Efficient Covariance Encoding

James R. Van Zandt

July 28, 2011

Abstract

We describe, optimize, and compare covariance encoding schemes. Several current systems encode three dimensional covariances in terms of their eigenvalues and Euler angles. We generalize this method to n dimensions. We precondition covariances to ensure that reconstructed matrices will be positive definite. We propose a new scalar measure of merit for covariance encodings, the Bhattacharyya distance. We compare the schemes in terms of the encoding error and the encoding length in bits. We recommend using enough bits to make the encoding error small compared to the error described by the covariance itself. The most efficient scheme uses logarithmic encoding of the variances and linear encoding of the Cholesky factor of the correlation matrix.

1 Introduction

In a networked tracking system for airborne or ballistic targets, it is usual to exchange not only track states (position, velocity, and possibly acceleration), but also their covariances, to facilitate cross-sensor cueing (determining what region to search for a target), track association (determining whether tracks are close enough to correspond to the same object), and track fusion (optimally combining two tracks). For covariance encoding, the most important requirement is that the recipient must always construct a positive definite matrix, which represents physically plausible track state uncertainties. Secondarily, the encoding should efficiently compromise between communications bandwidth used and error introduced in the covariance.

To characterize the error introduced by encoding, we propose using the Bhattacharyya distance [2, 5]. This scalar measure is easily calculated, and accounts for all the possible perturbations. It also permits the encoding error to be directly compared with the errors (from measurements, navigation, etc.) described by the covariance itself. We use this to establish a criterion for choosing field sizes: that the error introduced by the encoding be small compared to the error already present in the track state.

We discuss encoding methods that can be applied to more than three dimensions. Several current systems encode three dimensional covariances in terms of their eigenvalues and Euler angles. We generalize this method to n dimensions. If communications bandwidths are severely constrained, it may be appropriate to encode only partial covariance information (e.g. only horizontal components, or only variances). Such schemes are not considered here.

The remainder of this paper is organized as follows: Section 2 discusses measures of merit for covariance encoding. The testing method is described in Section 3. Section 4 describes several alternative encoding schemes and shows their individual performance. The schemes are compared in Section 5, and recommendations appear in Section 6.

2 Measures of Merit

Several measures of merit have been used in the past to evaluate covariance encodings [10, 11]. Some measures are most applicable to track association, and others to sensor cueing. I propose a new measure which accounts for all the possible effects of encoding on the covariance.

2.1 Association-Related Measures of Merit

Consider two tracks x_1 and x_2 with corresponding track covariances P_1 and P_2. The likelihood these reports should be associated (that is, that they plausibly represent the same object) may be estimated using the Mahalanobis distance D_m

$$D_m^2 = (x_1 - x_2)^T (P_1 + P_2)^{-1} (x_1 - x_2), \tag{1}$$

where T denotes the transpose, and D_m^2 has a Chi-square distribution.

D_m^2 is a function of the sum of the two track covariances. Assume one track is local and known very accurately, so the distance is dominated by the covariance of the remote track (the one with encoding error). Let P be the true covariance of the distant track, and P_2 be the covariance after encoding and decoding. One reasonable figure of merit is the ratio of the two Mahalanobis distances

$$\frac{x^T P_2 x}{x^T P x}, \tag{2}$$

averaged over test displacements x on the unit n-sphere [10]. The optimum value of this metric is one. This measure is more sensitive to changes in a small eigenvalue than a large one (e.g. a 10 percent change in the smallest eigenvalue makes a much larger difference than a 10 percent change in the largest eigenvalue).

The following MATLAB code estimates this metric using a Monte Carlo method.[1]:

```
function merit = mah(p,p2)
% this measure should be near one, can be either greater or
% smaller than one, and small values are especially bad
% 18 quasi-Monte Carlo trials
mah=zeros(3,6);
for k=1:3
  [r,q]=qr(randn(6,6));
  if det(r)<0, r(:,1)=-r(:,1); end % r is now a random 6D rotation
  for j=1:6
    x=r(:,j);   % x is uniformly distributed in the unit 6-sphere
    mah(k,j)=x'*(p2\x)/(x'*(p\x));
  end
end
merit=mean(mean(mah));
```

The first parameter of `mah` is the original covariance of the remote track, and the second parameter is the covariance after encoding and decoding.

The inverse of the error covariance is known as the Fisher information matrix [1]. It represents the amount of information in a measurement, in the following sense. Suppose we have an estimate of the current state of a system, and a linear measurement y with covariance P. We could use a Kalman filter to update the state with the measurement. As a thought experiment, imagine instead of the one measurement, we had two “sub-measurements”, the first with value y and covariance P_1, and the second with the same value y but with covariance P_2. Using the same Kalman filter, we could update the state with each sub-measurement in turn. It turns out that the effect on the state would be the same, if $P^{-1} = P_1^{-1} + P_2^{-1}$, i.e. the Fisher information matrix for the original measurement is the sum of the Fisher information matrices of the two sub-measurements.

[1]A random orthogonal matrix can be generated using the QR algorithm [8]

Recall that the trace of a matrix (the sum of the diagonal elements) is invariant with rotations. This suggests that the trace of the inverse of the covariance could be a useful scalar measure of the information [11]. If the state elements have different units, it is necessary to scale the covariance appropriately, e.g. with a matrix

$$S = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \tau & & \\ & & & & \tau & \\ & & & & & \tau \end{bmatrix}, \tag{3}$$

where τ is some suitable time constant (e.g. the expected latency of the track report, or half the update interval). Suppose a covariance P is encoded then decoded, resulting in a matrix P_2. If an encoding is guaranteed to be conservative, a suitable measure of the information lost is

$$1 - \frac{tr\left\{(SP_2S)^{-1}\right\}}{tr\left\{(SPS)^{-1}\right\}}. \tag{4}$$

If the encoding is not conservative, we may first scale the recovered covariance to make it conservative and then measure the information loss. Let U be the Cholesky factor of P^{-1}, so that $P^{-1} = U^T U$. Then the scaling constant k is the smallest eigenvalue of UP_2U^T, and the measure of information lost becomes

$$1 - k\frac{tr\left\{(SP_2S)^{-1}\right\}}{tr\left\{(SPS)^{-1}\right\}}. \tag{5}$$

For example:

```
% fraction of the information lost, so 0 is perfect
S=diag([1 1 1 6 6 6]);
u=chol(inv(p));
k=min(eig(u*p2*u'));
merit=1-k*trace(inv(S*p2*S'))/trace(inv(S*p*S'));
```

The information metric is largely determined by the best-known components of the track state (i.e., the small eigenvalues of P).

2.2 Cueing-Related Measures of Merit

Another family of measures is related to sensor cueing. Suppose a track is sent for the purpose of cueing a second sensor, which will perform a search in two dimensions to acquire the target. If the cue is too large, the sensor will waste effort on regions unlikely to contain the target ("waste"–see Figure 1). If the cue is too small, the sensor may fail to look in likely regions ("leakage"). If the measurement is characterized by the $2 \times n$ measurement matrix M, the most efficient search region is the ellipse defined by

$$xP_{proj}^{-1}x^T < \chi^2, \tag{6}$$

where P_{proj} is the two dimensional projection of the covariance

$$P_{proj} = MPM^T, \tag{7}$$

and χ is chosen according to the chi-square distribution so the target is included with the desired probability. For example, to include the target 95 percent of the time, we would choose $\chi = 2.4477$, since $P(\chi^2|\nu) =$

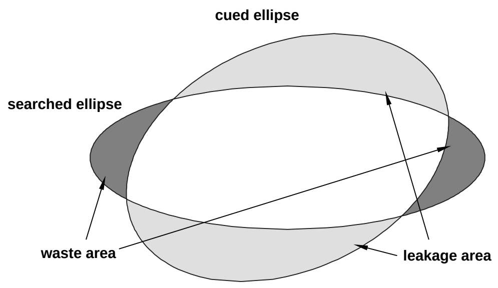

Figure 1: “Cookie Cutter” Model of Cueing.

$1 - Q(\chi^2|\nu) = .95$ for $\nu = 2$. Let the *cue ellipse* be derived from the original track covariance (for 95 percent inclusion probability), and the *search ellipse* be derived from the covariance reconstructed from the track report. The *leakage area ratio* is the area in the cue ellipse but outside the search ellipse, divided by the area of the cue ellipse [10]. E.g.:

```
function [leak,theta]=leakage(p,p2)
% leakage - calculate leakage area ratio (area in ellipse p but not in p2,
% divided by area in p)
  u=chol(inv(p));                      % whitening transformation
  eval=eig(u*p2*u');
  b=sqrt(min(eval));
  a=sqrt(max(eval));
  if 1<b
    leak=0;
  else
    if a<1
      leak=1-a*b;
    else
      theta=atan2(sqrt(a^2-1)*b,a*sqrt(1-b^2));
      leak=-2*(a^2*b*asin(cos(theta)/a)...
                -a*asin(cos(theta))...
                +b*cos(theta)*sqrt(a^2-cos(theta)^2)...
                -a*cos(theta)*sin(theta))/a/pi;
    end
  end
```

The *waste area ratio* is the area in the search ellipse but outside the cue ellipse, divided by the area of the search ellipse [10]. Each of these measures is non-negative, with optimal value of zero.

The leakage ratio and waste area ratio take a “cookie cutter” view, such that searching anywhere outside the cue ellipse is worthless, and failing to search anywhere inside it is inexcusable. However, the actual value of a given search region depends on its statistical distance from the center of the cue. Suppose the cue ellipse contains the target with probability f_c and the search ellipse contains it with probability f_s. The *leakage fraction* [10] is then

$$f_c - f_s. \tag{8}$$

It can be negative (indicating the sensor will acquire the target with more than the expected probability) as well as positive.

The following function calculates the fraction $q_s = 1 - f_s$ of the distribution outside the search ellipse:

```
function p=exclude(p1,p2,f,tol)
% exclude - return integral of a bivariate normal distribution with
% covariance p1, outside the ellipse defined by p2, scaled by factor k.
% That is, the integral over x=[x1 x2]' satisfying x'*p2^(-1)*x > 1
% of 1/(2*pi)*exp(-(1/2)*x'*p1^(-1)*x), where f = -k^2/2
%
% Inputs:
%   p1 = 2*2 covariance of the normal distribution function
%   p2 = 2*2 covariance defining the elliptical boundary of the
%        integration region
%   f = optional probability specifying the scaling factor for the
%       integration boundary.  The boundary ellipse is scaled by a factor
%       k according to the chi squared distribution for two degrees of
%       freedom, such that a fraction f would fall outside the radius
%       k. (default=.05)
%   tol = optional absolute integration error tolerance (default 1.e-6)

global axis_a axis_b
if nargin<3; f=.05; end
if nargin<4; tol=1.e-6; end
scale = sqrt(-2*log(f));
u = chol(inv(p1));                        % whiten the error
eval = eig(u*p2*u');
axis_a = scale*sqrt(eval(1));
axis_b = scale*sqrt(eval(2));
p = 4*quad('integrand',0,pi/2,tol);

function r=integrand(theta)
% integrand - calculate integrand for function exclude
global axis_a axis_b
R = 1./sqrt((cos(theta)/axis_a).^2 + (sin(theta)/axis_b).^2);
r = exp(-R.^2/2)/2/pi;
```

The area of the search ellipse is

$$A_s = \pi \sqrt{\det(P_s)}(-2\log(.05)), \tag{9}$$

where det(A) is the determinant of A. An optimal search area including a fraction f_s of the distribution would have area

$$A_0 = \pi \sqrt{\det(P_s)}(-2\log(q_s) \tag{10}$$

The *waste fraction* is the fraction by which the search area could be reduced, and still contain the target with the same probability as the cue ellipse [10]:

$$\frac{A_s - A_0}{A_s} \tag{11}$$

All of these waste and leakage metrics should be averaged over projection directions. If the track uncertainty is much smaller in some directions than others (as is common for single-radar tracks), then these metrics are sensitive mainly to the large dimensions of the uncertainty (i.e., the large eigenvalues of P). This is because the cued sensor rarely has the favorable geometry to take full advantage of the well-measured component of the target position. They are also sensitive to rotations of the uncertainty ellipsoid.

2.3 Bhattacharyya Distance

Each of the above metrics accounts for only a subset of possible perturbations in the covariance (e.g., rotation of the ellipsoid, or changes in small eigenvalues, or changes in large eigenvalues). I propose a new metric that accounts for all these perturbations: the *Bhattacharyya distance*[2] [2, 5].

The Bhattacharyya distance is a scalar separability measure between two probability distributions. If two multivariate normal distributions have mean m_i and covariance P_i, $i = 1, 2$, then the Bhattacharyya distance between them is

$$b = \frac{1}{8}\delta^T P^{-1}\delta + \frac{1}{2}\log\left(\frac{\det(P)}{\sqrt{\det(P_1)\det(P_2)}}\right), \tag{12}$$

where $\delta = m_1 - m_2$ and $P = (P_1 + P_2)/2$.

This distance summarizes all the possible ways an encoding could perturb a track report. The first term[3] is positive if the means of the two distributions differ. The second term is positive if their covariances differ, i.e., the uncertainty ellipsoids have different axes and/or orientations. It is invariant with rotations and translations of the coordinate system. It is equally sensitive to changes in the large or small eigenvalues.

The encoding error will be greater for some messages than others, due to the condition number of the covariance and (for some encoding methods) how well the axes of the uncertainty ellipsoid align with the coordinate axes. To account for this variability among messages, we suggest using as our metric the *90th percentile of the Bhattacharyya distance* over a representative sample of covariances.

To establish a reasonable criterion for this metric, we note that the Bhattacharyya distance depends on the mean of the track report, so it will be affected by measurement errors.[4] Let m_1 and m_2 be two n-dimensional estimated states, with the same covariance P, and $\delta = m_1 - m_2$. For example, they might be estimates from different Monte Carlo trials of a tracking simulation. Without loss of generality, we assume P is diagonal. The expected Bhattacharyya distance between the reports, due to measurement error alone, is then

$$\begin{aligned} b &= \frac{1}{8}[\delta_1 \quad \delta_2 \quad \cdots \quad \delta_n]\begin{bmatrix} 1/\sigma_1^2 & & & \\ & 1/\sigma_2^2 & & \\ & & \ddots & \\ & & & 1/\sigma_n^2 \end{bmatrix}\begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_n \end{bmatrix} + 0 \\ &= \frac{1}{8}\left(\frac{\delta_1^2}{\sigma_1^2} + \frac{\delta_2^2}{\sigma_2^2} + \cdots + \frac{\delta_n^2}{\sigma_n^2}\right) = \frac{2n}{8} = \frac{n}{4}. \end{aligned} \tag{13}$$

In the third expression, the sum of the normalized errors is $2n$ rather than n, because both measurements contribute errors to δ.

We suggest that sensor systems are typically much more expensive than the communication network used to report their measurements, so in trading off the investment in the two systems, the emphasis should be on preserving as much as possible of the measurement information. That is, we want the encoding error to be small compared to the measurement error.[5] We also recognize that our metric will not accurately

[2]Hint: "ch" is pronounced as in "charcoal", not as in "chaos".

[3]which is proportional to the Mahalanobis distance

[4]The mean is also affected by round-off errors during encoding. We assume the track state encoding has a high enough resolution that round-off errors are much smaller than measurement errors. Round-off errors are not considered in this analysis.

[5]In that case, we feel it is not necessary to ensure the encoding is conservative (that the decoded covariance completely encloses the original covariance [9]).

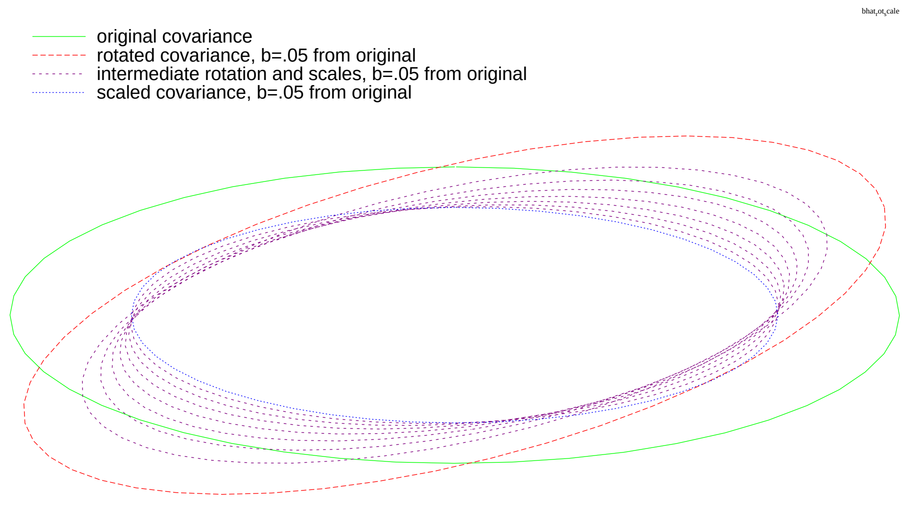

Figure 2: Bhattacharyya Distance = 0.05 due to Rotation and Scaling.

reflect the impact of the encoding error on each actual application. We suggest a factor of ten margin, giving us this criterion: *The 90th percentile Bhattacharyya distance should be less than* $n/40$.

In 2D, the suggested criterion would be $b = 2/40 = 0.05$. As an illustration, Figure 2 shows the "one sigma" uncertainty ellipses for an original covariance, a rotated version, a scaled version, and intermediate rotated and scaled versions. All the perturbed covariances are at a Bhattacharyya distance of 0.05 from the original. Similarly, ellipses perturbed by scaling and displacement are shown in Figure 3, and ellipses perturbed by displacement and changed aspect ratio are shown in Figure 4.

3 Testing

For best results, an encoding scheme should be matched to the expected covariances.

The difficulty of encoding a correlation matrix increases according to its condition number (the ratio of the largest to smallest eigenvalue). Single-sensor track covariances tend to have high condition numbers, because a typical radar can measure range much more accurately than angle. Composite track covariances tend to have lower condition numbers, because they are updated by several sensors.[6] Figures 5 and 6 show distributions of condition numbers and of eigenvalues of position and velocity covariance sub-matrices for two sets of covariances.

Each proposed encoding was subjected to Monte Carlo testing with 1000 test covariances. For example, in the 6×6 case, each test covariance P was generated as

$$P = R_{6\times6} S P_0 S^T R_{6\times6}^T, \tag{14}$$

[6]Similarly, in Global Positioning Satellite navigation, the Geometric Dilution of Precision (GDOP) is reduced when more satellites are visible.

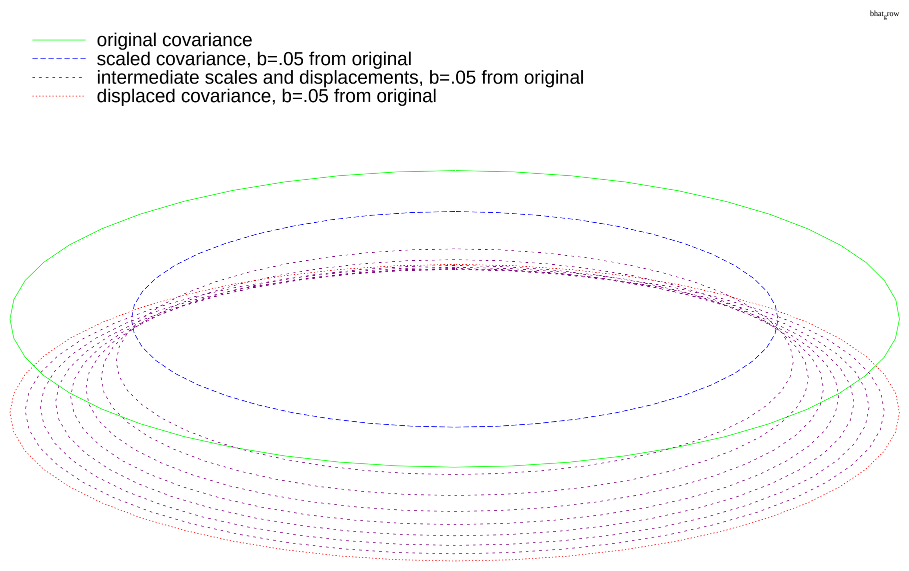

Figure 3: Bhattacharyya Distance= 0.05 due to Scaling and Displacement.

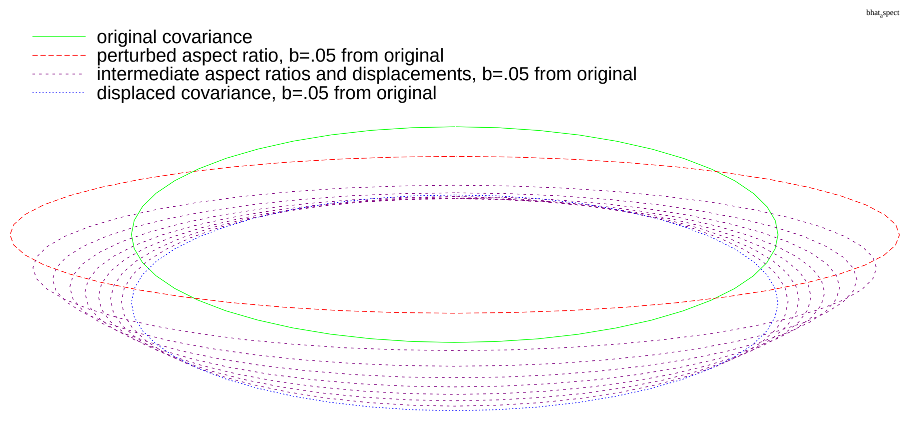

Figure 4: Bhattacharyya Distance= 0.05 due to Displacement and Aspect Ratio.

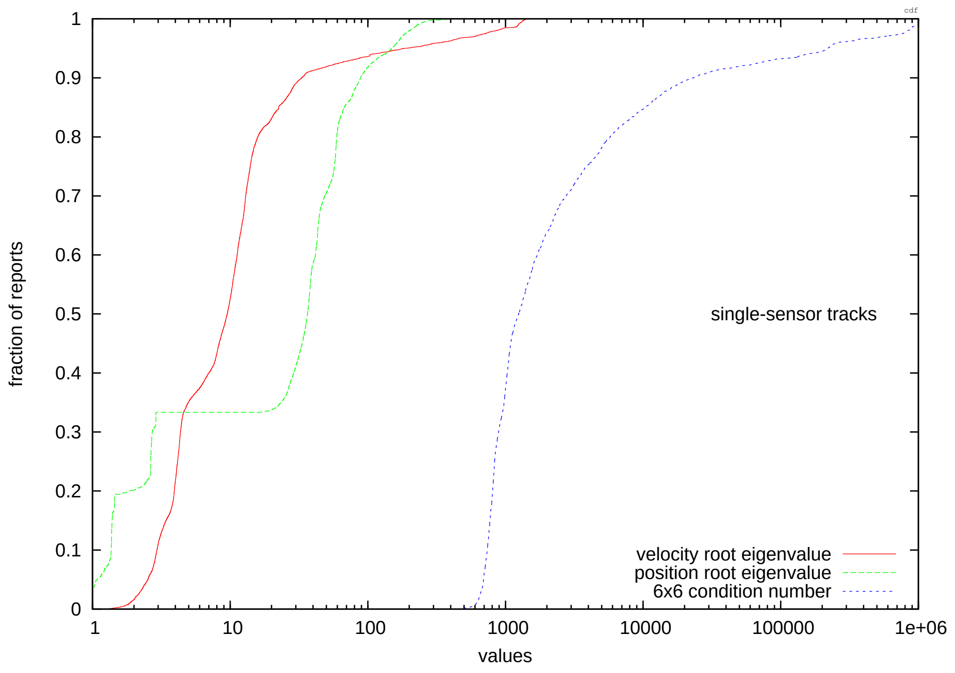

Figure 5: Covariance Matrix Characteristics for Single-Sensor Tracks.

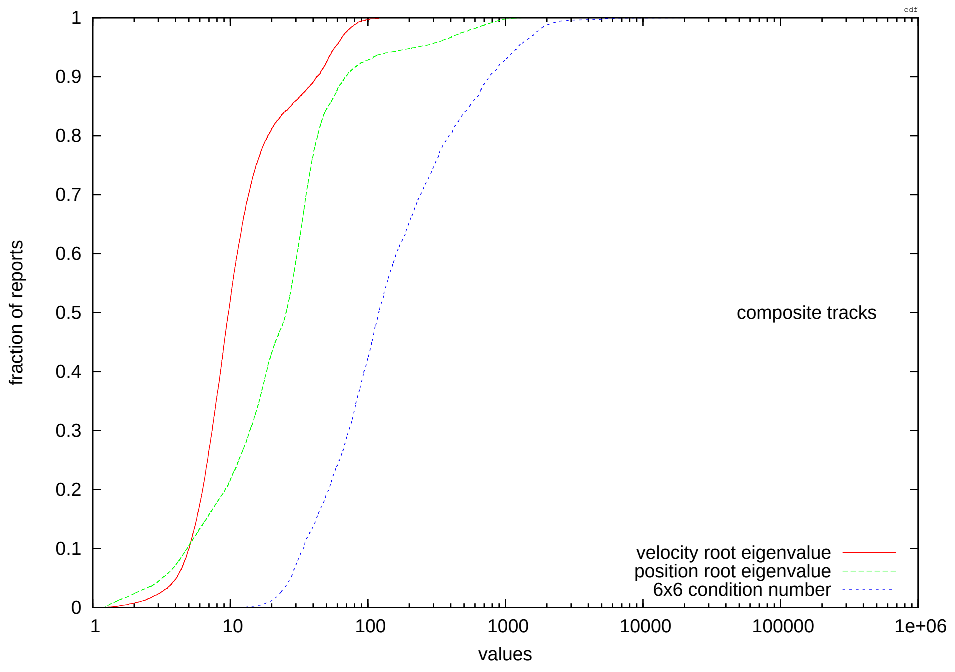

Figure 6: Covariance Matrix Characteristics for Composite Tracks.

Table 1: Axis Limits for Test Covariances

Axis	Lower Limit	Upper Limit	Units
i	l_i	u_i	
1	22	500	m
2	5	106	m
3	1	22	m
4	4	1000	m/s
5	4	64	m/s
6	4	16	m/s
7	12	50	m/s^2
8	6	25	m/s^2
9	3	12	m/s^2

where

$$P_0 = \begin{bmatrix} \sigma_1^2 & & & & & \\ & \sigma_2^2 & & & & \\ & & \sigma_3^2 & & & \\ & & & \sigma_4^2 & & \\ & & & & \sigma_5^2 & \\ & & & & & \sigma_6^2 \end{bmatrix}, \tag{15}$$

the ith semi-axis σ_i is log-uniformly distributed between l_i and u_i as shown Table 1,

$$S = \begin{bmatrix} 1 & 0 & 0 & \rho & 0 & 0 \\ 0 & 1 & 0 & 0 & \rho & 0 \\ 0 & 0 & 1 & 0 & 0 & \rho \\ \rho & 0 & 0 & 1 & 0 & 0 \\ 0 & \rho & 0 & 0 & 1 & 0 \\ 0 & 0 & \rho & 0 & 0 & 1 \end{bmatrix}, \tag{16}$$

ρ is uniformly distributed in [.5, .9], $R_{6\times6}$ is a block diagonal rotation matrix

$$R_{6\times6} = \begin{bmatrix} R_{3\times3} & 0 \\ 0 & R_{3\times3} \end{bmatrix}, \tag{17}$$

and $R_{3\times3}$ is a uniformly distributed random 3D rotation. Distributions of eigenvalues and condition numbers for the test covariances are shown in Figure 7.

Each of these test covariances was encoded, decoded, and the Bhattacharyya distance between the original and decoded covariances calculated. The random number generator seed was reset before each run, so the same covariances were used to test each encoding method.

4 Encoding Methods

Several covariance encoding methods have been suggested:

- Encode the covariance elements directly.
- Encode the variances and the correlation matrix.

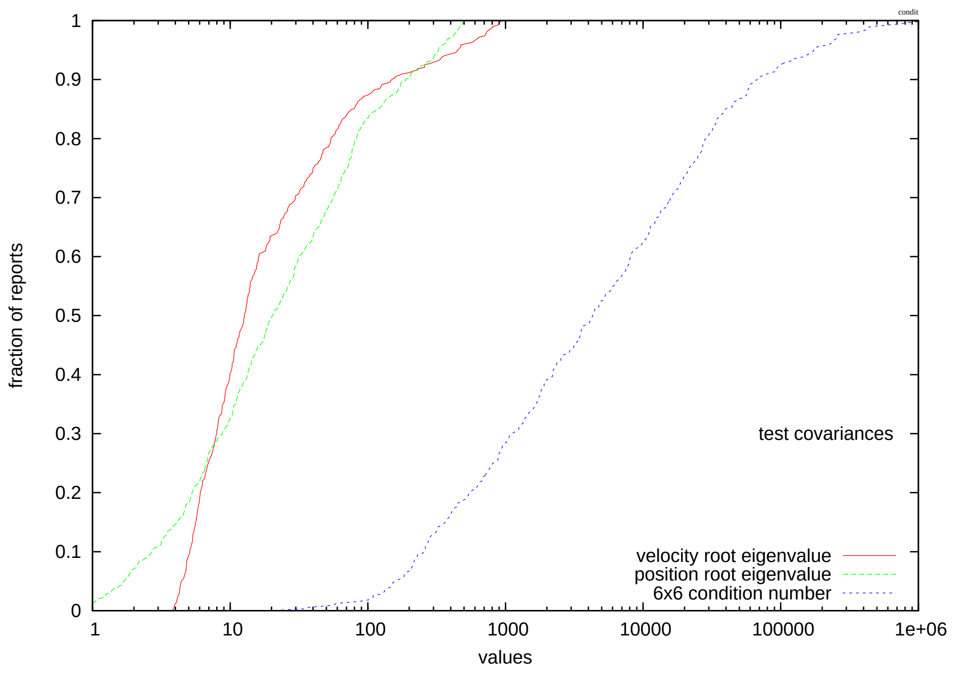

Figure 7: Characteristics of Test Covariances.

- For 3×3 covariance, encode the eigenvalues and the Euler angles needed to reconstruct the eigenvectors (i.e. the lengths and directions of the principle axes of the uncertainty ellipsoid).

The first two classes of encodings generalize immediately to more dimensions. We found two ways to extend the third class to more dimensions.

4.1 Covariance Element Encoding

The simplest scheme directly encodes each element of the covariance matrix. Logarithmic encoding is used, to cover a large dynamic range. The resolution depends on the field size and the dynamic range of the values being encoded. This is shown in Figure 8. We assume a dynamic range of 2000:1. For variances (which are always positive), it then takes 8 bits to give 3 percent resolution. I.e. the ratio between encoded values is

$$\left(\frac{P_{max}}{P_{min}}\right)^{1/2^8} = 2000^{1/256} = 1.030 \tag{18}$$

If b_m bits are used for each diagonal element, a value V is encoded by an integer

$$i = \left\lfloor c \frac{\log(V/P_{min})}{\log(P_{max}/P_{min})} + \frac{1}{2} \right\rfloor, \tag{19}$$

where $c = 2^{b_m} - 2$. The decoded value is

$$V_{decoded} = P_{min} \left(\frac{P_{max}}{P_{min}}\right)^{i/c}. \tag{20}$$

Off-diagonal elements of the covariance can be of either sign, so we need an extra bit to encode those elements with the same resolution. If we use b_c bits for each off-diagonal element, then the length of the

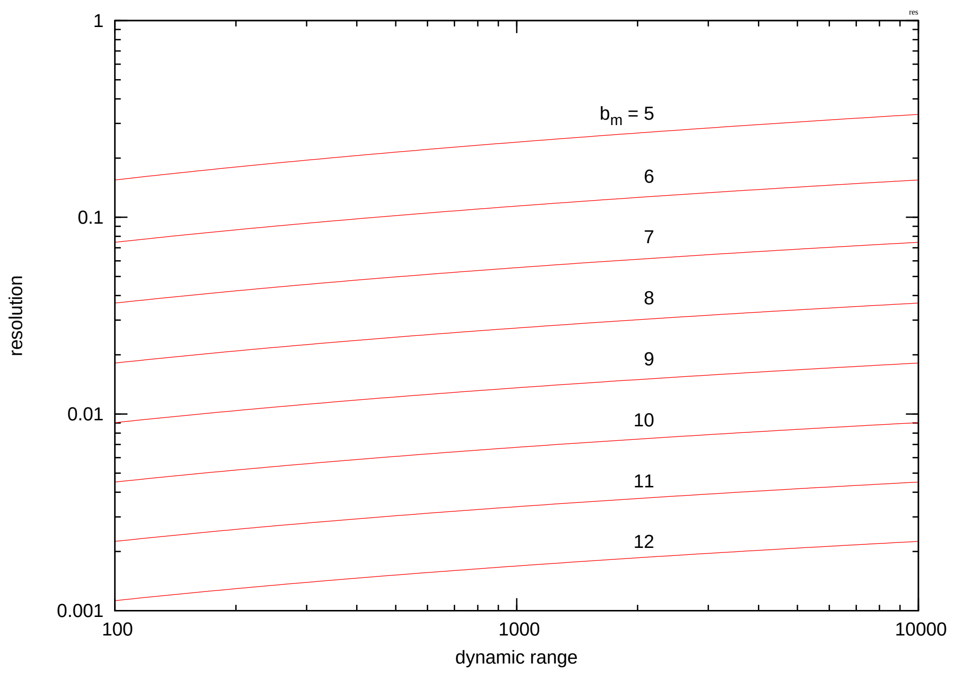

Figure 8: Resolution of Logarithmic Encoding.

encoding (i.e. the number of bits to encode the upper triangular portion of the symmetric covariance matrix) is

$$nb_m + \frac{n(n-1)}{2} b_c. \tag{21}$$

Here, we assume $b_c = b_m + 1$.

With this scheme, a large fraction of reconstructed matrices would not be positive definite, making them invalid covariance matrices (i.e. with *imaginary* uncertainties). Figure 9 shows distributions of ratios of eigenvalues (i.e. the largest eigenvalue of the reconstructed matrix divided by the largest eigenvalue of the original matrix, etc.) with $b_c = 9$ and $b_m = 8$. For our test matrices, 58 percent of the reconstructed matrices have at least one negative eigenvalue, and 15 percent have two!

To ensure the reconstruction of positive definite matrices, we can *precompensate* by multiplying each off-diagonal element of the covariance by $1 - \epsilon$. This kind of precompensation has the effect of inflating the uncertainty ellipsoid, more significantly for the smaller axes than the larger axes. We find empirically that $\epsilon = 2^{5-b_c}$ is sufficient.

The encoding efficiency of this scheme, with and without precompensation, is shown in Figure 10, in terms of the 90 percentile Bhattacharyya distance. Precompensation makes the encoding very conservative—the eigenvalues are never negative, but mostly they are too large.

The basic problem with this encoding scheme is that accommodating a large dynamic range in covariance element values forces us to use a coarse resolution, which makes it difficult to keep the reconstructed covariances positive definite.

4.2 Root Variance/Correlation Matrix Encoding

We may make the encoding more efficient by first expressing the covariance in terms of root variances (the square roots of diagonal elements of the covariance matrix) and a correlation matrix, as suggested by Jerardi [4]. The n root variances have a large dynamic range, and can be encoded logarithmically as above using

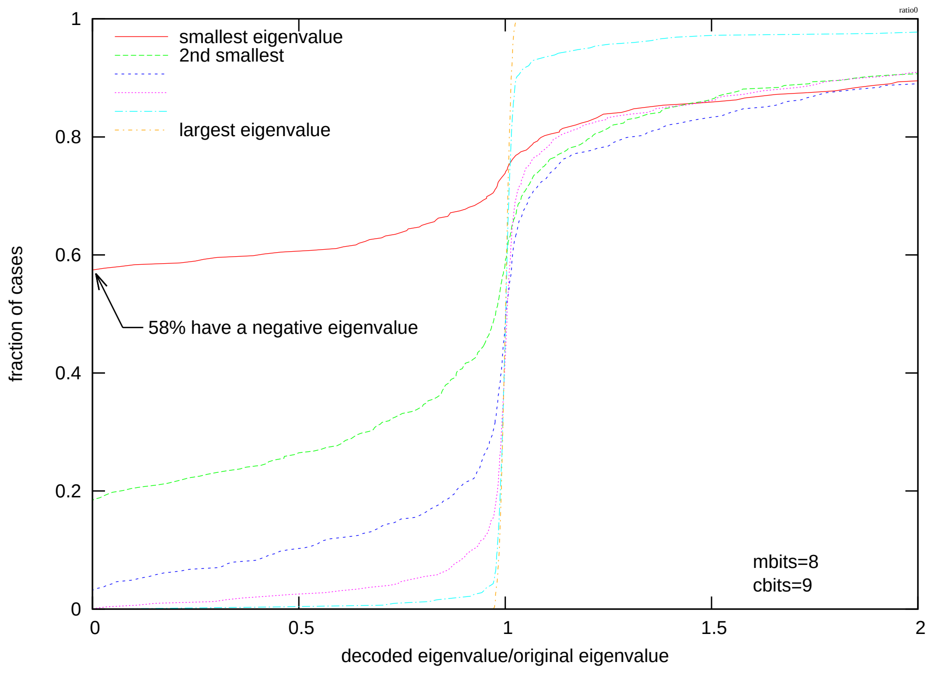

Figure 9: Distribution of Eigenvalue Ratios for Covariance Element Encoding.

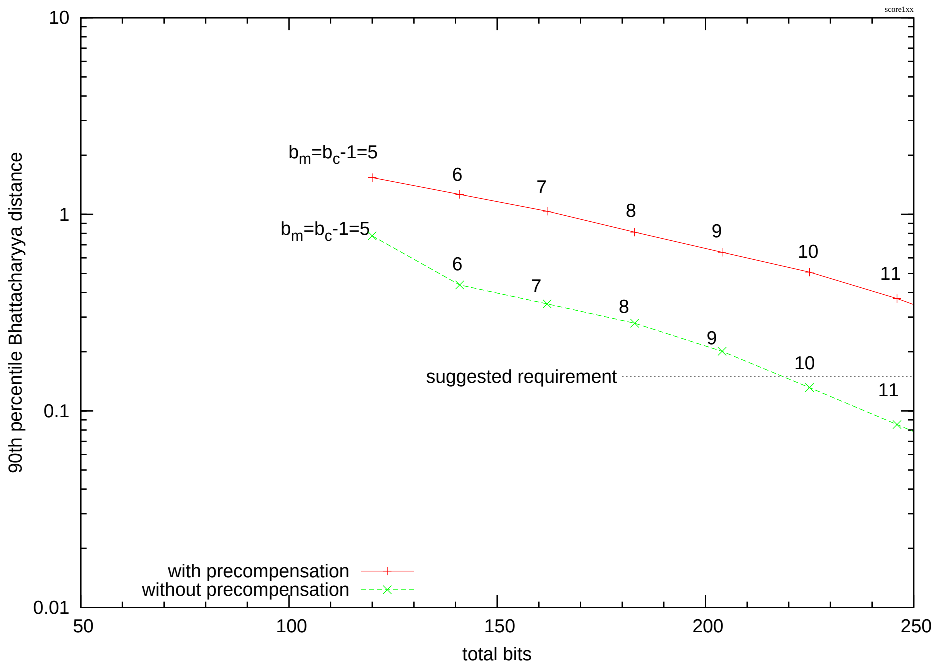

Figure 10: Efficiency of Encoding Covariance Matrix Elements.

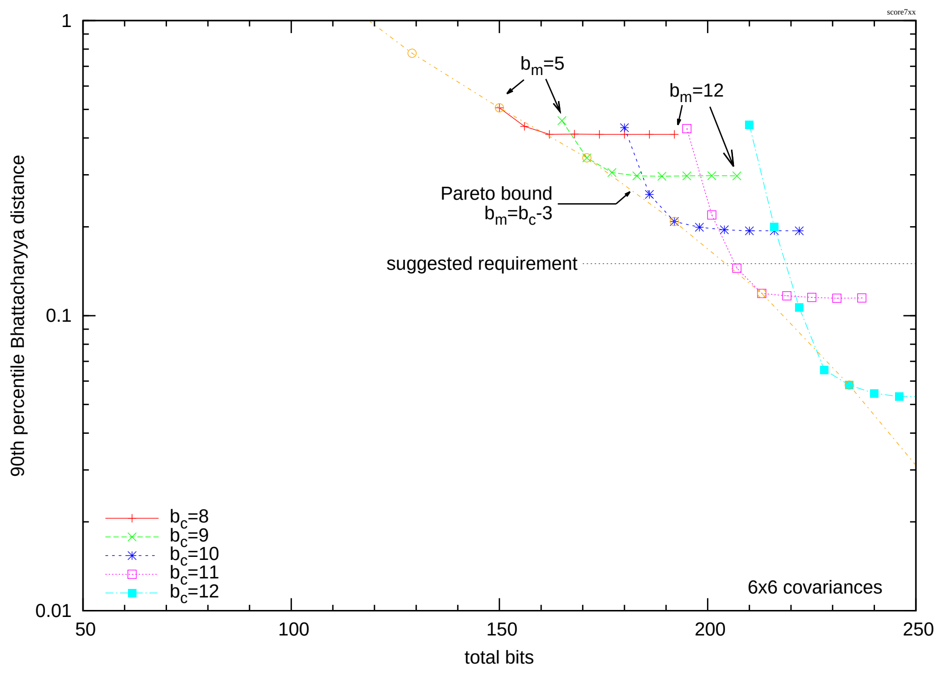

Figure 11: Efficiency of Root Variance/Correlation Matrix Encoding.

a relatively coarse resolution. The positive definiteness of the covariance depends only on the correlation matrix elements. These have a finite range $(-1, 1)$, and can be encoded linearly with a fine resolution. If b_c bits are used for each correlation element, a value C would be encoded by an integer

$$i = \min\{\lfloor cC \rfloor, c - 1\}, \tag{22}$$

where $c = 2^{b_c - 1}$, and the decoded value would be

$$C_{decoded} = \frac{i + 1/2}{c}. \tag{23}$$

The length of the encoding is again given by (21). We find it is adequate to precompensate using $\epsilon = 2^{1-b_c}$, which is much less than in Section 4.2. The efficiency is shown in Figure 11. Since there are two parameters, there is actually a family of curves. For the most efficient encodings (the efficient frontier, or Pareto bound), we have approximately $b_m = b_c - 3$.

4.3 Root Variance/Correlation Factor Encoding

As mentioned above, the difficulty of encoding a matrix increases with its condition number. Using the Cholesky decomposition, we may find an upper triangular matrix U such that $C = U^T U$. The condition number of U is the square root of the condition number of C, so U is much easier to encode than C [6].

U has $n(n + 1)/2$ nonzero elements. We encode all but the first row of U linearly using (22). The first row of U is the same as the first row of C, so the first element is always 1 and need not be encoded. We use $n - 1$ bits to encode only the signs of the remaining elements. Each column obeys the constraint

$$\sum_i U_{ij}^2 = 1, \tag{24}$$

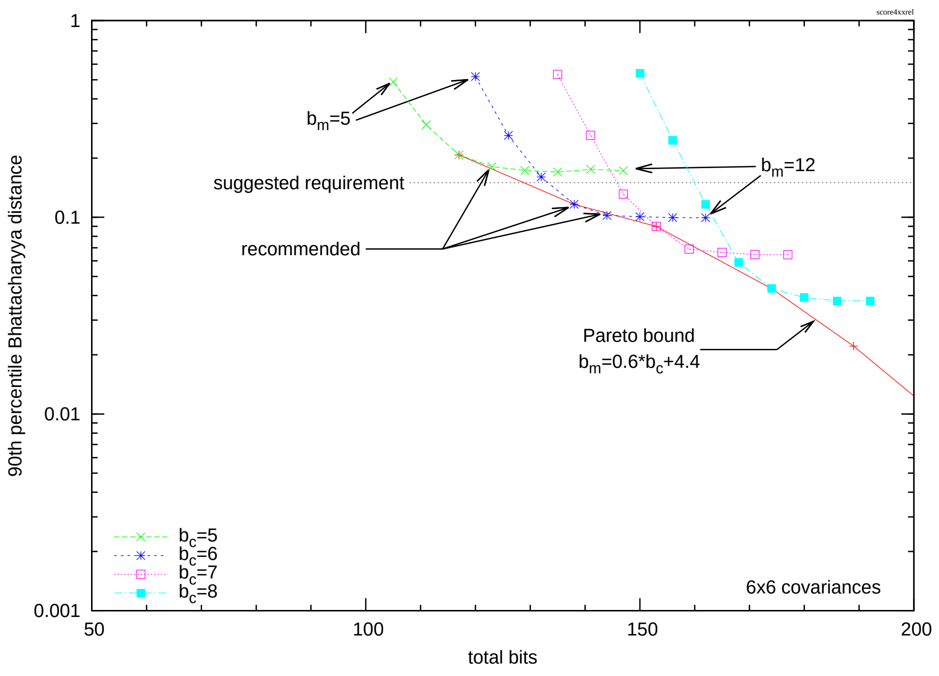

Figure 12: Efficiency of Root Variance/Correlation Factor Encoding.

and this can be used to find the magnitudes of those elements. The length of the encoding is again given by (21). For precompensation, we can use$\epsilon = 2^{-2b_c}$, which is much less than in Section 4.2 or 4.1. The encoding efficiency is shown in Figure 12. For the most efficient encodings, we have $b_m = 0.6b_c + 4.4$.

Applying this scheme to the 3×3 and 9×9 test covariances results in the performance shown in Figures 13 and 14. (The knees in the 3×3 figure are near $b_m = b_c + 1$, but the $b_m = 0.6b_c + 4.4$ points are still near the Pareto bound, and we expect 6×6 covariances will be sent more often.) In each case, the encoding with $b_c = 6$ and $b_m = 8$ meets the suggested requirement.

4.4 Eigenvalue/Rodrigues Parameter Encoding

In 3D, a covariance may be encoded using its three eigenvalues together with three Euler angles, as in a recent patent [9]. This can be generalized to n dimensions as follows. Using the singular value decomposition algorithm, we may factor a covariance P as

$$P = USU^T, \tag{25}$$

where S is diagonal (the diagonal elements are the eigenvalues of P) and U is orthogonal. In general, U has n^2 independent elements. However, we may use the *Cayley transform* [3] to parameterize it as a function of a skew-symmetric matrix Q

$$U = (I-Q)(I+Q)^{-1} = (I+Q)^{-1}(I-Q), \tag{26}$$

where

$$Q = (I-U)(I+U)^{-1} = (I+U)^{-1}(I-U). \tag{27}$$

(This fails if U has an eigenvalue at -1 — for example, in two dimensions, a 180 degree rotation.)

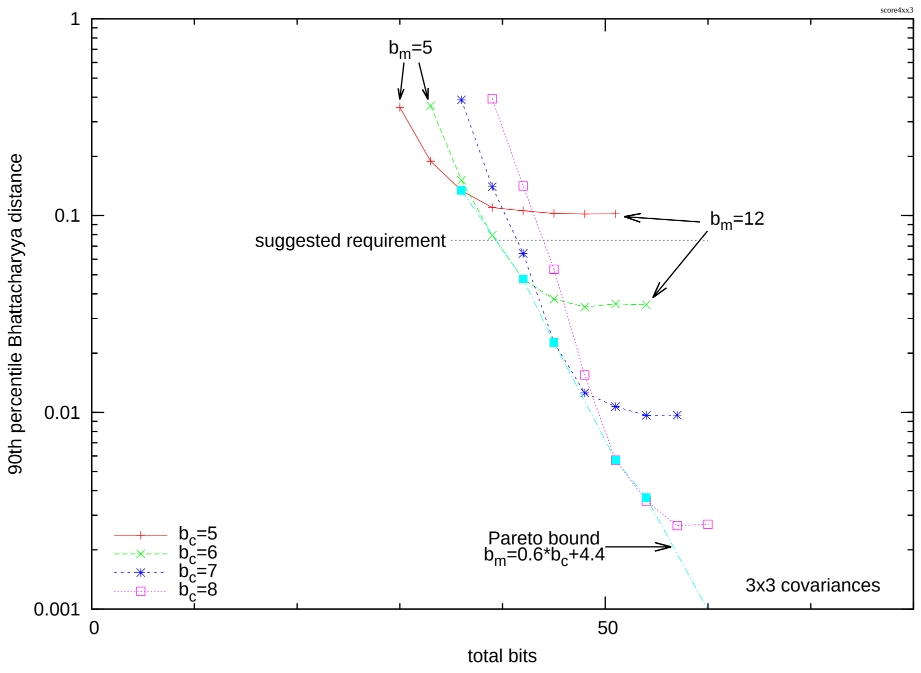

Figure 13: Efficiency of Root Variance/Correlation Factor Encoding for 3×3 Covariances.

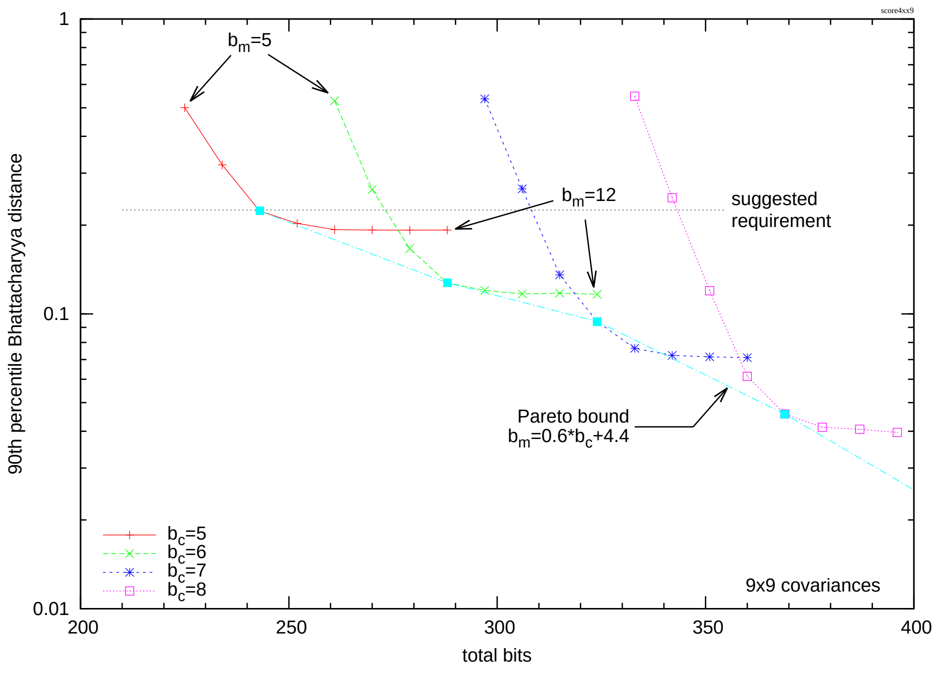

Figure 14: Efficiency of Root Variance/Correlation Factor Encoding for 9×9 Covariances.

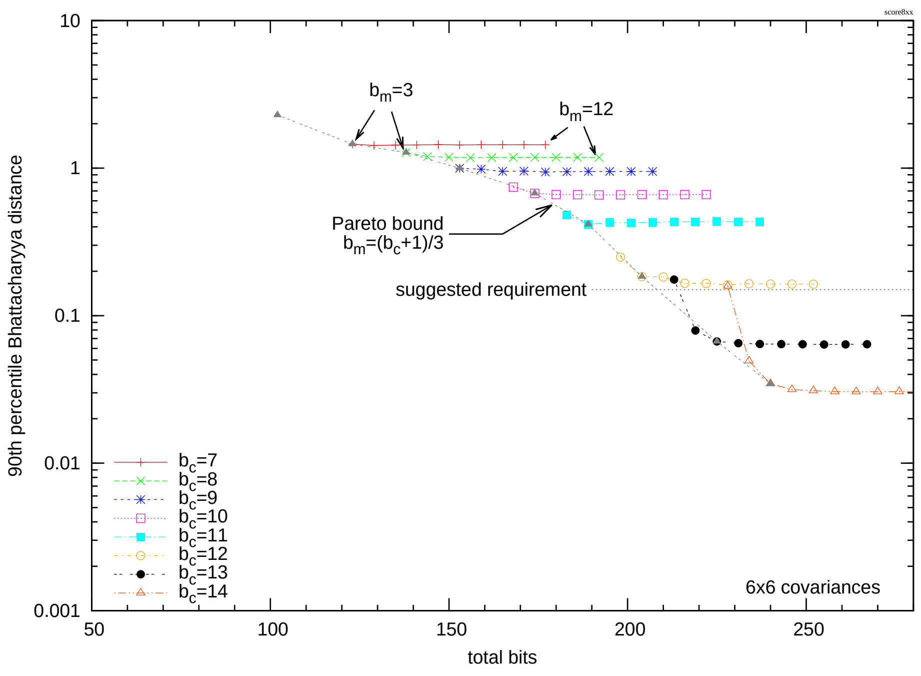

Figure 15: Efficiency of Encoding using Rodrigues Parameters.

The $M = n(n-1)/2$ independent elements of Q are the *extended Rodrigues parameters* [7]. Let q be the vector of extended Rodrigues parameters, e.g. in 3D

$$Q = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix}, \tag{28}$$

or in 6D

$$Q = \begin{bmatrix} 0 & -q_{15} & q_{14} & -q_{13} & q_{12} & -q_{11} \\ q_{15} & 0 & -q_{10} & q_9 & -q_8 & q_7 \\ -q_{14} & q_{10} & 0 & -q_6 & q_5 & -q_4 \\ q_{13} & -q_9 & q_6 & 0 & -q_3 & q_2 \\ -q_{12} & q_8 & -q_5 & q_3 & 0 & -q_1 \\ q_{11} & -q_7 & q_4 & -q_2 & q_1 & 0 \end{bmatrix}. \tag{29}$$

The q_i take on all values including infinity.[7] However, for a random orthogonal matrix U, the elements of Q have a tangent distribution—i.e. the arc tangents of elements of Q are uniformly distributed in $(-\pi/2, \pi/2]$. Therefore, we encode an element of Q using c_b bits as

$$m_i = 2^{c_b - 1} C \ \operatorname{atan}(q_i). \tag{30}$$

The encoding length is again given by (21). With this scheme, the reconstructed matrix is always positive definite, so no precompensation is necessary. Its efficiency is shown in Figure 15. For the most efficient encodings, we have approximately $b_m = (b_c + 1)/3$.

[7]Since any three-parameter parameterization of three-dimensional rotations has singularities, we should expect an M parameter description of U to have singularities too.

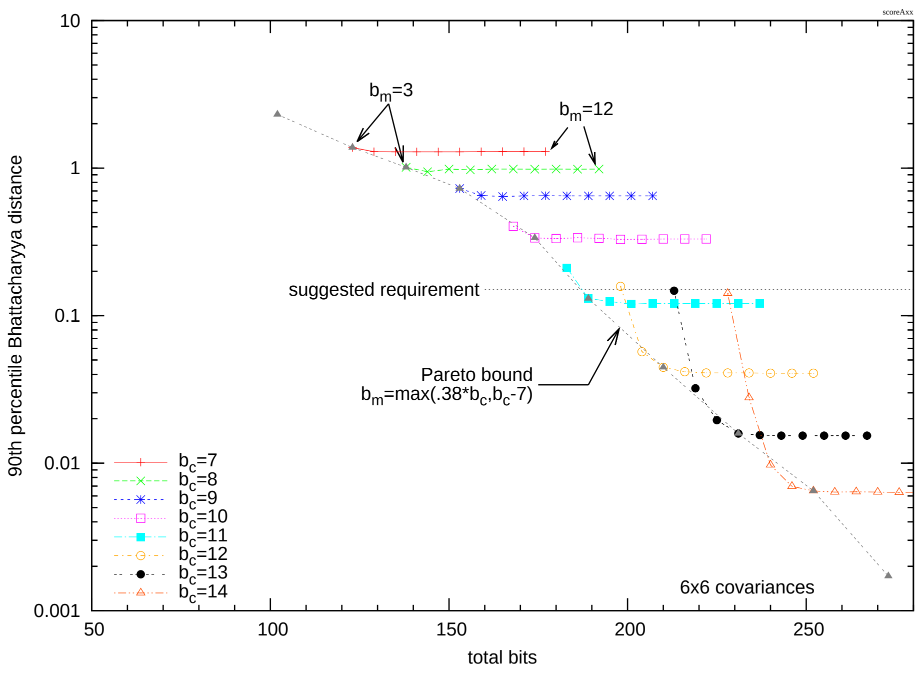

Figure 16: Efficiency of Encoding using Once-Redundant Parameters.

When Q has large elements, encoding using this scheme does not improve much as we increase the number of bits. The degradation may be reduced by encoding an element using

$$m_i = \begin{cases} 2^{c_b-1}C \ \ \mathrm{atan}(\sqrt{q_i}) & q_i \geq 0 \\ -2^{c_b-1}C \ \ \mathrm{atan}(\sqrt{|q_i|}) & q_i < 0 \end{cases} . \tag{31}$$

However, the scheme in the following section is even better.

4.5 Eigenvalue/Once-Redundant Encoding

A relatively efficient covariance encoding can be designed using the once-redundant encoding for an n-dimensional rotation introduced by Schaub, Tsiotras, and Junkins [7]. We start with SVD factoring and a Cayley transform as in the previous section. Let the Rodrigues parameters be q_i, $i = [1 \ldots M]$. Introduce a scaled set of parameters $\beta_i = \beta_0 q_i$, with the constraint

$$\beta_0^2 + \beta_1^2 + \cdots + \beta_M^2 = 1, \tag{32}$$

so that

$$\beta_0 = \frac{1}{\sqrt{1 + q_1^2 + q_2^2 + \cdots + q_M^2}}. \tag{33}$$

Like the Euler parameters (quaternions) in 3D, these once-redundant parameters have no singularities.[8] They have a finite range $(-1, 1)$, and can be encoded linearly. The efficiency of this method is shown in Figure 16. The most efficient encodings are for $b_m = \max\{.38b_c, b_c - 7\}$.

[8]This advantage is of course lost if, as here, they are calculated in terms of the Rodrigues parameters. It is possible to calculate them directly (see [7, equation 52]). However, as shown below, they do not turn out to be efficient enough to warrant the effort.

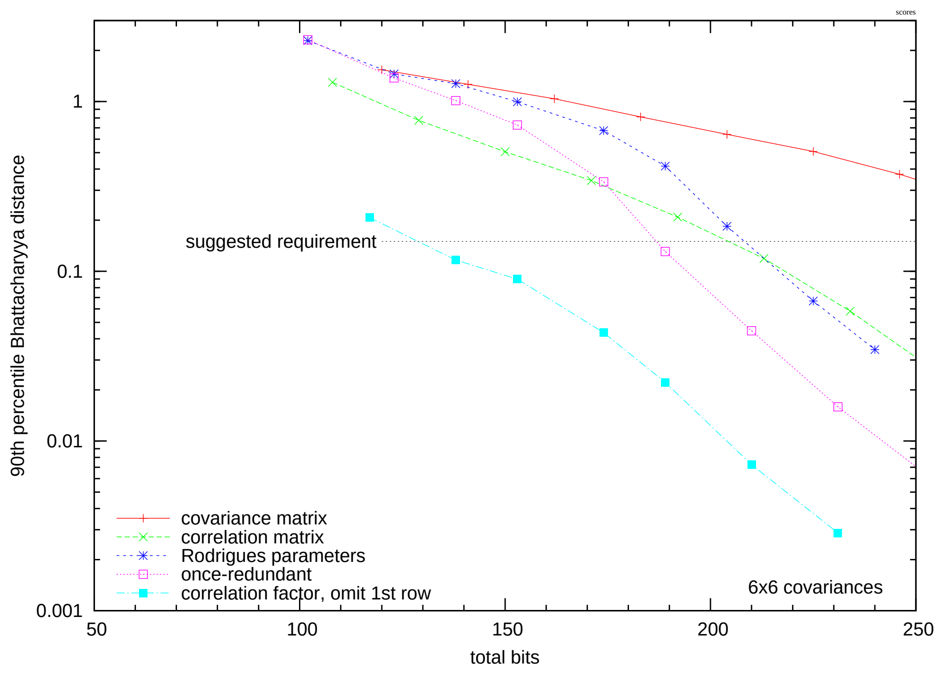

Figure 17: Summary of Covariance Encoding Efficiency.

5 Comparison of Encoding Methods

The Pareto bound efficiency curves for encoding 6D covariances using the encoding schemes described in Chapter 4 are collected in Figure 17. The most efficient scheme encodes the root variances and all but the first row of the correlation matrix factor.

The cumulative distributions of errors for the 6×6 test covariances are shown in Figure 18. The encoding with $b_m = 8$ and $b_c = 6$ meets the suggested criterion on the 90th percentile Bhattacharyya distance: $b < n/40 = 0.15$ for n=6 dimensions.

As mentioned above, even if the covariance were encoded perfectly, the received state would still differ from the truth due to measurement error. We would expect $4b$ to be chi-square distributed with n degrees of freedom. This contribution, $P(4b|6)$, is plotted in the same figure for reference.

6 Recommendations

We recommend that covariances be encoded using the scheme described in Section 4.3 (root variance/correlation factor encoding), with one of these parameter settings:

- $b_c = 5$ and $b_m = 8$,
- $b_c = 6$ and $b_m = 8$, or
- $b_c = 6$ and $b_m = 9$.

This recommendation is based on the set of test matrices adopted for testing. More bits may be needed for the magnitude if the dynamic range of encoded root variances is increased, or for the correlation matrix factor if covariance matrix condition numbers are found to be higher.

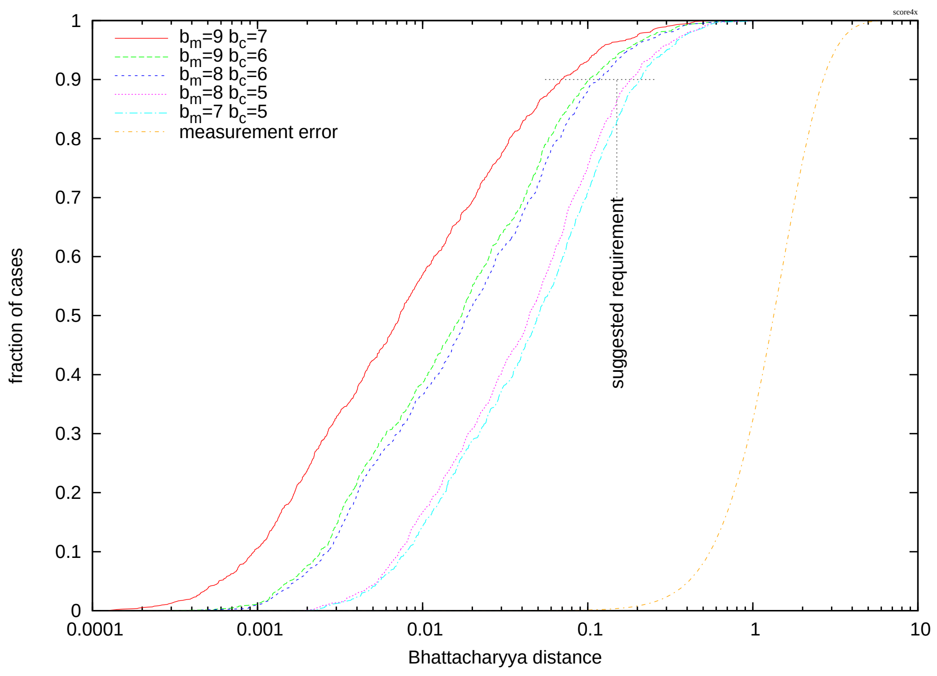

Figure 18: Error Distributions.

References

[1] Berger. *Statistical Decision Theory and Bayesian Analysis*, page 88. Springer-Verlag, New York, NY, 1985.

[2] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 35:99–109, 1943.

[3] A. Cayley. On the motion of rotation of a solid body. *Cambridge Mathematics Journal*, 3:224–232, 1843.

[4] Thomas Jerardi. Personal communication, 1992.

[5] T. Kailath. The divergence and Bhattacharyya distance measures in signal selection. *Communication Technology, IEEE Transactions on*, 15(1):52 –60, February 1967.

[6] John Nordmann. personal communication, 1992.

[7] Hanspeter Schaub, Panagiotis Tsiotras, and John L. Junkins. Principal rotation representations of proper $n \times n$ orthogonal matrices. *Intl. J. of Engineering Science*, 33(15):2277–2295, 1995.

[8] G. W. Stewart. The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM Journal on Numerical Analysis*, 17(3):pp. 403–409, June 1980.

[9] Robert E. Yang. Eigen-based method for covariance data compression. U. S. Patent 7574057B1, August 2009.

[10] J. R. Van Zandt. Error analysis for covariance encoding. Technical Report MTR93B0000014, MITRE Corporation, Bedford, MA, February 1993.

[11] J. R. Van Zandt. Tracklet covariance encoding. MITRE Paper MP97B0000095, MITRE Corporation, Bedford, MA, December 1997.

A Sample Program

The recommended encoding method (root variance/correlation factor encoding) was analyzed using the following MATLAB/Octave program:

```
function precomp4(mbits, cbits, n, oname, plotting)
% precomp4 - test precompensation when encoding all but first row of U
if nargin<3; n=6; end % # states (may be 3, 6, or 9)
if nargin<4; oname=[mfilename '.dat']; end
if nargin<5; plotting=0; end

% These values are only for illustration
if nargin<1 mbits = 8; end              % bits for magnitudes
if nargin<2 cbits = 8; end              % bits for covariance elements
pmin = [1 1 1 1 1 1 1 1 1]; % m, m/s, m/sˆ2
pmax = [2000 2000 2000  1000 1000 1000  100 100 100];
% upper and lower bounds for root variances
du = [ 500 106 22   1000 64 16    50 25 12];
dl=  [   22 5 1    4 4 4    12  6  3];

mbit_total=n*mbits;
% bits for signs, off-diagonal elements, diagonal elements:
cbit_total=n-1 + cbits*(n-1)*(n-2)/2 + (cbits-1)*(n-1);
bit_total=mbit_total+cbit_total;

fprintf([' ---------------- cbits=%d bit/element correlation encoding,'...
        'omitting first row, %d bits for correlation -----------\n'], ...
        cbits, cbit_total);
fprintf(' ---------------- mbits=%d, %d bits for magnitudes, %d bits total ---------\n', ...
        mbits, mbit_total, bit_total);

mcode = 2ˆmbits-2;      % root variance codes are in [0:mcode],
                        % 2ˆmbits-1 = "no statement"

ccode = 2ˆ(cbits-1);    % "no statement" not permitted, covariance
                        % codes are in [-ccode: ccode-1]
epsilon = 2ˆ(-2*cbits);

eval2_worst=1e9;

if exist('OCTAVE_VERSION')
  rand('state',42);
  randn('state',42);
else
  % rand('twister',5489); % MATLAB 7.7 accepts this syntax, but doesn't
  % actually reset the random number generator
  reset(RandStream.getDefaultStream); % reset random number generator
end
N=1e3;
ratio=zeros(N,n);
K=0;                                   % # valid test cases so far
for KK=1:N
  d = dl.*(du./dl).^rand(1,9); % log-uniformly distributed between bounds
  s = diag(d(1:n));
  rho=diag(.5*.4*rand*ones(1,3)); % position-velocity correlation
               % ranges from +0.5 to +0.9 (typically 0.7)
  [r,q]=qr(randn(3,3));
  if det(r)<0, r(:,1)=-r(:,1); end; % r is now a random 3D rotation matrix
  switch n
    case 3
      cr=eye(3);
    case 6
      cr=[eye(3) rho;rho eye(3)];
      r=blkdiag(r,r);
    case 9
      cr=[eye(3) rho zeros(3,3); rho eye(3) rho; zeros(3,3) rho eye(3)];
      r=blkdiag(r,r,r);
  end

  p=floor(r*s*cr*s*r');

  % encode the covariance
  sig=sqrt(diag(p));                   % root variances
  d=diag(1./sig);
  c=d*p*d;                             % correlation matrix

  if min(eig(c))>0                     % valid correlation matrix?

    K=K+1;
    eval1=sort(eig(p));

    % precompensate
    c=(1-epsilon)*c;
    for k=1:n
      c(k,k)=1;
    end

    u=chol(c);                  % Cholesky factor such that u'*u = c
                                % note k=n here
                                % linear encoding of Cholesky factor
    msg = zeros(1, n + n-1 + n*(n-1)/2); % n for root variances + n-1 for
                                          % signs + n*(n-1)/2 for U
    % log encoding of root variances
    for k=1:n
      msg(k)=floor( mcode*log(sig(k)/pmin(k))/log(pmax(k)/pmin(k))+1/2 );
      msg(k)=max(0, min(mcode, msg(k))); % sanitize
    end

    msg(n+1:2*n-1)=u(1,2:n)<0;          % signs of elements in top row

    % linear encoding of U
    k = 2*n;
    for j=2:n
      for i=2:j % omit first row
        msg(k)=min(floor(ccode*u(i,j)),ccode-1);
        k=k+1;
      end
    end

    % decode covariance
    u2 = zeros(n,n);
    u2(1,1)=1;
    k = 2*n;
    for j = 2:n
      sum = 1;
      for i = 2:j
        u2(i,j) = (msg(k)+1/2)/ccode;
        sum = sum - u2(i,j)^2;
        k = k+1;
      end
      if msg(n+j-1); sign=-1; else sign=1; end
      u2(1,j) = sign*sqrt(max(0, sum));
    end

    c2=u2'*u2;

    sig = pmin(1:n).*(pmax(1:n)./pmin(1:n)).^((msg(1:n))/mcode);
    s = diag(sig);
    p2 = s*c2*s;

    eval2=sort(eig(p2));
    ratio(K,:)=eval2./eval1; % assume the eigenvectors correspond
    if eval2(1)<eval2_worst
      eval2_worst=eval2(1);
      p_worst=p;
      c_worst=c;
      u_worst=u;
      msg_worst=msg;
      u2_worst=u2;
      c2_worst=c2;
```

```
    end

    pm=(p+p2)/2;
    bhat(K)=.5*log(det(pm)/sqrt(det(p)*det(p2)))/2; % Bhattacharyya distance
  end
end

fprintf('worst eigenvalue = %13.6g, corresponding to:\n',                  eval2_worst);
fprintf('p=');                                                             disp(p_worst);
fprintf('  eigenvalues= %10.3g %10.3g %10.3g %10.3g %10.3g %10.3g\n',  sort(eig(p_worst)));
fprintf('c=');                                                             disp(c_worst);
fprintf('  eigenvalues= %10.3g %10.3g %10.3g %10.3g %10.3g %10.3g\n',  sort(eig(c_worst)));
fprintf('\nu=');                                                           disp(u_worst);
fprintf('\nmsg=');                                                       disp(msg_worst);
fprintf('\nu2=');                                                         disp(u2_worst);
fprintf('\nc2=');                                                         disp(c2_worst);
fprintf('  eigenvalues= %10.3g %10.3g %10.3g %10.3g %10.3g %10.3g\n', sort(eig(c2_worst)));
fprintf('\n');
fprintf('K  eig(u3)  eig(c3)\n');

y=[.5:K]'/K;
ratio=sort(ratio(1:K,:));
if plotting>1
  figure(4);
  plot(ratio(:,1),y,ratio(:,2),y,ratio(:,3),y);
  axis([.95 1.05 0 1]);
end

bhat=sort(bhat);
if plotting>0, figure(24); semilogx(bhat,y); end

ofile=fopen(oname,'wt');
fprintf('# encoding Cholesky factor of correlation matrix, omitting first row\n');
fprintf('# cbits = %2d, %3d bits for correlation\n', cbits, cbit_total);
fprintf('# mbits = %2d, %3d bits for magnitudes\n', mbits, mbit_total);
fprintf('#             %3d bits total\n', bit_total);
fprintf('median Bhattacharyya distance = %f\n', bhat(round(K/2)));
fprintf('90th percentile Bhattacharyya distance = %f\n', bhat(round(K*.9)));
fprintf('%4d %f ##\n', bit_total, bhat(round(round(K*.9)))); % for score*.dat file
fprintf(ofile, '# %s(%d, %d, %d, ''%s'', %d)\n', mfilename, mbits, cbits, n, oname, plotting);
fprintf(ofile, '# encoding Cholesky factor of correlation matrix, omitting first row\n');
fprintf(ofile, '# cbits = %2d, %3d bits for correlation\n', cbits, cbit_total);
fprintf(ofile, '# mbits = %2d, %3d bits for magnitudes\n', mbits, mbit_total);
fprintf(ofile, '#             %3d bits total\n', bit_total);
fprintf(ofile, '# median Bhattacharyya distance = %f\n', bhat(round(K/2)));
fprintf(ofile, '# 90th percentile Bhattacharyya distance = %f\n', bhat(round(K*.9)));
fprintf(ofile, '# fraction    bhat  ratio(1)  ratio(2)  ratio(3)\n');
for i=1:K
```

```
  fprintf(ofile,'%f  %f  %f  %f  %f\n', y(i), bhat(i), ratio(i,1),ratio(i,2),ratio(i,3));
end
fclose(ofile);
```