Monte Carlo Simulation

Tentatively Due Tuesday November 15, 2004

The details of this first computer project are described below. This being a form of take-home exam means that each person is to do his/her own work. A second project will be assigned too, but for now you can get started on the first parts of this project. As we move forward in the lectures I will explain more details of the latter problems. How the grading on this project factors into your overall grade will be discussed in class.

Background Theory

Computer aided analysis of statistical communications/signal processing problems often leads to direct Monte Carlo simulation of the system under study. A Monte Carlo simulation requires one or more sources of pseudo random numbers, with each random number source producing samples of a random variable with a specified probability distribution.

Random Number Generation

For statistical modeling purposes we would like to assume that a random number generator produces a sequence of numbers corresponding to a particular outcome of an independent identically distributed (iid) sequence of random variables. Thus the random number generator output can be used to model a white sequence. The algorithms of interest are based on congruence relations. Random number generators of this type produce a long sequence of numbers eventually repeating, but approximately uniform on the interval (0,1).

Programming languages such as MATLAB, C/C++, Fortran, Pascal, and Basic have supplied functions which generate random sequences of this type. The user typically has the option of supplying a seed number to initialize the generator to a particular starting point. An excellent discussion of random number generator theory and practice for the C language can be found in the book by Press, et.al.

Random Sequence Modification

The random number sequence can be modified in a two fundamental ways: (1) shaping the probability distribution and (2) introducing dependence (correlation) between successive sequence values (i.e., filtering the sequence). Both types of sequence modification will be discussed below.

From random variable theory we know that a random variable $y$ transformed by

---

the function \( x = g(y) \), will result in \( x \) being uniform on the interval \((0,1)\) if \( g(y) = F_y(y) \). Thus given a random number generator which produces outcomes \( x \) of \( \text{rv} \ x \), uniform on \((0,1)\), we can in theory produce random numbers, \( y \), with pdf \( f_y(y) \) by letting

\[
y = F_y^{-1}(x)
\]

where \( F_y^{-1}(y) \) is the function inverse of the cdf \( F_y(y) \). As an example suppose we wish to generate random numbers which have a normal (Gaussian) distribution. The cdf of a normal \( \text{rv} \) can be written in terms of the error function. For the transformation from uniform to normal we will need the inverse error function. This approach is not very computationally efficient due to the difficulty in computing the needed transformation function. A popular approach to this problem involves performing more than one \( \text{rv} \) transformation. recall that if \( \text{rvs} \ x \) and \( y \) are normal, independent (uncorrelated is all we need here), with zero means and equal variance \( \sigma^2 \), then the transformation

\[
r = \sqrt{x^2 + y^2}, \theta = \text{atan}(y/x)
\]

yields independent \( \text{rvs} \ r \) and \( \theta \). The \( \text{rv} \ r \) has a Rayleigh pdf of the form

\[
f_r(r) = \frac{re^{-\frac{r^2}{2\sigma^2}}}{\sigma^2}, 0 \leq r < \infty
\]

and \( \theta \) is uniform on \((0, 2\pi)\). Working the above transformation backwards says that if we can form independent \( \text{rvs} \ r \) and \( \theta \) from uniform \( \text{rvs} \), then we can obtain \( x \) and \( y \) from

\[
x = r\cos \theta, \quad y = r\sin \theta
\]

Now if we treat the random number generator output as if it produces samples from an iid sequence, then consecutive outputs from a random number generator, \( x_n \) and \( x_{n+1} \), can be used to produce the samples

\[
r = \sqrt{-2\ln x_n}, \quad \text{assumes} \sigma^2 = 1
\]

\[
\theta = 2\pi x_{n+1}
\]

Two independent normal \( \text{rv} \) samples, with zero mean and variance \( \sigma_y^2 \) can then be generated from

\[
y(n) = y_n = r\sigma_y \cos \theta
\]

\[
y(n+1) = y_{n+1} = r\sigma_y \sin \theta
\]

Dependence between random number generator samples can be introduced via filtering. If we treat the random number sequence as a discrete-time signal then the filtering operations of interest are those corresponding to linear time invariant (LTI) systems. In particular we may consider LTI systems which satisfy linear constant coefficient difference equations (LCCDEs). The correlation between pairs of signal samples in the true statistical sense is given by the autocorrelation function \( R_{yy}(k) \)

\[
R_{yy}(k) = E\{y(n+k)y(n)\}
\]
As a simple example consider the first-order difference equation
\[ y(n) = ay(n - 1) + x(n), \quad |a| < 1 \]  
(8)

If the input \( x(n) \) is a white sequence (i.e., \( R_{xx}(k) = \sigma^2_x \delta(k) \)), then in steady-state the output autocorrelation function is given by
\[ R_{yy}(k) = \sigma^2_x \frac{|a|^k}{1 - a^2} \]  
(9)

Thus the random number generator sequence applied to this system will result in an output random number sequence with the above correlation property. The distribution of the random number is in general difficult to calculate unless the input sequence is normal. If the input sequence is normal, we know that in steady-state the output will also be normal.

**Sample Moments and Histograms**

There are many techniques for characterizing a random number generator. The sample mean, sample variance, and sample covariance are simple measurement techniques to implement. Block forms and recursive forms of these algorithms are available.

**Sample Mean**

For random number sequence \( x(n) \), a sample drawn from random variable \( x \), the sample mean is
\[ m_x = E\{x\} \equiv \frac{1}{N} \sum_{k=1}^{N} x(k) = \hat{m}_x \]  
(10)

A recursive formula for computing \( \hat{m}_x \) is
\[ \hat{m}_x(N) = \hat{m}_x(N-1) + \frac{1}{N}[x(N) - \hat{m}_x(N-1)] \]  
(11)

where \( \hat{m}_x(0) = 0 \).

**Sample Variance**

To estimate the quantity \( \sigma^2_x = E\{[x - m_x]^2\} \) use
\[ \sigma^2_x \equiv \frac{1}{N} \sum_{k=1}^{N} [x(k) - \hat{m}_x]^2 = \hat{\sigma}_x^2 \]  
(12)

Note this is a biased version of the variance estimator presented in the course notes.

A recursive formula for calculating \( \hat{\sigma}_x^2 \) is
\[ \hat{\sigma}_x^2(N) = \frac{N-1}{N} \hat{\sigma}_x^2(N-1) + \frac{N-1}{N^2} [x(N) - \hat{m}_x(N-1)]^2 \]  
(13)

where \( \hat{\sigma}_x^2(0) = 0 \) and \( \hat{\sigma}_x^2(1) = 0 \). With some work this can be converted to the unbiased estimator. Basically we need to replace \( N \) with \( N-1 \).
Sample Mean Vector
Consider the random vector \( \mathbf{X} = [x(1), x(2), \ldots, x(n)]' \) which has corresponding mean vector \( \mathbf{m}_x = [m_{x_1}, m_{x_2}, \ldots, m_{x_n}] \). The sample mean vector is given by
\[
\hat{\mathbf{m}}_x = \frac{1}{N} \sum_{k=1}^{N} \mathbf{X}(k)
\] (14)
The recursive formula presented earlier for the scalar rv case can also be extended to the vector case.

Sample Covariance Matrix
The random vector \( \mathbf{X} \) defined above has covariance matrix given by
\[
\mathbf{C} = E\{[\mathbf{X} - \mathbf{m}_x][\mathbf{X} - \mathbf{m}_x]'\} = 
\begin{bmatrix}
C_{11} & C_{12} & \cdots & C_{1n} \\
C_{21} & C_{22} & \cdots & C_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
C_{n1} & C_{n2} & \cdots & C_{nn}
\end{bmatrix}
\] (15)
An estimator (biased) for \( \mathbf{C} \) is
\[
\hat{\mathbf{C}} \equiv \frac{1}{N} \sum_{k=1}^{N} \mathbf{X}(k)\mathbf{X}'(k) - \hat{\mathbf{m}}_x\hat{\mathbf{m}}_x' = \hat{\mathbf{C}}
\] (16)
A recursive formula for \( \hat{\mathbf{C}} \) is
\[
\hat{\mathbf{C}}(k) = \frac{1}{N} [(N-1)(\hat{\mathbf{C}}(N-1) + \hat{\mathbf{m}}_x(N-1) + \mathbf{X}(N)\mathbf{X}'(N))]
- \frac{1}{N^2} [(N-1)\hat{\mathbf{m}}_x(N-1) + \mathbf{X}(N)][(N-1)\hat{\mathbf{m}}_x(N-1) + \mathbf{X}(N)]'
\] (17)
where \( \hat{\mathbf{C}}(0) = 0 \) and \( \hat{\mathbf{C}}(1) = 0 \). Note this biased estimator can be converted to an unbiased estimator if desired.

Sample Autocorrelation
If we view the random number sequence \( x(n) \) as sample function from a wide-sense stationary discrete-time random process, then we can write
\[
R_{xx}(k) \equiv \frac{1}{N} \sum_{n=0}^{N + |k| - 1} x(n+k)x(n) = \hat{R}_N(k)
\] (18)
In the above expression it is assumed that the available data is the real sequence \( x(0), x(1), \ldots, x(N-1) \), thus \( \hat{R}_N(k) = 0 \) for \( |k| \geq N \). Note that this sequence is also biased since
but $\hat{R}_N(k)$ is asymptotically unbiased since $\lim_{n \to \infty} (1 - |k|/N) = 1$. The bias is not that significant since $N >> |k|$ is needed for the estimate variance to be small.

A related quantity, the power spectral density of $x(n)$, is given by

$$S_{xx}(\omega) = \sum_{k = -\infty}^{\infty} R_{xx}(\omega)e^{-jk\omega}$$

(20)

An estimator for $S_{xx}(\omega)$ is

$$S_N(\omega) = \sum_{k = -\infty}^{\infty} R_N(k)e^{-jk\omega}$$

(21)

$$= \frac{1}{N} \left| \sum_{n = 0}^{N-1} x(n)e^{-jn\omega} \right|^2 = \frac{1}{N} |X_N(\omega)|^2$$

where $X_N(\omega)$ is the discrete Fourier transform (DFT) of $x(n)$. The spectral density estimate $S_N(\omega)$ is called the periodogram.

**Histograms**

A histogram can be used to estimate points on the pdf curve from samples of a random variable. The histogram pdf estimation approach is based on the fact that

$$f_\hat{x}(x) \equiv \frac{P[x - \varepsilon < x < x + \varepsilon]}{2\varepsilon}$$

(22)

If the random variable samples, $x(1), x(2), \ldots, x(N)$ are iid then,

$$P[x - \varepsilon < x < x + \varepsilon] \equiv \frac{\text{number of } x(n) \text{ between } x - \varepsilon \text{ and } x + \varepsilon}{N}$$

(23)

Define the indicator function

$$I_x(x(k)) = \begin{cases} 1, & x - \varepsilon < x(k) < x + \varepsilon \\ 0, & \text{otherwise} \end{cases}$$

(24)

A reasonable estimate of $f_\hat{x}(x)$ is

$$\hat{f}_x = \frac{1}{2N\varepsilon} \sum_{k = 1}^{N} I_x(x(k))$$

(25)

where $Q_x$ is the number of entries in the cell centered on $x$ (MATLAB’s `hist` function returns this). The accuracy of the histogram estimate is reflected in its mean and variance. It can be shown that $f_\hat{x}$ is approximately unbiased, with the approximation better if $\varepsilon$ is smaller. The vari-
ance of \( \hat{f}_x \) can be shown to be approximately
\[
\sigma_f^2 = \frac{f_x(x)}{2N\varepsilon}
\] (26)

Clearly it is desirable to have a small \( \varepsilon \) and \( N \) very large (unlimited), so that \( N\varepsilon \) is large and \( \sigma_f^2 \) is small.

A rule of thumb is to define the histogram cells or bins to cover a region of width \( 4\sigma_x \), where \( \sigma_x \) is the sample variance. An odd number of cells will allow the middle cell to be located at the sample mean, \( \hat{m}_x \). The cell width should be large enough to contain about 100 or more samples. Suppose the cell width is \( 2\varepsilon \), then we may choose \( \varepsilon \) and \( N \) such that
\[
2\varepsilon \left[ \frac{N}{100} \right] = 4\hat{\sigma}_x
\] (27)

Define the cell centers by
\[
x(k) = m_x + 2k\varepsilon, \quad k = 0, \pm 1, \pm 2, \ldots
\] (28)

and as shown in Figure 1. The histogram estimates are then given by
\[
\hat{f}_x(x(k)) = \frac{Q_x}{2N\varepsilon}
\] (29)

![Figure 1: Histogram pdf estimate](image)

**Exercises**

1. The random variable \( w \) is defined in terms of the random variables \( x, y, \) and \( z \), to be
\[
w = xy + z
\] (30)

The input rv are assumed to be mutually independent, with \( x \sim U(-1,1) \), \( y \sim U(-1,1) \), and \( z \sim U(0,1) \).

a.) Find the theoretical pdf \( f_w(w) \). Start by first finding the pdf on \( v = xy \) using text Example 6-26 p. 205 as a guide, then find the pdf on the sum \( w = v + z \) from a convolution. Note that the rv \( v \) and \( z \) are also independent. Why?

b.) Simulate the rv \( w \) using MATLAB and the uniform random number generator \( \text{rand()} \).

Estimate the theoretical pdf using a scaled histogram, using the technique described in the
handout or equivalently the notes page 5–25. The value of \( N \) should be at least 100,000. Plot both the pdf estimate and the theoretical pdf from part (a) on the same graph to see how well they compare.

2. Develop a transformation for converting uniform (0,1) rv samples into rv samples of \( y \) given that

\[
f_y(y) = \frac{4a^4 y}{(a^2 + y^2)^3} u(y), \quad a > 0
\]  

(31)

Note that \( a \) is just a parameter of the pdf. The area of \( f_y(y) \) is indeed unity independent of \( a \). An appendix containing useful integrals is available at the end of this handout.

a.) To begin with you need to find the cdf \( F_y(y) \) and then the inverse cdf so you can write

\[
y = T(x) = a \sqrt{\frac{1}{N} \left( 1 - x \right)} - 1, \quad 0 < x < 1
\]  

(32)

b.) Show that the theoretical mean is \( a\pi/4 \) and the variance is \( a^2 (16 - \pi^2)/16 \).

c.) Write a MATLAB program for generating rv samples of \( y \) using MATLAB’s \texttt{rand()} as the source of uniform random numbers.

d.) Using a reasonable number of samples (say 100,000 or more) find \( \hat{m}_x \) and \( \hat{\sigma}_x^2 \) for

\[
a = 0.5 \quad \text{and} \quad 1.5.
\]

Compare your results to the known theory of part b.

e.) Histogram the raw data used in (c) and then normalize it so that you have an estimate of the pdf. Compare your results to equation (31) by overlaying the true pdf. Do this for both values of \( a \).

3. In this problem you will generate correlated Gaussian random number pairs using a technique that is valid for random vectors of any distribution type and finite length. This problem will be restricted to the 2-D case, so the jointly normal random variate pairs will be of the form \( N(m_1, m_2, \sigma_1^2, \sigma_2^2, \rho) \) which corresponds to a jointly normal pdf of the form

\[
f_y(y) = \frac{\exp\left[-\frac{1}{2(1-\rho^2)}\left\{\left(\frac{y_1 - m_1}{\sigma_1}\right)^2 - 2\rho\left(\frac{y_1 - m_1}{\sigma_1}\right)\left(\frac{y_2 - m_2}{\sigma_2}\right) + \left(\frac{y_2 - m_2}{\sigma_2}\right)^2\right\}\right]}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}
\]  

(33)

where \( y = [y_1 \quad y_2]^T \).

Assume we are given an \( n \)-dimensional random vector \( X \) that has zero mean and covariance matrix \( C_x = I \), where \( I \) is the \( n \times n \) identity matrix. Our goal is to generate random variates corresponding to random vector \( Y \) such that they have a prescribed mean, \( \mathbf{m}_y \), and prescribed covariance matrix, \( C_y \). If we transform \( X \) with matrix \( A \) and add a constant vector \( \mathbf{m}_y \), i.e.

---

we obtain a new random vector with mean \( \mathbf{m}_y \) and covariance matrix \( \mathbf{C}_y = \mathbf{AIA}' = \mathbf{AA}' \).

Clearly getting the prescribed mean is easy, and simply requires adding a constant vector to the random number pairs resulting from the transformation \( \mathbf{AX} \). How do we choose \( \mathbf{A} \) to get the desired covariance matrix \( \mathbf{C}_y \)? Assuming that the desired \( \mathbf{C}_y \) is positive definite (determinant > 0 or all eigenvalues are positive), we can first expand \( \mathbf{C}_y \) as the following matrix product

\[
\mathbf{C}_y = \mathbf{PDP}'
\]  

(35)

where \( \mathbf{P} \) is a unitary matrix whose columns are the orthonormal eigenvectors of \( \mathbf{C}_y \) and \( \mathbf{D} \) is an \( n \times n \) diagonal matrix of the eigenvalues of \( \mathbf{C}_y \). We know that the eigenvalues are just the variances, \( \sigma_i^2 \), of the individual components of \( \mathbf{Y} \), and they must all be positive, so we can expand \( \mathbf{D} \) as

\[
\mathbf{D} = \mathbf{D}^{1/2} \mathbf{D}^{1/2}
\]  

(36)

where \( \mathbf{D}^{1/2} \) is an \( n \times n \) diagonal matrix with entries being the square root of the entries of \( \mathbf{D} \), or simply the standard deviations, \( \sigma_i \). The desired transformation we are seeking is finally

\[
\mathbf{A} = \mathbf{PD}^{1/2}
\]  

(37)

As a check note that

\[
\mathbf{AA}' = (\mathbf{PD}^{1/2})(\mathbf{PD}^{1/2})' = \mathbf{PD}^{1/2}\mathbf{D}^{1/2}\mathbf{P}' = \mathbf{PDP}' = \mathbf{C}_y
\]  

(38)

For the \( n = 2 \) case the covariance matrix is simply

\[
\mathbf{C}_y = \begin{bmatrix}
\sigma_1^2 & \rho \sigma_1 \sigma_2 \\
\rho \sigma_1 \sigma_2 & \sigma_2^2
\end{bmatrix}
\]  

(39)

In MATLAB obtaining \( \mathbf{PD}^{1/2} \) is easy since eigenvalue decomposition can be performed to obtain both \( \mathbf{P} \) and \( \mathbf{D} \) using \( [\mathbf{P}, \mathbf{D}] = \text{eig}(\mathbf{C}) \), where \( \mathbf{C} \) is a square matrix and \( \mathbf{P} \) and \( \mathbf{D} \) are as defined above in MATLAB terms.

a.) Write a MATLAB function that generates \( N \) Gaussian (normal) random number pairs using the technique described above. Zero mean unit variance iid normal random variates can be obtained in MATLAB from \( \mathbf{x} = \text{randn}(2,N) \) which produces matrix \( \mathbf{x} \) having 2 rows and \( N \) columns. The function should have an interface that takes as input the desired means, \( m_1 \) and \( m_2 \), the variances \( \sigma_1^2 \) and \( \sigma_2^2 \), and the correlation coefficient \( \rho \). The output should be a matrix, \( \mathbf{y} \), corresponding to the input variates \( \mathbf{x} \), e.g.,

```matlab
function y = corrgauss(mean, var, p, N)
% y = corrgauss(mean, var, p, N)
% Generate Correlated Gaussian RV's
% mean = mean vector [m1 m2]
% var = variance vector [var1 var2]
```
Exercises

% p = correlation coefficient
% N = number of random pairs to generate

x = randn(2,N);
y = zeros(size(x));
% Your code goes in here to compute y

b.) Test the function of part (a) using \( N = 1000 \) by plotting scatter plots of the data for

\[
\text{mean} = [1 1]; \quad \text{var} = [1 1]; \quad p = 0;
\]

When doing a scatter plot in MATLAB keep the same x-y axis units so the symmetry is obvious, e.g.,

\[
\begin{align*}
\text{plot}(y(1,:),y(2,:),',.'); & \quad \% \text{ Plot dots at each pair} \\
\text{axis('equal')}; & \quad \% \text{ make sure axis will be equal}
\end{align*}
\]

Comment on your results. Is this what you expected?

c.) Repeat part (b) using

\[
\text{mean} = [0 0]; \quad \text{var} = [1 1]; \quad p = 0.9;
\]

d.) Using the parameters part (c), but increasing the sample size to \( N = 100,000 \), compute the sample statistics for the mean vector, and covariance matrix. See how MATLAB can support this calculation or try the recursive forms.

e.) Using a sample size of \( N = 100,000 \) find experimentally the probability that the random vector \( \mathbf{Y} \) lies a square with unit length sides centered at the origin. Use the parameters of part (c). To check your answer numerically integrate the theoretical joint pdf, i.e., calculate

\[
P_{\text{square}} = \int_{-1/2}^{1/2} \int_{-1/2}^{1/2} f_Y(y_1, y_2) dy_1 dy_2
\]

f.) Repeat part (e) except now shift the square center to the point (2,2).

4. System Simulation: QPSK demodulation with an imperfect carrier reference. Shown below is a simplified block diagram of a quadrature phase-shift keyed (QPSK) coherent demodulator. Each of the four possible transmitted symbols can be viewed as a two dimensional vector with coordinated \((\pm A/\sqrt{2}, \pm A/\sqrt{2})\). For the modeling requirements of this problem we will assume that the received input signal vector is corrupted by an additive white Gaussian noise vector. The noise vector is assumed to have independent, zero mean, components \((n_1, n_2)\). Each component is taken to have variance \(\sigma_n^2/2\).

To coherently demodulate QPSK a coherent carrier reference is required. This is typically implemented using a phase-lock tracking loop. Since the coherent reference is typically derived from the noise corrupted received signal, the resulting phase estimate, \(\theta\), is actually a random variable. Ideally the rv \(\theta\) should be zero mean with a very small variance (i.e., \(\sigma_\theta^2 = 0\)). The result of \(\theta\) not being nonzero is a rotation of the received signal vector into a new vector \(\mathbf{Y}\).

The received symbol estimate is made by passing \(\mathbf{Y}\) through a decision rule. The rotated signal vector \(\mathbf{Y}\) is classified as being \(\xi_i\), \(i = 1, 2, 3, 4\) depending on which quadrant it lies
within.

a.) Find an analytical expression for the symbol error probability, \( P_e \), assuming that \( \theta = 0 \). Note, your solution should be in terms of the Gaussian Q-function. Note also that \( P_e \) is the average probability of making a symbol error, hence you need to consider all four symbol error probabilities or at least find out what symmetries exist.

b.) Show that the analytical expression for the conditional symbol error probability \( P_e(\theta) \), is given by

\[
P_e(\theta) = 1 - \left[ 1 - Q\left( \frac{2A^2}{\sigma_n^2} \cos(\theta + \frac{\pi}{4}) \right) \right] \left[ 1 - Q\left( \frac{2A^2}{\sigma_n^2} \sin(\theta + \frac{\pi}{4}) \right) \right]
\]  

(41)

Treating \( \theta \) as random variable we can use (41) to obtain the average symbol error probability, \( P_e \),

\[
P_e = \int_{-\infty}^{\infty} P_e(\theta) f_\theta(\theta) d\theta
\]  

(42)
The above expression is based on a result from back in Chapter 4 of the book and notes. In particular on page 105 of the text, equation (4-80), states a variation on total probability as

\[ \int_{-\infty}^{\infty} P(A|x = x)f_x(x)dx = P(A) \]  

(43)

Here we can think of the conditional probability \( P(A|x = x) \) as simply a probability as a function of some parameter. In practice the phase error \( \theta \) is modulo \( 2\pi \) so it makes sense for small \( \sigma_\theta \) that the integration limits just need to run from \(-\pi\) to \(\pi\), even though the density actually ranges from \(-\infty\) to \(\infty\). This will be applied in part (d).

c.) Using Monte-Carlo simulation obtain curves of estimated \( P_e \) for \( \sigma_\theta = 0, 1, \) and 10 degrees versus the input signal-to-noise ratio (SNR), which here is defined to be

\[ \text{SNR} = \frac{A^2}{\sigma_n^2} \]  

(44)

Assume that \( \theta \) is independent of the noise rv and is also Gaussian with zero mean.

Try to obtain reasonable estimates of \( P_e \) down to at least \( 10^{-5} \). Your error probability plots should use a semilogy format, or equivalently plot log base 10 of \( P_e \) versus SNR in dB (see below example).

\[ \theta \pi \leq \theta \leq \pi \]

Simulation Comments

- Note that you only need to simulate with the symbol \( (A/\sqrt{2}, A/\sqrt{2}) \) and then apply the noise vector and the rotation operation, which is controlled by the Gaussian random phase \( \theta \)
- A symbol error is made if the one or the other components becomes negative

d.) Check your simulation results using the analytical expressions you obtained in part (a) and (c). For the case of \( \sigma_\theta = 0 \) all you need is the Q-function which is given in the Appendix. When \( \sigma_\theta > 0 \) a numerical integration of the expression of (42) is required. When you average over \( \theta \) just consider \( -\pi \leq \theta \leq \pi \).
Appendix

Helpful Integrals for Problem 2

The following definite integrals are useful when working problem 2:

\[ \int_{0}^{\infty} \frac{x}{(x^2 + a^2)^3} \, dx = \frac{1}{4a^4}, \quad a > 0 \quad (45) \]

\[ \int_{0}^{\infty} \frac{x^2}{(x^2 + a^2)^3} \, dx = \frac{\pi}{16a^3}, \quad a > 0 \quad (46) \]

\[ \int_{0}^{\infty} \frac{x^3}{(x^2 + a^2)^3} \, dx = \frac{1}{4a^2}, \quad a > 0 \quad (47) \]

\[ \int_{0}^{\infty} \frac{x^4}{(x^2 + a^2)^3} \, dx = \frac{3\pi}{16a^3}, \quad a > 0 \quad (48) \]

A very powerful general result is

\[ \int_{0}^{\infty} \frac{x^m}{(x^n + a^n)^r} \, dx = \frac{(-1)^{r-1} \pi a^{m+1-nr} \Gamma \left( \frac{m+1}{n} \right)}{n \sin \left( \frac{(m+1)\pi}{n} \right) (r-1)! \Gamma \left( \frac{m+1}{n} - r + 1 \right)} \quad (49) \]

which is valid for \( 0 < m + 1 < nr \). Here \( \Gamma[n] \) is the gamma function which has special forms for different types of arguments, integers or half integers, in particular. Indeterminate results may require the use of LaHopital’s rule or something similar.

Gaussian Q-Function

The Gaussian Q-function was discussed earlier (pages 3–6 to 3–8 of the notes)

\[ Q(x) = \frac{1}{\sqrt{2\pi}} \int_{x}^{\infty} e^{-\frac{y^2}{2}} \, dy \quad (50) \]

Recall also that

\[ \text{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-u^2} \, du = 2Q(\sqrt{2}x) \quad (51) \]

or

\[ Q(x) = \frac{1}{2} \text{erfc}\left( \frac{x}{\sqrt{2}} \right) \quad (52) \]

The function \text{erfc} is available in MATLAB.
Numerical Integration in MATLAB

The MATLAB function of interest here is `quad()` or `quadl()` for 1-D integrals and `dblquad()` for 2-D integrations.

**Quad Help**

```matlab
>> help quad

QUAD Numerically evaluate integral, adaptive Simpson quadrature.
Q = QUAD(FUN,A,B) tries to approximate the integral of function
FUN from A to B to within an error of 1.e-6 using recursive
adaptive Simpson quadrature. The function Y = FUN(X) should
accept a vector argument X and return a vector result Y, the
integrand evaluated at each element of X.

Q = QUAD(FUN,A,B,TOL) uses an absolute error tolerance of TOL
instead of the default, which is 1.e-6. Larger values of TOL
result in fewer function evaluations and faster computation,
but less accurate results. The QUAD function in MATLAB 5.3 used
a less reliable algorithm and a default tolerance of 1.e-3.

[Q,FCNT] = QUAD(...) returns the number of function evaluations.

QUAD(FUN,A,B,TOL,TRACE) with non-zero TRACE shows the values
of [fcnt a b-a Q] during the recursion.

QUAD(FUN,A,B,TOL,TRACE,P1,P2,...) provides for additional
arguments P1, P2, ... to be passed directly to function FUN,
FUN(X,P1,P2,...). Pass empty matrices for TOL or TRACE to
use the default values.

Use array operators .* , ./ and .^ in the definition of FUN
so that it can be evaluated with a vector argument.

Function QUADL may be more efficient with high accuracies
and smooth integrands.

Example:
FUN can be specified three different ways.

A string expression involving a single variable:
Q = quad('1./(x.^3-2*x-5)',0,2);

An inline object:
F = inline('1./(x.^3-2*x-5)');
Q = quad(F,0,2);

A function handle:
Q = quad(@myfun,0,2);
where myfun.m is an M-file:
```
function y = myfun(x)
    y = 1./(x.^3-2*x-5);

See also QUADL, DBLQUAD, INLINE, @.

To numerically integrate a function, say a function which has an integrand of the form

$$ I(p_1, p_2) = \int_a^b g(x, p_1, p_2) \, dx, $$

we first write a MATLAB function file that evaluates the integrand in terms of \( x \) possibly being a vector and \( p_1 \) and \( p_2 \) being scalar parameters, e.g.,

```matlab
function y = g_integrand(x,p1,p2) %Stored in m-file g_integrand.m
...
    y = %%% formula for computing the integrand, may call other MATLAB functions
```

Now to evaluate for a particular parameter set \( p_1, p_2 \) we call quadl as follows

```matlab
% Assume a, b, p1, and p2 have been defined
I = quadl('g_integrand',a,b,[],[],p1,p2);
```

**Dblquad Help**

>> help dblquad

DBLQUAD Numerically evaluate double integral.

DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX) evaluates the double integral of

FUN(X,Y) over the rectangle XMIN <= X <= XMAX, YMIN <= Y <= YMAX.

FUN(X,Y) should accept a vector X and a scalar Y and return a

vector of values of the integrand.

DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX,TOL) uses a tolerance TOL

instead of the default, which is 1.e-6.

DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX,TOL,@QUADL) uses quadrature

function QUADL instead of the default QUAD.

DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX,TOL,@MYQUADF) uses your own

quadrature function MYQUADF instead of QUAD. MYQUADF should

have the same calling sequence as QUAD and QUADL.

DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX,TOL,@QUADL,P1,P2,...) passes

the extra parameters to FUN(X,Y,P1,P2,...).

DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX,[],[],P1,P2,...) is the same

as DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX,1.e-6,@QUAD,P1,P2,...)

Example:

FUN can be an inline object or a function handle.

```matlab
Q = dblquad(inline('y*sin(x)+x*cos(y)'), pi, 2*pi, 0, pi)
```

or

```matlab
Q = dblquad(@integrnd, pi, 2*pi, 0, pi)
```
where integrnd.m is an M-file:

```matlab
function z = integrnd(x, y)
  z = y*sin(x)+x*cos(y);
```

This integrates $y\sin(x)+x\cos(y)$ over the square $\pi \leq x \leq 2\pi$, $0 \leq y \leq \pi$. Note that the integrand can be evaluated with a vector $x$ and a scalar $y$.

Nonsquare regions can be handled by setting the integrand to zero outside of the region. The volume of a hemisphere is

```matlab
dblquad(inline('sqrt(max(1-(x.^2+y.^2),0))'),-1,1,-1,1)
```

or

```matlab
dblquad(inline('sqrt(1-(x.^2+y.^2)).*(x.^2+y.^2<=1)'),-1,1,-1,1)
```

See also QUAD, QUADL, INLINE, @.