. *Statistics and Computing*, 25(5):989–996.