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

Introduction

Introduction

- The computation of the multivariate normal (MVN) probability

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

  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 $\mathbf{\Sigma}$.

Motivation

The methods are based on

1. the bivariate conditioning method (Trinh and Genz, 2015) and
2. the hierarchical QMC method (Genton et al., 2018).

Multidimensional Conditioning Approximations

Quasi Monte Carlo Method

- MonteCarlo 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 $\mathbf{\Delta} \sim U[0,1]^n$,

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

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

  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

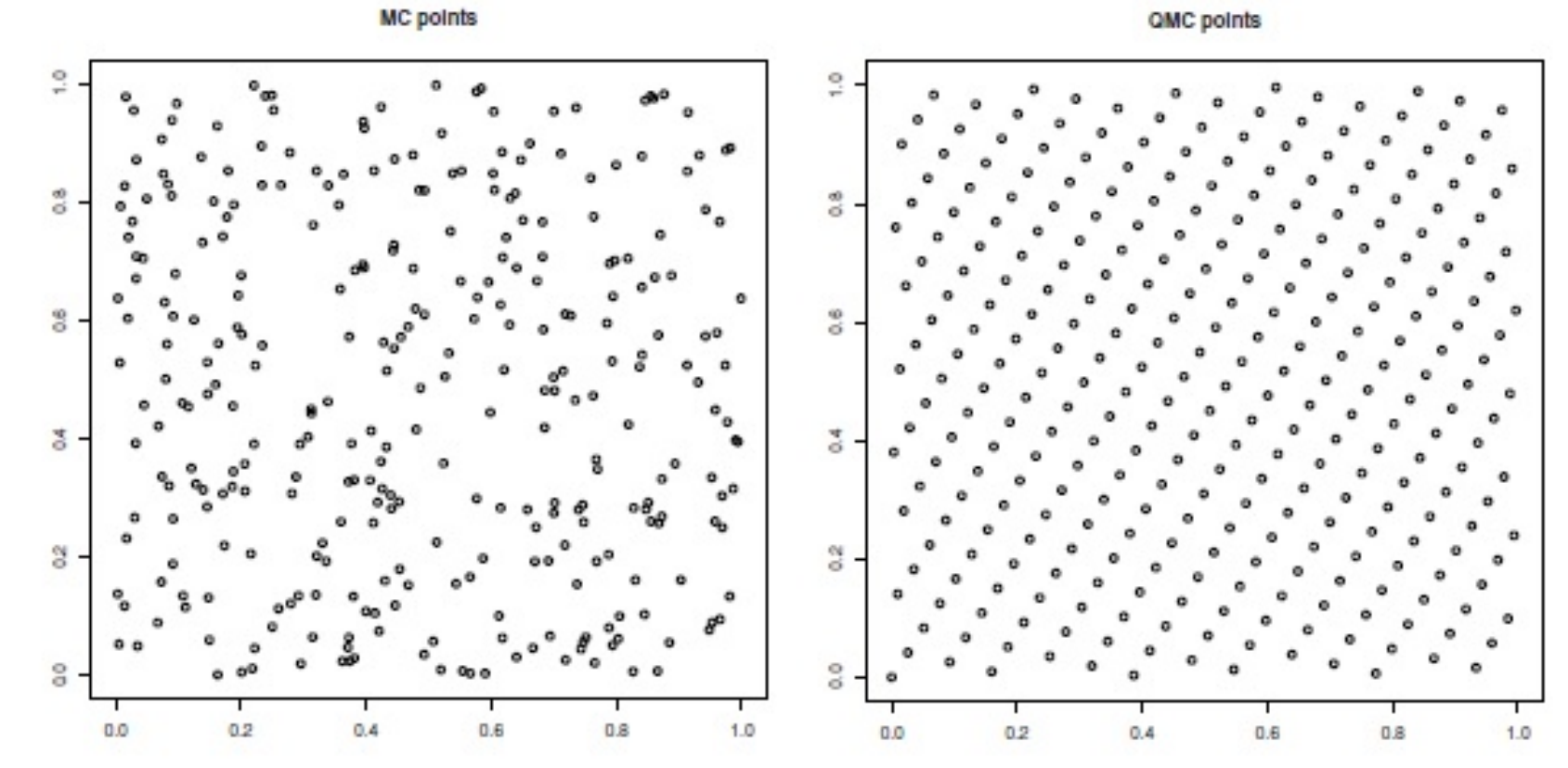


Figure 1: Comparison of MC and QMC sample points(Genz and Bretz, 2009)

Quasi Monte Carlo Method

$$\begin{aligned}
\mathbf{\Phi}_n(\mathbf{a} \le \mathbf{x} \le \mathbf{b}; \mathbf{\Sigma}) &= \mathbf{\Phi}_n(a \le \mathbf{L}\mathbf{y} \le \mathbf{b}; I_n) \\
&= \int_{a_1 \le l_{11} y_1 \le b_1} \phi(y_1) \cdots \int_{a_n \le \mathbf{l}_n^t \mathbf{y} \le b_n} \phi(y_n) d\mathbf{y} \\
&= \int_{\tilde{a}_1}^{\tilde{b}_1} \phi(y_1) \int_{\tilde{a}_2(y_1)}^{\tilde{b}_2(y_1)} \phi(y_2) \cdots \int_{\tilde{a}_n(y_1, \cdots, y_{n-1})}^{\tilde{b}_n(y_1, \cdots, y_{n-1})} \phi(y_n) d\mathbf{y}
\end{aligned}$$

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

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

$$\begin{aligned}
&= \int_{\Phi(\tilde{a}_1)}^{\Phi(\tilde{b}_1)} \int_{\Phi(\tilde{a}_2(\Phi^{-1}(z_1)))}^{\Phi(\tilde{b}_2(\Phi^{-1}(z_1)))} \cdots \int_{\Phi(\tilde{a}_n(\Phi^{-1}(z_1), \cdots, \Phi^{-1}(z_{n-1})))}^{\Phi(\tilde{b}_n(\Phi^{-1}(z_1), \cdots, \Phi^{-1}(z_{n-1})))} d\mathbf{z} (y_i = \Phi^{-1}(z_i)) \\
&= (e_1 - d_1) \int_0^1 (e_2(w_1) - d_2(w_1)) \cdots \\
&\int_0^1 (e_n(w_1, \cdots, w_{n-1}) - d_n(w_1, \cdots, w_{n-1}))) \int_0^1 d\mathbf{w}
\end{aligned}$$

$$\text{with } z_i = d_i + (e_i - d_i) w_i \qquad (2)$$

Quasi Monte Carlo Method

1: **procedure** MVN($\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, $\mathbf{a}$, $\mathbf{b}$, ns, N)
2: $\mathbf{L} = \text{cholesky}(\boldsymbol{\Sigma})$
3: $\mathbf{a} = \mathbf{a} - \boldsymbol{\mu}$; $\mathbf{b} = \mathbf{b} - \boldsymbol{\mu}$
4: $T = 0$, $N = 0$, $V = 0$
5: $\mathbf{p} = \text{vector of primes less than } \frac{5n \log n + 1}{4}$; $\mathbf{q} = \sqrt{\mathbf{p}}$
6: $\mathbf{P} = \mathbf{1}_{ns}$
7: $ans = 0$
8: **for** $i = 1, \cdots, ns$ **do**
9: $I_i = 0$, $\boldsymbol{\Delta} \sim U(0,1)^n$
10: **for** $j = 1, \cdots, N$ **do**
11: $\mathbf{X}[1:n, j] = (j+1)\mathbf{q} + \boldsymbol{\Delta}$
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** $j = 1, \cdots, n$ **do**
17: **if** $j > 1$ **then**
18: $c = \min(1, c + X[j-1, :] \odot dc)$
19: **sample**$[i-1, 1:N] = \Phi^{-1}(c)$
20: $s =$ **sample**$[1:i-1, 1:N]^T L[1:i-1, i]$
21: **end if**
22: $\mathbf{P}* = \Phi(\frac{b-s}{L[i,i]}) - \Phi(\frac{a-s}{L[i,i]})$
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. Biviarate example is follow.

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

,where $\mathbf{\Sigma}_{1,1}, \mathbf{D}_1$ is a 2×2 matrix. From $\mathbf{D}_1 = \mathbf{\Sigma_{1,1}}$, $\mathbf{M} = \mathbf{R}\mathbf{D}_1^{-1}$, $\widehat{\mathbf{D}} = \hat{\mathbf{\Sigma}} - \mathbf{M}\mathbf{D}_1\mathbf{M}^T$

$$\begin{aligned} \mathbf{\Phi}_n(\mathbf{a}, \mathbf{b}; \mathbf{0}, \mathbf{\Sigma}) = \frac{1}{\sqrt{|\mathbf{D}|(2\pi)^n}} \int_{\alpha_1}^{\beta_1} \int_{\alpha_2}^{\beta_2} e^{-\frac{1}{2}\mathbf{x_2}^T \mathbf{D}_1^{-1} \mathbf{x}_2} \\ \cdots \int_{\alpha_{2k-1}}^{\beta_{2k-1}} \int_{\alpha_{2k}}^{\beta_{2k}} e^{-\frac{1}{2}\mathbf{x_{2k}}^T \mathbf{D}_1^{-1} \mathbf{x}_{2k}} \end{aligned} \tag{3}$$

Conditioning Approximation

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

1: **procedure** LDL($\mathbf{\Sigma}$)
2: $\mathbf{L} \leftarrow \mathbf{I}_m, \mathbf{D} \leftarrow \mathbf{O}_m$
3: **for** $i = 1 : d : m - d + 1$ **do**
4: $\mathbf{D}[i : i + d - 1, i : i + d - 1] \leftarrow \mathbf{\Sigma}[i : i + d - 1, i : i + d - 1]$
5: $\mathbf{L}[i + d : m, i : i + d - 1] \leftarrow \mathbf{\Sigma}[i + d : m, i : i + d - 1]\mathbf{D}^{-1}[i : i + d - 1, i : i + d - 1]$
6: $\mathbf{\Sigma}[i + d : m, i + d : m] \leftarrow \mathbf{\Sigma}[i + d : m, i + d : m] - \mathbf{L}[i + d : m, i : i + d - 1]\mathbf{D}^{-1}[i : i + d - 1, i : i + d - 1]\mathbf{L}[i : i + d - 1, i + d : m]$
7: **if** $i + d < m$ **then**
8: $\mathbf{D}[i + d : m, i + d : m] \leftarrow \mathbf{\Sigma}[i + d : m, i + d : m]$
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 identitiy matrix $\mathbf{I}_d$ and d-dimensional zero matrix $\mathbf{O}_d$ and d-dimensional positive-definite matrix $\mathbf{D}_1, \cdots, \mathbf{D}_s$. As in (3), tranformation, $Y = LX$ provides m-dimensional multivariate normal prabability as the product of s d-dimensional multivariate normal probabilities as below.

$$\mathbf{\Phi}_{\boldsymbol{m}}(\mathbf{a}, \mathbf{b}; \mathbf{0}, \mathbf{\Sigma}) = \int_{\alpha_1}^{\beta_1} \phi_d(\mathbf{y}_1; \mathbf{D}_1) \int_{\alpha_2}^{\beta_2} \phi_d(\mathbf{y}_2; \mathbf{D}_2) \cdots \int_{\alpha_s}^{\beta_s} \phi_d(\mathbf{y}_s; \mathbf{D}_s) d\mathbf{y}_s \cdots d\mathbf{y}_2 d\mathbf{y}_1 \quad (4)$$

,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($\mathbf{\Sigma}, \mathbf{a}, \mathbf{b}, d$)
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: $P \leftarrow P \cdot \mathbf{\Phi}_d(\alpha, \beta; \mathbf{0}, \mathbf{D}')$
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

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

Theorem
(Kan and Robotti, 2017)

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

,where c is a vector with lth component defined as

$$c_l = \phi_1(a_l; \mu_l, \sigma_l^2) \Phi_{d-1}(\mathbf{a}_{-l}, \mathbf{b}_{-l}; \hat{\boldsymbol{\mu}}^1, \hat{\boldsymbol{\Sigma}}_l) - \phi_1(b_l; \mu_l, \sigma_l^2) \Phi_{d-1}(\mathbf{a}_{-l}, \mathbf{b}_{-l}; \hat{\boldsymbol{\mu}}^2, \hat{\boldsymbol{\Sigma}}_l)$$

$$\hat{\boldsymbol{\mu}}_l^1 = \mu_{-l} + \boldsymbol{\Sigma}_{-l,l} \frac{a_l - \mu_l}{\sigma_l^2}, \hat{\boldsymbol{\mu}}_l^2 = \mu_{-l} + \boldsymbol{\Sigma}_{-l,l} \frac{b_l - \mu_l}{\sigma_l^2},$$

$$\hat{\boldsymbol{\Sigma}}_l = \boldsymbol{\Sigma}_{-l,-l} - \frac{1}{\sigma_l^2} \boldsymbol{\Sigma}_{-l,l} \boldsymbol{\Sigma}_{l,-l}$$

Theorem 1 has same form with bivariate version of Trinh and Genz (2015) with $d = 2$ and it allows us to calculate $E[Y']$ in Algorithm 3 with $\boldsymbol{\Phi}$ which can be obtained with quasi monte calro method proposed by Genz (1992)

Multidimensional Conditioning Approximation with Univariate Reordering

Appropriate integration order on conditioning algorithm possibly improves estiation 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

procedure RCMVN($\mathbf{\Sigma}$, $\mathbf{a}$, $\mathbf{b}$, d)

  $\mathbf{y} \leftarrow \mathbf{0}$, $\mathbf{C} \leftarrow \mathbf{\Sigma}$

  for $i = 1 : m$ **do**

    if $i > 1$ **then**

      $\mathbf{y}[i-1] \leftarrow \frac{\phi(a') - \phi(b')}{\Phi(b') - \Phi(a')}$

    end if

    $j \leftarrow \text{argmin}_{i \le j \le m} \{ \Phi(\frac{\mathbf{b}[j] - \mathbf{C}[j, 1:i-1]\mathbf{y}[1:i-1]}{\sqrt{\mathbf{\Sigma}[j,j] - \mathbf{C}[j, 1:i-1]\mathbf{C}^T[j, 1:i-1]}}) - \Phi(\frac{\mathbf{a}[j] - \mathbf{C}[j, 1:i-1]\mathbf{y}[1:i-1]}{\sqrt{\mathbf{\Sigma}[j,j] - \mathbf{C}[j, 1:i-1]\mathbf{C}^T[j, 1:i-1]}}) \}$

    $\mathbf{\Sigma}[:, (i, j)] \leftarrow \mathbf{\Sigma}[:, (j, i)]$; $\mathbf{\Sigma}[(i, j), :] \leftarrow \mathbf{\Sigma}[(j, i), :]$

    $\mathbf{C}[:, (i, j)] \leftarrow \mathbf{C}[:, (j, i)]$; $\mathbf{C}[(i, j), :] \leftarrow \mathbf{C}[(j, i), :]$

    $\mathbf{a}[(i, j)] = \mathbf{a}[(j, i)]$

    $\mathbf{b}[(i, j)] = \mathbf{b}[(j, i)]$

    $\mathbf{C}[i, i] \leftarrow \sqrt{\mathbf{\Sigma}[i, i] - \mathbf{C}[i, 1 : i-1]\mathbf{C}^T[i, 1 : i-1]}$

    $\mathbf{C}[j, i] \leftarrow \frac{\mathbf{\Sigma}[j, i] - \mathbf{C}[i, 1:i-1]\mathbf{C}^T[j, 1:i-1]}{\mathbf{C}[i, i]}$, for $j = i+1, \cdots, m$

    $a' = \frac{\mathbf{a}[i] - \mathbf{C}[i, 1:i-1]y[1:i-1]}{\mathbf{C}[\mathbf{i}, \mathbf{i}]}$

    $b' = \frac{\mathbf{b}[i] - \mathbf{C}[i, 1:i-1]y[1:i-1]}{\mathbf{C}[\mathbf{i}, \mathbf{i}]}$

  end for

  return CMVN($\mathbf{\Sigma}$, $\mathbf{a}$, $\mathbf{b}$, d) as in Algorithm 3

end procedure

Algorithm 4: d-dimensional conditioning algorithm with univariate reordering

Hierarchical-Block Approximation

Hierarchical Cholesky Decomposition

Hackbusch (2015) proposed hiarchical 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

Hierachical 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: **U**, **D**, **V** = $lowrankSVD(A[xbegin + 1 : xbegin + nb, ybegin + 1 : ybegin + nb], rank)$
7: $\mathbf{A}[x + 1 : x + nb, y + 1 : y + rank] = \mathbf{UD}$
8: $\mathbf{A}[x + 1 : x + nb, y + rank + 1 : y + nb] = \mathbf{O}$
9: $\mathbf{A}[y + 1 : y + nb, x + 1 : x + rank] = \mathbf{VD}$
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: Hierachical cholesky decomposition

The Hierarchical-Block Conditioning Method

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

$$\Phi_n(\mathbf{a}, \mathbf{b}; \mathbf{0}, \mathbf{\Sigma}) = \int_{\mathbf{a}_1'}^{\mathbf{b}_1'} \phi_m(\mathbf{x}_1; \mathbf{B}_1\mathbf{B}_1^T) \cdots \int_{\mathbf{a}_r'}^{\mathbf{b}_r'} \phi_r(\mathbf{x}_r; \mathbf{B}_r\mathbf{B}_r^T) d\mathbf{x}_r \cdots d\mathbf{x}_1. \tag{5}$$

,where $\mathbf{a}'$, $\mathbf{b}'$, $i = 1, \cdots, r$, 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`, Method 1 in Cao et al. (2019))
2. d-dimensional conditioning algorithm (`HCMVN`, Method 2 in Cao et al. (2019))
3. d-dimensional conditioning algorithm with univariate reordering (`HRCMVN`, Method 3 in Cao et al. (2019)).

These methods are more effective and easily parallelizable than the classical methods.

The Hierarchical-Block Conditioning Method

1: **procedure** HMVN(a, b, $\mathbf{\Sigma}$, d)
2: $\mathbf{x} \leftarrow \mathbf{0}$ and $P \leftarrow 1$
3: $[\mathbf{B}, \mathbf{UV}] \leftarrow$ choldecomp_hmatrix($\mathbf{\Sigma}$)
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: $l \leftarrow \dim(\mathbf{U}_{i-1}\mathbf{V}_{i-1}^T)$
10: $\mathbf{g} \leftarrow \mathbf{U}_{i-1}\mathbf{V}_{i-1}^T\mathbf{x}[o_c + 1 : o_c + l]$
11: $\mathbf{a}[o_r + 1 : o_r + l] = \mathbf{a}[o_r + 1 : o_r + l] - \mathbf{g}$
12: $\mathbf{b}[o_r + 1 : o_r + l] = \mathbf{a}[o_r + 1 : o_r + l] - \mathbf{g}$
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

$M(\cdot)$ denotes the complexity of the QMC simulation in the given dimension. Table 1 shows that the time efficiency of the d-dimensional conditioning algorithm mainly comes from lowering the dimension in which the QMC simulation is performed.

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

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

- The updating cost is independent of the method.
- The complexity of the univariate reordering is $O(m^2 n)$, the same as the complexity of computing the MVN probabilities in `HCMVN`
- Since `HCMVN` and `HRCMVN` perform the QMC simulation in d-dimensions, these two methods are not greatly affected by the choice of m.

Block Reordering

Block Reordering

- The cdf value for n-dimensioned multivariate normal 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 `HMVN`, `HCMVN`, `HRCMVN` with/without block reordering.

Block Reordering

```
procedure BLOCKREORDER(G, ρ, a, b, m, ind)
    G, ρ, a, b, m, ind given, P ← 0
    for i = 1 : m : n − m + 1 do
        s ← ind[i : i + m − 1]
        A ← ρ(G, s)
        a′ ← a[s]
        b′ ← b[s]
        P ← [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, Σ, d)
  $\mathbf{x} \leftarrow \mathbf{0}$, $P \leftarrow 1$, ind $\leftarrow [1, \ldots, n]$
  $[\mathbf{B}, \mathbf{UV}] \leftarrow$ choldecomp_hmatrix(Σ)
  $\mathbf{B} \leftarrow$ Blockreorder(G, ρ, a, b, m, ind)
  for $i = 1 : r$ **do**
    $j \leftarrow (i-1)m$
    if $i > 1$ **then**
      $o_r \leftarrow$ row offset of $\mathbf{U}_{i-1}\mathbf{V}_{i-1}^T$
      $o_c \leftarrow$ column offset of $\mathbf{U}_{i-1}\mathbf{V}_{i-1}^T$
      $l \leftarrow \dim(\mathbf{U}_{i-1}\mathbf{V}_{i-1}^T)$
      $\mathbf{g} \leftarrow \mathbf{U}_{i-1}\mathbf{V}_{i-1}^T\mathbf{x}[o_c+1 : o_c+l]$
      $\mathbf{a}[o_r+1 : o_r+l] = \mathbf{a}[o_r+1 : o_r+l] - \mathbf{g}$
      $\mathbf{b}[o_r+1 : o_r+l] = \mathbf{a}[o_r+1 : o_r+l] - \mathbf{g}$
    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

- The *chol* function from **LinearAlgebra** package
- The *dpotrf* from **LAPACK** package
- Hierarchical cholesky decomposition which suggested by Hackbusch (2015) are implemented.

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

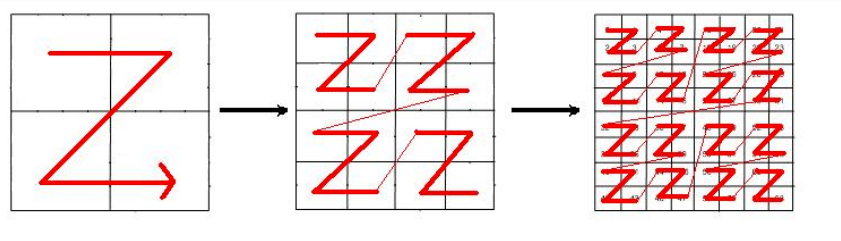


Figure 2: Morton's order(Salem and Arab, 2016)

Cholesky Factorization

With various n, three Cholesky methods are applied and results are below table 2. In low rank approximation at algorithm 5, rank is about $n^{1/4}$.

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: Excution times for Cholesky factorization

- Hierarchical cholesky decomposition is more efficient than other classical cholesky method with large dimension.
- Hierarchical cholesky decomposition provides $\mathbf{\Sigma} \approx L_H L_H^T$. Its relative error is defined as $\frac{\|\mathbf{\Sigma} - L_H L_H^T\|_2}{\|\mathbf{\Sigma}\|_2}$
- Table 2 ensures accuracy of hierarchical cholesky decomposition proposed by Hackbusch (2015).

Multivariate Normal Probabilities

- We implement `mvn`, the function that calculate multivariate normal probabilities using Richtmyer QMC method introduced in subsection 1.
- Varying sample size N and dimension d, two Monte Carlo methods are compared and results are below table 3 and 4.
- We generate N samples from $N(0, I_d)$ and set $\mathbf{a}_i = -\infty$ and $\mathbf{b}_i = 0$, i.e. true probabilities are $1/2^d$s, and repeat 20 times.
- To summarize the results, QMC is superior to MC in every criterion.

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% 0.058ms	0.0% 0.003ms	0.0% 0.006ms	0.0% 0.006ms	0.0% 0.011ms
1000	0.0% 0.003ms	0.0% 0.009ms	0.0% 0.013ms	0.0% 0.017ms	0.0% 0.020ms
1500	0.0% 0.006ms	0.0% 0.011ms	0.0% 0.016ms	0.0% 0.019ms	0.0% 0.030ms
2000	0.0% 0.011ms	0.0% 0.012ms	0.0% 0.013ms	0.0% 0.025ms	0.0% 0.036ms
2500	0.0% 0.009ms	0.0% 0.022ms	0.0% 0.033ms	0.0% 0.038ms	0.0% 0.056ms

Table 4: Results for the Richtmyer Quasi-Monte Carlo

d-dimensional Conditioning Algorithm without/with Reordering

Theorem

Stewart (1980) Let the independent vectors $x_1, \cdots, x_n$ *be distributed* $N(0, \sigma^2 \mathbf{I})$. *For* $j = 1, 2, \cdots, n-1$, *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* $[x_1, \cdots, x_n]$ *Let* $\mathbf{H}_j = diag(\mathbf{I}_{j-1}, \bar{\mathbf{H}}_j)$. *Let* $\mathbf{D} = diag(sign(r_{11}), \cdots, sign(r_{nn}))$. *Then the product* $\mathbf{Q} = \mathbf{D}\mathbf{H_1} \cdots \mathbf{H_{n-1}}$ *follows Haar Distribution.*

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

d-dimensional Conditioning Algorithm without/with Reordering

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

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

Estimation error tended to decrease as d increases with each m since lager d implers 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 univarite 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.

Hierarchical-Block Approximations

Settings

- Simulation size $= 20$
- Integration limits $a_i = -\infty$ and $b_i \sim (U, n)$ for $i = 1 \cdots, n$
- $\theta = 0.7$, $d_{ij} = 1$, 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

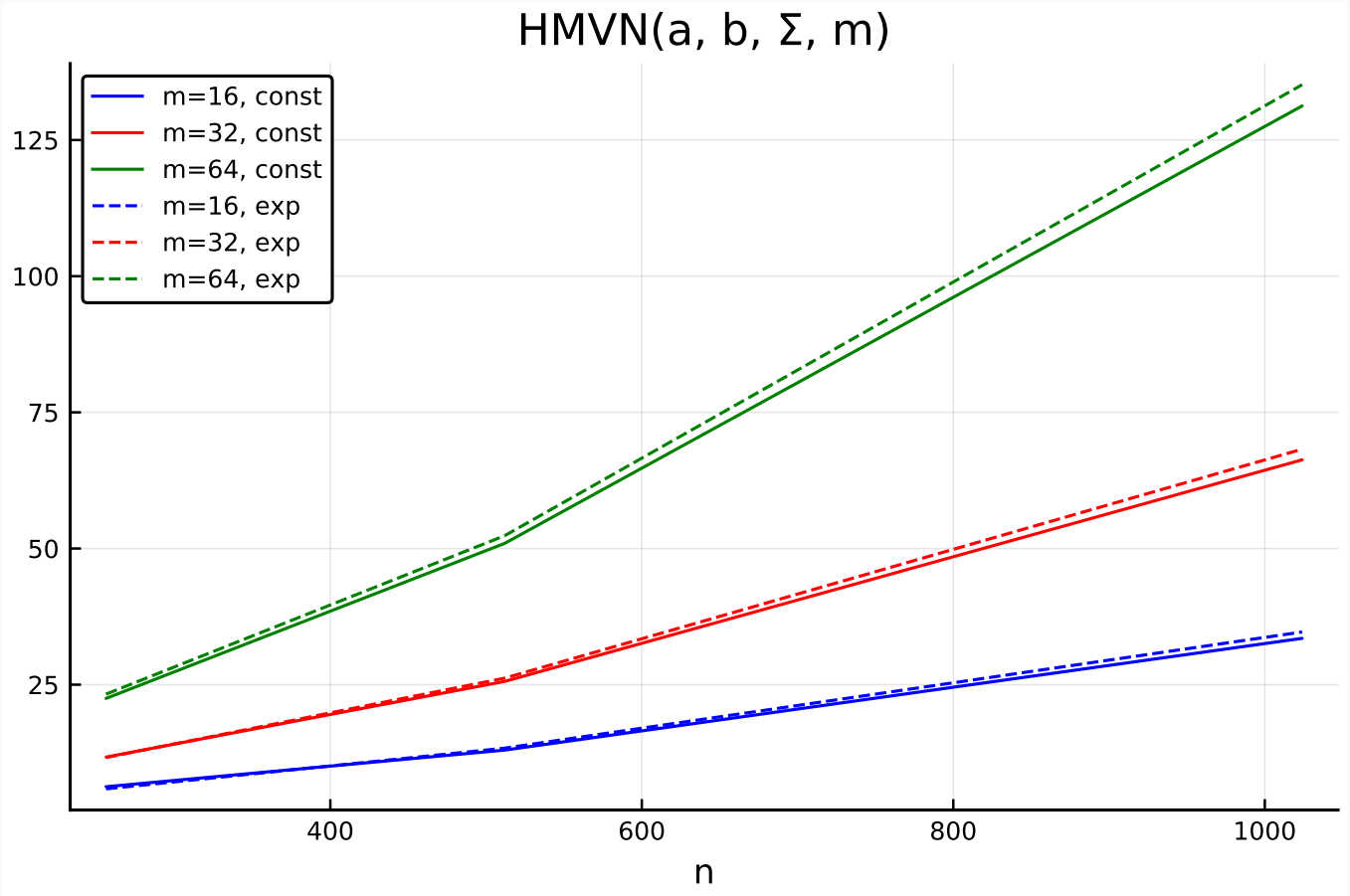
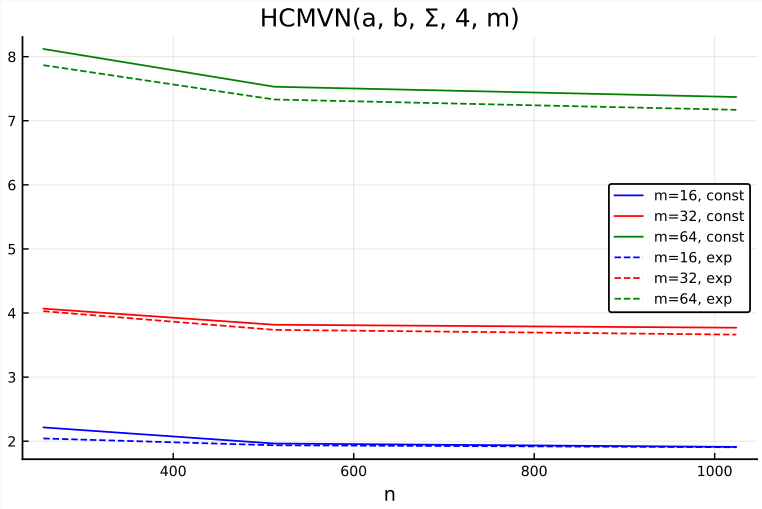
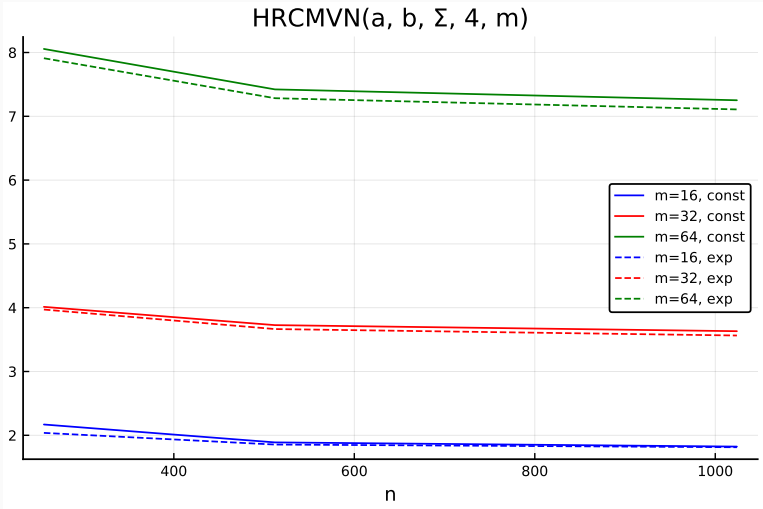


Figure 3: Execution time (seconds) for the hierarchical-block approximation

Hierarchical-Block Approximations- Execution Time ii



(a) `HCMVN`



(b) `HRCMVN`

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

The excution times of `HCMVN` and `HRCMVN` are significantly smaller than of `HMVN` even their performances are similar

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.

References ii

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.

References iii

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.