Numerical Computation of Multivariate Normal Probabilities *

Alan Genz
Department of Pure and Applied Mathematics
Washington State University
Pullman, WA 99164-3113 USA
Email: alangenz@wsu.edu

Abstract

The numerical computation of a multivariate normal probability is often a difficult problem. This article describes a transformation that simplifies the problem and places it into a form that allows efficient calculation using standard numerical multiple integration algorithms. Test results are presented that compare implementations of two algorithms that use the transformation, with currently available software.

KEY WORDS: multivariate normal distribution, Monte-Carlo, adaptive integration.

1 Introduction

A problem that arises in many statistics applications is that of computing the multivariate normal distribution function

$$F(\mathbf{a},\mathbf{b}) = \frac{1}{\sqrt{|\Sigma|(2\pi)^m}} \int_{a_1}^{b_1} \int_{a_2}^{b_2} \cdots \int_{a_m}^{b_m} e^{-\frac{1}{2}\boldsymbol{\theta}^t \Sigma^{-1} \boldsymbol{\theta}} d\boldsymbol{\theta},$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_m)^t$ and Σ is an $m \times m$ symmetric positive definite covariance matrix. If, for some i, a_i is $-\infty$ and b_i is ∞, an appropriate transformation allows the ith variable to be integrated explicitly and reduces the number of variables in the problem, so, for each i, at least one of a_i and b_i is assumed to be finite.

There is reliable and efficient software available for the cases $m = 1$ and $m = 2$ (for $m = 2$ see [DON 73], [DW 90] and [CW 91]), so assume $m > 2$. An algorithm by Schervish [SCHRV 84] is implemented for $m < 8$. This algorithm uses a locally adaptive numerical integration strategy based on Simpson's rule. It also provides an error bound for F. Tests reported in Section 5 of this article, however, show that this algorithm can require very long computation times for $m > 4$.

An obvious approach to computing F is to try currently available numerical integration software (e.g Monte-Carlo methods or subregion adaptive methods) directly. However, the infinite integration limits need to be handled either by using some type of transformation to a finite region or, by using carefully selected cutoff values. In either case, any numerical integration method that does not take account of the peaked character of the integrand can waste significant amounts of computational effort for many problems.

The purpose of this article is to show that a relatively straightforward sequence of transformations can be used to put the problem into a form that allows reliable and efficient computation of F using either simple Monte-Carlo or subregion adaptive numerical integration algorithms. These transformations will be described in the Section 2. Section 3 gives a complete description of a simple Monte-Carlo algorithm and Section 4 provides an illustrative example. Test results comparing the Schervish algorithm with Monte-Carlo and subregion adaptive algorithms that use the transformations will be presented in the Section 5.

2 Transformations

A sequence of three transformations will be used to transform the original integral into an integral over a unit hyper-cube. This sequence begins with a Cholesky decomposition transformation $\boldsymbol{\theta} = C\mathbf{y}$, where CC^t is the

*Revised version published in *J. Comp. Graph Stat.* 1 (1992), pp. 141-149. This work was supported in part by NSF grant DMS-9008125.

Cholesky decomposition of the covariance matrix Σ. Now $\boldsymbol{\theta}^t\Sigma^{-1}\boldsymbol{\theta} = \mathbf{y}^t C^t C^{-t} C^{-1} C\mathbf{y} = \mathbf{y}^t\mathbf{y}$, and $d\boldsymbol{\theta} = |C|d\mathbf{y} = |\Sigma|^{\frac{1}{2}}d\mathbf{y}$. Since $\mathbf{a} \le \boldsymbol{\theta} = C\mathbf{y} \le \mathbf{b}$ implies $(a_i - \sum_{j=1}^{i-1} c_{ij}y_j)/c_{ii} \le y_i \le (b_i - \sum_{j=1}^{i-1} c_{ij}y_j)/c_{ii}$ for $i = 1, 2, ..., m$, we have

$$F(\mathbf{a}, \mathbf{b}) = \frac{1}{\sqrt{(2\pi)^m}} \int_{a_1'}^{b_1'} e^{-\frac{y_1^2}{2}} \int_{a_2'(y_1)}^{b_2'(y_1)} e^{-\frac{y_2^2}{2}} ... \int_{a_m'(y_1,...,y_{m-1})}^{b_m'(y_1,...,y_{m-1})} e^{-\frac{y_m^2}{2}} d\mathbf{y},$$

with $a_i'(y_1, ..., y_{i-1}) = (a_i - \sum_{j=1}^{i-1} c_{ij}y_j)/c_{ii}$ and $b_i'(y_1, ..., y_{i-1}) = (b_i - \sum_{j=1}^{i-1} c_{ij}y_j)/c_{ii}$.

Now each of the y_i's can be transformed separately using $y_i = \Phi^{-1}(z_i)$, where

$$\Phi(y) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{y} e^{-\frac{1}{2}\theta^2} d\theta$$

is the standard univariate normal distribution function. After these transformations, $F(\mathbf{a}, \mathbf{b})$ becomes

$$F(\mathbf{a}, \mathbf{b}) = \int_{d_1}^{e_1} \int_{d_2(z_1)}^{e_2(z_1)} ... \int_{d_m(z_1,...,z_{m-1})}^{e_m(z_1,...,z_{m-1})} d\mathbf{z},$$

with $d_i(z_1, ..., z_{i-1}) = \Phi((a_i - \sum_{j=1}^{i-1} c_{ij}\Phi^{-1}(z_j))/c_{ii})$ and $e_i(z_1, ..., z_{i-1}) = \Phi((b_i - \sum_{j=1}^{i-1} c_{ij}\Phi^{-1}(z_j))/c_{ii})$.

The integrand in this form is much simpler than the original integrand. The integration region is more complicated, however, and cannot be handled directly with standard numerical multiple integration algorithms. A solution to this problem is to put the integral into a constant limit form using $z_i = d_i + w_i(e_i - d_i)$. After this final set of transformations,

$$F(\mathbf{a}, \mathbf{b}) = (e_1 - d_1) \int_0^1 (e_2 - d_2) ... \int_0^1 (e_m - d_m) \int_0^1 d\mathbf{w},$$

with $d_i = \Phi((a_i - \sum_{j=1}^{i-1} c_{ij}\Phi^{-1}(d_j + w_j(e_j - d_j)))/c_{ii})$ and $e_i = \Phi((b_i - \sum_{j=1}^{i-1} c_{ij}\Phi^{-1}(d_j + w_j(e_j - d_j)))/c_{ii})$.

The innermost integral over w_m can be done explicitly because d_m and e_m have no dependence on w_m, so the complete sequence of transformations has reduced the number of integration variables by one.

The sequence of transformations described here might appear to have made the final integrand more complicated. However, the overall transformation has forced a priority ordering on the integration variables. The w_1 variable is the most important one, because all of the integrand factors $e_i - d_i$ for $i > 1$, depend on it. The w_2 variable is the next most important one, and so on. This priority ordering actually makes the problem more suitable for the type of subregion adaptive algorithm used for some of the tests described in Section 5. This type of adaptive algorithm subdivides along one coordinate axis at a time and therefore works most efficiently when the integrand has a few priority variables that define the directions of most variation in the integrand.

3 An Algorithm

The problem is now in a form that could be presented as input for a variety of different numerical integration algorithms. Test results which will be reported in the next section show that a simple Monte-Carlo algorithm is surprisingly effective so the details of such an algorithm, which incorporates the transformations discussed in the previous section, are now given.

Multivariate Normal Probabilities Algorithm

1. **Input** Σ, $\mathbf{a}$, $\mathbf{b}$, ϵ, α and N_{max}.
2. Compute lower triangular Cholesky factor C for Σ.
3. Initialize $Intsum = 0, N = 0, Varsum = 0$,
   $d_1 = \Phi(a_1/c_{1,1})$, $e_1 = \Phi(b_1/c_{1,1})$ and $f_1 = e_1 - d_1$.
4. **Repeat**
   (a) Generate uniform random $w_1, w_2, ..., w_{m-1} \epsilon [0,1]$.
   (b) For $i = 2, 3, ..., m$ set $y_{i-1} = \Phi^{-1}(d_{i-1} + w_{i-1}(e_{i-1} - d_{i-1}))$,
   $d_i = \Phi((a_i - \sum_{j=1}^{i-1} c_{i,j} y_j)/c_{i,i})$, $e_i = \Phi((b_i - \sum_{j=1}^{i-1} c_{i,j} y_j)/c_{i,i})$
   and $f_i = (e_i - d_i) f_{i-1}$.
   (c) Set $N = N + 1$, $\delta = (f_m - Intsum)/N$, $Intsum = Intsum + \delta$,
   $Varsum = (N-2)Varsum/N + \delta^2$ and $Error = \alpha\sqrt{Varsum}$.
   Until $Error < \epsilon$ or $N = N_{max}$.
5. **Output** $F = Intsum/N, Error$, and N.

The input parameter α is the usual Monte-Carlo confidence factor for the standard error. If for example, $\alpha = 2.5$ is used, we expect the actual error in F to be less than $Error$, 99% of the time. N_{max} is an input parameter that limits the total amount of time allowed for the computation.

For problems where for some i's either a_i is $-\infty$ or b_i is ∞, any implementation of the algorithm should explicitly set $d_i = 0$ or $e_i = 1$ to avoid wasteful evaluation of Φ. Preliminary tests with the algorithm showed that there was usually some reduction in the computation time if the variables were reordered (along with appropriate rows and columns of Σ) so that the variables associated with the largest integration intervals were the innermost variables (this sorting of the variables was suggested by Schervish [SCHRV 84]). Bexause this reordering operation does not take much time compared to the total computation time it was added to the algorithm as an additional step between steps 1 and 2 and used for the tests described in the Section 5.

In some applications it is necessary to compute integrals in the form

$$G(g) = \frac{1}{\sqrt{|\Sigma|(2\pi)^m}} \int_{a_1}^{b_1} \int_{a_2}^{b_2} ... \int_{a_m}^{b_m} e^{-\frac{1}{2}\boldsymbol{\theta}^t \Sigma^{-1} \boldsymbol{\theta}} g(\boldsymbol{\theta}) d\boldsymbol{\theta},$$

where $g(\boldsymbol{\theta})$ might be one or more application specific expectation functions. For these integrals only minor modifications to the algorithm are needed. Because $g(\boldsymbol{\theta})$ might depend on θ_m, w_m would need to be generated as part of step 4(a) and $y_m = \Phi^{-1}(d_m + w_m(e_m - d_m))$ would need to be added to step 4(b), along with an additional substep to explicitly compute $\boldsymbol{\theta} = C\mathbf{y}$. Finally, in step 4(c), f_m would need to be replaced (twice) by $f_m g(\boldsymbol{\theta})$. If the application requires an integral value for more than one $g(\boldsymbol{\theta})$, then all of the g functions should be done in the same calculation by further expanding steps 3, 4(c) and the error test, in order to avoid duplicating the work required for computing the transformations.

4 An Example

This section gives an example illustrating the ideas discussed in the previous two sections. Assume $m = 3$, $\mathbf{a} = (-\infty, -\infty, -\infty)$, $\mathbf{b} = (1, 4, 2)$ and

$$\Sigma = \begin{pmatrix} 1 & \frac{3}{5} & \frac{1}{3} \\ \frac{3}{5} & 1 & \frac{11}{15} \\ \frac{1}{3} & \frac{11}{15} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{3}{5} & \frac{4}{5} & 0 \\ \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \end{pmatrix} \begin{pmatrix} 1 & \frac{3}{5} & \frac{1}{3} \\ 0 & \frac{4}{5} & \frac{2}{3} \\ 0 & 0 & \frac{2}{3} \end{pmatrix}.$$

Then, after the first transformation,

$$F(\mathbf{a}, \mathbf{b}) = \frac{1}{\sqrt{(2\pi)^3}} \int_{-\infty}^{1} e^{-\frac{y_1^2}{2}} \int_{-\infty}^{5-3y_1/4} e^{-\frac{y_2^2}{2}} \int_{-\infty}^{3-y_1/2-y_2} e^{-\frac{y_3^2}{2}} d\mathbf{y}.$$

After the second transformation,

$$F(\mathbf{a}, \mathbf{b}) = \int_0^{\Phi(1)} \int_0^{\Phi(5-3\Phi^{-1}(z_1)/4)} \int_0^{\Phi(3-\Phi^{-1}(z_1)/2-\Phi^{-1}(z_2))} d\mathbf{z}.$$

Finally,

$$F(\mathbf{a},\mathbf{b}) = \Phi(1)\int_0^1 \Phi(5-3\Phi^{-1}(w_1\Phi(1))/4) \int_0^1 \Phi(3-\Phi^{-1}(w_1\Phi(1))/2-\Phi^{-1}(w_2\Phi(5-3\Phi^{-1}(w_1\Phi(1))/4)))\int_0^1 d\mathbf{w}.$$

If variables θ_2 and θ_3 are interchanged to sort $\mathbf{b}$, then

$$\Sigma = \begin{pmatrix} 1 & \frac{1}{3} & \frac{3}{5} \\ \frac{1}{3} & 1 & \frac{11}{15} \\ \frac{3}{5} & \frac{11}{15} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{3} & \frac{2\sqrt{2}}{3} & 0 \\ \frac{3}{5} & \frac{2\sqrt{2}}{5} & \frac{2\sqrt{2}}{5} \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{3} & \frac{3}{5} \\ 0 & \frac{2\sqrt{2}}{3} & \frac{2\sqrt{2}}{5} \\ 0 & 0 & \frac{2\sqrt{2}}{5} \end{pmatrix}.$$

After the first transformation,

$$F(\mathbf{a},\mathbf{b}) = \frac{1}{\sqrt{(2\pi)^3}}\int_{-\infty}^1 e^{-\frac{y_1^2}{2}}\int_{-\infty}^{(3-y_1/2)/\sqrt{2}} e^{-\frac{y_2^2}{2}}\int_{-\infty}^{(10-3y_1/2-\sqrt{2}y_2)/\sqrt{2}} e^{-\frac{y_3^2}{2}}d\mathbf{y}.$$

And after the final transformation,

$$F(\mathbf{a},\mathbf{b}) = \Phi(1)\int_0^1 \Phi\left(\frac{3-\Phi^{-1}(w_1\Phi(1))/2}{\sqrt{2}}\right) \int_0^1 \Phi\left(\frac{10-3\Phi^{-1}(w_1\Phi(1))/2-\sqrt{2}\Phi^{-1}(w_2\Phi(\frac{3-\Phi^{-1}(w_1\Phi(1))/2}{\sqrt{2}})\sqrt{2})}{\sqrt{2}}\right)\int_0^1 d\mathbf{w}.$$

If the Monte-Carlo algorithm is used with this example (where $F(\mathbf{a},\mathbf{b}) \doteq 0.82798$), the computed sample variance for the original problem is approximately 0.0016, while the computed sample variance for the reordered variable version of the problem is approximately 0.000064. These results imply that in order to compute $F(\mathbf{a},\mathbf{b})$ to some prescribed accuracy level, the original form of the problem would require approximately 25 times more computation than the reordered variable version of the problem.

5 Test Results

Three types of tests were used for integration methods based on the transformations described in Section 2. The first and third sets of tests used covariance matrices with a simple structure, where a different method could be used to compute F and independently verify the accuracy of the algorithms. The second set of test used random covariance matrices, where no simple independent method of verifying the accuracy of the algorithms is available. In all of the tests the lower limits $\mathbf{a}$ were set equal to $-\infty$. For the first two test sets the requested absolute accuracy level ϵ was fixed, and for the third test set three different ϵ values were used.

A FORTRAN implementation of the algorithm in the previous section was compared with the Schervish algorithm and, an algorithm which uses a subregion adaptive multiple integration method for the numerical integration. This method is described in an article by Berntsen, Espelid and Genz [BEG 91]. The single precision FORTRAN implementation that was used for the tests reported in this section is called SCUHRE. This algorithm begins by applying a low degree (basic) integration rule to the whole integration region (a unit cube for these problems). If the error estimate is too large, the initial integration region is divided in half. The cut is made along the coordinate axis where the integrand is most rapidly changing. Next, the basic rule is applied to the two new subregions. If the total error estimate is still too large, the subregion with largest error is cut in half. The algorithm continues in this manner by subdividing, at each stage, the subregion with largest estimated error, until the sum of all of the subregion errors is less than ϵ or a limit on the total number of integrand evaluations has been reached. This type of algorithm concentrates the integrand evaluations in the subregions and along the directions where the integrand is most rapidly changing.

Initial tests with the SCUHRE code showed that although the degree seven basic rule used by SCUHRE often provided very accurate results, it required too much time for larger m values. A degree five basic rule was therefore added to the code and used as the basic integration rule for the following tests. Initial tests with the modified SCUHRE code also showed that the error estimates provided by SCUHRE were usually much too conservative. A driver subroutine was therefore written for SCUHRE. The driver initially called SCUHRE

with $N_0 = 100 + 10m^2$ allowed integrand evaluations. Repeated calls of SCUHRE were made, with $N_k = 2N_{k-1}$ new integrand evaluations allowed for the k^{th} call. For each new call of SCUHRE, the computation began where the previous computation had been stopped. If A_k and E_{A_k} were used to denote the respective estimates for F and the error in F provided by SCUHRE, the driver subroutine continued the computation until $|A_k - A_{k-1}| + E_{A_k} / \sum_{j=0}^{k} N_j < \epsilon$. The results given in Tables 1-3 were obtained using this modified version of SCUHRE (with driver).

A driver was also written for the crude Monte-Carlo algorithm described in the previous section and this driver was used for the tests reported in this section. For this driver $N_0 = 100 + 10m^2$ was used for the initial N_{max} in the Monte-Carlo algorithm. Subsequent calls to the Monte-Carlo algorithm used $N_{max} = N_k = 3N_{k-1}/2$. Let M_k denote the Monte-Carlo estimate for F using N_k samples and $E^2_{M_k} = \sigma^2_k / N_k$, where σ^2_k is the sample variance for M_k. This driver subroutine continues the computation until $2.5/(\sum_{j=0}^{k} 1/E^2_{M_j})^{\frac{1}{2}} < \epsilon$ and, for an estimate for F it returns the weighted average $(\sum_{j=0}^{k} M_j/E^2_{M_j})/(\sum_{j=0}^{k} 1/E^2_{M_j})$.

For the test results in this section, S, M and A are used to denote results from the Schervish algorithm, Monte-Carlo algorithm (with driver subroutine) and subregion adaptive algorithm (modified SCUHRE with driver subroutine); E_S, E_M and E_A are used to denote average absolute errors and T_S, T_M and T_A are used to denote average running times (in seconds) for the three algorithms. A Digital DECstation 3100 ($\simeq$ 14 mips) was used for the tests.

In the first set of tests, a constant correlation family of covariance matrices, defined by $\sigma_{i,j} = \rho$, for $i \neq j$ and $\sigma_{i,i} = 1$, was used. With these covariance matrices, F has a value that can be accurately computed independently ([TONG 90], p. 192). This is given by

$$F(\mathbf{b}) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}t^2} \prod_{i=1}^{m} \Phi((b_i + \sqrt{\rho}t)/\sqrt{1-\rho})\, dt$$

For each $m = 3, 4, ..., 10, 15, 20$, a sample of 50 different covariance matrices was used, with random ρ values, chosen uniformly from (0,1). The upper limits were also random, chosen uniformly from $[0, \sqrt{m}]$. For $m > 6$, the Schervish software required rapidly increasing computation times and so results are not given for these cases. For $m = 5$ and $m = 6$, the time distributions were skewed, with approximately 80% of the particular computation times less than T_S and one or two more five times larger than T_S. The results are reported in Table 1. Under each row in the table that is labeled with an m value is a row of standard deviation values for the corresponding entries in row above.

Table 1: Constant $\sigma_{i,j}$ Covariance Matrix Results, $\epsilon = 0.005$

m	E_S	E_M	E_A	T_S	T_M	T_A	T_S/T_M
3	0.00035	0.00108	0.00002	0.01	0.11	0.08	0.06
	0.00022	0.00095	0.00003	0.02	0.12	0.05	
4	0.00025	0.00116	0.00007	0.23	0.18	0.17	1.27
	0.00021	0.00121	0.00007	0.31	0.16	0.08	
5	0.00025	0.00142	0.00012	10.91	0.36	0.29	30.50
	0.00016	0.00096	0.00011	22.25	0.22	0.09	
6	0.00018	0.00118	0.00016	334.20	0.80	0.56	419.98
	0.00013	0.00102	0.00015	481.89	0.93	0.42	
7	-	0.00095	0.00018	-	1.13	0.93	
	-	0.00074	0.00019	-	0.83	0.11	
8	-	0.00104	0.00020	-	1.44	1.47	
	-	0.00088	0.00021	-	0.94	0.16	
9	-	0.00118	0.00021	-	1.59	2.08	
	-	0.00090	0.00021	-	0.82	0.19	
10	-	0.00118	0.00022	-	2.18	3.03	
	-	0.00105	0.00027	-	1.19	0.69	
15	-	0.00103	0.00032	-	6.13	10.03	
	-	0.00094	0.00032	-	2.81	0.90	
20	-	0.00081	0.00044	-	8.26	19.18	
	-	0.00065	0.00042	-	2.50	4.41	

A second set of tests used 50 random covariance matrices. These random covariance matrices were generated with a method described in an article by Marsaglia and Olkin [MO 94], for $m = 3, 4, ..., 10$. This method consists

of the following steps. First, a random lower triangular matrix is generated with entries chosen uniformly from [-1,1]. Then, the rows of this matrix are normalized so that the sum of the squares of the elements in each row add to one. Finally this lower triangular matrix is multiplied by it's transpose, to produce a random covariance matrix. The upper integration limits for these tests were chosen in the same way that they were chosen for the previous tests. For $m = 3$ and $m = 4$ the Schervish software [SCHRV 84] was used for comparison. For $m > 4$, however, the Schervish software again required widely varying times for each test and would not terminate after 50 hours for one of the tests with $m = 5$ so it could not be used for $m > 4$. Since accurate values for F were not known, the values used were obtained using the subregion adaptive software with requested absolute accuracy $\epsilon = 0.00025$. In Table 2 the results for these tests are reported.

Table 2: Random Covariance Matrix Results, $\epsilon = 0.005$

m	E_S	E_M	E_A	T_S	T_M	T_A	T_S/T_M
3	0.00028	0.00166	0.00006	0.02	0.48	0.12	0.03
	0.00053	0.00146	0.00007	0.03	0.52	0.10	
4	0.00109	0.00164	0.00055	574.60	1.09	0.48	527.64
	0.00377	0.00152	0.00086	3889.45	1.02	0.36	
5	-	0.00146	0.00130	-	1.61	1.00	
	-	0.00127	0.00166	-	1.36	0.91	
6	-	0.00166	0.00174	-	0.81	0.87	
	-	0.00134	0.00186	-	0.80	0.91	
7	-	0.00194	0.00248	-	1.05	1.22	
	-	0.00121	0.00199	-	1.04	1.32	
8	-	0.00199	0.00269	-	0.80	1.48	
	-	0.00173	0.00225	-	0.58	1.32	
9	-	0.00188	0.00246	-	0.79	2.04	
	-	0.00146	0.00213	-	0.62	2.67	
10	-	0.00169	0.00239	-	1.35	2.25	
	-	0.00136	0.00212	-	1.08	1.78	

In order to investigate the effect of the size of the requested accuracy on the computation times for the the three algorithms, a third set of tests were performed. These tests used the same sample size and matrices as the first set of tests, but also used a decreasing sequence of three ϵ values ($\epsilon = 0.01, 0.001, 0.0001$), for $m = 3, 4, 5, 6$. Results obtained without excessive workstation time are given in Table 3.

Table 3: Constant $\sigma_{i,j}$ Covariance Matrix Results, Decreasing ϵ

m	ϵ	E_S	E_M	E_A	T_S	T_M	T_A
3	0.01	0.000515	0.00160	0.000025	0.006	0.03	0.06
	0.001	0.000062	0.00029	0.000023	0.007	1.22	0.07
	0.0001	0.000004	-	0.000004	0.009	-	0.21
4	0.01	0.000403	0.00170	0.000071	0.112	0.05	0.09
	0.001	0.000040	0.00032	0.000048	0.174	2.01	0.19
	0.0001	0.000004	-	0.000015	0.382	-	0.77
5	0.01	0.000445	0.00166	0.000118	3.978	0.07	0.14
	0.001	0.000040	0.00029	0.000089	9.255	3.87	0.28
	0.0001	0.000005	-	0.000020	26.82	-	1.68
6	0.01	-	0.00155	0.000166	-	0.13	0.23
	0.001	-	0.00031	0.000117	-	6.09	0.50
	0.0001	-	-	0.000037	-	-	2.92

For all of the tests the computation times for the transformation based algorithms increased slowly with m. For the smaller m values, the subregion adaptive algorithm took less time than the Monte-Carlo algorithm and was more accurate, but for m greater than 6 or 7 the Monte-Carlo algorithm usually took less time and had comparable accuracy. For the $m = 3$ the Schervish algorithm was faster than the other two algorithms, but these tests show that for m greater than 4 or 5, it might not be feasible to use the Schervish algorithm for practical computations. For the random covariance matrix tests, the results suggest that either the subregion adaptive or Monte-Carlo algorithm can reliably produce multivariate normal probabilities accurate to two or three decimal

digits in one or two seconds of DECstation 3100 time, for problems with as many as ten dimensions. Both the Schervish and the Monte-Carlo algorithms require rapidly increasing amounts of computation time when ϵ is decreased.

6 Concluding Remarks

The results reported in this article suggest that it is possible to reliably compute moderately accurate multivariate normal probabilities for practical problems with as many as ten variables, in a few seconds or less of workstation time, if relatively straightforward transformations of the integrals are initially carried out. It is then possible to use standard numerical integration software for the computations. For higher accuracy work, the simple Monte-Carlo algorithm converges slowly (this is expected theoretically) and the use of some type of adaptive algorithm is probably necessary.

References

[BEG 91] Berntsen, J., Espelid, T.O. and Genz, A. (1991) 'Algorithm 698: DCUHRE-An Adaptive Multidimensional Integration Routine for a Vector of Integrals', *ACM Transactions on Mathematical Software* **17**, pp. 452-456.

[CW 91] Cox, D. R. and Wermuth, N. (1991) 'A Simple Approximation for Bivariate and Trivariate Normal Integrals', *International Statistical Review* **59**, pp. 263-269.

[DON 73] Donnelly, T. G. (1973) 'Algorithm 462: Bivariate Normal Distribution', *Communications of the ACM* **16**, p. 638.

[DW 90] Drezner, Z. and Wesolowsky, G. O. (1990) 'On the Computation of the Bivariate Normal Integral', *Journal of Statistical Computation and Simulation* **35**, pp. 101-107.

[MO 94] Marsaglia, G. and Olkin, I. (1984) 'Generating Correlation Matrices', *SIAM Journal of Scientific and Statistical Computing* **5**, pp. 470-475.

[SCHRV 84] Schervish, M. (1984) 'Multivariate Normal Probabilities with Error Bound', *Applied Statistics* **33**, pp. 81-87.

[TONG 90] Tong, Y. L. (1990) *The Multivariate Normal Distribution*, Springer-Verlag, New York.