A Joint Receiver-Decoder for Convolutionally Coded BPSK

Jon Hamkins

May 27, 1999

Abstract

This report presents a method to jointly estimate phase and data from a convolutionally coded Binary Phase-Shift Keying (BPSK) signal with random phase noise and Additive White Gaussian Noise (AWGN). The joint receiver-decoder successfully decodes fully suppressed carrier signals in harsh phase noise environments. A complete description is given of the software implementation, including the generation of statistically accurate phase noise samples. Numerical results are given for the NASA standard (7,1/2) convolutional code and frequency flicker dominated phase noise. For phase noise levels of -10.58, -1.55 and 7.48 dB rad^2/Hz at a 1 Hz offset, a data rate of 40bps, and a Bit Error Rate (BER) of 0.01, the joint receiver decoder saves 3 to 4.25 dB of power over a nonjoint approach.

1 Introduction

This report presents a method to perform phase-tracking and data decoding of a convolutionally coded BPSK signal corrupted by random phase noise and AWGN. Current receivers and decoders accomplish these two tasks in an *isolated* manner: the receiver performs phase-tracking and the decoder produces the decoded data sequence. The sole communication between the conventional receiver and decoder is via a sequence of soft symbols at the output of the receiver. In contrast, the technique given in this report estimates the phase and data *jointly*, which improves both the phase estimate and the decoded data sequence.

Some amount of optimality can be claimed for the isolated-blocks architecture, because the phase tracking loop is motivated by the Maximum A Posteriori (MAP) estimate, and a Viterbi decoder also gives the MAP estimate of the bits. However, the optimality of the phase tracking loop is based on the assumption that transmitted symbols are independent, and the optimality of the Viterbi decoder is based on assumptions of perfect timing and phase synchronization. Neither assumption is true, especially in the high phase noise environments that occur at low data rates.

In these harsh cases, it is possible to improve system performance with an integrated architecture that jointly tracks phase and decodes data. The phase tracker can take advantage of the structure

Table 1: Properties of various phase tracking loops.

	Can operate without residual carrier	Makes use of carrier info in modulation sidebands	Performance can be maximized without SNR estimate	Makes use of a posteriori bit probabilities	Makes use of underlying error-correcting code
PLL	No	No	Yes	No	No
Integrate/Dump loop	Yes	Yes	No	No	No
Squaring loop	Yes	Yes	No*	No	No
Costas loop	Yes	Yes	No*	No	No
Decision-directed loop	Yes	Yes	No*	Yes	No
Joint receiver-decoder	Yes	Yes	Yes	Yes	Yes

* Ideal filter coefficients depend on SNR, but performance is often insensitive to SNR mismatch.

of the channel code, and the decoder can take advantage of improved tracking capability. Further improvements might also be obtained by moving the external symbol synchronization block into the joint receiver-decoder. This report does not tackle this issue, and assumes perfect timing.

Each of the various conventional receiver phase-tracking loops utilizes slightly different amounts of information from the modulation sidebands [LS73], summarized in Table 1. Technically, the decision-directed loop can be viewed as a joint receiver-decoder, because it estimates both phase and data, but its feedback is a sequence of memoryless, symbol-by-symbol decisions with no operational dependence on the the underlying error-correcting code. The other loops operate without making any data decisions at all: a phase-locked loop (PLL) requires an unmodulated residual carrier signal, the MAP-based integrate-and-dump loops average the phase estimate over the "unknown" data, and the squaring and Costas loops nullify the modulation in the sidebands by squaring (or effectively squaring) the signal.

The joint receiver-decoder uses a trellis-based algorithm to jointly estimate the phase and data. Corresponding to each state of the trellis is a hypothesized data sequence leading to that state, *and also a hypothesized phase estimate conditioned on that data sequence.* This conditioning is the key advantage over the other loops. When conditioned on the correct data path, the phase estimate performance of any loop of the type described above is the same as for a pilot tone with no data modulation. Furthermore, the phase estimate can be updated one or more times per symbol; there is no requirement that the phase remain constant throughout the constraint length of the code. As a result, the joint receiver-decoder offers the potential to operate on a suppressed carrier signal in phase noise regions too harsh for a conventional receiver and decoder, and to do so without incurring a squaring loss typified by loops that remove modulation by squaring the signal.

2 Preliminaries

2.1 The signal model

We consider a noiseless received signal of the form

$$s(t) = A \sin[\omega_c t + \theta(t) + \beta m(t)], \tag{1}$$

where $\theta(t)$ is the randomly varying phase, ω_c is the carrier frequency, β is the mod index or mod angle, $m(t) = \sum_k c_k p(t-kT)$ is the convolutionally encoded waveform, $c_k \in \{-1, +1\}$, and $p(t)$ is a rectangular pulse of length T. We do not consider subcarrier modulation in this report. Note that when $\beta = \pi/2$ the signaling is antipodal, reducing to an ordinary suppressed carrier BPSK signal. The total received signal is

$$r(t) = s(t) + n(t),$$

where $n(t)$ is an AWGN process with one-sided power spectral density N_0.

2.1.1 Complex baseband

For convenience, we work primarily in complex baseband, by defining in the usual way $s(t) \triangleq Re[\tilde{s}(t)e^{j\omega_c t}]$ and $n(t) \triangleq Re[\tilde{n}(t)e^{j\omega_c t}]$. Using these definitions and Eq. (1), we have

$$\begin{aligned} \tilde{s}(t) &= -jAe^{j[\theta(t)+\beta m(t)]}, \\ \tilde{n}(t) &= n_c(t) + jn_s(t), \quad \text{and} \\ \tilde{r}(t) &= \tilde{s}(t) + \tilde{n}(t). \end{aligned} \tag{2}$$

In this representation, the noise components $n_c(t)$ and $n_s(t)$ are independent white Gaussian noise processes with one-sided power spectral density $S_{n_c}(f) = S_{n_s}(f) = 2N_0$. For the complex baseband signals, the inner product is defined by $\langle x, y \rangle \triangleq \int_T x(t)y(t)^* dt$, where * denotes the complex conjugate. Also, $\|x\|^2 \triangleq \langle x, x \rangle = \int_T x(t)x(t)^* dt = \int_T |x(t)|^2 dt$.

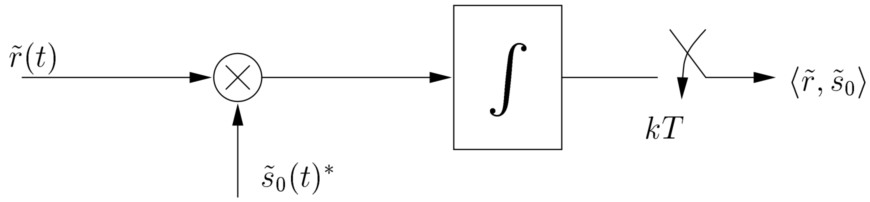


Figure 1: A matched filter sampled once per symbol.

2.1.2 Conversion to discrete-time

A sampled system is used. A Matched Filter (MF) at the front-end of the receiver has a continuous-time input and produces a discrete-time output used by the tracking loops, which are implemented digitally.

The continuous-time received signal $\tilde{r}(t) = \tilde{s}(t) + \tilde{n}(t)$ is the input to a complex MF, as shown in Fig. 1 Let $\tilde{s}_0(t) = -jA$ denote the signal $\tilde{s}(t)$ with no modulation or phase offset present. The output of the MF for the kth transmitted symbol is

$$\langle \tilde{r}, \tilde{s}_0 \rangle = \int_{(k-1)T}^{kT} \tilde{r}(t) jA dt = A^2 e^{j\beta c_k} \int_{(k-1)T}^{kT} e^{j\theta(t)} dt + N, \tag{3}$$

where $N = N_R + jN_I$, and where N_R, N_I each have a distribution of $N(0, N_0 A^2 T)$. Because a rectangular pulse shape is used, the MF does nothing more than integrate and dump.

2.2 The channel code

The coded bits $\{c_k\}$ are obtained from an uncoded information stream $\{a_k\}$ by the standard NASA rate 1/2, constraint length 7 convolutional code shown in Fig. 2. This is the convolutional code with

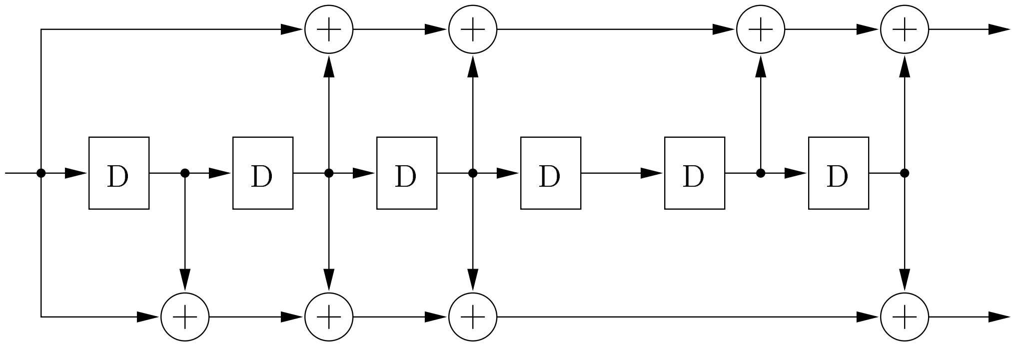


Figure 2: NASA standard (7,1/2) convolutional code.

generators $(133)_{oct}$ and $(171)_{oct}$, where the most significant bit represents D^6, i.e.,

$$\begin{aligned} g^1(D) &= 1 + D^2 + D^3 + D^5 + D^6 \\ g^2(D) &= 1 + D + D^2 + D^3 + D^6. \end{aligned}$$

We have striven to be consistent with the literature, but there are others that use different octal representations for generators. For example, in [LC83] the *least* significant bit represents D^6.

2.3 The maximum likelihood (ML) receiver

The joint Maximum Likelihood (ML) estimate for the kth bit of an uncoded signal and the unknown phase is given by

$$(\hat{c}_k^{ML}, \hat{\theta}(t)) \triangleq \arg \max_{c_k, \theta(t)} p(r(t)|\theta(t), c_k).$$

Using a Karhunen-Loeve expansion for $r(t)$, it follows (see, e.g., [Pro95, page 335] or [Sim97]) that

$$p(r(t)|\theta(t), c_k) = C \exp \left[\frac{-1}{N_0} \int_{(k-1)T}^{kT} (r(t) - s(t))^2 dt \right],$$

or in complex baseband,

$$p(r(t)|\theta, c_k) = C \exp \left[\frac{-1}{2N_0} \|\tilde{r} - \tilde{s}\|^2 \right]. \tag{4}$$

Eq. (4) is referred to as the *likelihood function.* Expanding,

$$\|\tilde{r} - \tilde{s}\|^2 = \langle \tilde{r} - \tilde{s}, \tilde{r} - \tilde{s} \rangle = \|\tilde{r}\|^2 - 2Re\langle \tilde{r}, \tilde{s} \rangle + \|\tilde{s}\|^2. \tag{5}$$

Plugging (5) into (4), taking the logarithm, and discarding terms which do do not depend on c_k, we have the joint ML estimate of the phase and data

$$(\hat{c}_k^{ML}, \hat{\theta}(t)^{ML}) = \arg \max_{c_k, \theta(t)} Re\langle \tilde{r}, \tilde{s} \rangle. \tag{6}$$

The maximum can be found by computing $\langle \tilde{r}, \tilde{s} \rangle$ under different hypotheses for c_k and $\theta(t)$ and choosing the maximum. This is accomplished in practice by conditioning on a value c_k first, and using the best resulting phase estimate under that hypothesis.

2.3.1 Single sample per symbol receiver

For the kth symbol and hypothesized phase $\hat{\theta}(t)$ and data $\hat{c}_k$, we have hypothesized signal $\tilde{s}(t) = \tilde{s}_0 e^{j[\hat{\theta}(t)+\beta\hat{c}_k]}$. If the true phase $\theta(t)$ does not significantly vary over a single symbol epoch, we may write $\theta_k \triangleq \theta(t)$ for $(k-1)T \leq t < kT$ and thus

$$\begin{aligned} \langle \tilde{r}, \tilde{s} \rangle &= e^{-j[\hat{\theta}_k+\beta\hat{c}_k]}\langle \tilde{r}, \tilde{s}_0 \rangle & (7) \\ &= A^2 T \exp[-j(\theta_k - \hat{\theta}_k) + \beta(c_k - \hat{c}_k))] + N e^{-j(\hat{\theta}_k+\hat{c}_k)} & (8) \end{aligned}$$

may be used in Eq. (6) to determine the ML estimates. Eq. (7) demonstrates that the MF output $\langle \tilde{r}, \tilde{s}_0 \rangle$ is a sufficient statistic for ML detection.

2.3.2 Multiple samples per symbol receiver

If $\theta(t)$ varies significantly over the symbol epoch, then estimating the phase once per symbol can result in a poorly performing receiver. To solve the problem we may sample the output of the MF $m > 1$ times during the integrating interval, and compute a phase estimate more than once per symbol. We choose m sufficiently large so that $\theta(t)$ may be approximated as a constant $\theta_k[i]$ within the ith interval of the kth symbol. The ith discrete sample of the kth symbol at the output of the MF is denoted by

$$r_k[i] \triangleq \int_{t_1}^{t_1+T/m} \tilde{r}(t)\tilde{s}_0(t)^* dt = \frac{A^2 T}{m} e^{j[\theta_k[i]+\beta c_k]} + N_k[i] \tag{9}$$

where $t_1 = (k - 1 + (i/m))T$, and $N_k[i]$ contains independent real and imaginary components each Gaussian and with variance $N_0 A^2 T/m)$. Thus, for a multiple samples per symbol system,

$$\langle \tilde{r}, \tilde{s} \rangle = \sum_{i=0}^{m-1} r_k[i] e^{-j(\hat{\theta}_k[i]+\beta\hat{c}_k)} = \sum_{i=0}^{m-1} \frac{A^2 T}{m} \exp[j(\theta_k[i] - \hat{\theta}_k[i] + \beta(c_k - \hat{c}_k))] + N_k[i] e^{-j(\hat{\theta}_k[i]+\beta\hat{c}_k)}, \tag{10}$$

which can be plugged into Eq. (6) to obtain the ML phase and data estimates. Note that Eq. (10) is identical to Eq. (7) in the case of $m = 1$, in which case $r_k[0] = \langle \tilde{r}, \tilde{s}_0 \rangle$ and $N_k[0] = N$.

3 Phase Noise

This section describes a phase noise model representative of real oscillators, and an efficient method to simulate the phase noise process.

3.1 Phase noise PSD

The one-sided PSD of oscillator phase is often modeled as [Gag95]

$$S(f) = \frac{S_3}{f^3} + \frac{S_2}{f^2} + S_0 \quad \mathrm{rad}^2/\mathrm{Hz}. \tag{11}$$

The S_3/f^3 term is called frequency flicker noise, the S_2/f^2 term is white frequency noise, and the S_0 term is white phase noise. If $S_3 > 0$ or $S_2 > 0$, then this model implies that the power of the signal is infinite, even within a small band about the zero frequency: $\int_0^c S(f)df = \infty$. To avoid this problem, in this report it is assumed that the model above is accurate only for frequencies above some lower frequency f_l.

Oscillator phase noise characteristics are also commonly specified by $L(f)$, which is defined as the ratio of the phase noise power in a sideband of bandwidth 1 Hz to the total signal power, at an offset of f Hz from the carrier frequency. The normalized phase noise power $L(f)$ is related to $S(f)$ in the following way [HSR73]:

$$L(f) \approx 10 \log_{10}\left(\frac{S(f)}{2}\right). \tag{12}$$

See Appendix A for a more detailed discussion of this approximation. Eq. (12) is only valid for small levels of noise; however, Eq. (12) is often treated as a definition, and it will be used as such in this report. The units of $L(f)$ are dBc/Hz, or dB rad^2/Hz.

3.2 Computer generation of phase noise

The numerical results in this report are based on simulations using frequency flicker dominated phase noise, in which all coefficients in Eq. (11) were set to zero except S_3, which is scaled to the desired value. For example, to achieve "-13 dB rad^2/Hz at 1 Hz offset," we have $S(f) = S_3/f^3$ and set S_3 so that $10\log_{10}(S(f)/2) = -13$ dB rad^2/Hz at $f = 1$ Hz. In this example, $S_3 = 1^3 \times 2 \times 10^{-13/10}$, or 0.1002.

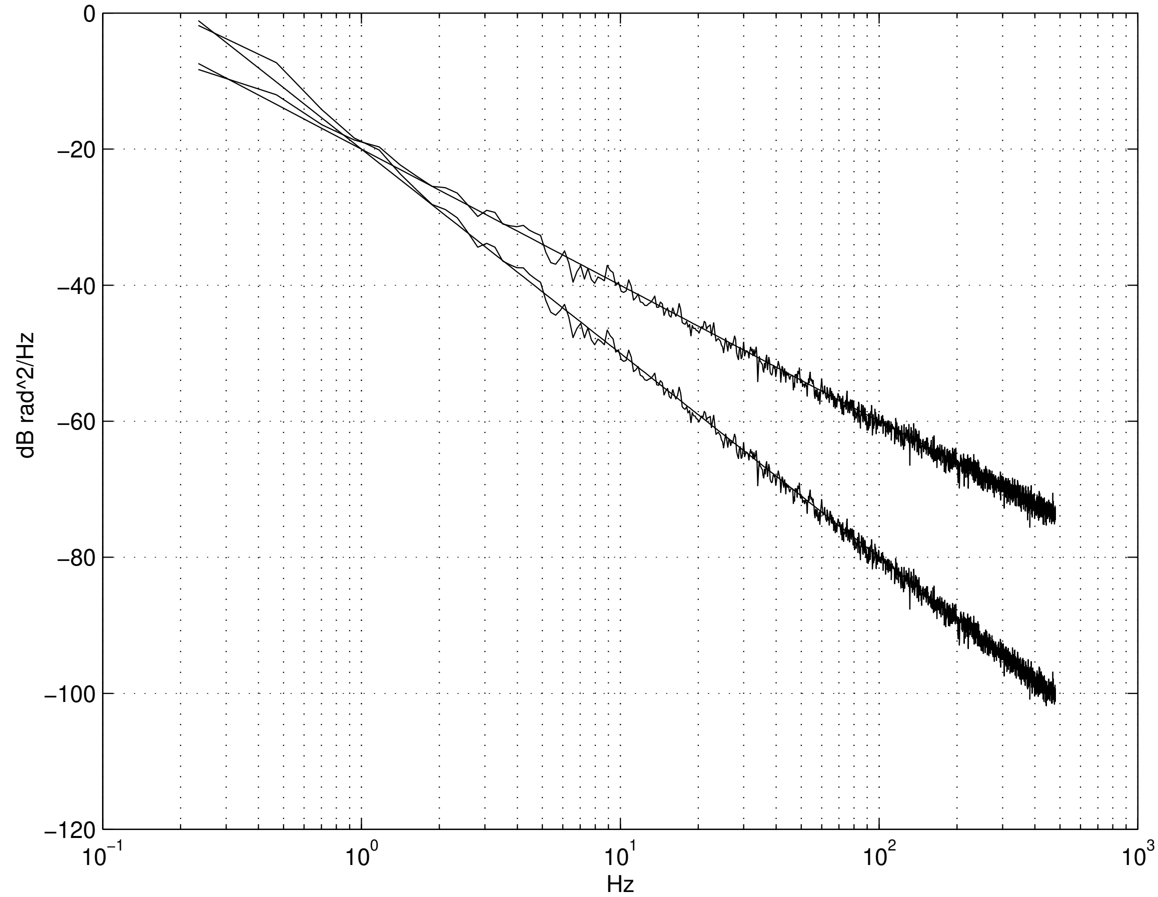


Figure 3: Measured PSD's of 100,000 computer generated phase noise samples, compared to the desired PSD's $S(f) \sim 1/f^3$ and $S(f) \sim 1/f^2$.

This is representative of the phase noise seen on the Cassini Aux Osc [Mak94].

A C program was used to simulate the phase noise. The program allows specification $S(f)$ in the form of Eq. (11), and generates phase samples. The theoretical basis and implementation of this program is discussed in Appendix B. Fig. 3 shows the measured spectral density of computer generated phase noise samples, for two cases. In both cases, -20 dB rad^2/Hz at a 1 Hz offset was desired. In one case, the phase noise was assumed to be dominated by frequency flicker noise $S(f) = S_3/f^3$, and in the other by white frequency noise $S(f) = S_2/f^2$. Note the 30 and 20 dB per decade slopes, respectively. The computer generated phase samples accurately follow the desired spectrum down to -100dBc/Hz.

Alternatively, the program can also calibrate the phase noise by scaling $S(f)$ such that the variance of the innovations sequence, i.e., the incremental variance between phase samples, is a given value. This allows comparison to previous results [Ham98a] that use a Gauss-Markov process for the phase noise.

4 The Joint Receiver-Decoder

In a conventional receiver, incoming samples from a matched filter are fed into a single phase recovery loop, and then into a correlator for each of the hypothesized data values, -1 and +1. In a convolutionally coded system, the receiver outputs are used as partial branch metrics on any branch having the given data hypothesis. This is shown in Fig. 4.

By contrast the joint receiver-decoder uses Per Survivor Processing (PSP) phase tracking within a Viterbi decoder, as shown in Fig. 5. The PSP approach avoids averaging over unknown data, does not incur a squaring loss, and allows for phase estimate updates one or more times per symbol. It involves computing a phase estimate at each state in the Viterbi trellis, taking full advantage of the strength of the code in performing a strong type of data-aided phase tracking. The method is not the maximum likelihood estimator, because the estimates and data are coupled, and the Viterbi algorithm excludes certain data sequences; however, it should exhibit superior performance over uncoupled methods. For a general explanation of Viterbi decoding of convolutional codes, see [Pro95]; for PSP, see [RPT95].

Here is how it works. At each state in the trellis, the Viterbi decoder stores an associated hypothesized data sequence according to the surviving path to that state, as usual. This hypothesized data sequence can be used to unmodulate the signal. If the hypothesized data sequence is the correct one, then the data-wiped signal is a CW signal with phase noise and AWGN, trackable by a PLL or other loop in the usual way.

For each incorrect surviving path, the phase estimate might not be accurate and indeed may be significantly worse than a phase tracker based on averaging over unknown data. For the true data path, however, the phase estimate will be better. Furthermore, the phase estimate at each state requires no decoding delay, so updates from one trellis state to the next can track phases that vary somewhat rapidly.

A second and significant benefit of the approach is that the data-wiped signal is never squared, thereby avoiding the "squaring loss" normally associated with tracking suppressed carrier signals.

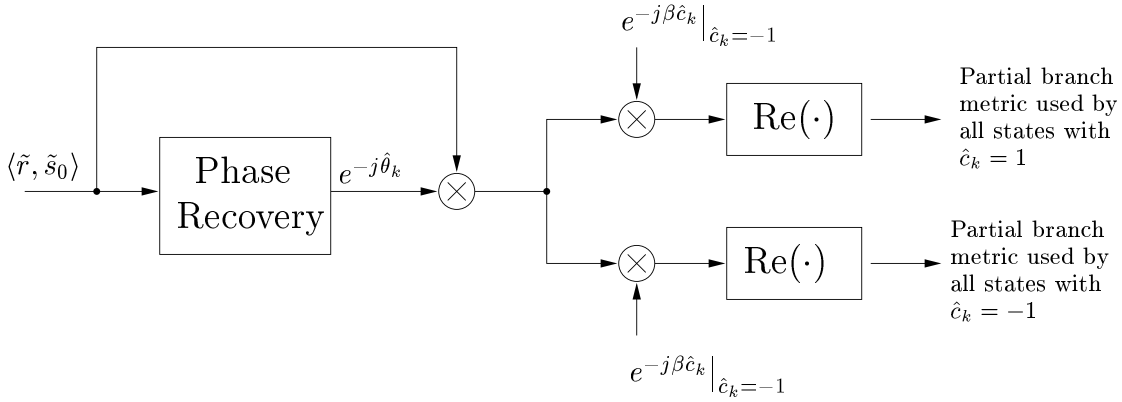


Figure 4: Conventional digital receiver for convolutionally coded BPSK.

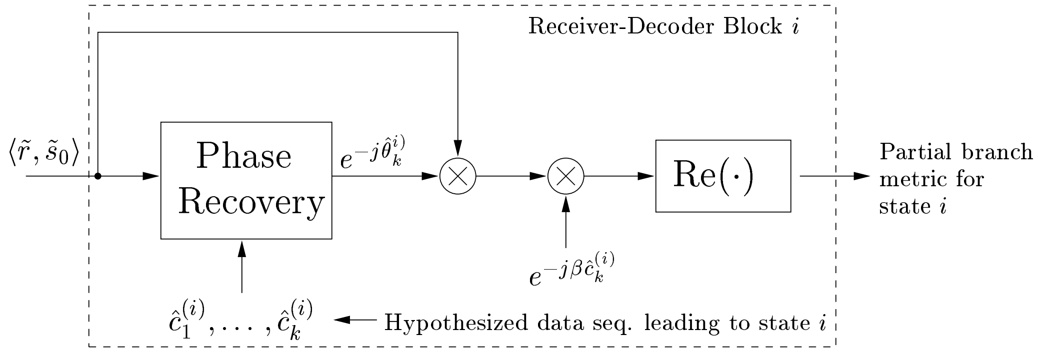


Figure 5: Joint receiver-decoder block for state i. There is such a block for each $i = (1,\ldots,n)$. Phase recovery uses the hypothesized data sequence leading to state i at time k.

4.1 Phase estimation

In principle, any phase tracking method that can be used for a CW signal can be used within the joint receiver-decoder. Because multiple instances of the tracker are required, there is a complexity limitation. Most loops are simple enough that this is not a problem, however. In the following sections we describe the implementation of a PLL, an open loop MAP phase estimator, and a Kalman filter estimator.

4.1.1 PLL

The phase error variance (mean squared phase error, in radians^2) of the PLL is given by the sum of the variance due to thermal noise and the variance due to the phase noise. Under the assumption that the data-wipe has been performed correctly, the suppressed carrier signal becomes a CW signal with carrier power equal to the original total signal power P_t. Thus, the phase error variance due to thermal noise is given by

$$\sigma^2_{\text{AWGN}} \approx \frac{1}{\rho_L} \triangleq \frac{B_L N_0}{P_t} = \frac{B_L}{R_d(E_b/N_0)},$$

where ρ_L is the loop SNR, R_d is the data rate, and B_L is the loop bandwidth. The thermal noise term is increasing in B_L, which represents the fact that opening up the loop bandwidth lets in more noise power to the loop.

The phase error variance due to the phase noise is given by [Kin96]

$$\sigma^2_{\text{phase-noise}} \approx \int_0^\infty S(f)|1 - H(f)|^2 df.$$

For a second order underdamped PLL tracking of frequency flicker dominated noise, this has been shown to be accurately approximated by $\sigma^2_{\text{phase-noise}} = kS_3/B_L^2$ [Kin96], with $k = 8.7$. The phase noise term is decreasing in B_L, indicating that opening the loop bandwidth allows one to better track the dynamics of the phase noise. As can be seen from Fig. 6, the approximations are extremely accurate throughout the entire range of loop bandwidths. A discrepancy begins to appear as the loop bandwidth approaches the Nyquist rate, as aliasing begins to occur.

The total phase error variance is given by

$$\sigma^2_\phi \triangleq \sigma^2_{\text{AWGN}} + \sigma^2_{\text{phase-noise}} \approx \frac{B_L}{R_d(E_b/N_0)} + \frac{kS_3}{B_L^2}. \tag{13}$$

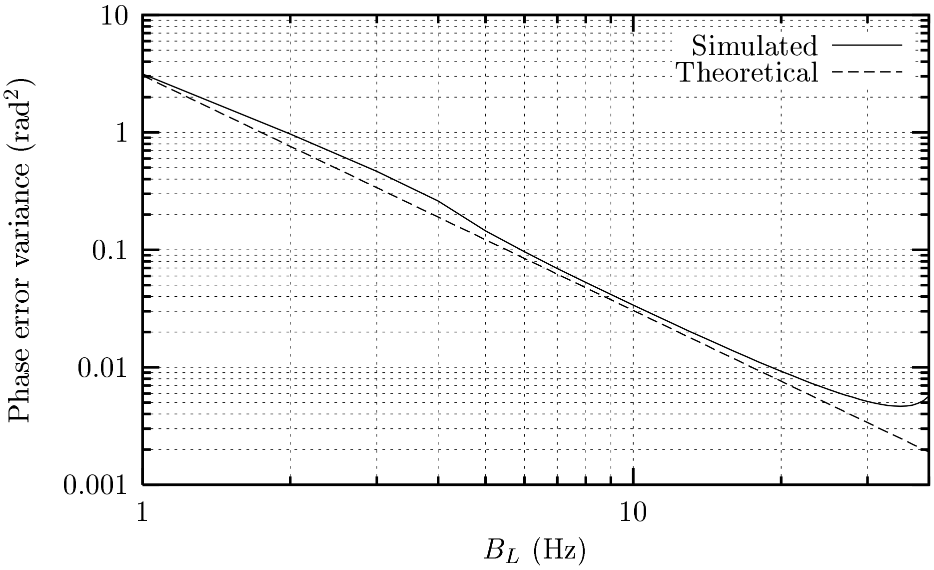


Figure 6: Simulated vs. theoretical performance of a second order underdamped PLL on a CW signal with -7.57 dB rad^2/Hz at 1 Hz offset and no AWGN, as a function of loop bandwidth. The sample rate is $F_s = 80$ Hz.

The optimal loop bandwidth can be found by differentiating Eq. (13) and solving for B_L. This gives

$$B_L^{opt} \approx (2 R_d k S_3 E_b / N_0)^{1/3}. \tag{14}$$

This optimal loop bandwidth is shown in Fig. 7, for a data rate of 40bps and phase noise levels considered in the numeric portions of the report.

4.1.2 MAP estimation

The classical derivation of the MAP estimate of the phase assumes that the data is unknown. Thus, the estimate involves an averaging over the two possibilities of the data, -1 and $+1$. When known data is hypothesized, a new MAP phase estimate emerges.

We now derive the MAP phase estimate following the general approach of [LS73], but accounting for data which is now hypothetically known. Begin by assuming that the value of $\theta(t)$ remains constant over a period of K bits, $K \geq 1$. The MAP phase estimate is that value of θ that maximizes the conditional

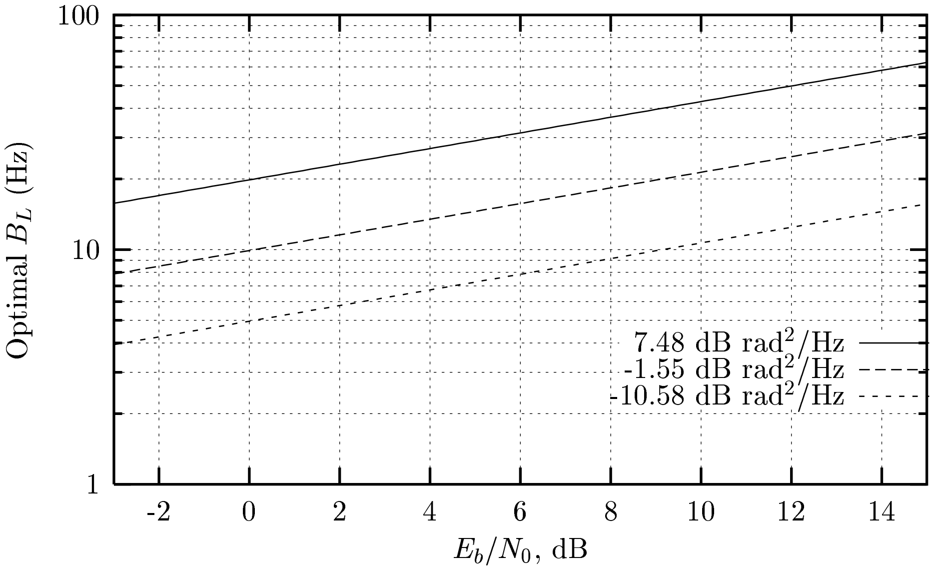


Figure 7: The optimal loop bandwidth for a data rate of 40bps, as a function of E_b/N_0 and phase noise level.

density $p(\theta|r(t), \mathbf{c})$, where $\mathbf{c} = (c_o, \ldots, c_{K-1})$. Using Bayes rule,

$$p(\theta|r(t), \mathbf{c}) = \frac{p(\theta|\mathbf{c})p(r(t)|\theta, \mathbf{c})}{p(r(t)|\mathbf{c})}.$$

We reasonably assume that θ and $\mathbf{c}$ are independent and that θ is uniformly distributed in $[0, 2\pi)$, and thus $p(\theta|\mathbf{c})$ and $p(r(t)|\mathbf{c})$ are independent of θ. Hence, the MAP estimate is given by

$$\hat{\theta}^{MAP} = \arg\max_{\theta} p(r(t)|\theta, \mathbf{c}).$$

That is, the MAP estimate is the same as the ML estimate. Following the analysis in Section 2.3, it follows that

$$\hat{\theta}^{MAP} = \arg\max_{\theta} Re\langle\tilde{r}, \tilde{s}\rangle. \tag{15}$$

In a typical implementation, in which $\tilde{s}$ is not completely determined because $\mathbf{c}$ is unknown, the MAP estimate is obtained from Eq. (15) by averaging over the possible values of $\mathbf{c}$. When operating with $\beta = \pi/2$ and equiprobable signals, it can be shown [Pro95, page 351] that this results in a MAP estimate of

$$\begin{aligned}
\hat{\theta}^{MAP} &= \arg\max_{\theta} \sum_{k=1}^{K} \ln\cosh\left[\frac{2A}{N_0}\int_{(k-1)T}^{kT} r(t)\cos(\omega_c t+\theta)dt\right], \text{ in passband notation} \\
&= \arg\max_{\theta} \sum_{k=1}^{K} \ln\cosh\left[\frac{A}{N_0} Im \int_{(k-1)T}^{kT} \tilde{r}(t)e^{-j\theta}dt\right], \text{ in complex baseband notation.}
\end{aligned}$$

Note that this estimate does not contain a reference to the data vector $\mathbf{c}$. Also, it requires an external estimate of the SNR, and the argmax may have to be accomplished by choosing among multiple correlator outputs.

In contrast, the PSP MAP estimate is given by substituting a hypothesized $\hat{m}(t)$ into Eq. (2) and then Eq. (15), which gives

$$\hat{\theta}^{MAP} = \arg\max_{\theta} Re \int_0^{KT} \tilde{r}(t)\exp\left[-j(\theta - \pi/2 + \beta\hat{m}(t))\right]dt. \tag{16}$$

A necessary condition for achieving the maximum is that the derivative of (16) with respect to θ is zero, i.e.,

$$Re \int_0^{KT} \tilde{r}(t)(-j)\exp\left[-j(\hat{\theta}^{MAP} - \pi/2 + \beta\hat{m}(t))\right]dt = 0.$$

After a few lines of simplification via trigonometric formulas we obtain

$$\hat{\theta}^{MAP} = -\tan^{-1}\left[\frac{Re\int_0^{KT}\tilde{r}(t)e^{-j\beta\hat{m}(t)}dt}{Im\int_0^{KT}\tilde{r}(t)e^{-j\beta\hat{m}(t)}dt}\right]. \tag{17}$$

This PSP MAP estimate should result in improved tracking because it is conditioned on a hypothesized data sequence, does not require an external estimate of SNR, and can be implemented exactly with a single complex baseband correlator.

4.1.3 Kalman filter estimation

The MAP phase estimator of the previous section gives the best performance possible with an observation interval of K bits, *under the assumption that the phase is constant within that interval.* However, we are most interested in improving receiver performance for rapidly varying phase noise cases, in which the phase can be viewed as constant for only very short intervals. Tracking loops or filters have the potential to do better than a one-shot MAP estimate if they are able to make effective use of an observation interval longer than that in which the phase may be accurately assumed to be constant.

One such method is a Kalman filter, which gives optimal MSE tracking performance when the observable is linear in the phase error and the the phase varies according to a Gauss-Markov random sequence. In this model, the phase is constant within a single codeword interval (two bits, for the (7,1/2) code) and between these intervals varies according to

$$\theta_{k+1} = \theta_k + U_k, \tag{18}$$

where $\theta_k = \theta(2kT)$ and where $U_1, U_2, \ldots$ are i.i.d. zero-mean Gaussian random variables with variance σ_θ^2. Assuming known data and that the observable is the imaginary part of the correlation output in (9), except that we are now integrating over two bits,

$$Y_k = 2A^2T \sin(\theta_k - \hat{\theta}_{k|k-1}) + V_k \approx 2A^2T(\theta_k - \hat{\theta}_k) + V_k, \tag{19}$$

where $\hat{\theta}_{k|k-1}$ is the estimate of θ_k computed from observables $Y_1, \ldots, Y_{k-1}$, and where $V_1, V_2, \ldots$ are i.i.d. zero-mean Gaussian random variables with variance $2N_0A^2T$ and are independent of $U_1, U_2, \ldots$. The goal is to determine the following quantities:

$$\begin{aligned}
\hat{\theta}_{k|k} &\triangleq E[\theta_k | Y_1, \ldots, Y_k] \\
\hat{\theta}_{k+1|k} &\triangleq E[\theta_{k+1} | Y_1, \ldots, Y_k] \\
\hat{\Sigma}_{k|k-1} &\triangleq \mathrm{Var}(\theta_k | Y_1, \ldots, Y_{k-1}) \\
\hat{\Sigma}_{k|k} &\triangleq \mathrm{Var}(\theta_k | Y_1, \ldots, Y_k).
\end{aligned}$$

By the orthogonality principle, $\hat{\theta}_{k|k}$ as defined above minimizes the mean-squared error $E\|\theta_k - \hat{\theta}_{k|k}\|^2$.

The solution is given by (see, e.g., [Poo88, page 292-293])

$$
\begin{aligned}
\hat{\theta}_{k|k} &= \hat{\theta}_{k|k-1} + \frac{\Sigma_{k|k-1} Y_k}{2A^2 T \Sigma_{k|k-1} + N_0} \\
\theta_{k+1|k} &= \hat{\theta}_{k|k} \\
\Sigma_{k|k} &= \frac{\Sigma_{k|k-1} N_0}{2A^2 T \Sigma_{k|k-1} + N_0} \\
\Sigma_{k+1|k} &= \Sigma_{k|k} + \sigma_\theta^2,
\end{aligned}
$$

with the initialization $\hat{\theta}_{1|0} = E[\theta_1] = \pi$ (assuming θ_1 is uniform on $[0, 2\pi)$) and $\Sigma_{1|0} = \mathrm{Var}[\theta_1^2] = \sigma_\theta^2$.

5 Nonjoint receiver-decoder performance

The performance of a nonjoint phase tracking and decoding system would be quite bad for a suppressed carrier signal in a high phase noise environment, and would result in a lopsided comparison to the joint receiver-decoder. However, using the same total power the performance of a nonjoint system can be improved by reallocating power to a residual carrier.

The amount of power to reallocate to the residual carrier is chosen as that which minimizes the bit error rate, given by [Kin96]

$$
P_b = \int_{-\pi/2}^{\pi/2} f\left(\frac{E_b}{N_0}\cos^2\phi\right) p_\phi(\phi)\,d\phi = \int_{-\pi/2}^{\pi/2} f\left(\frac{P_t \sin^2\beta}{N_0 R_d}\cos^2\phi\right) p_\phi(\phi)\,d\phi, \tag{20}
$$

where $f(\cdot)$ is the baseline BER vs. E_b/N_0 performance when the phase is known exactly, $p_\phi(\phi)$ is the probability density function of the phase error ϕ given by

$$
p_\phi(\phi) = \frac{\exp\left(\frac{\cos\phi}{\sigma_\phi^2}\right)}{\int_{-\pi/2}^{\pi/2} \exp\left(\frac{\cos\psi}{\sigma_\phi^2}\right) d\psi}, \quad |\phi| \le \pi/2, \tag{21}
$$

and σ_ϕ^2 is the phase error variance of the receiver. Assuming that a subcarrier is used to move the modulation sidebands away from the residual carrier, and a second order underdamped PLL is used to track the frequency flicker dominated phase, the phase error variance is given by

$$\sigma_\phi^2 = \sigma_{\text{AWGN}}^2 + \sigma_{\text{phase-noise}}^2 \approx \frac{B_L N_0}{P_t \sin^2 \beta} + \frac{k S_3}{B_L^2}, \tag{22}$$

with $k = 8.7$. If sideband data aiding is used the phase error variance may be lower, but we shall not consider this case in this report. The loop bandwidth B_L is optimized by taking the derivative of Eq. (22) w.r.t. B_L and setting it equal to zero, which gives

$$B_L^{opt} = (2kS_3(\sin^2 \beta) P_t/N_0)^{1/3}. \tag{23}$$

After plugging Eq. (23) into Eq. (22), and then Eq. (21) and finally Eq. (20), the error rate is expressed as a function

$$P_b = P_b(P_t/(N_0 R_d), \beta).$$

For a fixed $P_t/(N_0 R_d)$, the β which minimizes P_b is declared the optimal modulation index, and the resulting BER performance and phase error variance are reported.

The method above is essentially the method used by [Sha95], approached from another direction. In [Sha95], the minimum P_t/N_0 is found for a given R_d and fixed error rate.

6 Performance Simulations

The joint receiver decoder with PLL phase tracking was implemented in C. For each data point, 10,000 errors or 10,000,000 information bits were simulated, whichever came first. To simplify the simulations, we assume perfect timing synchronization, no pulse spreading, and AWGN. The decision statistics are outputs of a MF implemented in complex baseband, as in Eq. (9). Soft-decision decoding is used. Coarser soft-decision quantization more typical of the high speed DSN receivers will perform slightly worse. Fig. 2 lists the parameters used in the simulations. The nonjoint results are from a spreadsheet [Sha99], using the method in Section 5.

Time constraints did not allow simulations for the Kalman filter or MAP algorithms, which are expected to perform about the same or slightly worse than the PLL.

Fig.s 8-10 show BER performance and phase error variance performance for phase noise levels of -10.58, -1.55, and 7.48 dB rad^2/Hz, respectively. Loosely speaking, these represent harsh, very harsh,

Table 2: Parameters used in the high phase-noise BPSK simulations

Encoder parameters:	
Code:	Convolutional
Constraint length:	7
Code rate:	1/2
Generator polynomials:	133 171 (D^0=lsb)
Number of trellis states:	64
Differential encoding:	Yes
Modulation parameters:	
Amplitude:	1.000000
Bit period (coded bit):	0.012500 sec
Data rate:	40 bps
Carrier frequency:	0 Hz
Initial carrier phase:	0.000000 radians
Mod index (angle):	1.570796 radians (90 degrees)
Pulse shape:	Rectangular
Channel parameters:	
Channel type:	AWGN
Eb/No:	1-15 dB
Phase noise:	
Type:	Frequency flicker dominated
PSD (1-sided):	$0.175/f^3$, $1.4/f^3$, $11.2/f^3$ rad^2/Hz
Increment variance:	5.30e-04, 4.24e-03, 3.39e-02 (b/w samples)
Power at 1 Hz offset:	-10.58, -1.55, 7.48 dB rad^2/Hz
Phase stored modulo 2π:	Yes
Length of phase filter:	4096
Samples per code bit:	1, 4 (F_s = 80 or 320)
Receiver/decoder parameters:	
Phase tracking method:	Second-order standard underdamped PLL
PLL loop bandwidth:	5-60 Hz (optimized by Eq. (14))
Decoding delay:	100
Number of errors to simulate:	10,000, or 10 million bits

and extremely harsh phase noise environments.

Because a Reed-Solomon outer code is usually used with the inner NASA standard (7,1/2) code, the BER region of interest is in the vicinity of 10^{-3} to 10^{-1}. For example, when using a RS(255,223) outer code, a bit error rate of 0.01 from the inner code will result in an end-to-end BER of less than 10^{-6}. As can be seen from Fig.s 8-10, the joint receiver-decoder saves 3 dB, 3 dB, and 4.25 dB at a BER of 0.01 and phase noise levels of -10.58, -1.55, and 7.48 dB rad^2/Hz, respectively.

The phase error variance for the joint and nonjoint receiver-decoders are shown in Fig.s 8(b), 9(b), and 10(b). The performance of the nonjoint receiver is a function of the carrier power to noise ratio, P_c/N_0. The flat region is one in which P_c/N_0 is constant, a consequence of the requirement that a loop SNR of 10 dB must be maintained [Sha99]. In Fig. 8(b), we see that at about 4 dB, all power in the nonjoint system is devoted to the carrier, i.e., $P_t = P_c$. The joint receiver-decoder operates on a suppressed carrier signal ($P_c = 0$), and slowly approaches the performance of a CW signal as $P_t/(N_0 R_d)$ increases.

As can also be seen in Fig.s 8(a), 9(a), and 10(a), the gain of the joint approach over a nonjoint approach decreases as $P_t/(N_0 R_d)$ increases. The reason for this is unknown, but may possibly be the way in which the PLL for the joint receiver-decoder is implemented. Two approaches were used, one in which phase samples were generated once per symbol, and one in which the phase samples were generated four times per symbol. For the multiple samples per symbol case, the joint receiver-decoder also produced phase updates four times per symbol. For a phase noise level of -10.58 dB rad^2/Hz and low SNR, the multiple updates per symbol was slightly worse than a single update per symbol, representing the fact that the PLL is tracking fluctuations due to AWGN, not true phase changes. On the other hand, at 7.48 dB rad^2/Hz, the multiple updates/symbol helps quite a bit.

There is a cross-over point for which one method works better than the other. This is illustrated in Fig. 11, for a phase noise level of -1.55 dB rad^2/Hz. As can be seen, below approximately 7 dB, a single phase update per symbol works better. Above 7 dB, four updates per symbol works better. Thus, there is some amount of PLL optimization, beyond optimizing the loop bandwidth B_L, which can yield improved results. We expect that when the phase tracking is optimized, the joint receiver-decoder performance curve will have the same slope as the baseline performance curve with no phase error.

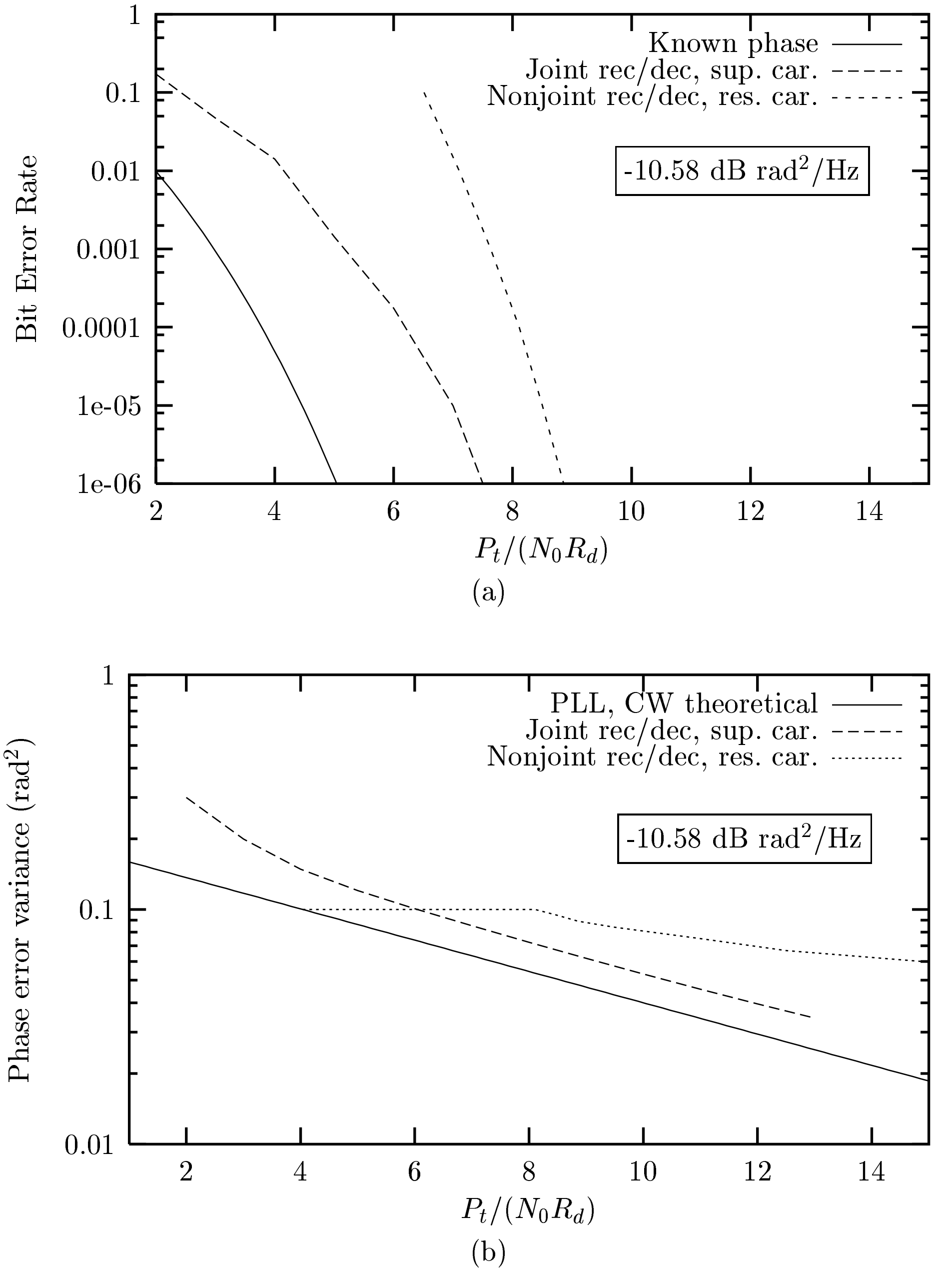
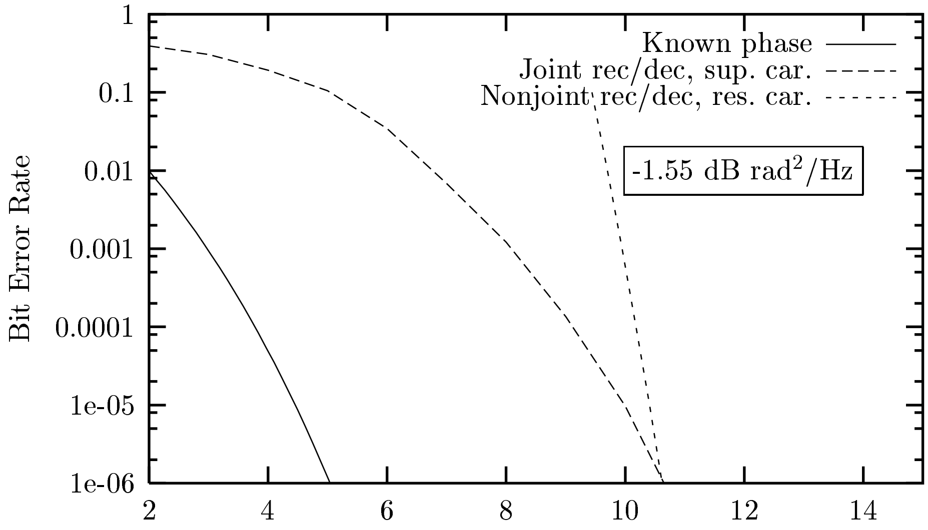
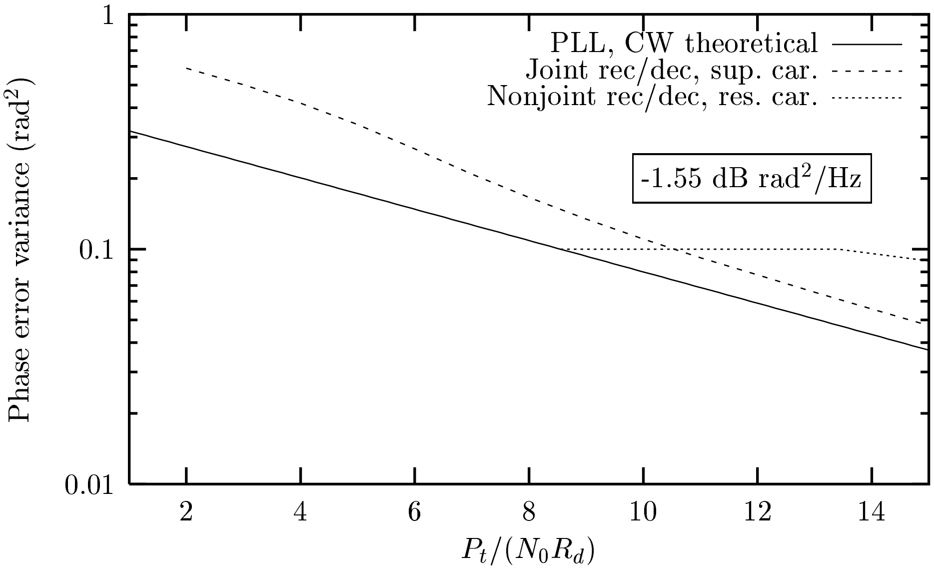


Figure 8: Comparison of (a) BER performance and (b) phase error performance, for joint and nonjoint receiver-decoders, when the phase noise level is -10.58 dB rad^2/Hz at 1Hz offset. The known phase and joint receiver-decoder curves are for a suppressed carrier signal, where $P_t/(N_0 R_d) = E_b/N_0$. The nonjoint receiver operates on a signal of the same total power, a fraction of which is allocated to a residual carrier in a manner that optimizes BER vs P_t/N_0 performance.

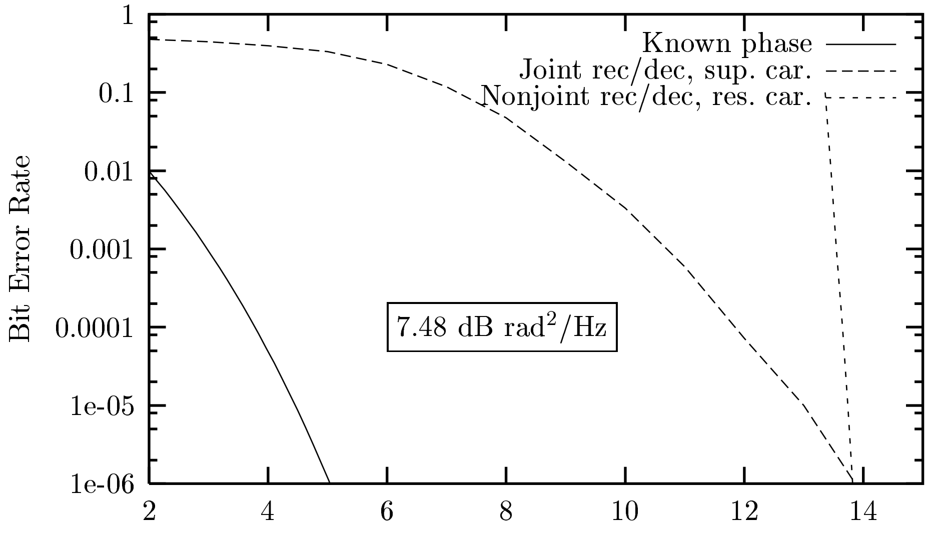


(a)

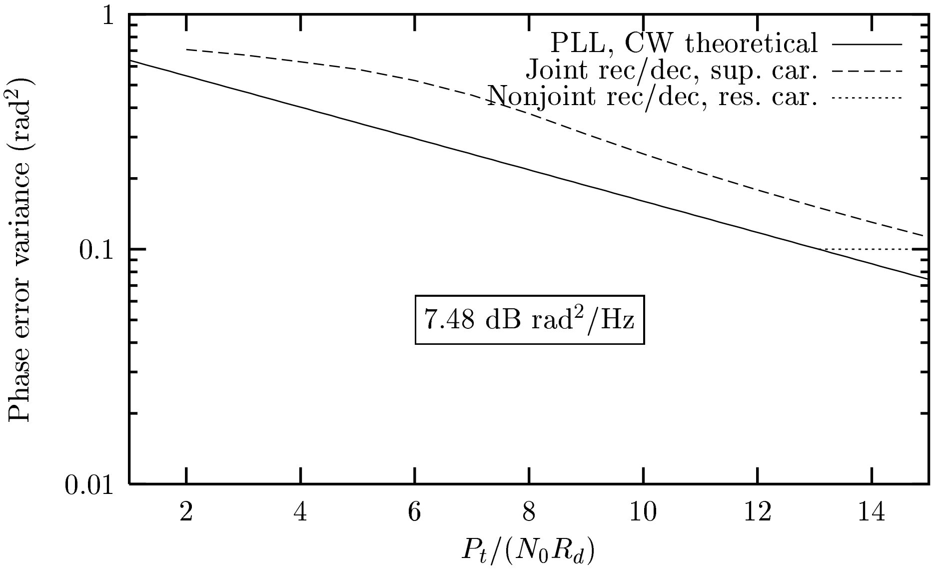


(b)

Figure 9: Comparison of (a) BER performance and (b) phase error performance, for joint and nonjoint receiver-decoders, when the phase noise level is -1.55 dB rad^2/Hz at 1Hz offset. The known phase and joint receiver-decoder curves are for a suppressed carrier signal, where $P_t/(N_0R_d) = E_b/N_0$. The nonjoint receiver operates on a signal of the same total power, a fraction of which is allocated to a residual carrier in a manner that optimizes BER vs P_t/N_0 performance.



(a)



(b)

Figure 10: Comparison of (a) BER performance and (b) phase error performance, for joint and nonjoint receiver-decoders, when the phase noise level is 7.48 dB rad^2/Hz at 1Hz offset. The known phase and joint receiver-decoder curves are for a suppressed carrier signal, where $P_t/(N_0 R_d) = E_b/N_0$. The nonjoint receiver operates on a signal of the same total power, a fraction of which is allocated to a residual carrier in a manner that optimizes BER vs P_t/N_0 performance.

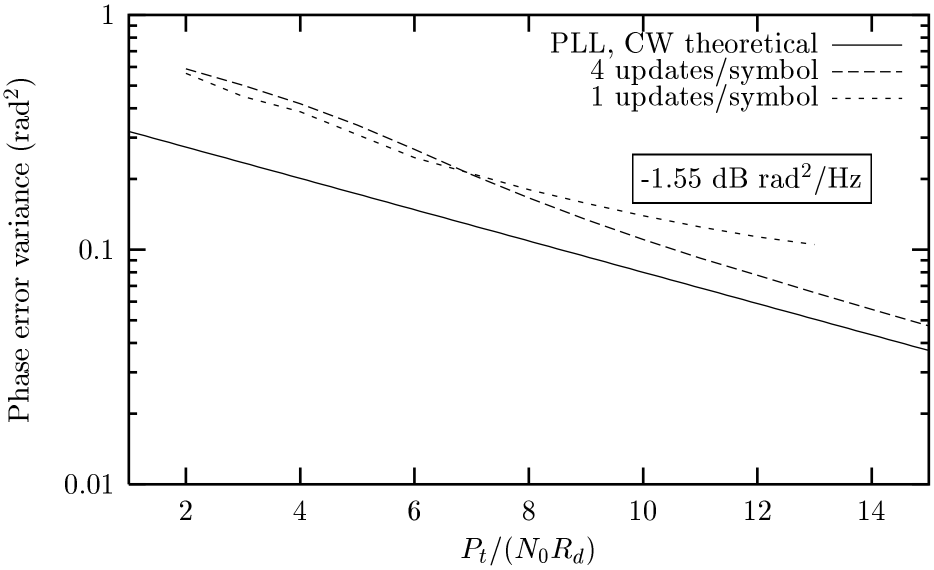


Figure 11: Comparison of joint receiver-decoder phase tracking for one and four phase estimate updates per symbol. For low $P_t/(N_0R_d)$, more updates per symbol results in tracking of unwanted AWGN effects on the observed phase, while for high $P_t/(N_0R_d)$ it results in better tracking of the true phase.

7 Conclusions

This report presented a method to jointly estimate phase and data from a convolutionally coded BPSK signal with random phase noise and AWGN. The joint receiver-decoder successfully decodes fully suppressed carrier signals in harsh phase noise environments. A complete description was given of the software implementation, including the generation of statistically accurate phase noise samples.

Numerical results were given for the NASA standard (7,1/2) convolutional code and frequency flicker dominated phase noise. Simulations indicated that for phase noise levels of -10.58, -1.55 and 7.48 dB $\mathrm{rad}^2/\mathrm{Hz}$ at a 1 Hz offset— all three are harsh phase noise environments— a data rate of 40bps, and a BER of 0.01, that the joint receiver decoder saves 3 to 4.25 dB of power over a nonjoint approach. This is so despite operating on a fully suppressed carrier signal, instead of the residual signal (with optimal mod index) used in the comparisons with the nonjoint receiver-decoder.

It should be emphasized that the nonjoint receiver we compare to in this report does not use sideband data-aiding. Gains possible from sideband data-aiding would reduce the reported gains of the joint receiver-decoder.

Future work should include a proper study of the optimization of the phase tracking. This includes development of better tracking loops, the effect of single vs. multiple phase updates per symbol, and methods to combat phase estimate errors that result from burst decoding errors. Also, there is as yet no analytic development of performance bounds, which would be very helpful in generating quick performance estimates. The joint receiver-decoder presented here can also be extended to QPSK and OQPSK signalling, and to joint symbol-synchronization, phase-tracking, and decoding.

References

[Gag95] R. M. Gagliardi. *Introduction to Communications Engineering.* John Wiley & Sons, New York, 1995.

[Ham98a] J. Hamkins. PSP phase-tracking of convolutionally coded BPSK. JPL IOM 3315-99-02, September 1998.

[Ham98b] J. Hamkins. Remarks on noise generation. JPL IOM 331.98.12.007, December 1998.

[HSR73] D. Halford, J. H. Shoaf, and A. S. Risley. Frequency domain specification and measurement of signal stability. In *Proc. 27th Ann. Symp. Freq. Control*, pages 421–431, 1973.

[Kin96] P. W. Kinman. TLM-21 DSN telemetry system, Block-V Receiver, 810-5, rev. D. JPL document, December 1996.

[LC83] S. Lin and D. J. Costello Jr. *Error Control Coding: Fundamentals and Applications.* Prentice-Hall, New Jersey, 1983.

[LS73] W. C. Lindsey and M. K. Simon. *Telecommunication Systems Engineering.* Dover, Toronto, Ontario, 1973.

[Mak94] A. Makovsky. Effects of Cassini aux osc phase noise on telemetry BER. JPL IOM 3391-94-100, December 1994.

[OS89] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing.* Prentice Hall, Englewood Cliffs, NJ, 1989.

[Poo88] H. V. Poor. *An Introduction to Signal Detection and Estimation.* Springer-Verlag, New York, 1988.

[Pro95] J. G. Proakis. *Digital Communications.* McGraw Hill, Inc., third edition, 1995.

[PVTF92] W. H. Press, W. T. Vetterling, S. A. Teukolsky, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, New York, 1992.

[RPT95] R. Raheli, A. Polydoros, and C.-K. Tzou. Per-survivor processing: a general approach to MLSE in uncertain environments. *IEEE Trans. Commun.*, 43(2/3/4):354–364, February/March/April 1995.

[Sha95] S. Shambayati. Preliminary results on optimum settings for BVR for tracking of Voyager 1. JPL IOM 3315-95-SS08, December 1995.

[Sha99] S. Shambayati. Private communication, May 1999.

[Sim97] M. K. Simon. Synchronization for space communications. JPL seminar series, 1997.

Appendix A
Relationship of $S(f)$ to $L(f)$

Oscillator phase noise characteristics are also commonly specified by $L(f)$, which is defined as the ratio of the phase noise power in a sideband of bandwidth 1 Hz to the total signal power, at an offset of f Hz from the carrier frequency. The normalized phase noise power $L(f)$ is related to $S(f)$ in the following way [HSR73]. let $s(t)$ be a phase-noisy signal of the form

$$s(t) = A\exp[j(\theta(t) + \phi(t))], \tag{24}$$

where $\theta(t)$ is the phase of an oscillator (plus angle modulation, if any) and where $\phi(t)$ is the phase variation due to phase noise. The "small angle condition" holds at frequency f if the phase fluctuations occurring at rates f Hz or faster are small compared to one radian, i.e.

$$\int_f^\infty S_\phi(f')df' << 1 \quad \mathrm{rad}^2. \tag{25}$$

If this condition holds, then $\sin\phi(t) \approx \phi(t)$ and $\cos\phi(t) \approx 1$, and we may rewrite Eq. (24) as

$$s(t) = Ae^{j\theta(t)} \cdot (\cos\phi(t) + j\sin\phi(t)) \approx Ae^{j\theta(t)}(1 + j\phi(t)) = Ae^{j\theta(t)} + A\phi(t)e^{j(\theta(t)+\pi/2)}. \tag{26}$$

The first term is the signal with no phase noise present, and its power is A^2. The second term is the phase noise contribution, and its average power in a 1 Hz bandwidth centered at frequency f is

$$\int_{f-\frac{1}{2}}^{f+\frac{1}{2}} A^2 \left(\frac{S_\phi(f)}{2}\right) df \approx \frac{A^2 S_\phi(f)}{2}, \tag{27}$$

assuming $S_\phi(f)$ does not vary much within the integration region. The factor of 1/2 is introduced in Eq. (27) because $S_\phi(f)$ is a one-sided power spectral density. Thus, the ratio of the noise power in the 1 Hz sideband to the signal power, in decibels, is approximately [Kin96, HSR73]

$$L(f) \approx 10\log_{10}\left(\frac{A^2 S_\phi(f)/2}{A^2}\right) = 10\log_{10}\left(\frac{S_\phi(f)}{2}\right). \tag{28}$$

There are a couple of areas of confusion regarding this formulation. First, the units of $L(f)$ are usually given as “dBc/Hz”, which indicates the number of decibels the phase noise power is below the carrier power. This is somewhat of a misnomer because, for example, a suppressed-carrier modulated signal might have very little power at the carrier frequency. It is the decibels below the *total* signal power that $L(f)$ measures, not the decibels below the carrier power [HSR73]. Second, the small angle condition does not always hold, particularly for the high phase noise cases this report considers. The approximation in Eq. (28) is not valid in those cases.

For these reasons, this report does not refer to $L(f)$ explicitly; instead, phase noise is given by the right hand side of Eq. (28) and specifying the type of phase noise, e.g., frequency flicker dominated phase noise of the form $1/f^3$. The units of $10\log_{10}(S(f)/2)$ are dB rad^2/Hz. When the small angle condition holds, this is the same as specifying the phase noise in dBc/Hz and using $L(f)$.

Appendix B
Computer generation of phase noise

B.1 The basic idea

The basic idea of the simulation is to put a sequence of i.i.d. Gaussian random deviates through a finite impulse response (FIR) filter, thereby changing the white PSD into the desired PSD. It is easier to first describe this process in continuous time, and then describe how this is approximated in discrete-time.

In continuous-time, a random process $X(t)$ with one-sided power spectral density $S_{XX}(f) = N_0$ is the input to a linear filter with transfer function $H(f)$. The output $Y(t)$ has PSD $S_{YY}(f) = |H(f)|^2 S_{XX}(f) = |H(f)|^2 N_0$, and thus any $H(f)$ that satisfies $|H(f)|^2 = S_{desired}(f)/N_0$ will result in $Y(t)$ having the desired spectrum. In particular, $H(f)$ may be chosen real, which gives $H(f) = \sqrt{\frac{S_{desired}(f)}{N_0}}$. The impulse response is given by the inverse Fourier transform[1]

$$h(t) = \int_{-\infty}^{\infty} \sqrt{\frac{S_{desired}(f)}{N_0}} e^{-2\pi j f t} df. \tag{29}$$

The discrete impulse response is a sampled version of the continuous version, $h_k \triangleq h(t_k)$, where $t_k = k\Delta$ are the sampling times and $\Delta = 1/F_s$ is the sampling interval. The relationship between h_k and H_n is given by the discrete Fourier transform (or fast Fourier transform (FFT)), defined by

$$h_k \triangleq \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi j k n/N}, \tag{30}$$

$$H_n \triangleq \sum_{n=0}^{N-1} h_k e^{2\pi j k n/N}. \tag{31}$$

One can demonstrate (see [Ham98b]) that $\Delta H_n \approx H(f_n)$ for $n = -N/2, \ldots, N/2$, where $f_n = n/(N\Delta)$, provided that (1) $h(t) \approx 0$ outside of $(0, (N-1)\Delta)$, and (2) $H(f) \approx 0$ for all $|f| > F_s/2$. Furthermore, H_n is periodic with period N. Rather than letting n vary between $-N/2$ and $N/2$, it is customary when using the FFT to vary n from 0 to $N-1$. In this way, h_k and H_n have the same set of indices,

[1] Usually the inverse Fourier transform is defined as $h(t) = \int_{-\infty}^{\infty} H(f) e^{2\pi j f t} df$, which differs from Eq. (29) by the negative sign in the exponent. This is a small difference, however, and this less common definition is used in some standard references, e.g., [Pro95, PVTF92].

$0, \ldots, N-1$. In the frequency domain, the zero frequency corresponds to $n = 0$, positive frequencies correspond to $1 \leq n \leq N/2 - 1$ and negative frequencies correspond to $N/2 \leq n \leq N-1$. The value $N/2$ corresponds to both $F_s/2$ and $-F_s/2$.

B.2 The FFT size and the sample rate

The proper FFT size N to use is that which is sufficient to obtain the dynamic range in the PSD at the output. For phase noise of the form $S(f) = c/f^{\alpha}$, the minimum FFT size is given by

$$N = 2 \cdot 10^{(A-B)/(10\alpha)},$$

where A and B are the desired upper and lower dBc/Hz at which the simulation to is desired to operate. The sample rate must be at least twice as high as the frequency for which the PSD is B dBc/Hz.

As an example, suppose we want to generate a frequency-flicker dominated phase noise with -20 dBc/Hz at a 1 Hz offset, and we want the simulation to be accurate within the entire dynamic range from 0dBc/Hz down to -100dBc/Hz. This implies $L(1) = 10\log_{10}(S(1)/2) = -20$, or $S(1) = 0.02$ and $S(f) = 0.02/f^3$. The frequency for which $10\log_{10}(S(f)/2) = -100$ is $f = 10^{8/3} = 464$ Hz. By the Nyquist criterion, the simulation should produce outputs at a sample rate at least twice that frequency, or 928 Hz. At the other extreme, $10\log_{10}(S(f)/2) = 0$ is satisfied when $f = (0.02)^{1/3} = 0.27$ Hz. Therefore, the FFT should have at least $1/0.27$ frequency bins per Hz, or roughly $928/0.27 = 3420$ bins overall. FFT's work much faster when their size is a power of two, so a value of 4096 could be used.

B.3 The FIR coefficients

An initialization routine is used to define the impulse response an FIR filter, using the following method. Beginning with the continuous-time transfer function $H(f) = \sqrt{S_{desired}(f)/N_0}$, the approximation $H_n \approx (1/\Delta)H(f_n)$ is used to initialize the discrete-time transfer function. Then, for each odd n we multiply H_n by -1. This does not affect the PSD of the filter output, because the PSD is a function of the magnitude of the transfer function; however, it has the beneficial effect of shifting the largest FIR coefficients of the impulse response towards the middle ($k = N/2$) and away from the tails ($k = 0$ and $k = N-1$) [Ham98b]. The FIR coefficients are computed from H_n by taking the inverse FFT, as defined in Eq. (31).

The method does not guarantee that the impulse response tends to zero at the tails. If the tails do not tend to zero, the generated samples will tend to not have the desired PSD. To combat this problem, we subtract off the DC part of the impulse response, i.e., subtract h_0 from every impulse coefficient $h_0, \ldots, h_{N-1}$. The FFT of this modified impulse response has the same $H_1, \ldots, H_{N-1}$ as before, and only the zero-frequency H_0 is different. This is a reasonable modification; from the discussion in Section 1 we know H_0 cannot conform to the frequency flicker type noise because it would have to be infinite to do so. Thus, by assigning the zero frequency response H_0 exactly right, one can guarantee that h_k goes to zero at the tails. Without knowing a priori what this value of H_0 is, we can assign an arbitrary value to H_0, take the inverse FFT, and subtract off the DC part h_0 from all the coefficients $h_0, \ldots, h_{N-1}$.

B.4 The Gaussian random inputs

Gaussian random deviates may be generated from a uniform deviate generator by using the Box-Muller method [PVTF92]. Namely, if x_1 and x_2 are independent and uniformly distributed on $(0, 1)$, then $y_1 = \sqrt{-2\ln x_1}\cos(2\pi x_2)$ and $y_2 = \sqrt{-2\ln x_1}\sin(2\pi x_2)$ are i.i.d. zero mean, unit variance normal deviates.

B.5 The FIR output

B.5.1 With a convolution

A sequence of i.i.d. Gaussian random deviates is generated and used as input to the filter. The ith filter output is determined by a convolution:

$$y_k = \sum_{n=0}^{N-1} h_n x_{k-n}. \tag{32}$$

The first N filter outputs are thrown out $(y_1, \ldots, y_N)$, to give time for a full input sequence to feed into the FIR. After initialization, this method requires N multiplies and $N-1$ additions per output sample, i.e., a computational complexity of $O(N)$ per output sample.

B.5.2 With an FFT and the overlap-add method

For large filter lengths, a convolution can be time consuming. For example, each point on the performance curves in this report was based on the simulation of 80 million phase noise samples. (The 10 million information bits results in 20 million coded bits, each of which was sampled up to four times.) A more efficient method than direct convolution is to multiply the discrete filter transfer function H_n by the FFT of the first N filter inputs, and then take the inverse FFT. Appropriate zero-padding is necessary for this to work properly, and the beginning and ending outputs must be thrown out. This will generate $N/2$ data points.

One disadvantage of this method, however, is that it does not allow for simulations of arbitrary lengths, because there is a practical limit to the size of an FFT that can be performed. This problem may be overcome by using a smaller FFT size and stitching together multiple data sets in the appropriate way. This is known as the "overlap-add" method of generating filter outputs. It is described in more detail in [OS89, p. 558] and [PVTF92]. To quote from [PVTF92],

> If your data set is so long that you do not want to fit it into memory all at once, then you must break it up into sections and convolve each section separately. Now, however, the treatment of end effects is a bit different. You have to worry not only about spurious wrap-around effects, but also about the fact that the ends of each section of data should have been influenced by data at the nearby ends of the immediately preceding and following sections of data, but were not so influenced since only one section of data is in the machine at a time... [One solution is the overlap-add method.] Here you don't overlap the input data. Each section of data is disjoint from the others and is used exactly once. However, you carefully zero-pad it at both ends so that there is no wrap-around ambiguity in the output convolution or deconvolution. Now you overlap and add these sections of output. Thus, an output point near the end of one section will have the response due to the input points at the beginning of the next section of data properly added in to it, and likewise for an output point near the beginning of a section, *mutatis mutandis.*

When computed with FFT's, the computational complexity of the FIR output is dominated by the FFT calculations, which take $O(N \log N)$ time. Thus, for each point, only $O(\log N)$ time is needed, a substantial savings over the $O(N)$ required for convolution computations. Table 3 indicates the time

Table 3: Seconds needed to generate 10 million FIR outputs on a 333MHz Pentium II, for various FIR lengths N.

N	Required time, FFT	Required time, convolution	Speed-up
64	33.0	137	4.2
256	34.4	515	15.0
1024	59.1	1076	18.2
4096	65.7	4253	64.7
16384	102	179502	176

savings in seconds, based on computer simulations.