Advanced Statistical Computing Proejct

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

Hyunseok Yang, Kyeongwon Lee, Songhyun Kim

Department of Statistics, Seoul National University

December 8, 2019

1 Introduction

The computation of multivariate normal probability appears various fields. For instance, the inferences based on the central limit theorem, which holds when the sample size is large enough, is widely used in the social sciences and engineering as well as in the natural sciences. Recently, the dimensionality of data and models has been grown significantly, and in this respect, so does a need for the methodology to efficiently calculate high-dimensional multivariate normal probability.

Cao, Genton, Keyes, and Turkiyyah (2019) proposes new approaches to approximate high-dimensional multivariate normal probability

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

using the hierarchical matrix $\mathcal{H}$ (Hackbusch, 2015) for the covariance matrix $\mathbf{\Sigma}$. The methods are based on two state-of-arts methods, among others, are the bivariate conditioning method (Trinh & Genz, 2015) and the hierarchical Quasi-Monte Carlo method (Genton, Keyes, & Turkiyyah, 2018). Specifically, Cao et al. (2019) generalize the bivariate conditioning method to a d-dimension and combine it with the hierarchical representation of the covariance matrix.

2 Multidimensional Conditioning Approximations

2.1 Conditioning Approximation

We can exploit Cholesky factors from LDL decomposition rather than dealing with original covariance matrix. Mendell and Elston (1974) and Kamakura (1989) developed conditioning method to calculated cdf of multivariate truncated normal distribution. Trinh and Genz (2015) employ bivariate blocking method for efficient calculation while accuracy is preserved.

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

,where $\mathbf{\Sigma}_{1,1}, \mathbf{D}_1$ is a 2×2 matrix. From $\mathbf{D}_1 = \mathbf{\Sigma_{1,1}}$, $\mathbf{M} = \mathbf{R}\mathbf{D}_1^{-1}$, $\widehat{\mathbf{D}} = \hat{\mathbf{\Sigma}} - \mathbf{M}\mathbf{D}_1\mathbf{M}^T$, we can obtain bivariate LDL decomposition of $\mathbf{\Sigma}$ inductively.
With transformation $\mathbf{y} = L\mathbf{x}$, $\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}$ is tranformed to $a_j - \sum_{m=1}^{j-1} l_{jm}x_m = \alpha_j \leq x_j \leq b_j - \sum_{m=1}^{j-1} l_{jm}x_m = \beta_j$ for $j = 1, \cdots, n$. Then, with $k = \frac{n}{2}$ and $\mathbf{x}_{2k} = (x_{2k-1}, x_{2k})^T$

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

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

Algorithm 1 LDL decomposition

procedure LDL($\mathbf{\Sigma}$)
  $\mathbf{L} \leftarrow \mathbf{I}_m, \mathbf{D} \leftarrow \mathbf{O}_m$
  for $i = 1 : d : m - d + 1$ **do**
    $\mathbf{D}[i : i + d - 1, i : i + d - 1] \leftarrow \mathbf{\Sigma}[i : i + d - 1, i : i + d - 1]$
    $\mathbf{L}[i + d : m, i : i + d - 1] \leftarrow \mathbf{\Sigma}[i + d : m, i : i + d - 1]\mathbf{D}^{-1}[i : i + d - 1, i : i + d - 1]$
    $\mathbf{\Sigma}[i + d : m, i + d : m] \leftarrow \mathbf{\Sigma}[i + d : m, i + d : m] - \mathbf{L}[i + d : m, i : i + d - 1]\mathbf{D}^{-1}[i : i + d - 1, i : i + d - 1]\mathbf{L}[i : i + d - 1, i + d : m]$
    if $i + d < m$ **then**
      $\mathbf{D}[i + d : m, i + d : m] \leftarrow \mathbf{\Sigma}[i + d : m, i + d : m]$
    end if
  end for
  return L and **D**
end procedure

When $s = \frac{m}{d}$ is integer, results of Algorithm 1, $\mathbf{L}, \mathbf{D}$ 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 $\mathbf{D}_1, \cdots, \mathbf{D}_s$. Algorithm 1 is still valid when m is not multiple of d if we allow $\mathbf{L}, \mathbf{D}$ to have non-d dimensional matrix block as last row.

As in (2), tranformation, $Y = LX$ provides m-dimensional multivariate normal prabability as the product of s d-dimensional multivariate normal probabilities as below.

$$\mathbf{\Phi_m}(\mathbf{a}, \mathbf{b}; \mathbf{0}, \mathbf{\Sigma}) = \int_{\alpha_1}^{\beta_1} \phi_d(\mathbf{y}_1; \mathbf{D}_1) \int_{\alpha_2}^{\beta_2} \phi_d(\mathbf{y}_2; \mathbf{D}_2) \cdots \int_{\alpha_s}^{\beta_s} \phi_d(\mathbf{y}_s; \mathbf{D}_s) d\mathbf{y}_s \cdots d\mathbf{y}_2 d\mathbf{y}_1 \tag{3}$$

,where $\boldsymbol{\alpha}_i = \mathbf{a}_i - \sum_{j=1}^{i-1} \mathbf{L}_{ij}\mathbf{y}_j, \boldsymbol{\beta}_i = \mathbf{b}_i - \sum_{j=1}^{i-1} \mathbf{L}_{ij}\mathbf{y}_j$ Equation (3) is implemented as below.

Algorithm 2 d-dimensional conditioning algorithm

procedure CMVN($\mathbf{\Sigma}, \mathbf{a}, \mathbf{b}, d$)
  $\mathbf{y} \leftarrow \mathbf{0}, P \leftarrow 1$
  for $i = 1 : s$ **do**
    $j \leftarrow (i-1)d$
    $\mathbf{g} \leftarrow \mathbf{L}[j+1 : j+d, 1 : j]\mathbf{y}[1 : j]$
    $\boldsymbol{\alpha} \leftarrow \mathbf{a}[j+1 : j+d] - \mathbf{g}$
    $\boldsymbol{\beta} \leftarrow \mathbf{b}[j+1 : j+d] - \mathbf{g}$
    $\mathbf{D}' \leftarrow \mathbf{D}[j+1 : j+d, j+1 : j+d]$
    $P \leftarrow P \cdot \mathbf{\Phi}_d(\boldsymbol{\alpha}, \boldsymbol{\beta}; \mathbf{0}, \mathbf{D}')$
    $\mathbf{y}[j+1 : j+d] \leftarrow E[\mathbf{Y}']$
  end for
  return P and $\mathbf{y}$
end procedure

2.2 Multidimensional Truncated Expectations

In algorithm 2 needs approximation of $\mathbf{\Phi}_d$ and $E[Y']$. $\mathbf{\Phi}_d$ is possibly obtained with quasi monte calro method proposed by Genz and Bretz (2009), and Kan and Robotti (2017) provides methods to calculate $E[Y']$. The truncated expectation is expressed as

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

Theorem 1. *(Kan & 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

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

Proof. Derivative of the multivariate normal density satisfies below

$$-\frac{\partial \phi_n(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \mathbf{x}} = \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \mu)\phi_n(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{4}$$

With integration (4) from $\mathbf{a}$ to $\mathbf{b}$,

$$\boldsymbol{c} = \boldsymbol{\Sigma}^{-1} \begin{bmatrix} F_1^d - \mu_1 \Phi_{d-1} \\ F_2^d - \mu_1 \Phi_{d-1} \\ \vdots \\ F_d^d - \mu_1 \Phi_{d-1} \end{bmatrix} \tag{5}$$

Using the fact that

$$\begin{aligned}
\phi_n(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})|_{x_j = a_j} &= \phi_1(a_j; \mu_j, \sigma_j^2)\phi_{n-1}(\mathbf{x}_{-j}; \hat{\boldsymbol{\mu}}_j^1 \hat{\boldsymbol{\Sigma}}^1) \\
\phi_n(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})|_{x_j = b_j} &= \phi_1(b_j; \mu_j, \sigma_j^2)\phi_{n-1}(\mathbf{x}_{-j}; \hat{\boldsymbol{\mu}}_j^2 \hat{\boldsymbol{\Sigma}}^1),
\end{aligned}$$

(5) becomes

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

□

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 2 with $\boldsymbol{\Phi}$ which can be obtained with quasi monte calro method proposed by Genz and Bretz (2009)

2.3 Multidimensional Conditioning Approximation with Univariate Reordering

It is known that appropriate integration order on conditioning algorithm possibly improves estiation accuracy. Schervish (1984) originally proposed integral with shortest integration interval widths be the outermost integration variables to reduce overall variation of integrand and Gibson, Glasbey, and Elston (1994) suggested 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.

Algorithm 3 d-dimensional conditioning algorithm 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 2
end procedure

3 Hierarchical-Block Approximations

In this section, we suggest methods to solve the n-dimensional MVN problem with the hierarchical covariance matrix using the d-dimensional conditioning method with that of the Monte Carlo-based method for solving the m-dimensional MVN problems presented by the diagonal blocks.

3.1 Hierarchical Cholesky Decomposition

Hackbusch (2015) proposed hiarchical matrix and its cholesky decomposition method. We have applied low rank approximation to each block of its decomposition to make implementation efficiently and save storage while accuracy is preserved. $A = LU$have the structure

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & O \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{12}^2 \\ O & L_{22} \end{pmatrix}$$

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 $A_{22} - L_{21}L_{21}^T$

(a) and (d) is solved with hierachical cholesky decomposition it self, and (b) is easy since it has triangular form. (c) is implemented with low-rank approximation of SVD, i.e. $A = UDV^T = \sum_{i=1}^{n} d_i u_i v_i^T = \sum_{i=1}^{k} d_i u_i v_i^T$. Hierachical cholesky decomposition of $n \times n$ matrix into $m \times m$ blocks is implemented like below.

Algorithm 4 Hierachical cholesky decomposition

procedure HCHOL(A, n,m,rank)
  for $i = 1 : log_2(\frac{n}{m})$ **do**
    $nb = n/2^i$
    x = 0, y = nb
    for $j = 1 : 2^{i-1}$ **do**
      $\mathbf{U}, \mathbf{D}, \mathbf{V} = lowrankSVD(A[xbegin + 1 : xbegin + nb, ybegin + 1 : ybegin + nb], rank)$
      $\mathbf{A}[x + 1 : x + nb, y + 1 : y + rank] = \mathbf{UD}$
      $\mathbf{A}[x + 1 : x + nb, y + rank + 1 : y + nb] = \mathbf{O}$
      $\mathbf{A}[y + 1 : y + nb, x + 1 : x + rank] = \mathbf{VD}$
      $\mathbf{A}[y + 1 : y + nb, x + rank + 1 : x + nb] = \mathbf{O}$
      $x+ = 2nb, y+ = 2nb$
    end for
  end for
end procedure

3.2 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, we can express (1) as

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

Where $\mathbf{a}'$, $\mathbf{b}'$, $i=1,\cdots,r$, are the corresponding segments of the updated $\mathbf{a}$ and $\mathbf{b}$. Specifficaly, we can compute n-dimensional MVN problem using hierarchical structure as algorithm 5.

Algorithm 5 Hierarchical-block conditioning algorithm

procedure `HMVN`($a,\ b,\ \mathbf{\Sigma},\ d$)
  $\mathbf{x} \leftarrow \mathbf{0}$ and $P \leftarrow 1$
  $[\mathbf{B},\mathbf{UV}] \leftarrow$ `choldecomp_hmatrix`($\mathbf{\Sigma}$)
  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
    $\mathbf{a}_i \leftarrow \mathbf{a}[j+1:j+m]$
    $\mathbf{b}_i \leftarrow \mathbf{b}[j+1:j+m]$
    $P = P * \Phi_m(\mathbf{a}_i,\mathbf{b}_i;\mathbf{0},\mathbf{B}_i\mathbf{B}_i^T)$
    $\mathbf{x}[j+1:j+m] \leftarrow \mathbf{B}_i^{-1}\mathbb{E}[\mathbf{X}_i]$
  end for
end procedure

Note the probabilities $\Phi_m(\mathbf{a}_i,\mathbf{b}_i;\mathbf{0},\mathbf{B}_i\mathbf{B}_i^T)$ can be computed using Quasi-Monte Carlo method (`HMVN`, Method 1 in Cao et al. (2019)), d-dimensional conditioning algorithm (`HCMVN`, Method 2 in Cao et al. (2019)) or with 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.

3.3 Computational Complexity

For a clearer comparison of the complexities, we decompose the complexity of Algorithm 5 into three parts and list the complexity for each part in Table 1, where $M(\cdot)$ denotes the complexity

of the QMC simulation in the given dimension.

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

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

The three parts of the complexity are the calculation of the MVN probability (MVN prob), the calculation of the truncated expectations (Trunc exp), and the update of the integration limits with truncated expectations (Upd limits). The latter two share the same asymptotic order in all three complexity terms. The updating cost is independent of the method. The complexity of the univariate reordering is $O(m^2n)$, the same as the complexity of computing the MVN probabilities in `HCMVN`, resulting in an identical major complexity component for `HCMVN` and `HRCMVN`. Since `HCMVN` and `HRCMVN` perform the QMC simulation in d-dimensions, these two methods are not greatly affected by the choice of m.

4 Block Reordering

Summarizing the previous computations, the cdf value for n-dimensioned multivariate normal variable comprises of m multiplications of d-dimensional integrals. The idea arose from the construction of hierarchical Cholesky decomposition of covariance matrix, which enables concentration of high correlation values in the block diagonal area. As a consequence, correlation within blocks(or, group of compositions) are high while correlation between blocks(between groups) are low. Recall the RCMVN algorithm(2) : as computing each d-dimensional integral values, integration variables were arranged in order of increasing order of CMVN probability values, from outer to inner. Trinh and Genz (2015) discovered that this reordering improves overall accuracy. In this sense, the authors adopted block reordering procedure. Briefly speaking, this procedure permutes the block of LDL-decomposed covariance matrix, in order of RCMVN probability values of each blocks. On the top left corner, the block with minimal probability value get its place. The block reordering algorithm introduced in the paper is as following:

Here, sort function arranges elements in the vector 'ind' in an increasing order based on $\mathbf{P}$. Also, G means the intrinsic geometry of data. In practice, we set the geometry as 2D isotropic exponential covariance model, assuming data observation occured in unit square grd, indices

Algorithm 6 Blockwise reordering

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

arranged based on Morton order of level 1. The paper implemented the comparison between computing HMVN, HCMVN, HRCMVN values with or without block reordering procedure. Commonly, block reordering procedure places on the way before implementing hierarchical decomposition. For example, one can derive the algorithms of HMVN, including block reordering procedure as following:

Block reordering aims for improving accuracy rather than shortening computation time. According to the simulation result introduced in the paper, the extra time used for running block reordering is relatively small compared with the total time costs. The table below displays the running time for block reordering procedure and its condition.

5 Numerical Examples

In this section, we show you the results of implementing and experimenting with the methods described above. The implementation was done in julia=1.2.0. Code is made available in the github repository[1].

5.1 Cholesky Factorization

The *chol* function from **LinearAlgebra** package, the *dpotrf* from **LAPACK** package, and 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 1.

[1] `https://github.com/kw-lee/AdvStatComp_HDP`

Algorithm 7 Hierarchical-block conditioning algorithm with Block Reordering

procedure `HCMVN_BRO`$(a,\ b,\ \Sigma,\ d)$
  $\mathbf{x} \leftarrow \mathbf{0}$, $P \leftarrow 1$, 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
    $\mathbf{a}_i \leftarrow \mathbf{a}[j + 1 : j + m]$
    $\mathbf{b}_i \leftarrow \mathbf{b}[j + 1 : j + m]$
    $P = P * \Phi_m(\mathbf{a}_i, \mathbf{b}_i; \mathbf{0}, \mathbf{B}_i\mathbf{B}_i^T)$
    $\mathbf{x}[j + 1 : j + m] \leftarrow \mathbf{B}_i^{-1}\mathbb{E}[\mathbf{X}_i]$
  end for
end procedure

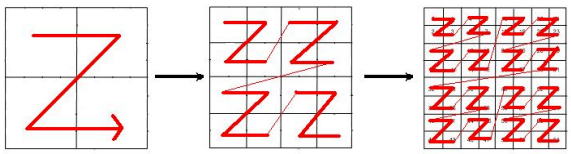


Figure 1: Morton's order(Salem & Arab, 2016)

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

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 $\frac{\|\mathbf{\Sigma} - L_H L_H^T\|_2}{\|\mathbf{\Sigma}\|_2}$, and table 2 ensures accuracy of hierarchical cholesky decomposition proposed by Hackbusch (2015).

5.2 Multivariate Normal Probabilities

To implement `*MVN` functions, we need to calculate n-dimensional normal probability (1),

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

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

numerically. We implement `mvn`, the function that calculate multivariate normal probabilities using Richtmyer Quasi-Monte Carlo(QMC) method proposed by Genz and Bretz (2009). It is well-known that QMC methods is more effective than classical Monte Carlo(MC) method. Varying sample size N and dimension d, two Monte Carlo methods are compared and results are below table 3. 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. Relative errors and computation times of each method are formulated.

Note QMC is superior to MC in every criterion. All the multivariate normal distribution probabilities required in the next algorithms are calculated using the `mvn` function.

5.3 d-dimensional Conditioning Algorithm without/with Reordering

Haar distribution is defined with a uniform distribution in the unitary $N \times N$ matrices group, $U(N)$. Stewart (1980) provides how to sample from Haar distribution with theorem 2

Theorem 2. *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.*

We simulates 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 $i = 1 \cdots, m$ are chosen. 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}, and relative error and spent time is formulated below.

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.

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

5.4 Hierarchical-Block Approximations

In this section, we implement three methods in the section 3 and compare theirs performance

- 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

We use 20 as the sample size instead of 250 as in Table 4 because the covariance structure is fixed, leading to a much smaller standard deviation for the estimators. We simulates two spatial problems with various values of m and n.

(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 4: Errors and execution times of the d-dimensional conditioning method

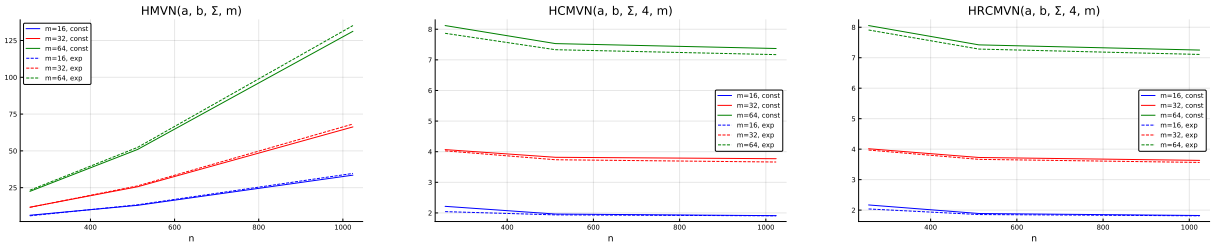
1. Constant covariance matrix: $k(x_i, x_j) = \theta + (1 - \theta)\delta_{ij}$ for some $|\theta| < 1$.

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.

We set the 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). Estimated value is compared with approximated value obtained via QMC with a sample size of 10^4. In this simulation, we fix $d = 4$ for `HCMVN` and `HRCMVN`. Table 5 and Figure 2 are errors and execution times under the constant covariance structure and 1D exponential covariance structure respectively.

Note the excution times of M2 and M3 are significantly smaller than of M1 even their performances are similar.

Next, we give simulation with a 2D exponential covariance matrix using morton's order for indexing the sample points on the plane. We fix $m = 8$ for the 2D covariance structure and examine the effectiveness of our algorithm varying $d = 1, 2, 4$. To test the sensitivity with respect to the correlation strength, we perform the estimation under $\beta = 0.3, 0.1$, and 0.03, representing strong, medium, and weak correlation strengths. We set n to 16, 64, 256 and other settings are same as above. Table 6 presents the results.



(a) M1 (b) M2 (c) M3

Figure 2: Execution time (seconds) under the constant covariance structure and 1D exponential covariance structure

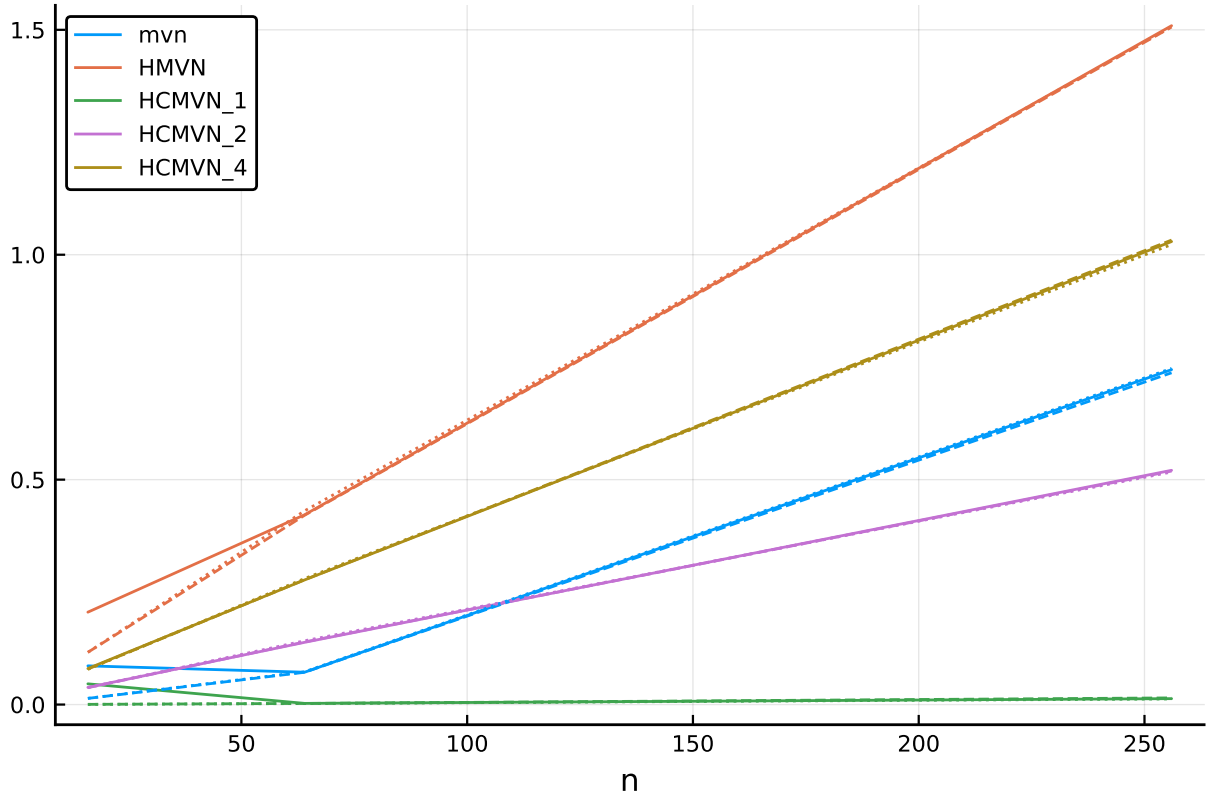


Figure 3: Execution time (seconds) under the constant covariance structure and 2D exponential covariance structure; solid line: $\beta = 0.3$, dashed line: $\beta = 0.1$, dotted line: $\beta = 0.03$

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

In Table 6, $\texttt{HCMVN}_d$ denotes `HCMVN` function with d and

$$compress = \frac{\text{Total memory size of the } \mathcal{H} \text{ matrix}}{\text{Memory size of the covariance matrix } \boldsymbol{\Sigma}}.$$

In this experiment, Cao et al. (2019) mentioned

> It is worth mentioning that the correlation strength is essentially increased when n increases while β remains unchanged because the samples are from the unit hypercube. Also the increase of d will reduce the estimation error but is still unable to reach a satisfactory level. The method, when used without a reordering strategy, does not produce sufficiently accurate results. This motivated the development of the reordering strategy described in the next section

with extremely large size simulation. But, in our experiment, this phenomenon could not be found due to restriction of the computing resource.

6 Conclusion

References

Cao, J., Genton, M. G., Keyes, D. E., & Turkiyyah, G. M. (2019). Hierarchical-block conditioning approximations for high-dimensional multivariate normal probabilities. *Statistics and Computing*, *29*(3), 585–598.

n	*compress*	`mvn`	`HMVN`	$\texttt{HCMVN}_1$	$\texttt{HCMVN}_2$	$\texttt{HCMVN}_4$
$\beta = 0.3$						
16	0.89	0.00ppm	53.68ppm	2139.98ppm	1723.46ppm	276.37ppm
64	0.56	0.00ppm	143.88ppm	167.26ppm	167.69ppm	169.31ppm
256	0.28	0.00ppm	215.10ppm	213.41ppm	213.41ppm	213.41ppm
$\beta = 0.1$						
16	0.89	0.00ppm	0.00ppm	0.84ppm	0.60ppm	0.02ppm
64	0.56	0.00ppm	0.00ppm	1.61ppm	0.09ppm	0.01ppm
256	0.28	0.00ppm	0.00ppm	0.02ppm	0.02ppm	0.02ppm
$\beta = 0.03$						
16	0.89	0.00ppm	0.00ppm	0.00ppm	0.00ppm	0.00ppm
64	0.42	0.00ppm	0.00ppm	0.00ppm	0.00ppm	0.00ppm
256	0.20	0.00ppm	0.00ppm	0.00ppm	0.00ppm	0.00ppm

Table 6: Relative errors and efficiency of the two-level hierarchical-block conditioning methods

Genton, M. G., Keyes, D. E., & 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., & Bretz, F. (2009). *Computation of multivariate normal and t probabilities* (Vol. 195). Springer Science & Business Media.

Gibson, G., Glasbey, C., & Elston, D. (1994). Monte carlo evaluation of multivariate normal integrals and sensitivity to variate ordering. *Advances in Numerical Methods and Applications*, 120–126.

Hackbusch, W. (2015). *Hierarchical matrices: algorithms and analysis* (Vol. 49). Springer.

Kamakura, W. A. (1989). The estimation of multinomial probit models: A new calibration algorithm. *Transportation Science*, *23*(4), 253–265.

Kan, R., & 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., & Elston, R. (1974). Multifactorial qualitative traits: genetic analysis and prediction of recurrence risks. *Biometrics*, 41–57.

Salem, F. K. A., & 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., & Genz, A. (2015). Bivariate conditioning approximations for multivariate normal probabilities. *Statistics and Computing*, *25*(5), 989–996.