Simulating Normal Rectangle Probabilities and Their Derivatives: The Effects of Vectorization

by

Vassilis Argyrou Hajivassiliou*

September 1992, Revised July 1993

Abstract

An extensive literature in econometrics and in numerical analysis has considered the computationally difficult problem of evaluating the multiple integral representing the probability of a multivariate normal random vector constrained to lie in a rectangular region. A leading case of such an integral is the negative orthant probability, implied by the multinomial probit (MNP) model used in econometrics and biometrics. Classical parametric estimation of this model requires, for each trial parameter vector and each observation in a sample, evaluation of a normal orthant probability and its derivatives with respect to the mean vector and the variance-covariance matrix. Several Monte Carlo simulators have been developed to approximate the orthant probability integral and its linear and logarithmic derivatives that limit computation while possessing properties that facilitate their use in iterative calculations for statistical inference. In this paper, I discuss Gauss and FORTRAN implementations of 13 simulation algorithms, and I present results on the impact of vectorization on the relative computational performance of the simulation algorithms. I show that the 13 simulators differ greatly with respect to the degree of vectorizability: in some cases activating the CRAY-Y/MP4 vector facility achieves a speed-up factor in excess of 10 times, while in others the gains in speed are negligible. Evaluating the algorithms in terms of lowest simulation root-mean-squared-error for given computation time, I find that (1) GHK, an importance sampling recursive triangularization simulator, remains the best method for simulating probabilities irrespective of vectorization; (2) the crude Monte-Carlo simulator CFS offers the greatest benefits from vectorization; and (3) the GSS algorithm, based on "Gibbs resampling," emerges as one of the preferred methods for simulating logarithmic derivatives, especially in the absence of vectorization.

E-mail: vassilis@econ.yale.edu
Address: Cowles Foundation for Research in Economics
Yale University
30 Hillhouse Ave.
New Haven, CT 06520

[*]I am indebted to Yoosoon Chang and to Anne Royalty for expert research assistance, and to Jim Heckman for very useful comments. This research was partly financed through NSF grant SES-92-11913. NSF also provided the supercomputer resources for this project through access to the CRAY-Y/MP4 at the National Center for Supercomputer Applications at the University of Illinois at Urbana-Champaign.

Simulating Normal Rectangle Probabilities and Their Derivatives: The Effects of Vectorization

1 Introduction

An extensive literature in econometrics and in numerical analysis[1] has considered the problem of evaluating the multiple integral

$$P \equiv P(\mathbf{B}; \mu, \Omega) = \int_a^b n(v-\mu, \Omega)dv \equiv \mathbf{E_V 1}(V \in \mathbf{B}), \tag{1}$$

where V is a m-dimensional normal random vector with mean μ, covariance matrix Ω, and density $n(v-\mu,\ \Omega)$, and $\mathbf{1}(V \in \mathbf{B})$ is an indicator for the event $\mathbf{B} = \{V \mid a < V < b\}$. A leading case of such an integral is the negative orthant probability, where $\mathbf{B} = \{V \mid V < 0\}$.[2] The problem is computationally difficult unless the dimension of the integral is less than four or the covariance matrix Ω has a special structure, such as a factorial structure with a small number of factors.

The *multinomial probit* (MNP) model, which is of particular interest in econometrics and biometrics for modelling discrete response behaviour, has cell probabilities that are negative orthant probabilities, with μ and Ω depending on unknown parameters and, in general, on covariates.[3] In this model, a random sample of observations on N individuals, indexed by $i = 1, \cdots, N$, is available. Each individual evaluates all J (finite) available (mutually exclusive and exhaustive) choices, and selects the one that gives the highest utility. Alternative j has observable attributes X_j and yields

[1]See Clark (1961), Daganzo (1980), Davis and Rabinowitz (1984), Dutt (1973, 1976), Fishman (1973), Hammersley and Handscomb (1964), Horowitz, Sparmonn, and Daganzo (1981), Moran (1984), Owen (1956), Rubinstein (1981), Stroud (1971), and Thisted (1988).

[2]Where convenient, I write $P(\mathbf{B}; \mu, \Omega)$ as $P(a, b; \mu, \Omega)$, or when $a = -\infty$, as $P(b; \mu, \Omega)$. Note that $P(b; \mu, \Omega) \equiv P(0; \mu - b, \Omega)$ is the cumulative multivariate normal distribution, also denoted $\mathcal{N}(b; \mu, \Omega)$. This setup covers all cases of interest, since components V_i for which both limits are infinite can be margined out analytically, and components V_i with a_i finite and $b_i = +\infty$ can be converted to the previous case by a reversal of sign.

[3]For example, see McFadden (1986).

(random) latent utility $y_j^* = X_j\beta + \epsilon_j$. Observed choice is represented by the index k such that $\{k | y_k^* = max\{y_1^*, \cdots, y_J^*\}\}$.

Classical estimation of this model requires, for each trial parameter vector and each observation in a sample, evaluation of (1) and of its linear and logarithmic derivatives with respect to μ and Ω.[4] Hajivassiliou (1993) explains that in a typical example with $J = 16$, $N = 1000$, and X consisting of 20 explanatory variables, classical estimation of β and $\Omega \equiv E\epsilon\epsilon'$ by standard numerical quadrature[5] would require longer than 3 months of CRAY-1 CPU!

Before the advent of simulation estimation methods, researchers tried to ease this computational burden by relying on distributional assumptions, which unfortunately implied typically unrealistic restrictions on the allowed choice pattern. Examples are the *extreme-value* assumptions of McFadden (1973) leading to analytic expressions for $P(\mathbf{B}; \mu, \Omega)$ in the form of the *multinomial logit* model; the scalar Ω structure of Hausman and Wise (1978), and the factor-analytic structure of Heckman (1981), either implying computationally tractable (1). McFadden (1989) and Pakes and Pollard (1989) developed the method of simulated moments, which requires only *unbiased*, though not necessarily highly accurate, simulations for (1). In view of this, an extensive survey study by Hajivassiliou, McFadden, and Ruud (1992) examined the performance of 13 simulation algorithms that have been developed for approximation of (1) and its derivatives that limit computation while possessing properties that facilitate their use in iterative calculations for statistical inference. Section 2 overviews these simulation methods and summarizes some of the Hajivassiliou et al (1992) findings, which were obtained on PC-486 microcomputers using the GAUSS matrix language. Section 3 describes the test problems used to investigate the operational properties of the methods and outlines GAUSS and FORTRAN implementations of them. Section 4 discusses the issue of vectorizability of the various simulators using the coding of three

[4]Alternative Bayesian estimation methods exist for this problem as well. These methods require the evaluation of integrals like (1) only for the calculation of the posterior density, instead of repeatedly. The dimension of the integration needed is now augmented from m by the number of unknown parameters that are to be estimated. See McCulloch and Rossi (1993) and Geweke, Keane, and Runkle (1993).

[5]which would be *highly* inaccurate in such context

particular algorithms as illustration.[6] This Section also discusses timing experiments that suggest that vectorization can have a very significant impact on the computational performance of the various algorithms. Section 5 overviews the comparative root-mean-square ($RMSE$) characteristics of the 13 algorithms for a given expenditure in computation time and investigates the impact of vectorization on their relative performance. A key finding is that the rankings change significantly depending on whether vectorization is activated or not. Section 6 concludes the paper.

2 Simulation Methods

Subsection 2.1 gives a brief overview of the characteristics of the 13 simulators. The linear and logarithmic derivatives of choice probabilities are listed in Subsection 2.2. Subsection 2.3 discusses very briefly the simulation approaches considered in this paper. A more detailed presentation of the algorithms and their properties is given in Hajivassiliou (1993), Section 4.

2.1 Overview

The general principles involved in the construction of the simulation algorithms are the following: sequential sampling, acceptance/rejection, the distinction between unbiased and asymptotically unbiased simulation, the method of importance sampling, and variance-reduction methods, specifically antithetic variates and control variates. See Hendry (1984) for a discussion of fundamental concepts in Monte-Carlo integration and its use in econometrics. Table 1 summarizes the methods and the mnemonics adopted. A complete listing of all acronyms used in this paper appears in the last table (8). The reader is referred to Hajivassiliou (1993) for more detailed descriptions of the simulation algorithms and their properties. In summary, the first method, termed the crude frequency simulator, is the most direct one in that it computes the sample

[6]See Modi (1988) for a detailed discussion of the concepts of "vectorization" and "parallel-processing".

frequency of simulated draws and uses them as approximations for the true probabilities. Next, a generalization of the CFS employs the technique of importance sampling, thereby converting the sample frequency to a weighted sample frequency that can have a smaller sampling variance. This is called the normal importance sampling (NIS) simulator. The kernel-smoothed frequency simulator (KFS) generalizes the CFS by replacing the discrete zero-one outcome of the binomial experiment by a continuous outcome on the [0,1] interval. This simulator is designed to overcome the discontinuities in the CFS with respect to the parameters of the underlying normal distribution. The fourth through seventh simulators, the Stern-decomposition simulator (SDS), the Geweke-Hajivassiliou-Keane simulator (GHK), the parabolic cylinder function (PCF) simulator, and the Deák Chi-Square Simulator (DCS), are also applications of importance sampling like the NIS and possess the property of smoothness, also possessed by the KFS. Whereas the KFS is generally a biased simulator, these simulators are both smooth and unbiased. These simulators differ according to the importance sampling distribution that they use. All of the simulators, with the exception of ARS and GSS, are simulators of P and its derivatives. Methods GSS and ARS do not produce estimates for probabilities and derivatives, but only of the logarithmic derivatives.[7]

The simulators in a second group apply specifically to the logarithmic derivatives of P, because these simulators address directly the problem of sampling from a truncated normal distribution. The acceptance/rejection simulator (ARS) is another importance sampling technique which additionally filters out draws that fall outside an acceptance region determined by the truncation. The Gibbs sampler simulator (GSS) is an alternative method for sampling from the truncated multivariate normal distribution. The GSS is smooth in the distribution parameters, but the ARS is not.

A third approach is taken in the sequentially unbiased simulators (SUS), which construct unbiased simulators of $1/P$. The last method, approximately unbiased simulators (AUS), comprises a family of simulators that are approximately unbiased for $1/P$. The members of this family can be constructed from most of the simulators of P

[7]Two versions of ARS are actually studied, ARSE, based on an exponential comparison density, and ARST, based on a truncated normal comparison density.

Table 1

Simulators for P, $\nabla_\mu P$, $\nabla_\Omega P$, $\nabla_\mu \log P$, and $\nabla_\Omega \log P$

Name of Simulator	**Mnemonic**	**Unbiased for P**	**Unbiased for ∇P**	**Unbiased for $\nabla \log P$**
Crude Frequency Simulator	CFS	y	n	n
Normal Importance Sampling Simulator	NIS	y	n	n
Kernel-Smoothed Frequency Simulator	KFS	y$*$	n	n
Stern Decomposition Simulator	SDS	y	n	n
Geweke-Hajivassiliou-Keane Simulator	GHK	y	n	n
Parabolic Cylinder Function Simulator	PCF	y	n	n
Deák Chi-square Simulator	DCS	y	n	n
Acceptance/Rejection Simulator	ARS	–	–	y
Gibbs Sampler Simulator	GSS	–	–	y$**$
Sequentially Unbiased Simulator	SUS	n	n	n
Approximately Unbiased Simulator	AUS	n	n	n

$*$ Window parameter must approach 0.

$**$ Number of Gibbs resamplings must approach ∞.

in the first group.

2.2 Derivatives of Rectangle Probabilities

The derivatives of (1) with respect to μ and Ω can be written:

$$\nabla_\mu P(\mathbf{B};\mu,\Omega) = \tag{2}$$

$$\Omega^{-1}\int_{-\infty}^{+\infty} \mathbf{1}(v\in\mathbf{B})(v-\mu)n(v-\mu,\Omega)dv \equiv \Omega^{-1}\mathbf{E}_{\mathbf{V}}\mathbf{1}(V\in\mathbf{B})(V-\mu),$$

$$\nabla_\Omega P(\mathbf{B};\mu,\Omega) = \tag{3}$$

$$(1/2)\Omega^{-1}\int_{-\infty}^{+\infty} \mathbf{1}(v\in\mathbf{B})[(v-\mu)(v-\mu)'-\Omega]n(v-\mu,\Omega)dv\cdot\Omega^{-1} \equiv$$

$$(1/2)\Omega^{-1}\mathbf{E}_{\mathbf{V}}\mathbf{1}(V\in\mathbf{B})[(V-\mu)(V-\mu)'-\Omega]\cdot\Omega^{-1}.$$

These formulas imply

$$\nabla_\mu \log\ P(\mathbf{B};\mu,\Omega) \equiv \nabla_\mu P(\mathbf{B};\mu,\Omega)/P(\mathbf{B};\mu,\Omega) = \Omega^{-1}\mathbf{E}_{V|\mathbf{B}}(V-\mu), \tag{4}$$

$$\nabla_\Omega \log \; P(\mathbf{B};\mu,\Omega) \equiv \nabla_\mu P(\mathbf{B};\mu,\Omega)/P(\mathbf{B};\mu,\Omega) =$$
$$(1/2)\Omega^{-1}\mathbf{E}_{V|\mathbf{B}}[(V-\mu)(V-\mu)' - \Omega]\cdot\Omega^{-1}, \tag{5}$$

where "$\mathbf{E}_{V|\mathbf{B}}$" denotes expectation with respect to the conditional density $n(v-\mu,\Omega,\mathbf{B}) \equiv \mathbf{1}(v \in \mathbf{B})n(v-\mu,\Omega)/P(\mathbf{B};\mu,\Omega)$. Note that (2) and (3) are partial moments of the density, and (4) and (5) are conditional moments. A full derivation of these formulae appears in Hajivassiliou and McFadden (1990).

It is useful for statistical applications to develop techniques for approximating (4)–(5) as well as (1)–(3). See Hajivassiliou (1993) for a review of available simulation estimation methods for LDV models, where it is explained how these estimators rely on simulators for expressions (1)–(5). Table 2 summarizes these results. I use the following acronyms: MSM refers to the Method of Simulated Moments of McFadden (1989) and Pakes and Pollard (1989); SSML denotes the Smoothly Simulated Maximum Likelihood estimator of Börsch-Supan and Hajivassiliou (1992); MSS1 is the Method of Simulated Scores approach of Hajivassiliou and McFadden (1990); SEM1 represents the Simulated EM algorithm of Ruud (1991); finally, MSS2 refers to an alternative MSS estimator due to van Praag et al (1986) and Hajivassiliou and McFadden (1990); and SEM2 is an alternative SEM algorithm proposed by van Praag et al (1992).

Table 2: Simulation Estimators for LDV Models

Estimator	**Simulated Functions**
MSM	(1)
SSML	(1)
MSS1	(4)–(5)
SEM1	(4)–(5)
MSS2	(1)–(3)
SEM2	(1)–(3)

Define $h(v)$ to be the polynomial array

$$h(v) = \begin{bmatrix} 1 & (v-\mu)'\Omega^{-1} \\ \Omega^{-1}(v-\mu) & \frac{1}{2}[\Omega^{-1}(v-\mu)(v-\mu)'\Omega^{-1} - \Omega^{-1}] \end{bmatrix}. \tag{6}$$

Then, equations (1)–(3) can be written

$$H \equiv H(\mathbf{B}; \mu, \Omega) = \int_{-\infty}^{+\infty} \mathbf{1}(v \in \mathbf{B}) h(v) n(v - \mu, \Omega) dv \equiv \mathbf{E}_{\mathbf{V}} \mathbf{1}(V \in \mathbf{B}) h(V), \quad (7)$$

and equations (4) and (5) can be written

$$H_C = \mathbf{E}_{V|\mathbf{B}} h(V) \equiv H/P(\mathbf{B}; \mu, \Omega), \quad (8)$$

This implies that the northwest element of H gives (1), the remainder of the first row gives (2), and the southeast subarray gives (3), as follows:

$$H = \begin{bmatrix} (1) & (2) \\ (2)' & (3) \end{bmatrix} \quad (9)$$

The analogous elements of H_C give one, (4), and (5):

$$H_C = \begin{bmatrix} 1 & (4) \\ (4)' & (5) \end{bmatrix} \quad (10)$$

Both the GAUSS and FORTRAN implementations of the thirteen simulation algorithms discussed in subsection 3.2 below were designed to return H and H_C as defined here.

2.3 Simulation Procedures

For statistical inference, it is often unnecessary to achieve high numerical accuracy in evaluating (1)–(5). For example, simulating P by the frequency of the event $\mathbf{1}(v \in \mathbf{B})$ in a number of Monte Carlo draws comparable to sample size will tend to produce statistics in which the variance introduced by simulation is at worst of the same magnitude as the variance due to the observed data. Further, when probabilities appear linearly across observations in an estimation criterion, independent unbiased simulation errors are averaged out. Then, a small, fixed number of draws per probability to be evaluated will be sufficient with increasing sample size to reduce simulation noise at the same rate as noise from the observed data.[8] This makes it computationally feasible to

[8]In outline, suppose $\hat{\theta}$ is an M-estimator that solves

$$0 = N^{1/2} \mathbf{E}_N s(\hat{\theta}, \eta),$$

treat statistical problems that require repeated evaluation of high-dimensional normal rectangle probabilities. McFadden (1989), Pakes and Pollard (1989), and McFadden and Ruud (1990) analyze the statistical properties of such estimators, and Hajivassiliou (1993) and Hajivassiliou and Ruud (1993) survey simulation estimation methods for LDV models.

The first seven methods considered here, CFS through DCS, simulate H. Methods ARS and GSS simulate H_C by drawing from the conditional distribution of V given $V \in \mathbf{B}$. Method SUS approximates $H_C = H/P$ using independent unbiased simulators of H and $1/P$. Method AUS is similar but uses a biased simulator of $1/P$ to speed computation. Some versions of AUS require a positive simulator of P. This is guaranteed by NIS, SDS, GHK, PCF, DCS, and by KFS if a positive kernel is used. The number of draws required in ARS is random. The remaining methods will in general use a fixed number of repetitions, which may in statistical applications increase with sample size.

To understand what is perhaps the most intuitive simulation method, write the random vector V as

$$V = \mu + \Gamma\eta, \tag{11}$$

where η is an independent standard normal vector of dimension m and Γ is a lower triangular Choleski factor of Ω, so $\Omega = \Gamma\Gamma'$. A simple approach to approximating (1) is

where s is an approximation (involving Monte Carlo elements η) to a function $\sigma(\theta)$ of P and its derivatives that has expectation zero at the true parameter θ^o, and $\mathbf{E}_N$ denotes empirical expectation over an independent sample of size N. Then, one can write

$$0 = N^{1/2}\mathbf{E}_N s(\hat{\theta},\eta) \equiv N^{1/2}\mathbf{E}_N\sigma(\theta^o) + N^{1/2}\mathbf{E}_N[s(\theta^o,\eta) - \sigma(\theta^o)] + N^{1/2}\mathbf{E}_N[\sigma(\hat{\theta}) - \sigma(\theta^o)]$$
$$+N^{1/2}\mathbf{E}_N[s(\hat{\theta},\eta) - \sigma(\hat{\theta}) - s(\theta^o,\eta) + \sigma(\theta^o)].$$

Under standard regularity conditions, the first term is asymptotically normal, reflecting the noise in the observations, and the third term is proportional to $\sqrt{N}(\hat{\theta} - \theta^o)$. The last term will be of order $o_p(1)$ for simulators that satisfy a stochastic equicontinuity condition. When s is a smooth function of crude frequency simulators of P, $\nabla_\mu P$, etc., obtained using R Monte Carlo draws, the second term will behave like $\sqrt{N/R}$ times an expression that is asymptotically normal, so that it will be comparable in magnitude to the first term when R and N are proportional. If, in addition, there is any averaging out of simulation noise across observations, the second term may be of order $o_p(1)$ when R and N are proportional, or comparable in magnitude to the first term for fixed R. This argument summarizes Theorem 1, Section 5 of Hajivassiliou and McFadden (1990).

to make repeated Monte Carlo draws for η, use (11) to calculate V for each parameter vector, and then form an empirical analogue of the expectation in (7). Below I call this the *crude frequency simulator* (CFS) of $P(\mathbf{B};\mu,\Omega)$ and its derivatives. Similarly, a crude frequency simulator for H_C can be formed by rejecting draws of V that do not satisfy the conditioning event $V \in \mathbf{B}$, and then forming an empirical analogue of the conditional expectation in (8) using the accepted draws. Crude frequency simulators have both advantages and disadvantages. Their main advantage is that they are quick to compute and ideal for vectorization. Section 4 below justifies this claim.[9] They are *not* continuous in parameters, however, as they exhibit jumps at parameter values yielding draws of V on the boundary of $\mathbf{B}$.[10] These discontinuities, however, can slow iterative parameter search.[11] In principle, the accuracy of the CFS can be improved by use of antithetic and control variates, which are general simulation variance-reduction techniques. See Hendry (1984) for a description of these variance-reduction approaches, and Hajivassiliou et al (1992) for explicit constructions of random antithetic grids and of suitable control variates.

3 Monte Carlo Experiments

Subsection 3.1 describes the test problems used to evaluate the operational characteristics of these algorithms, and subsection 3.2 summarizes their implementations in a series of GAUSS and FORTRAN procedures for simulation of multivariate normal rectangle probabilities and the derivatives of these probabilities with respect to the

[9]To ensure this property, one should use direct methods for the generation of normal variates, e.g., the inverse c.d.f. technique, as opposed to accept/reject techniques. See Devroye (1986) for an overview of such methods.

[10]These discontinuities do not prevent use of these simulators for statistical inference. If η is not redrawn when parameters change so that "chatter" is avoided, then these simulators are piecewise constant in parameters, and the manifolds on which discontinuities occur are linear. These properties imply a stochastic equicontinuity property that is sufficient to make the simulators well-behaved in statistical inference; see McFadden (1989).

[11]See Quandt(1984) for a discussion of iterative parameter search algorithms and their requirements in terms of continuity and differentiability of the function to be searched over.

mean and covariances of the normal distribution.[12]

3.1 Description of the Test Problems

A case that yields multivariate normal orthant probabilities that are easily calculated analytically or by quadrature is the one-factor model,

$$V = \mu + S\eta + \Lambda\epsilon, \tag{12}$$

where S is an $m \times m$ diagonal matrix with diagonal elements s_i, Λ is an $m \times 1$ array of factor loadings, μ is an $m \times 1$ vector of means, η is an $m \times 1$ vector of independent standard normal variates, and ϵ is an independent standard normal scalar variate. Given ϵ, the constraints require

$$S^{-1}(a - \mu - \Lambda\epsilon) < \eta < S^{-1}(b - \mu - \Lambda\epsilon). \tag{13}$$

The expressions for P, $\partial P/\partial\mu_m$, and $\partial P/\partial\lambda_m$ can be evaluated by one-dimensional Gaussian quadrature, and in a few cases evaluated analytically, which will provide benchmarks to gauge the accuracy of the 13 simulation algorithms in the experiments below. One analytic case occurs when the factor is loaded only on the last alternative, so that it is equivalent to a change in the scale of V_m. Hajivassiliou et al (1992) show how the transformation s_m to $(s_m^2 + \lambda_m^2)^{1/2}$ and λ_m to zero yields analytic expressions for P and its derivatives.

A second analytic case occurs when $a = -\infty$, $b = 0$, $\mu = 0$, S is the identity matrix, and Λ is a vector of ones. This corresponds, of course, to the independent normal distribution restricted on the negative orthant. Then,[13]

$$P = \int_{-\infty}^{+\infty} \Phi(-\epsilon)^m \phi(-\epsilon) d\epsilon = 1/(m+1), \tag{14}$$

[12] Both versions of the programs are available from the author upon request, or by anonymous `FTP` from the site `econ.yale.edu`, subdirectory `pub/vassilis/simulation`.

[13] The Hajivassiliou et al (1992) study also considered the case of a random effect combined with an autoregressive structure of order one, for which no analytic solution exists. I do not consider this model here, since it does not offer any significant additional insights with respect to the importance of vectorization, which is the focus of the present paper. The key feature of this model is to allow one to consider increasing the dimension m. Typically, however, m is *much* smaller than the number of repetitions NR, and so the potential impact of vectorization is primarily over NR.

$$\partial P/\partial \mu_m = \int_{-\infty}^{+\infty} \Phi(-\epsilon)^{m-1}\phi(-\epsilon)^2 d\epsilon,$$

$$\partial P/\partial \lambda_m = \int_{-\infty}^{+\infty} \Phi(-\epsilon)^{m-1}\phi(-\epsilon)^2 \epsilon d\epsilon.$$

3.2 Description of Simulation Algorithms

The simulation methods presented in this paper have been coded in GAUSS and in FORTRAN. Both versions of the programs are available from the author upon request or by anonymous `FTP` from the site `econ.yale.edu:pub/vassilis/simulation`. Each simulator procedure requires the following standard inputs; the interpretation of some inputs may vary from routine to routine, and not all are used in all routines:

`M`	Dimension of the multivariate normal
`VMU`	Mean of multivariate normal, an $\mathtt{M} \times 1$ vector
`W`	Covariance matrix of multivariate normal, an $\mathtt{M} \times \mathtt{M}$ array
`WI`	Inverse of `W`
`C`	Lower triangular Choleski factor of `W`, an $\mathtt{M} \times \mathtt{M}$ array
`A`	Lower bound of rectangle, an $\mathtt{M} \times 1$ vector. {When the lower bound is $-\infty$, set $\mathtt{A} = (-1.0E10) * \mathtt{ONES(M,1)}$.}
`B`	Upper bound of rectangle, an $\mathtt{M} \times 1$ vector. {When the upper bound is ∞, set $\mathtt{B} = (1.0E10) * \mathtt{ONES(M,1)}$.}
`NR`	Number of repetitions
`U`	Random variates, an $\mathtt{M} \times \mathtt{R}$ array
`PARM`	Parameters and constants for the simulation

The simulators all return $\{\mathtt{P}, \mathtt{HU}, \mathtt{HC}\}$, where `P` is the scalar rectangle probability, `HU` is the $(\mathtt{M}+1) \times (\mathtt{M}+1)$ array of unconditional partial moments (6), and `HC` is the $(\mathtt{M}+1) \times (\mathtt{M}+1)$ array of conditional moments (7). Parts of the output not provided by a simulator are set to $-\mathtt{999}$.

In the FORTRAN implementation, two additional inputs are required, `MMAX` and `NRMAX`, specifying the maximum values of `M` and `NR` allocated at compilation time.

The programs include code for all statistical functions, spherical transformations, and antithetics routines that are required by the simulation algorithms, and hence are self-contained.

3.3 Comparative Performance

For each computational experiment I used five hundred Monte Carlo repetitions of all thirteen simulation algorithms. The number of simulations in calculating empirical expectations of the H matrix function was chosen endogenously by the programs, so as to require approximately the same time for each simulation method. Hence, the simulators can be ranked in terms of lowest $RMSE$ for a given investment in computation time. The specific results summarized in this Section were obtained through the GAUSS implementation of the routines on 486/33MHz Personal Computers. The FORTRAN timings reported in Section 4 were obtained on the CRAY-Y/MP4 supercomputer with 4 processors at the National Center for Supercomputer Applications (NCSA) of the University of Illinois at Urbana-Champaign.

Figures 1 and 2 describe the first series of experiments I consider here, in which truncated normal vectors V of dimension $m = 2$ were generated having the factor structure (12). Figure 1 gives the 6 types of variance/covariance structure[14] studied, with $\omega_1 =1$, $\omega_2 =\{1 \text{ or } 8\}$, and $\rho_{12} =\{0, .6, \text{or } .9\}$, where $var\ V_i \equiv \omega_i$ and $cov(V_1, V_2) \equiv \rho_{12} \cdot \omega_1 \cdot \omega_2$. Figure 2 describes the 14 different rectangles/restrictions[15] investigated. These rectangles were chosen so as to analyze the effect of symmetry around either or both axes, as well as the location of them either close to the center of the distribution or far out in the tails. Hence, the experiments incorporate the 84 cases $\{A1, A2, \ldots, N5, N6\}$ obtained by combining these 6 correlation structures with the 14 sets of restrictions.

Table 3 gives the true probability $P(\mathbf{B}; \mu, \Omega)$ for each of the 84 cases, calculated analytically when possible or by 40-point Gaussian Hermite quadrature. It also gives other characteristics of the experiments, e.g., the condition number and determinant of each of the six Ω's studied. The probabilities range from about 10^{-12} to above 0.95.

[14]indexed by a number from 1 to 6

[15]indexed by a letter from A to N

Table 3: Characteristics of 84 Experiments
Restrictions {A-N} $\times$ Variance-Covariances {1-6}
Exact Probability of Each Case[*]

	V.-Cov.1	V.-Cov.2	V.-Cov.3	V.-Cov.4	V.-Cov.5	V.-Cov.6
Restr. A (3×2)	0.01133	0.02113	0.02143	0.04625	0.05656	0.07206
Restr. B (3×2)	0.34042	0.34112	0.34135	0.04960	0.04973	0.04974
Restr. C (3×2)	0.01133	0.00095	8.601e-08	0.04625	0.03617	0.02069
Restr. D (3×2)	4.101e-05	0.00081	0.00174	0.00017	3.674e-05	9.799e-11
Restr. E (3×2)	0.00092	0.00023	2.662e-08	0.00013	1.366e-05	1.118e-11
Restr. F (3×2)	4.101e-05	1.036e-09	1.123e-13	0.00017	4.088e-06	1.633e-12
Restr. G (5×5)	0.12977	0.15259	0.15422	0.03720	0.03134	0.00792
Restr. H (5×5)	0.12977	0.08309	0.04057	0.03720	0.01960	0.00129
Restr. I (10×2)	0.02272	0.02272	0.02272	0.09276	0.09276	0.09276
Restr. J (10×2)	0.68269	0.68269	0.68269	0.09948	0.09948	0.09948
Restr. K (10×2)	0.02272	0.02272	0.02272	0.09276	0.09276	0.09276
Restr. L (6×4)	0.00135	0.00121	0.00074	0.16260	0.16295	0.16304
Restr. M (6×4)	0.95192	0.95311	0.95444	0.19688	0.19738	0.19741
Restr. N (6×4)	0.00135	0.00121	0.00074	0.16260	0.16295	0.16304
Cond.Num.(Ω)	1.000	3.999	18.999	63.999	101.139	345.446
Det.(Ω)	1.000	0.640	0.190	64.000	40.960	12.160

[*] After the name of each restriction type, the dimensions of the rectangle appear in parentheses.

For each one of the 84 cases studied, the methods were rated in terms of root-mean-squared-error relative to the best method for that case, e.g., a $RMSE$ Rating of 0.5 means that the method in question exhibited double the $RMSE$ of the method with the lowest $RMSE$ for that case.[16] Overall, in the 84 cases studied the average ratings of the various methods are obtained for simulating the probability expression (1) and the logarithmic derivatives (4)–(5).[17]

Each figure presenting the relative $RMSE$ ratings consists of fours parts: In the

[16]Interested readers may request from the authors considerably more detailed tables that report bias, variance, mean-squared-error, quantiles, robust statistics, and timing results for all cases studied.

[17]Simulation of the linear derivatives (2)–(3) is more akin to simulating the probability (1) itself compared to simulating the *logarithmic* derivatives. Hence, the linear derivative results are not reported here. Interested readers can request them from the author.

first, average ratings are given for each of the 6 variance-covariance structures across all 14 restrictions. Hence this part allows one to select the simulator with the best $RMSE$ performance irrespective of the type of restriction for specific variance-covariance structure. The second and third parts of each figure allow one to do the converse, namely select the best simulator for specific types of restrictions, irrespective of the variance-covariance structure. The fourth and final part of each figure reports summary ratings of the simulators averaged across *all* 84 restriction and covariance types.

Table 4: Average $RMSE$ Ratings* across 84 Cases
PC Gauss Results

Probabilities		**Logarithmic Derivatives**	
GSS	**	ARSE	0.021
ARSR	**	AUS	0.087
ARSE	**	SUS	0.088
KFS	0.025	ARSR	0.122
SDS	0.143	DCS	0.138
CFS	0.203	KFS	0.169
DCS	0.204	CFS	0.250
PCF	0.297	SDS	0.367
NISE	0.357	NIST	0.410
NIST	0.610	NISE	0.430
SUS	0.762	GHK	0.442
AUS	0.762	GSS	0.461
GHK	0.934	PCF	0.669

* Average $RMSE$ Rating $\equiv \frac{1}{84}\sum_{k=1}^{84} RMSE(\text{given method in case k})/RMSE(\text{best method for case k})$.

** ARSE, ARSR, and GSS not applicable.

Figures 3 and 4 summarize the $RMSE$ results from these 84 experiments first reported in Hajivassiliou et al (1992), performed on 486/33 PC microcomputers using the GAUSS matrix language. The main findings are: GHK can be recommended as unambiguously the best performing method for simulating normal orthant probabilities, achieving an overall $RMSE$ rating of 93%, compared to about only 76% for the next best methods, AUS and SUS. For simulating derivative expressions, the PCF method exhibited the highest overall rating, 67%, while GSS and GHK came next. It is interesting to note that the normal importance sampling methods, $NISE$ and

$NIST$, seemed to perform fairly well, bettered only by 3 methods in the case of probabilities, and by 2 methods in the case of derivatives. GHK appeared more robust than all other methods, in that it performed at or near the top in each one of the 84 cases studied.[18] In particular, it performed even more impressively relative to the other algorithms in the most difficult cases of either high correlation among the elements of V and/or very low probability mass in the restriction region. The consequences for the performance of the simulation methods of increasing the number of simulations used is another important issue studied by Hajivassiliou et al (1992).

In the next two Sections, I discuss the concept of vectorization and investigate the impact vectorization has on the questions raised above.

4 Vectorization Issues

The 13 simulation algorithms discussed in Section 3.2 differ greatly in the extent to which they can be "vectorized." This term refers to certain characteristics of an algorithm that lead to greater speed benefits on modern supercomputers equipped with a so-called "vector facility," as, for example, the one on the CRAY-Y/MP4 at the NCSA in Illinois. Such a mechanism achieves quicker calculations on adjacent elements of a vector. For a good introduction to the concepts of "vectorization" and "parallel-processing," see Modi (1988).

Without delving deeply into computer science issues, a simple illustration is the following. Of the 13 simulation algorithms studied, an algorithm that is highly vectorizable is CFS, since there exists complete independence between the generation of each draw. In contrast to this, the GSS method is highly non-vectorizable because, while it requires independent operations across different simulations, for a given simulation the operations are dependent in the Markov chain structure that defines the Gibbs resampling technique.

It is important to note that the PC-GAUSS results discussed in the previous section discriminate against methods like GSS, that are *not substantially vectorizable*, since

[18]Indeed, GHK achieved a first-place rating in over 70 of the 84 cases studied.

GAUSS is particularly efficient for vector operations. In timings reported in Table 5 below using GAUSS, vectorized-, and non-vectorized-FORTRAN codes, I confirm that methods that are difficult to vectorize then gain in relative speed. The impact of vectorization techniques on the performance of the simulation algorithms studied is potentially a very important issue that I investigate in this paper.

To illustrate the differing degrees of vectorizability of algorithms, I present pseudo-code in matrix `GAUSS`-like language for three of the thirteen simulators considered in this paper, `CFS`, `GHK`, and `GSS`. First, I give some notation:

- Problem: Draw random deviates from the truncated density $\underset{M \times 1}{y^*} \sim N(\underset{M \times 1}{\texttt{MU}}, \underset{M \times M}{\texttt{W}})$

  such that $\underset{M \times 1}{\texttt{A}} \leq y^* \leq \underset{M \times 1}{\texttt{B}}$.
- $\texttt{R}$ = number of simulations, $\texttt{G}$ = number of Gibbs resamplings.
- $\underset{(M-1) \times 1}{\texttt{MKVEC}}$: vector $\{1, 2, \cdots, M\}$, *excluding* K.
- $\texttt{RMATU} \equiv \{M \times R$ matrix of $U[0,1]$ random variates$\}$,
  $\texttt{RMATN} \equiv \{M \times R$ matrix of $N[0,1]$ random variates$\}$.
- Cholesky factor $\underset{M \times M}{\texttt{C}}$: defined by $\texttt{W} = \texttt{C} * \texttt{C}'$ and $\texttt{WI} \equiv \texttt{W}^{-1}$.
- `SUMC(X)`, `MAXC(X)`, `MINC=(X)`: calculate $M \times 1$ vectors of the SUMS, MAXIMA, and MINIMA respectively of the columns of the $M \times N$ matrix X.
- `PHEE(X)`: calculate the N(0,1) c.d.f., i.e., $\Phi(X)$.
- `PHEEINV(X)`: calculate the N(0,1) *inverse* c.d.f., i.e., $\Phi^{-1}(X)$.

As the following shows, the `CFS` algorithm requires only vector operations, without the need for any `DO`-loops:

The `CFS` Algorithm

```
V=C*U;
AA=(A-MU)*ONES(1,NR);
BB=(B-MU)*ONES(1,NR);

FF=(V.<BB) .* (V.>AA);
FF=MINC(FF);
P=SUMC(FF)/NR;
```

The GHK algorithm is also quite vectorizable, only requiring a DO-loop over the (small) dimension M:

The GHK Algorithm

```
J=1;
IVEC=1;
TA=PHEE((A1-MU1/C1)*ONES(1,NR);
TB=PHEE((B1-MU1/C1)*ONES(1,NR);
TT=PHEEINV(U[1,.] .* TA + (1-U[1,.]) .* TB);
WGT=TB-TA;
DO WHILE J<M;
   J=J+1;
   TA=PHEE((AJ-MUJ/CJ)*ONES(1,NR);
   TB=PHEE((BJ-MUJ/CJ)*ONES(1,NR);
   IVEC=IVEC|J;
   WGT=WGT .* (TB-TA);
ENDO;
P=SUMC(WGT')/NR;
```

In complete contrast to the two algorithms above, the GSS method cannot be vectorized to any significant degree: it requires three DO-loops, over simulations R, Gibbs resamplings G, and dimension M. The key problem is that unlike M, which is “small,” the outer loops are “large” (over NR and G). This is because Gibbs resamplings follow a Markov scheme, and are thus by nature not independent:

The GSS Algorithm

```
I=0;
V=(A+B)/2-MU;                                       % initial vector
DO WHILE I<R;                                       % loop over repetitions
   I=I+1;
   J=0;
   DO WHILE J<G;                                    % loop over resamplings
      J=J+1;
      K=K+0;
      DO WHILE K<M;                                 % loop over k
         K=K+1;
         MKVEC=(1,...,M excluding K);
         CSDk=sigma(k given -k);                    % see text
         CMUk=mu(k given -k);                       % see text
         TA=PHEE((Ak-MUk-CMUk)/CSDk);
         TB=PHEE((Bk-MUk-CMUk)/CSDk);
         UKJ=U[k,j];
         Vk=CMUk + CSDk*PHEEINV(UKJ*TA + (1-UKJ)*TB);
      ENDO;
   ENDO;
   VBAR=VBAR+V;
ENDO;
VBAR=INV(W)*VBAR/NR;
```

To investigate the impact different vectorizability has on the operational characteristics of the 13 simulators, I performed the following timing experiment: The 13 algorithms were run over all the 84 correlation/restrictions configurations described above and their computational performance was noted. Two different implementations of the algorithms were compared, the first in the (vectorized) matrix language GAUSS, and the second in FORTRAN. In the FORTRAN case, the experiments were repeated once with and once without activating the vector-processor on the CRAY-Y/MP4. Table 5 gives the relative speed characteristics of the various simulators in both the vectorized GAUSS implementation as well as the vectorized and non-vectorized FORTRAN versions, averaged over all 84 correlation/restriction configurations described above. The common characteristic of all the numbers reported is that the time required to generate x simulations using the CFS algorithm is the same along a given column, where $x = \{500, 1000, 2000, 5000, \text{and } 10000\}$. For example, I found that with GAUSS vectorization one can generate on the average 148 GHK, 12 GSS, and 69 ARSRE simulations in the same amount of time taken for 10,000 CFS simulations. With FORTRAN vectorized on the CRAY-Y/MP4 these numbers remain comparable (123, 56, and 10 respectively). These numbers, however, change dramatically when the vector facility is switch off: they now become 1651 for GHK, 144 for GSS, and 1,000 for ARSE simulations. Hence, this simple experiment suggests that the $RMSE$ rankings could change substantially without vectorization. In particular, note that without vectorization GSS rises considerably in prominence, which accords with the findings in Hajivassiliou and Ruud (1993) of very satisfactory performance of the MSS variants based on GSS simulations. This question is the focus of the next Section.

Number of simulations matching the time taken by a given number of CFS simulation

Method	**PC-GAUSS-V**					**Cray-FORTRAN-S**					**Cray-FORTRAN-V**				
CFS	500	1000	2000	5000	10000	500	1000	2000	5000	10000	500	1000	2000	5000	10000
NISE	26	40	92	218	429	129	257	513	1283	2572	13	24	44	105	202
NIST	10	11	21	50	98	50	99	199	497	993	10	10	15	37	73
KFS	24	42	84	208	409	220	437	874	2184	4377	20	34	67	160	287
SDS	10	10	15	35	70	52	104	206	515	1036	10	10	13	31	59
GHK	10	16	31	73	148	83	165	329	822	1651	10	14	26	63	123
PCF	20	36	59	169	333	134	270	542	1350	2697	14	21	46	110	184
DCS	10	10	14	35	64	47	103	192	511	932	10	10	11	29	53
ARSE	10	10	14	35	69	50	100	200	498	1000	10	10	12	28	56
ARSR	10	10	10	24	48	30	61	122	305	611	10	10	10	20	40
GSS	10	10	10	10	12	14	14	29	71	144	10	10	10	10	10
AUS	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
SUS	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

PC-GAUSS-V: Results on a 486/33 PC using the matrix language GAUSS
Cray-FORTRAN-S: Results on CRAY-Y/MP4 in FORTRAN77, Vector Facility not active
Cray-FORTRAN-V: Results on CRAY-Y/MP4 in FORTRAN77, Vector Facility active

Table 5

5 Results

The 13 simulation algorithms in the FORTRAN implementation were run on the CRAY-Y/MP4 of NCSA. Each parameter configuration was always run in pairs: one with vectorization activated achieved by using the `-Zp` switch on the `CF77` compiler, and another with vectorization deactivated (option `-Zc`)[19]. The effects on the performance of the algorithms of activating and deactivating vectorization on the CRAY are summarized in Tables 6 and 7, to be contrasted to the results reported in Table 4, obtained on the 486/33 PC's using GAUSS. Recall that the ratings are in terms of simulation $RMSE$ for given investment of CPU time.

The first major finding is that GHK, the importance sampling recursive triangularization simulator, remains the best method for simulating probabilities irrespective of vectorization: Its overall $RMSE$ performance rating is always at the top, being 0.934 with PC-GAUSS, dropping to 0.761 with Cray-Scalar, and rising to 0.835 with Cray-Vector. The second major finding is that the crude Monte-Carlo simulator CFS offers the greatest benefits from full vectorization: Activating the vector-facility on the Cray moves CFS from 7th best with a rating of only 0.187 all the way to 2nd best, with a rating of 0.404. The final key finding is that the GSS algorithm, based on "Gibbs resampling," emerges as the preferred method for simulating logarithmic derivatives in the absence of vectorization: In the PC-GAUSS and Cray-Vectorized cases it comes out 2nd and 3rd best with ratings of 0.461 and 0.321 respectively, while in the Cray-Scalar case it jumps to the top of the rankings with a rating of 0.742.

Figures 3–8 give more analytic results, with the PC-Gauss results summarized in Figures 3–4, the Cray-Scalar in Figures 5–6, and the Cray-Vectorized results in Figures 7–8. Figures 3, 5, and 7, present $RMSE$ performance results on simulating the probability expression (1), while the last figure of each set, namely Figures 4, 6, and

[19]In addition to vectorization, the option `-Zp` also allows "micro-tasking" across the 4 processors. I experimented with turning this feature off through the switch `-Zv` but this had only insignificant effects on the results.

8, give the results for simulating the logarithmic derivatives (4) and (5). As explained earlier, in each such figure the results are presented in four parts, (a) to (d). The first two report performance ratings for each of the 14 restriction types, averaging across all 6 correlation structures, with parts (a) and (b) giving cases A–G and H–N respectively. Part (c) reports performance for each of the 6 types of correlation, averaging across the 14 restriction types. Finally, in part (d) I report overall performance, averaging across all 84 correlation/restriction cases. These Figures allow one to assess the performance of the various simulation algorithms for the different correlation/restriction configurations, as well as the summary performance averaged across all configurations. The impact vectorization has on the relative $RMSE$ rankings of the simulators is quite striking.

Table 6: Average $RMSE$ Ratings* across 84 Cases
CRAY-Y/MP4 FORTRAN, Vector Facility Active

Probabilities		Logarithmic Derivatives	
GSS	**	ARSE	0.011
ARSR	**	SUS	0.024
ARSE	**	AUS	0.026
KFS	0.006	DCS	0.041
SDS	0.033	ARSR	0.049
DCS	0.045	KFS	0.058
NISE	0.113	SDS	0.165
SUS	0.185	NISE	0.177
AUS	0.200	NIST	0.200
PCF	0.254	GHK	0.310
NIST	0.256	GSS	0.321
CFS	0.404	CFS	0.610
GHK	0.835	PCF	0.726

* Average $RMSE$ Rating $\equiv \frac{1}{84}\sum_{k=1}^{84} RMSE$(given method in case k)/$RMSE$(best method for case k).

** ARSE, ARSR, and GSS not applicable.

Table 7: Average $RMSE$ Ratings[*] across 84 Cases
CRAY-Y/MP4 FORTRAN, Vector Facility Active

Probabilities		Logarithmic Derivatives	
GSS	**	ARSE	0.016
ARSR	**	AUS	0.038
ARSE	**	SUS	0.045
KFS	0.001	DCS	0.059
SDS	0.106	KFS	0.061
DCS	0.114	ARSR	0.092
CFS	0.187	CFS	0.182
PCF	0.366	GHK	0.252
NISE	0.384	SDS	0.276
AUS	0.473	NIST	0.300
SUS	0.551	NISE	0.359
NIST	0.591	PCF	0.697
GHK	0.761	GSS	0.742

* Average $RMSE$ Rating $\equiv \frac{1}{84}\sum_{k=1}^{84} RMSE(\text{given method in case k})/RMSE(\text{best method for case k})$.

** ARSE, ARSR, and GSS not applicable.

6 Conclusions

The problem of evaluating multivariate normal probabilities and their derivatives is an important one in econometrics and biometrics because such expressions appear in leading econometric models, such as the *multinomial probit* (MNP) and other limited dependent variable models based on normality. Classical estimation of these models requires, for each trial parameter vector and each observation in a sample, evaluation of such probability expressions and their derivatives. The problem is computationally difficult unless the dimension of the integral is less than four or the covariance matrix Ω has a special structure, such as a factorial structure with a low number of factors.

This paper examined the vectorizability properties of Gauss and FORTRAN implementations of the 13 Monte Carlo techniques surveyed in Hajivassiliou et al (1992), and presented results on the impact of vectorization on the relative computational performance of these simulation algorithms. I used several test problems to investigate the operational properties of the methods, focussing on $RMSE$ rankings for a given expenditure of CPU time, and summarized the computational experience with them. I also examined the impact of increasing the number of simulations NR. I confirmed the main finding in Hajivassiliou et al (1992) that for simulating orthant probabilities the GHK simulator, based on an importance sampling recursive triangularization scheme, appears overall the best method. This result holds irrespective of vectorization. In addition, I found that the crude Monte-Carlo algorithm offers the greatest benefits from vectorization, while the GSS simulator performs excellently in simulating logarithmic derivatives unbiasedly, especially in the absence of vectorization.

In this paper I concentrated on the importance of the degree of vectorizability of the simulation algorithms for their relative performance rankings. Another interesting issue to study would be the computational advantages of using simultaneous processing on massively parallel computers, like the Connection Machine CM5 at the NCSA, to distribute the computations of each of the NR repetitions. This could open up the calculation of econometric models of hitherto unmanageable complexity. I leave this for future research.

Table 8: List of Acronyms

Acronym	Description
ARSE	Acceptance/Rejection Simulator, Exponential Comparison Density
ARSR	Acceptance/Rejection Simulator, Recursive Normal Comparison Density
AUS	Approximately Unbiased Simulator
CFS	Crude Frequency Simulator
DCS	Deàk Chi-squared Simulator
GHK	Geweke-Hajivassiliou-Keane Simulator
GSS	Gibbs Sampling Simulator
KFS	Kernel-smoothed Frequency Simulator
LDV	Limited Dependent Variable Model
MNP	Multinomial Probit Model
MSM	Method of Simulated Moments[1]
MSS1	Method of Simulated Scores[2]
MSS2	Alternative Method of Simulated Scores[3]
NCSA	National Center for Supercomputer Applications, University of Illinois[4]
NISE	Normal Importance Sampling Simulator with Exponential Sampler
NIST	Normal Importance Sampling Simulator with Truncated Normal Sampler
PCF	Parabolic Cylinder Function Simulator
RMSE	Root Mean Squared Error
SDS	Stern Decomposition Simulator[5]
SEM1	Simulated EM-algorithm[6]
SEM2	Simulated EM-algorithm[7]
SSML	Smoothly Simulated Maximum Likelihood Estimation[8]
SUS	Sequentially Unbiased Simulator

[1] McFadden (1989), Pakes and Pollard (1989)
[2] Hajivassiliou and McFadden (1990)
[3] Hajivassiliou and McFadden (1990), van Praag and Hop (1987)
[4] Urbana-Champaign
[5] Stern (1992)
[6] Ruud (1992)
[7] van Praag et al (1992)
[8] Börsch-Supan and Hajivassiliou (1993)

References

[1] Börsch-Supan, A. and V. Hajivassiliou 1990. Smooth Unbiased Multivariate Probability Simulators for Maximum Likelihood Estimation of Limited Dependent Variable Models. Cowles Foundation Discussion Paper No. 960., forthcoming in the *Journal of Econometrics*.

[2] Clark, C. 1961. The Greatest of a Finite Set of Random Variables. *Oper. Res.* **9**:145-162.

[3] Daganzo, C. 1980. *Multinomial Probit.* New York: Academic Press.

[4] Davis, P. and P. Rabinowitz 1984. *Methods of Numerical Integration.* New York: Academic Press.

[5] Devroye, L. 1986. *Non-Uniform Random Variate Generation.* New York: Springer.

[6] Dutt, J. 1973. A Representation of Multivariate Normal Probability Integrals by Integral Transforms. *Biometrika* **60**:637-645.

[7] Dutt, J. 1976. Numerical Aspects of Multivariate Normal Probabilities in Econometric Models. *Ann. Econ. Social Measurement* **5**:547-562.

[8] Fishman, G. 1973. *Concepts and Methods of Digital Simulation.* New York: Wiley.

[9] Geweke, J. 1989a. Bayesian Inference in Econometric Models Using Monte Carlo Integration. *Econometrica* **57**:1317-1339.

[10] Geweke, J. 1989b. Efficient Simulation from the Multivariate Normal Distribution Subject to Linear Inequality Constraints and the Evaluation of Constraint Probabilities. Mimeo, Duke University.

[11] Geweke, J. 1991. Efficient Simulation from the Multivariate Normal and Student$-t$ Distributions Subject to Linear Constraints. *Computing Science and Statistics: Proc. 23rd Sympos.* pp.571-578.

[12] Geweke, J., M. Keane, and D. Runkle 1993. Alternative Computational Approaches to Inference in the Multinomial Probit Model. Working Paper, University of Minnesota.

[13] Hajivassiliou, V. and D. McFadden 1990. The Method of Simulated Scores, with Application to Models of External Debt Crises. Cowles Foundation Discussion Paper No. 967.

[14] Hajivassiliou, V., D. McFadden, and P. Ruud 1992. Simulation of Multivariate Normal Orthant Probabilities: Methods and Programs. Cowles Foundation Discussion Paper No. 1021, Yale University.

[15] Hajivassiliou, V. 1993. Simulation Estimation Methods for Limited Dependent Variable Models. In *Handbook of Statistics, Vol.11.* Edited by G.S. Maddala, C.R. Rao, and H.D. Vinod. Amsterdam: North-Holland.

[16] Hajivassiliou, V. and P. Ruud 1993. Classical Simulation Estimation Methods. In *Handbook of Econometrics, Vol. 4.* Edited by R. Engle and D. McFadden. Amsterdam: North-Holland.

[17] Hammersley, J. and D. Handscomb 1964. *Monte Carlo Methods.* London: Methuen.

[18] Hausman, J. and D. Wise 1978. A Conditional Probit Model for Qualitative Choice: Discrete Decisions Recognizing Interdependence and Heterogeneous Preferences. *Econometrica* **46**:403-426

[19] Hendry, D. 1984. Monte Carlo Experimentation in Econometrics. In *Handbook of Econometrics, Vol.2.* Edited by Z. Griliches and M. Intriligator. Amsterdam: North Holland, pp.937-976.

[20] Horowitz, J., J. Sparmonn, and C. Daganzo 1981. An Investigation of the Accuracy of the Clark Approximation for the Multinomial Probit Model, *Transportation Sci.* **16**:382-401.

[21] Keane, M. 1990. A Computationally Efficient Practical Simulation Estimator for Panel Data, with Applications to Estimating Temporal Dependence in Employment and Wages. Mimeo, University of Minnesota.

[22] McFadden, D. 1986. Econometric Analysis of Qualitative Response Models. In *Handbook of Econometrics, Vol.2.* Edited by Z. Griliches and M. Intriligator. Amsterdam: North Holland, pp.1395-1457.

[23] McFadden, D. 1981. Econometric Models of Probabilistic Choice. In *Structural Analysis of Discrete Data with Econometric Applications.* Edited by C. Manski and D. McFadden. Cambridge: MIT Press, pp. 198-272.

[24] McFadden, D. 1989. A Method of Simulated Moments for Estimation of Discrete Response Models without Numerical Integration. *Econometrica* **57**:995-1026.

[25] McFadden, D. and P. Ruud 1990. Estimation by Simulation. Working Paper, M.I.T.

[26] Modi, J.J. 1988. *Parallel Algorithms and Matrix Computation.* New York: Oxford University Press.

[27] Moran, P. 1984. The Monte Carlo Evaluation of Orthant Probabilities for Multivariate Normal Distributions. *Austral. J. Statist.* **26**:39-44.

[28] Owen, D. 1956. Tables for Computing Bivariate Normal Probabilities. *Ann. Math. Statist.* **27**:1075-1090.

[29] Pakes, A. and D. Pollard 1989. Simulation and the Asymptotics of Optimization Estimators. *Econometrica* **57**:1027-1057.

[30] Quandt, R. 1984. Computational Problems in Econometrics. In *Handbook of Econometrics, Vol.1.* Edited by Z. Griliches and M. Intriligator. Amsterdam: North Holland, pp. 1395-1457.

[31] McCulloch, R. and P. E. Rossi 1992. An Exact Likelihood Analysis of the Multinomial Probit Model. Working Paper 91-102, Graduate School of Business, University of Chicago.

[32] Rubinstein, R. 1981. *Simulation and the Monte Carlo Method.* New York: Wiley.

[33] Ruud, P. 1986. On the Method of Simulated Moments for the Estimation of Limited Dependent Variable Models. Mimeo, University of California at Berkeley.

[34] Ruud, P. 1990. A Note on Computing Multinomial Probit Estimators by Simulation. Mimeo, Department of Economics, M.I.T.

[35] Ruud, P. 1991. Extensions of Estimation Methods Using the EM Algorithm. *J. Econometrics* **49**:305-341.

[36] Stern, S. 1992. A Method for Smoothing Simulated Moments of Discrete Probabilities in Multinomial Probit Models. *Econometrica* **60**:943-952

[37] Stroud, A. 1971. *Approximate Calculation of Multiple Integrals.* New York: Prentice Hall.

[38] Thisted, R. 1988. *Elements of Statistical Computing.* Chapman-Hall.

[39] van Praag, B.M.S., and J.P. Hop 1987. Estimation of Continuous Models on the Basis of Set-Valued Observations. Working Paper, Erasmus University.

[40] van Praag, B.M.S., J.P. Hop and E. Eggink 1991. A Symmetric Approach to the Labor Market by Means of the Simulated EM-Algorithm with an Application to Married Females. Working Paper, Erasmus University.

Figure 1
Variance/Covariance Structures Studied

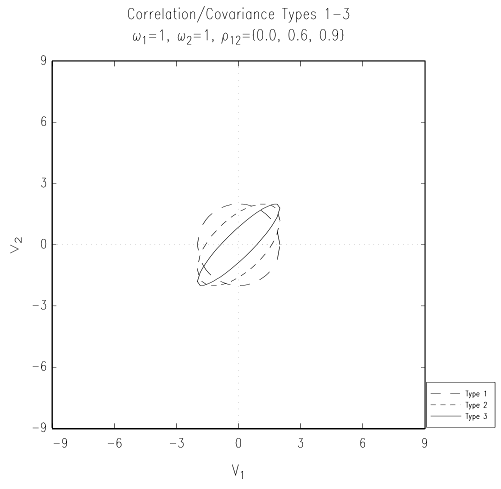


Figure 2
Rectangles/Restrictions Studied

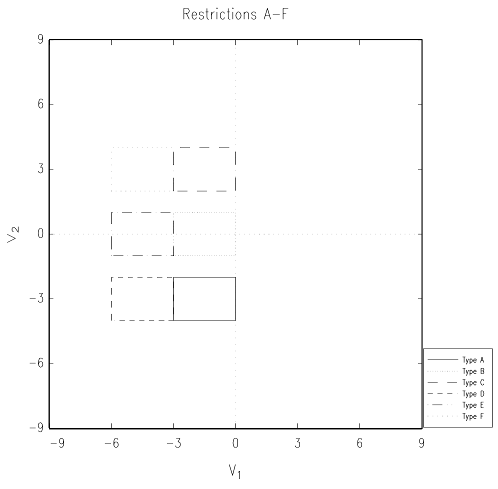


Figure 2 (continued)
Rectangles/Restrictions Studied

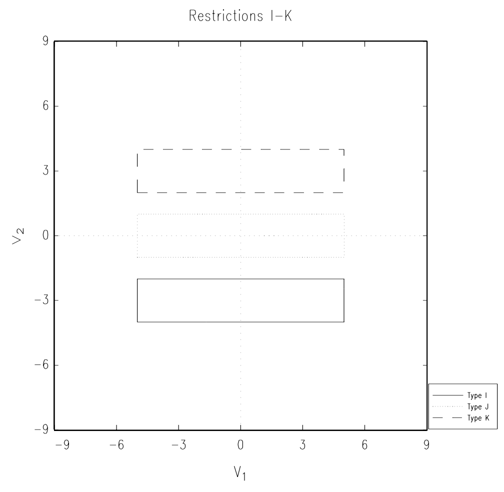
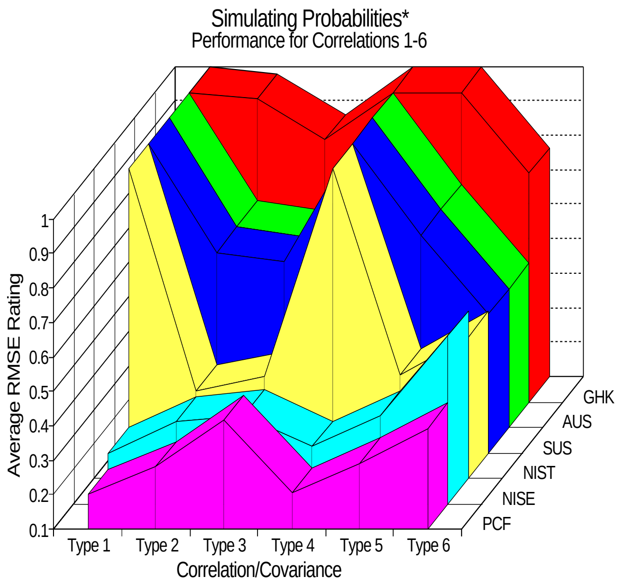


Figure 3
RMSE Ratings: PC-Gauss Results



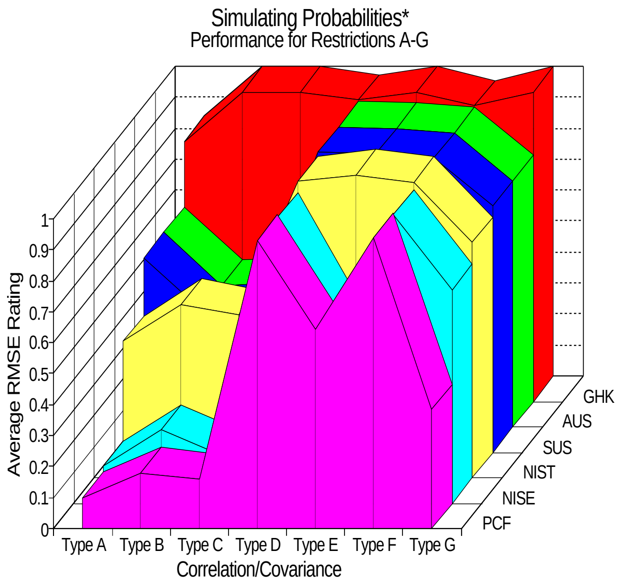
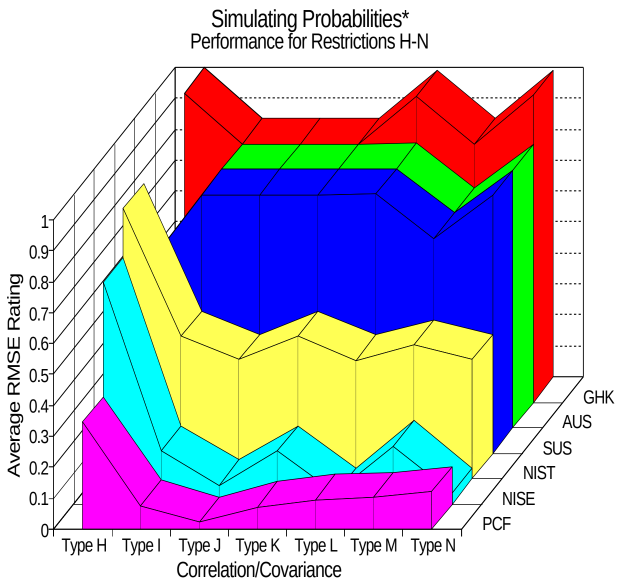


Figure 3 (continued)
RMSE Ratings: PC-Gauss Results



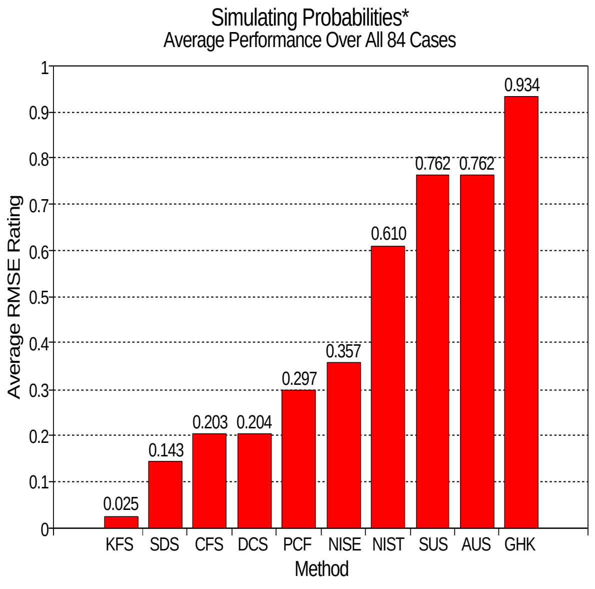
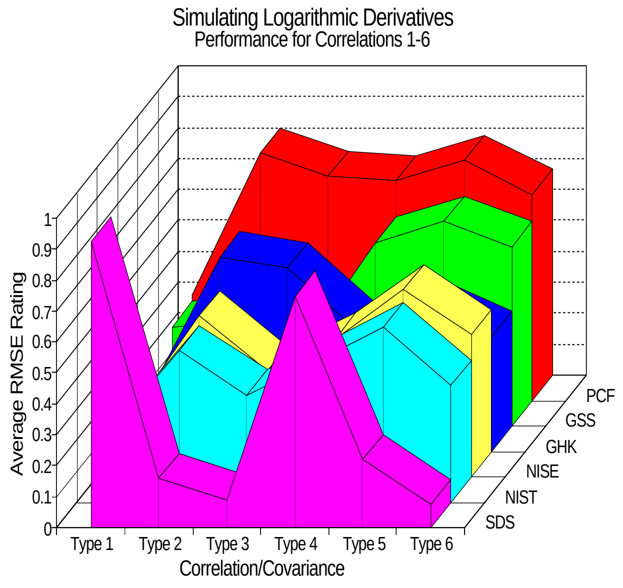


Figure 4
RMSE Ratings: PC-Gauss Results



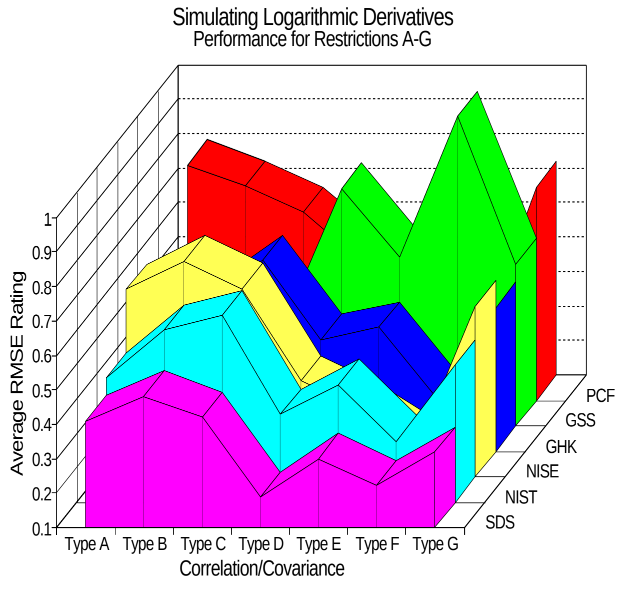
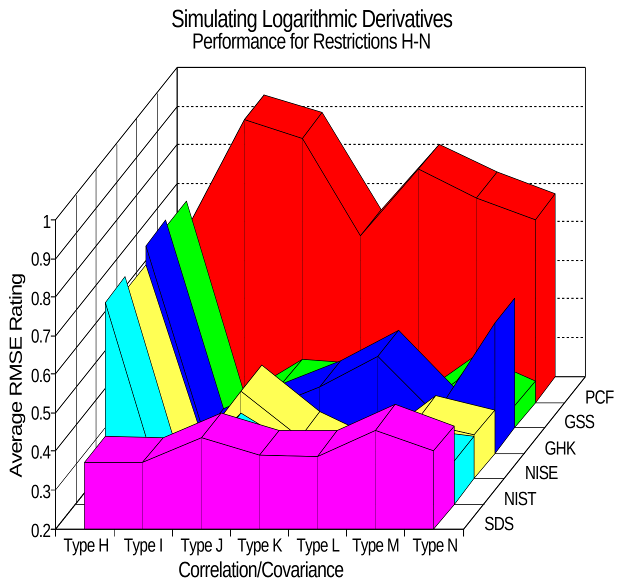


Figure 4 (continued)
RMSE Ratings: PC-Gauss Results



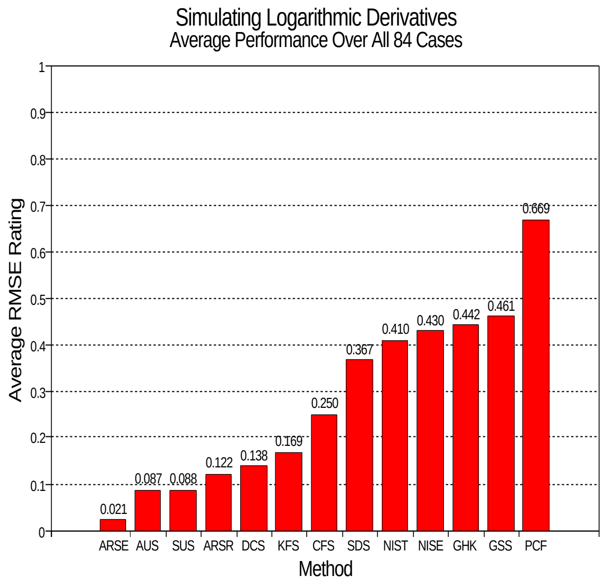
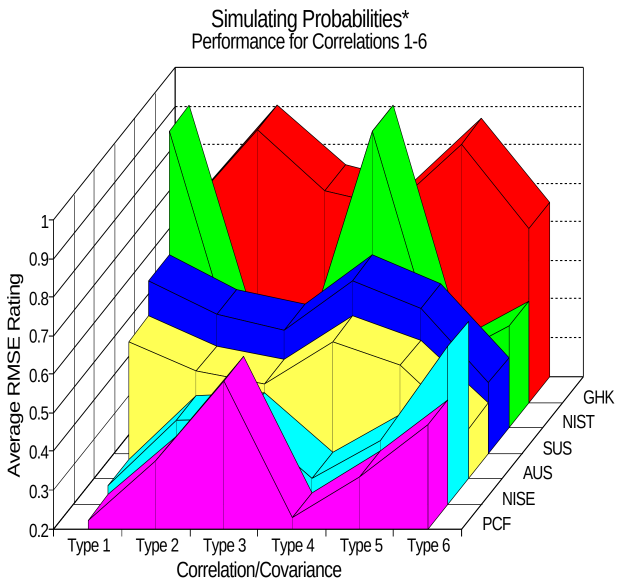


Figure 5
RMSE Ratings: Cray-Y/MP Scalar Results



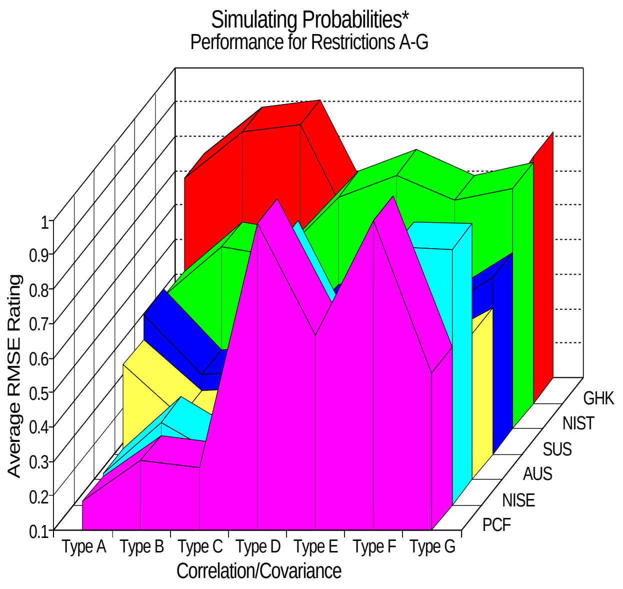
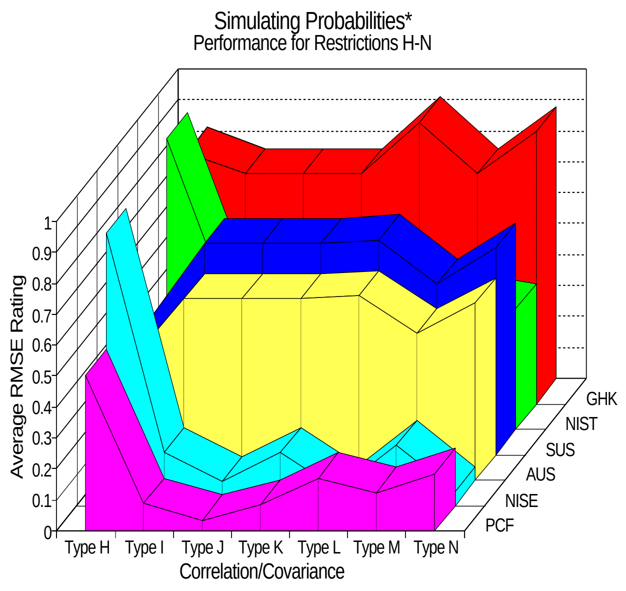


Figure 5 (continued)
RMSE Ratings: Cray-Y/MP Scalar Results



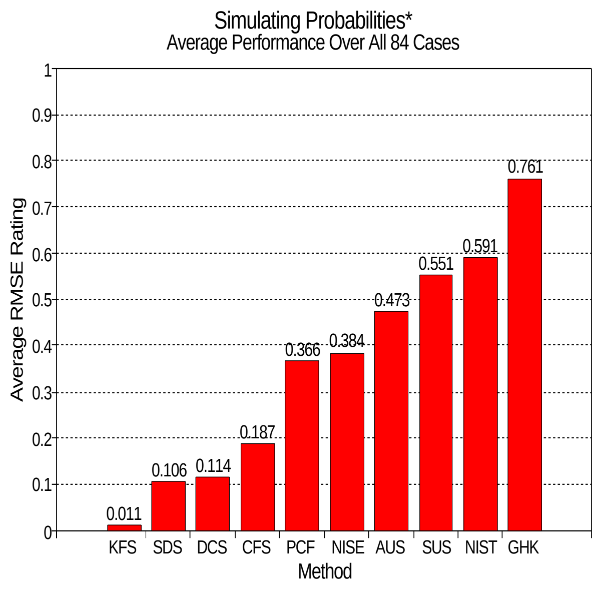
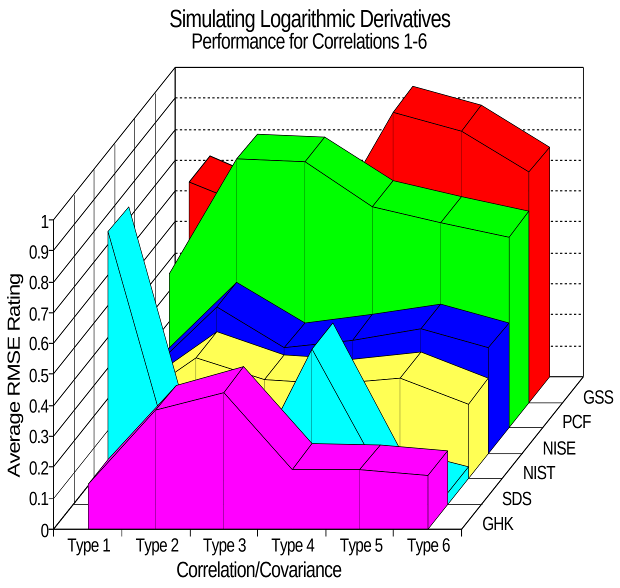


Figure 6
RMSE Ratings: Cray-Y/MP Scalar Results



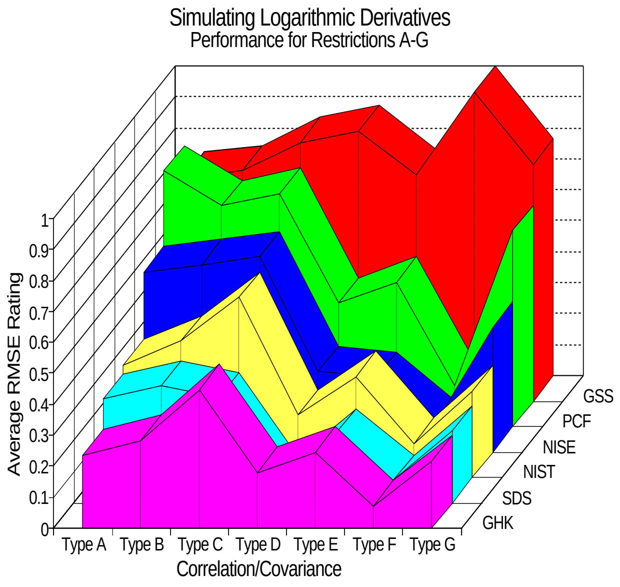
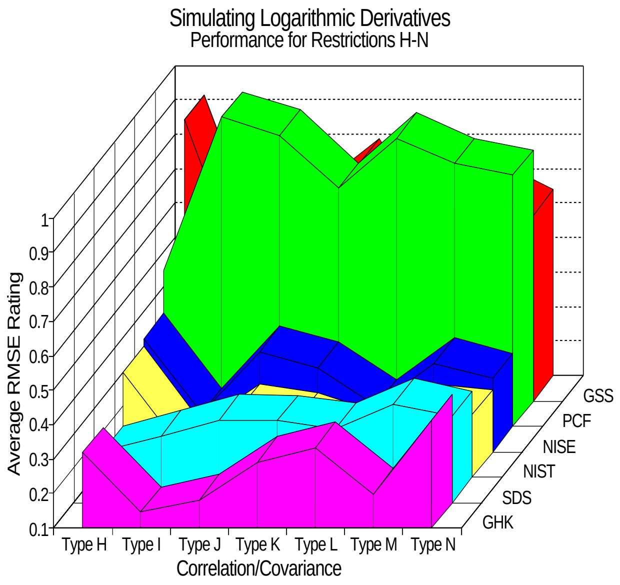


Figure 6 (continued)
RMSE Ratings: Cray-Y/MP Scalar Results



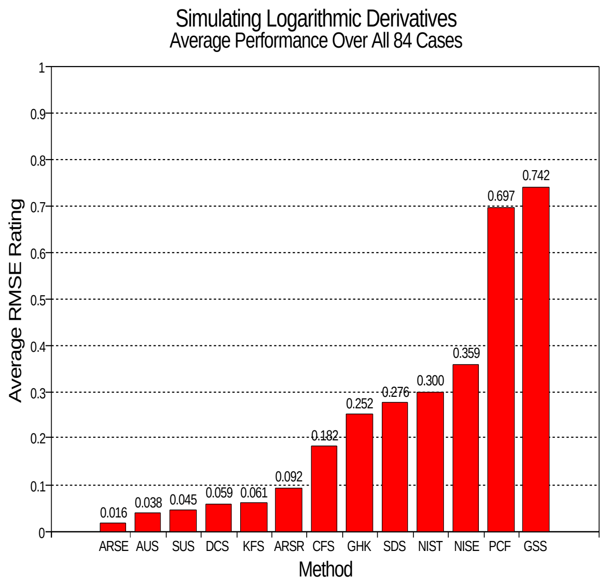
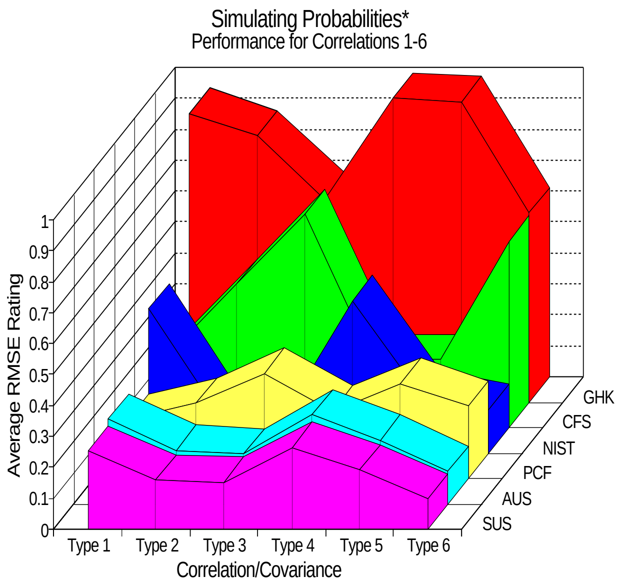


Figure 7
RMSE Ratings: Cray-Y/MP Vector Results



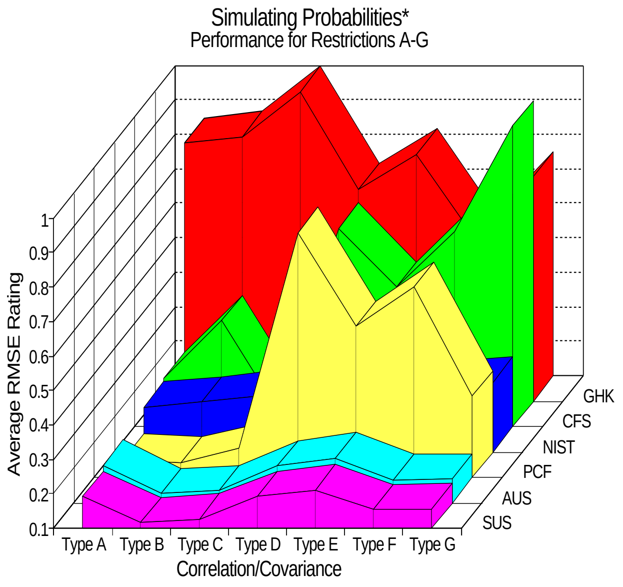
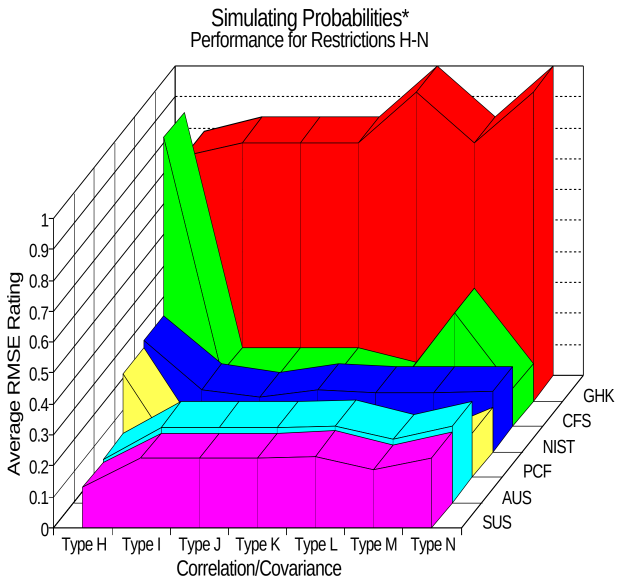


Figure 7 (continued)
RMSE Ratings: Cray-Y/MP Vector Results



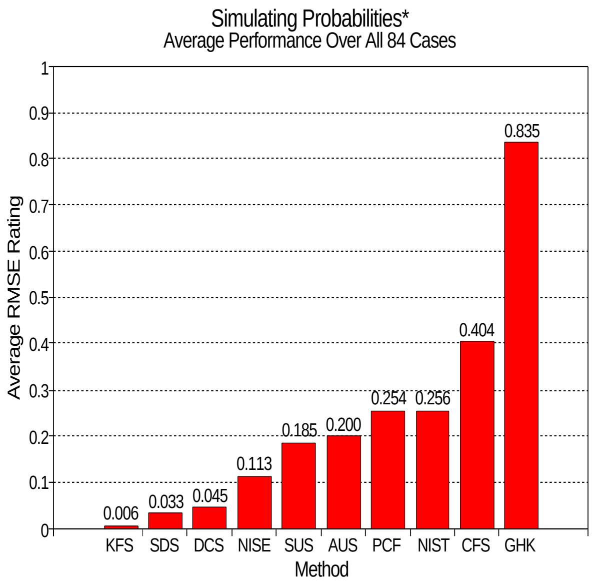
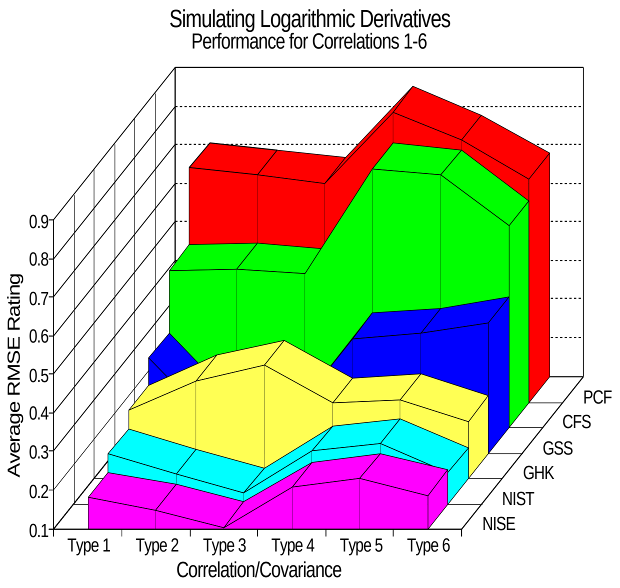


Figure 8
RMSE Ratings: Cray-Y/MP Vector Results



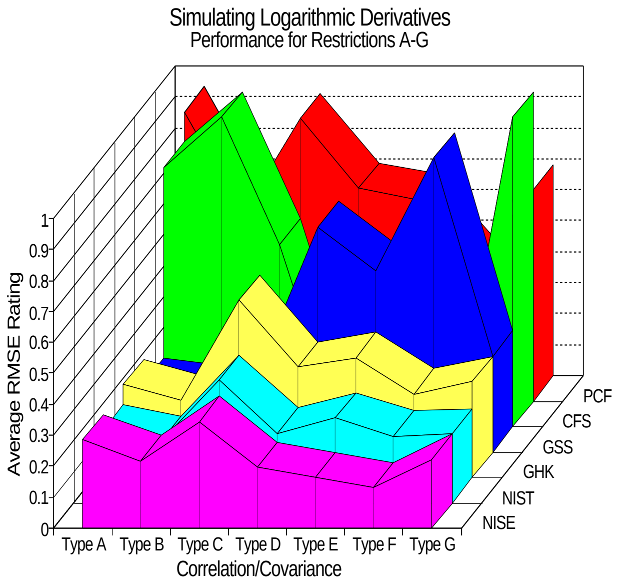
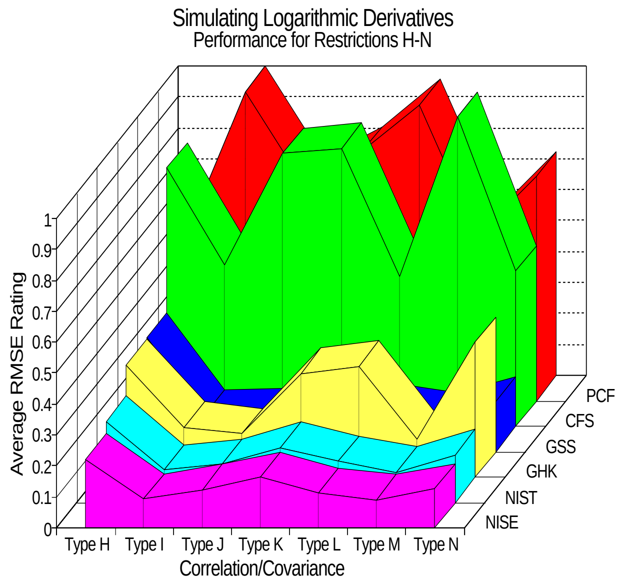
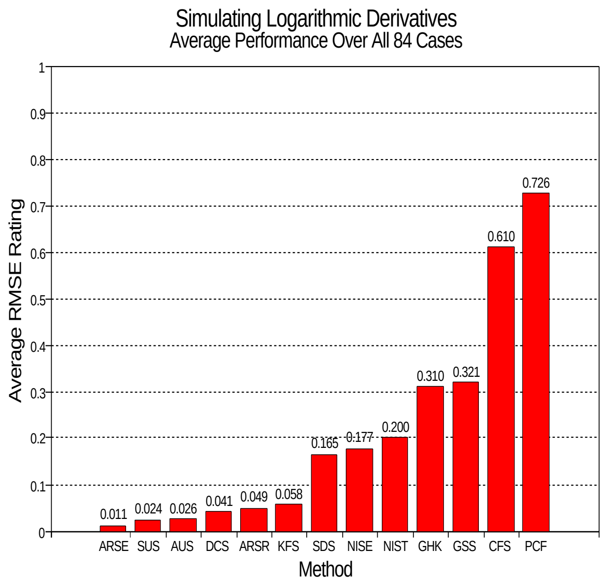


Figure 8 (continued)
RMSE Ratings: Cray-Y/MP Vector Results


Simulating Logarithmic Derivatives
Performance for Restrictions H-N
Average RMSE Rating
Type H
Type I
Type J
Type K
Type L
Type M
Type N
Correlation/Covariance
PCF
CFS
GSS
GHK
NIST
NISE


Simulating Logarithmic Derivatives
Average Performance Over All 84 Cases
Average RMSE Rating
ARSE
SUS
AUS
DCS
ARSR
KFS
SDS
NISE
NIST
GHK
GSS
CFS
PCF
Method