

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

---

Hyunseok Yang, Kyeongwon Lee, and Songhyun Kim

December 8, 2019

Department of Statistics, SNU

# Table of contents

1. Introduction
2. Multidimensional Conditioning Approximations  
Quasi Monte Carlo Method
3. Hierarchical-Block Approximation
4. Block Reordering
5. Elements
6. Conclusion

# 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  $\mu$  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

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  $\Delta \sim U[0, 1]^n$ ,

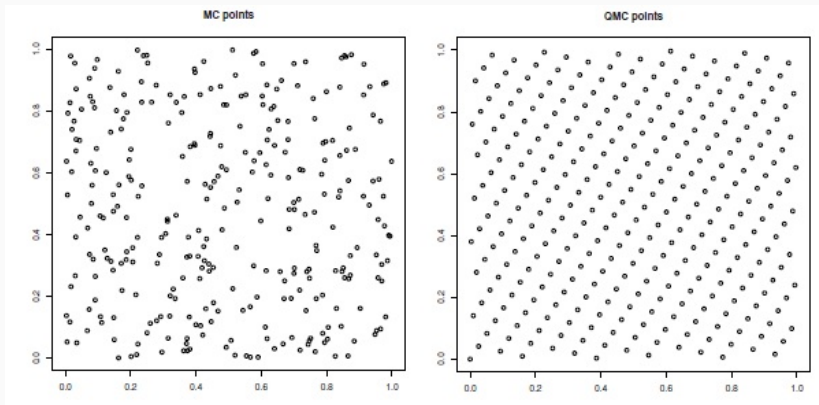
$$L_N = \{\mathbf{z} + \Delta \bmod 1 : \mathbf{z} \in K_N\}$$

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

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

**lined 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  $2 \times 2$  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. Algorithms and details are following.

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

lined 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), tranformation,  $\mathbf{Y} = \mathbf{L}\mathbf{X}$  provides  $m$ -dimensional multivariate normal prabability as the product of  $s$   $d$ -dimensional multivariate normal probabilities as below.

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

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

lined 3: d-dimensional conditioning algorithm

# Multidimensional Truncated Expectations

The truncated expectation is expressed as

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

## 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

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

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

```

**lined 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  $A_{22} - L_{21}L_{11}^T$

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:        $U, D, V = \text{lowrankSVD}(A[xbegin + 1 : xbegin + nb, ybegin + 1 : ybegin + nb], rank)$ 
7:        $A[x + 1 : x + nb, y + 1 : y + rank] = UD$ 
8:        $A[x + 1 : x + nb, y + rank + 1 : y + nb] = O$ 
9:        $A[y + 1 : y + nb, x + 1 : x + rank] = VD$ 
10:       $A[y + 1 : y + nb, x + rank + 1 : x + nb] = O$ 
11:       $x+ = 2nb, y+ = 2nb$ 
12:    end for
13:  end for
14: end procedure
```

**lined 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  $\Sigma$ . Then,

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

,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 (HBMV, Method 1 in Cao et al. (2019))
2.  $d$ -dimensional conditioning algorithm (HCMV, Method 2 in Cao et al. (2019))
3.  $d$ -dimensional conditioning algorithm with univariate reordering (HRCMV, 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( $a$ ,  $b$ ,  $\Sigma$ ,  $d$ )
2:    $x \leftarrow 0$  and  $P \leftarrow 1$ 
3:    $[B, UV] \leftarrow \text{choldecomp\_hmatrix}(\Sigma)$ 
4:   for  $i = 1 : r$  do
5:      $j \leftarrow (i - 1)m$ 
6:     if  $i > 1$  then
7:        $o_r \leftarrow \text{row offset of } U_{i-1} V_{i-1}^T$ 
8:        $o_c \leftarrow \text{column offset of } U_{i-1} V_{i-1}^T$ 
9:        $l \leftarrow \text{dim}(U_{i-1} V_{i-1}^T)$ 
10:       $g \leftarrow U_{i-1} V_{i-1}^T x[o_c + 1 : o_c + l]$ 
11:       $a[o_r + 1 : o_r + l] = a[o_r + 1 : o_r + l] - g$ 
12:       $b[o_r + 1 : o_r + l] = a[o_r + 1 : o_r + l] - g$ 
13:    end if
14:     $a_j \leftarrow a[j + 1 : j + m]$ 
15:     $b_j \leftarrow b[j + 1 : j + m]$ 
16:     $P = P * \Phi_m(a_j, b_j; 0, B_j B_j^T)$ 
17:     $x[j + 1 : j + m] \leftarrow B_j^{-1} E(x_j)$ 
18:  end for
19: end procedure
```

**lined 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	$\frac{n}{m} M(m)$	$2nM(m) + O(nm^2)$	$O(mn + kn\log(n/m))$
HCMVN	$\frac{n}{d} M(d) + O(m^2 n)$	$2nM(d) + O(nd^2)$	$O(mn + kn\log(n/m))$
HRCMVN	$\frac{n}{d} M(d) + O(m^2 n)$	$2nM(d) + O(nd^2)$	$O(mn + kn\log(n/m))$

**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



# Elements

---

The theme provides sensible defaults to  
`\emph{emphasize}` text, `\alert{accent}` parts  
or show `\textbf{bold}` results.

becomes

The theme provides sensible defaults to *emphasize* text, **accent** parts or  
show **bold** results.

# Font feature test

- Regular
- *Italic*
- SMALLCAPS
- **Bold**
- ***Bold Italic***
- **SmallCaps**
- Monospace
- *Monospace Italic*
- Monospace **Bold**
- *Monospace Bold Italic*

## Items

- Milk
- Eggs
- Potatos

## Enumerations

1. First,
2. Second and
3. Last.

## Descriptions

**PowerPoint** Meeh.  
**Beamer** Yeeeha.

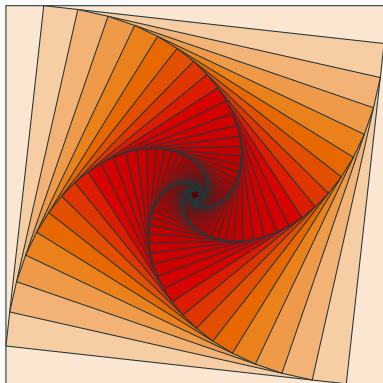
- This is important

- This is important
- Now this



- This is important
- Now this
- And now this

- This is really important
- Now this
- And now this



**Figure 2:** Rotated square from texample.net.

**Table 2:** Largest cities in the world (source: Wikipedia)

City	Population
Mexico City	20,116,842
Shanghai	19,210,000
Peking	15,796,450
Istanbul	14,160,467

Three different block environments are pre-defined and may be styled with an optional background color.

## **Default**

Block content.

## **Alert**

Block content.

## **Example**

Block content.

## **Default**

Block content.

## **Alert**

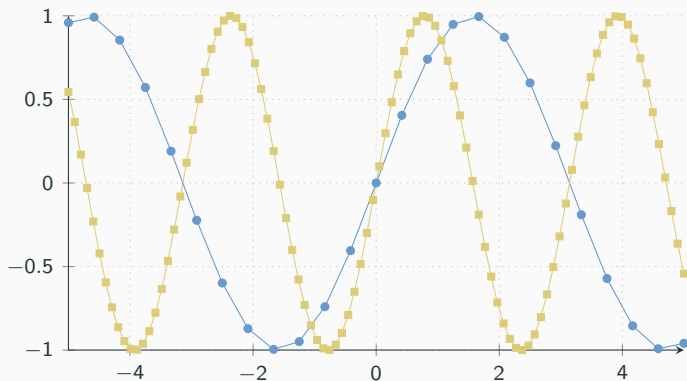
Block content.

## **Example**

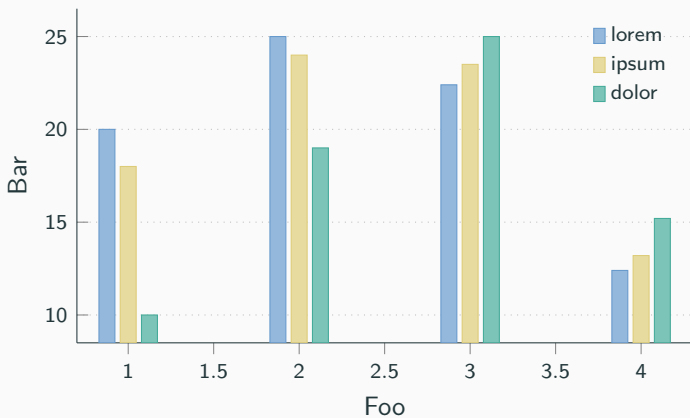
Block content.

$$e = \lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n$$

# Line plots



# Bar charts





*Veni, Vidi, Vici*

**metropolis** defines a custom beamer template to add a text to the footer. It can be set via

```
\setbeamertemplate{frame footer}{My custom footer}
```

Some references to showcase `[allowframebreaks]` ?????

# Conclusion

---

Get the source of this theme and the demo presentation from

`github.com/matze/mtheme`

The theme *itself* is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.



**Questions?**

# Backup slides

Sometimes, it is useful to add slides at the end of your presentation to refer to during audience questions.

The best way to do this is to include the `appendixnumberbeamer` package in your preamble and call `\appendix` before your backup slides.

**metropolis** will automatically turn off slide numbering and progress bars for slides in the appendix.

# 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.
- 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.
- Trinh, G. and Genz, A. (2015). Bivariate conditioning approximations for multivariate normal probabilities. *Statistics and Computing*, 25(5):989–996.