JPL PUBLICATION 85-68

A Proposed Technique for the Venus Balloon Telemetry and Doppler Frequency Recovery

Raymond F. Jurgens
Dariush Divsalar

April 15, 1985

NASA
National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL PUBLICATION 85-68

A Proposed Technique for the Venus Balloon Telemetry and Doppler Frequency Recovery

Raymond F. Jurgens
Dariush Divsalar

April 15, 1985

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

TECHNICAL REPORT STANDARD TITLE PAG

1. Report No. JPL PUB 85-68	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle A Proposed Technique for the Venus Balloon Telemetry and Doppler Frequency Recovery		5. Report Date April 15, 1985 6. Performing Organization Code
7. Author(s) R.F. Jurgens and D. Divsalar		8. Performing Organization Report No
9. Performing Organization Name and Address JET PROPULSION LABORATORY California Institute of Technology 4800 Oak Grove Drive Pasadena, California 91109		10. Work Unit No. 11. Contract or Grant No. NAS7-918 13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address NATIONAL AERONAUTICS AND SPACE ADMINISTRATION Washington, D.C. 20546		JPL Publication 14. Sponsoring Agency Code BG-314-40-51-01-03

15. Supplementary Notes

16. Abstract

In this report a technique has been proposed to accurately estimate the Doppler frequency and demodulate the digitally encoded telemetry signal that contains the measurements from the balloon instruments. Since the data are prerecorded, one can take advantage of noncausal estimators that are both simpler and more computationally efficient than the usual closed-loop or real-time estimators for signal detection and carrier tracking. Algorithms for carrier frequency estimation, subcarrier demodulation, bit and frame synchronization are described. A Viterbi decoder algorithm using a branch indexing technique has been devised to decode constraint length 6, rate 1/2 convolutional code that is being used by the balloon transmitter. These algorithms are memory efficient and can be implemented on microcomputer systems.

17. Key Words (Selected by Author(s)) Spacecraft Communications, Command, and Tracking Communications Space Sciences (General) Lunar and Planetary Exploration (Advanced)	18. Distribution Statement Unclassified; unlimited

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) unclassified	21. No. of Pages	22. Price

JPL 0184 R 9/8

ABSTRACT

In this report a technique has been proposed to accurately estimate the Doppler frequency and demodulate the digitally encoded telemetry signal that contains the measurements from the balloon instruments. Since the data are prerecorded, one can take advantage of noncausal estimators that are both simpler and more computationally efficient than the usual closed-loop or real-time estimators for signal detection and carrier tracking. Algorithms for carrier frequency estimation, subcarrier demodulation, bit and frame synchronization are described. A Viterbi decoder algorithm using a branch indexing technique has been devised to decode constraint length 6, rate 1/2 convolutional code that is being used by the balloon transmitter. These algorithms are memory efficient and can be implemented on microcomputer systems.

CONTENTS

CONTENTS (continued)

I. INTRODUCTION

The Venus Balloon (Vega) Mission is a joint French-Soviet-and US project to place two balloons on the atmosphere of Venus. The balloons will be tracked by Very Long Baseline Interferometry (VLBI) to determine their positions and velocities on the disc of Venus. Besides this, the gondola contains an instrumentation package that sends telemetry data directly to earth using an L-band transmitter. This mission is described more fully in reference [1].

The scientific data recovery of the Venus Balloon mission requires accurate estimates of the Doppler frequency of the balloon transmission as well as the demodulation of the digitally encoded telemetry signal that contains the measurements from the balloon instruments. Toward this end, the balloon signal has been designed to provide an easy method to determine adequate starting parameters from this weak signal. Specifically, an unmodulated carrier wave is transmitted at the beginning and end of the data transmissions for a period of 30 seconds. A known bit sequence follows the carrier wave such that the starting time can be estimated accurately. To aid in carrier tracking through the frame, 45° modulation is used such that half the transmitted power remains in the carrier wave. The subcarrier wave is separated far enough from the carrier that the information spectrum does not approach the carrier frequency; thus, it is easy to filter the carrier out of the signal for accurate Doppler extraction.

The Doppler shift of the balloon transmission depends upon the balloon's location in the atmosphere and upon the motion of the balloon within the atmosphere. As the balloon drifts across the disc of Venus, the Doppler shift should drift from high to low frequency. The total drift should be constrained within 2 kHz. The width of the modulation spectrum requires an extra 500 Hz. Thus a total spectral width of 2.5 kHz needs to be preserved if there is no a priori knowledge of the initial frequency. As there is some a priori knowledge, probably it is safe to preserve 2 kHz which requires that roughly 4k samples per second be recorded. Eight or twelve bit samples are fully adequate as no appreciable quantizing errors will exist, since the signal to noise ratio in the 2 kHz bandwidth is well below unity. Telemetry transmissions occur at a minimum period of one half hour. The total recorded telemetry data represents less than 10^9 unprocessed bits.

The signal to noise ratio is given by

$$P_S/P_N = \frac{P_T G_{min} A_R}{4\pi D^2} /kTB$$

where

P_T	=	transmitter power in watts
G_{min}	=	minimum transmitting antenna gain
A_R	=	the effective receiving antenna collecting area in sq. meter
D	=	the distance in meters
k	=	Boltzmann's constant in watts/K/Hz
T	=	the effective receiver system temperature in K
B	=	the receiver bandwidth in Hz

where the full 2 kHz bandwidth is used. After filtering to a 1 Hz bandwidth, the signal to noise ratio increases to roughly 63. In this narrow bandwidth, the signal should be clearly visible on a spectrum by spectrum basis. For example, using DSS-14 with an L-band FET amplifier would give $P_S/P_N = 3.16 \times 10^{-12}$ where

P_T	=	5
G_{min}	=	0.5
A_R	=	1.63×10^3
D	=	1.03×10^{11}
k	=	1.38×10^{-23}
T	=	35
B	=	2×10^3

II. INITIAL DETECTION TECHNIQUE

a. Determination of Filter Passband Shape

Prior to the arrival of the actual Balloon signal and periodically while no signal exists, the receiver passband should be measured accurately. Since the passband shape is determined by the offset baseband filter and any aliasing due to the discrete sampling, it is not expected that the shape will vary significantly over any time scale of interest to the detection of a single telemetry frame. An accurate enough passband spectrum can be obtained by averaging 200 signal free power spectra to get roughly 7% statistics. If each spectrum uses 4096 samples, the time to acquire this average noise spectrum is $200 \times \frac{4096}{4000} = 204.8$ s or about 3.4 minutes.

Procedure:

Using the signal free region of the recorded data, perform the Real Fourier Transform (RFT) algorithm. (Details of the theory of the RFT have been supplied as NASA Tech Brief No. NPO-11649.) The real Fourier transform results in place of the data array that contained 4096 points. Next form the power spectrum from these data. The power spectrum will result in $N/2 + 1$ points (2049) as follows: The first point in the transformed array is the a_0 term, the second point is the $a_{N/2}$ term, the remaining $N-2$ points are complex, thus $a_1 = d_3 + j\ d_4$, $a_2 = d_5 + j\ d_6$, $a_k = d_{2k+1} + j\ d_{2k+2}$ ..., $a_{N/2-1} = d_{N-1} + j\ d_N$. $a_0 = d_1 + j\ 0$, $a_{N/2} = d_2 + j\ 0$ where a_k is a complex component of the transformed array and d_k are the result of the RFT program.

To begin, clear the power array P_n which contains $N/2+1$ points. Next accumulate 200 spectra points to this array by forming the power from the array, where

$$P_n(k) \leftarrow P_n(k) + a_k\ a_k^* \quad ; \quad k = 0,\ldots,N/2$$

Save the P_n array to use as a baseline reference to detect the unmodulated carrier. Note that the standard deviation of the noise spectrum will be about 7% of the mean value.

b. Baseline Removal

Spectra measured during the carrier portion should contain a strong line at the carrier frequency. Due to acceleration on the balloon, the carrier frequency could vary by as much as 0.5 Hz/s but rather uniformly for periods as long as the data transmission period. Thus it is not advisable to accumulate the spectra for any appreciable period as smearing would result. The signal power spectra should be determined in the same way as the noise power spectra. Let P_s be a signal spectrum resulting from a single RFT. Then the effects of the passband shape can be removed by division.

$$P_s(k) \leftarrow P_s(k)/P_n(k) \qquad k = 0, \ldots, N/2$$

c. Threshold Detection

The object of this section is to provide a way to automatically search for the spectral line and to determine if a spectral line has been detected in the present data frame. Let k_ℓ and k_m represent the limits of the search where k_ℓ could start at zero and k_m could range to N/2. Let there be M spectra containing pure carrier. In reality a single spectrum is formed from 1.02 s of data, and there are 30 s of pure carrier, so there will be roughly 29 such spectra. Let $L(n)$, $1 \leq n \leq M$ be an array that stores the location of the detected line for each spectrum.

Procedure:

1. Since each spectrum contains noise having statistics of Chi^2, and some point will always be maximum, a threshold must be established that assures that the signal power is significantly larger than the maximum noise power. This can be done only within some confidence level since the noise statistics are not limited in range. Note that the signal is normally divided between two spectral points. Search the normalized power spectrum for the max power. This point is then $P_s(k_m) = P_{max}$. Now form the following statistics for each spectrum:

$$\mu = \frac{1}{N/2} \sum_{k=0\ ,\ k \neq k_m}^{N/2} P_s(k)$$

$$\sigma^2 = \frac{1}{N/2} \sum_{k=0\ ,\ k \neq k_m}^{N/2} P_s^2(k) \ - \mu^2$$

Test $P_{max} > \mu + 9\sigma$

If the above is true, this is a tentative detection. If not true continue to the next spectrum. To be statistically certain of a detection, save the value of k_m and process the next two spectra. If all three yield a value of k_m within a range of two, the detection is reasonably assured. This can be checked by finding the maximum and minimum values of k_m. If the difference between the max and min is 2, 1 or 0, the detection is established. Record the starting sample number associated with the first point of the first spectrum and maintain it as a starting reference point for further processing.

2. Beginning at this time reference, use the above procedure to fill the array L(n) with the successive values of k_m. Process 29 spectra.

d. <u>Carrier Frequency Estimation</u>

The array L(n) contains the trace of the cw line as a function of time. It is expected that this will form a straight line having a small slope. There is some probability that bad estimates of k_m exist. It is likely that these will be widely separated from the main sequence of points. Thus it is necessary to attempt to eliminate the bad points if they exist.

Procedure:

1. Find the mean of L, and mark all points deviating from the mean by more than 5 by replacing the value of L(n) with -1.
2. Find a line of regression by linear least squares that represents L(n). Skip all points containing -1 in this analysis. The result of this analysis is an equation that gives $\hat{L} = a\,x + b$. In the above analysis, the values of x should be the sample number. For example, x_1 should be associated with the center sample number for the first spectrum. That is, x_1 is sample number 2048 counting from the sample reference number. In this way the variation of the instantaneous frequency can be predicted relative to the reference sample number. Thus if i is the sample number and i_{ref} is the reference sample number, the frequency (in frequency number) is given by

$$L = a\,(i - i_{ref}) + b$$

and

$$x = (i_{ref} - 1) + \frac{N}{2} + n\,N$$

where n is the spectrum number and L is a measure of frequency cell number as a function of sample number. It is useful to know both phase and frequency as a function of sample number where real time is referenced to the first sample (i_{ref}). If S_R is the sample rate and N the number of samples in the RFT, then

Let $$w(i) = 2\pi \frac{S_R}{N} [(i - i_{ref}) a + b]$$

$$t = (i - i_{ref})/S_R$$

Then $$w(t) = 2\pi \frac{S_R}{N} [a\, S_R\, t + b]$$

$$\phi(t) = 2\pi \frac{S_R}{N} [a\, S_R\, t^2/2 + bt]$$

Thus $$w = \frac{2\pi S_R\, b}{N} \text{ and } \dot{w} = \frac{2\pi S_R^2\, a}{N}$$

3. Repeat the above procedure for the end-of-frame cw spectra. This forms a second equation of frequency versus sample number. Let L_A be the first and L_B the second, then some measure of frequency stability over the data frame can be determined by evaluating the difference of L_A and L_B at a point near the center of the frame. If this difference is too large, it may be difficult to track the carrier through the frame.

III. DOPPLER RECOVERY AND THE EQUATION OF TIME

a. Filtering and Data Reduction

All balloon motion relative to the center of Venus is unmodeled in the frequency drifting of the first local oscillator. Thus it is important to know the Doppler shift accurately to know the radial velocity and acceleration of the balloon as well as to be able to demodulate the carrier wave. L_A and L_B give a first order prediction of that Doppler, but furtner prediction may be carried out by filtering out the sidebands and tracking the carrier through the frame. The frame consists of 330 seconds of data, and there are 1.32×10^6 samples through the frame. Since this is far too many to store in the computer memory, it is important to perform some data reduction. We are interested in frequency departures relative to our initial frequency line of regression. Normally, departures of more than a few Hz from this line are unexpected (note each frequency bin has a width of 4000/4096 = 0.9765 Hz). To get some bound upon this, determine the maximum difference between $|L_A - L_B|$ over the whole time span. The width of the filter can be set at twice this value. Call this bandwidth B. Note that the Nyquist sampling requires only 2B samples per second. Since B will be only a few Hz, only four or five samples per second are required to specify the carrier signal. Thus, roughly two thousand memory locations are needed to store the information for the entire frame. This data reduction is achieved by heterodyning the input signal (samples) with a reference carrier based on L_A and filtering. The filter can be a simple running average of 1/2B seconds of data; however, it is convenient to form complex samples at a rate of 1/B using a complex mixer. Figure 1 below shows the block diagram for a complex mixer.

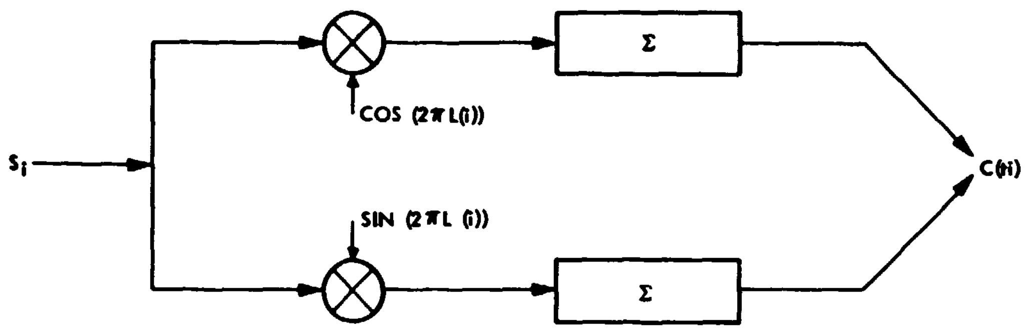


Figure 1. Baseband Filter for Carrier Separation

Procedure:

1. Determine B as discussed above.

2. Let T = 1/B. The number of samples to average in the filter is then 4000 T = M.

3. Beginning at the reference sample form the products

$$S_i \, \text{Cos}[2\pi(wt + \frac{\dot{w}}{2} t^2)] \qquad t \text{ is time}$$

and

$$S_i \, \text{Sin}[2\pi(wt + \frac{\dot{w}}{2} t^2)]$$

Then produce the average of M such samples such that

$$C_k = \sum_{i=(k-1)\frac{M}{2} + i_{ref}}^{(k+1)\frac{M}{2} + i_{ref}} S_i \, \text{Cos} \, [2\pi(wt + \frac{\dot{w}}{2} t^2)]$$

$$- j \sum S_i \, \text{Sin} \, [2\pi(wt + \frac{\dot{w}}{2} t^2)]$$

4. Save C_k for further analysis.

b. Estimation of Doppler Model

In this section, we attempt to construct an accurate model of the carrier wave to be used as a phase coherent reference for demodulation of the subcarrier signal. This is essential to prevent the true carrier and the estimated local carrier from forming a beat that modulates the subcarrier amplitude. The phase coherent local oscillator is constructed from the carrier samples C_k. The extent to which the carrier model M_k can represent the data depends upon many factors such as the signal to noise ratio, the phase stability of the transmitter, the motion of the balloon, the atmospheric turbulence and many other stochastic errors and unmodeled effects. Thus it is not possible to know a priori the level to which coherence can be established. For that reason, it is necessary to know when the carrier tracking loop is locked as tightly as possible. A simple criterion is to establish whether the estimated carrier signal has fallen significantly below the estimated incoherent power level. Thus we attempt to lock the carrier model to the carrier samples as tightly as possible by narrowing the effective bandwidth until the signal level begins to drop. The effective bandwidth is controlled by the duration of the data span used in the processing. Increasing the span to 30s should be possible most of the time unless the balloon is undergoing severe turbulence or accelerations.

Procedure:

Nonlinear Least Squares Regression

Let the carrier signal model be given by

$$M_k = A\, e^{j(\phi + wt_k - \dot{w}/2t_k^2)}$$

where

A is an amplitude coefficient

ϕ is an initial phase reference

t_k is the time of the kth sample and k is centered in the span of data

w is the frequency

$\dot{w}$ is the rate of change of frequency

Let the error between the measured samples and the model be given by

$$\varepsilon = \sum_{k=-N}^{N} W_k \, |C_k - M_k|^2$$

where W_k is a set of weights that are normally set equal to the reciprocal of the noise variance, and 2N+1 is the number of samples in 30 seconds. In this case the weights can be set equal to unity unless the true covariance matrix is needed. We desire to minimize ε by adjusting A, ϕ, w, and $\dot{w}$. Let x represent the vector A, ϕ, w, $\dot{w}$ and Δx represent a vector of small corrections to x.

$$\Delta x = [\Delta A, \ \phi, \ \Delta w \ \Delta \dot{w}]$$

Then

$$\frac{\partial M_k}{\partial A} = \frac{\partial M_k}{\partial x_1} = e^{j(\phi + wt_k + \dot{w}/2t_k^2)}$$

$$\frac{\partial M_k}{\partial \phi} = \frac{\partial M_k}{\partial x_2} = j \ A \ e^{j(\phi + wt_k + \dot{w}/2t_k^2)}$$

$$\frac{\partial M_k}{\partial w} = \frac{\partial M_k}{\partial x_3} = j \ A \ t_k \ e^{j(\phi + wt_k + \dot{w}/2t_k^2)}$$

$$\frac{\partial M_k}{\partial \dot{w}} = \frac{\partial M_k}{\partial x_4} = j \ A \ t_k^2/2e^{j(\phi + wt_k + \dot{w}/2t_k^2)}$$

A minimum in ε requires that

$$\frac{\partial \varepsilon}{\partial x_i} = 0 \qquad i = 1,\ldots,4$$

Thus

$$-\sum_{k=-N}^{N} 2\, R_e\left\{W_k(C_k - M_k)\,\frac{\partial M_k^*}{\partial x_i}\right\} = 0 \quad i = 1,\ldots,4$$

M_k may be expanded about the initial value of x. If we substitute only the linear terms of the series for M_k, we arrive at the normal equations. Let

$$M_k \cong M_{ok} + \sum_{j=1}^{4} \left.\frac{\partial M_k}{\partial x_j}\right|_{x_j = x_{oj}} \Delta x_j$$

Then

$$\sum_{k=-N}^{N} 2\, R_e\left\{W_k(C_k - M_{ok})\,\frac{\partial M_k^*}{\partial x_i}\right\} = \sum_{j=1}^{4}\sum_{k=-N}^{N} 2\, R_e\left\{W_k\,\frac{\partial M_k}{\partial x_j}\,\frac{\partial M_k^*}{\partial x_i}\right\}\Delta x_j$$

evaluate $\frac{\partial M_k}{\partial x_j}$ at $x = x_o$

The normal equation forms four equations with four unknowns where Δx_j are the unknowns. This solution must be iterated until the Δx_j are suitably small. Thus the next estimate of $x_o = x_o + \Delta x$. The meaning of suitably small is either defined from the limits of the measurements as defined by the covariance matrix (the inverse of the normal equation matrix) or by what is reasonable. Reasonable in this case is determined by theoretical limits or by how much error we can stand. In general a phase error of 10° is about as large as acceptable. This constrains $\Delta\phi < 0.17$, $\Delta w < 0.01$ and $\Delta\dot{w} < 0.0015$ for a 30s span. In general, the iteration should be driven to about a tenth of these values.

Accurate starting estimates are usually required to begin the nonlinear least squares regression analysis. The prediction equation given by L_A should normally determine the frequency to an accuracy slightly better than 0.5 Hz. In such a case the maximum integrated phase error approaches 360° in two seconds. The least squares procedure requires that the phase error be less than 180° and preferably less than 90° over the data span if convergence is to be assured. Assuring the phase error is small enough either requires that the preliminary estimates of w and $\dot{w}$ are adequate for the time span considered or that the time span is short enough. If the time span is too small, the signal to noise ratio will be too small for adequate detection. Since most of the $\dot{w}$ term has been heterodyned out by the L_A model, it normally can be initialized to zero. Determination of w can be done in several ways, each of which has merit under certain conditions. We suggest the following possibilities:

1. Under normal conditions, the NLLSR can be used with a short span of data (about 1s). Once convergence is established, the span can be doubled using the new estimate of w. This can be continued until the 30s span either converges or the estimate of A begins to drop, indicating inadequacy in the model or phase incoherence of the signal.
2. A sequence of 1s estimates can be made over the 30s data span using NLLSR. The resulting $\hat{w}$ then can be used to form a linear equation of frequency giving w and $\dot{w}$ as starting parameters. This procedure has been tested and works well if the signal to noise ratio is adequate.

3. This last procedure may be useful if convergence cannot be obtained using 1 and 2 above. Use the complex fast Fourier transform (FFT) to form the power spectrum of the C_k's. The span of data could extend to 30s giving 1/30 Hz resolution. A peak detection scheme similar to that of Section IIc can be used to find the maximum power and its location. This procedure will not work if $\dot{w}$ is too large, but does work for smaller signal to noise ratios than procedure 2.

A first estimate of $Ae^{j\phi}$ can be formed by simple correlation of the model with the data C_k.

$$A_o = \frac{1}{2N+1} \sum_{k=-N}^{N} C_k \, e^{-j(wt_k + \frac{\dot{w}}{2} t_k^2)}$$

Thus

$\hat{A} = |A_o|$

$\hat{\phi} = \text{Arctan } (\text{Im } \hat{A}/\text{Re } \hat{A})$

As we progress through the data, it is best to determine the starting values from the results of the previous data. However, A and ϕ can always be computed quickly from the procedure above.

Experimentation with some carrier models has shown that some procedure for detecting nonconvergence is necessary. A suitable estimate is to assure that $\hat{A}^2$ does not fall significantly below the estimated carrier power level. An estimate of the incoherent power is given by $\hat{P}$ where

$$\hat{P} = \frac{1}{2N+1} \sum_{k=-N}^{N} |C_k|^2 - \hat{N}$$

$\hat{N}$ is an estimate of the noise power which can be formed from the original signal statistics by noting that the power in the 2 kHz band is mostly noise. Thus the value of $\hat{N}$ can be related simply to the statistics of S_i by noting that M samples of the original signal were added together and the gain of the complex mixer is 1/2 per side.

$$\hat{N} = M\,\sigma_s^2 = M\,\frac{1}{2K+1}\sum_{i=-K}^{K} S_i^2$$

Here σ_s^2 is the variance of the samples, S_i in the data span currently being analyzed. This assumes S_i are zero mean and nearly white.

The results of the above analysis provide a series of equations for estimating $\phi(t)$ that are valid over a 30 second strip of data. The 30 second spans are formed every 15 seconds. These data are intrinsically valuable as scientific output and are used for the carrier demodulation discussed in the next section. Clearly, the previous operations provide a great data reduction in specifying the phase function since each 30 second strip of data is represented by four numbers and there are only 22 such strips over the data frame.

The above procedure provides no guarantee that the phase is continuous across the thirty second boundaries. However, since the equations are valid over a 15 second period overlapping consecutive strips, it is possible to force the phase to be continuous by using a weighted combination of two adjacent equations to form the phase function. Figure 2 below shows the form of the weights.

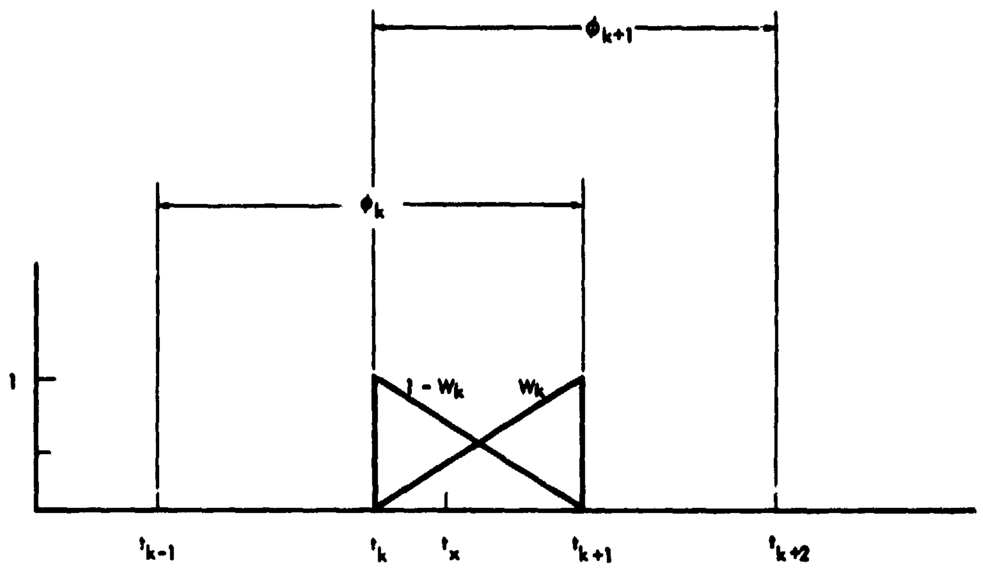


Figure 2. Linear Weights for Interpolation of Phase

The interpolation is formed as shown in Figure 2 where

$$W_k(t_x) = \frac{t_x - t_k}{\Delta t} \; ; \quad \Delta t = t_{k+1} - t_k$$

and

$$\hat{\phi}(t_x) = (1-W_k(t_x))\phi_k(t_x) + W_k(t_x)\phi_{k+1}(t_x)$$

where

$$\phi_k(t_x) = w_k t_x + \frac{\dot{w}_k}{2} t_x^{\,2}$$

and $w_k, \dot{w}_k$ are from least squares regression. Thus the continuity of phase across the boundaries is assured.

IV. CARRIER DEMODULATION

The carrier demodulation is easily carried out using the phase function of the previous section. If we presume that the binary (square wave modulation) is B(t), then the modulated signal is of the form

$$S(t) = S\, e^{j[\pi/4\, B(t) + \theta(t)]}$$

where $B(t) = \underline{+}1$ and S is a real amplitude weight. $\theta(t)$ is presumed to be composed of drifts in the transmitter and Doppler shift due to balloon motion. $\hat{\theta}(t)$ estimates both phase components, thus the recovered modulation r(t) is given by

$$r(t) = S(t)\, e^{-j\,\hat{\theta}(t)}$$

$$\hat{\theta}(t) = \hat{\phi}(t) + w_A t + \frac{\dot{w}_A}{2} t^2$$

*Note 1

When $\theta_d - \hat{\theta}_d$ is small as it should be,

$$r(t) \cong S\left[\frac{\sqrt{2}}{2} + j \sin \frac{\pi}{4} B(t)\right]$$

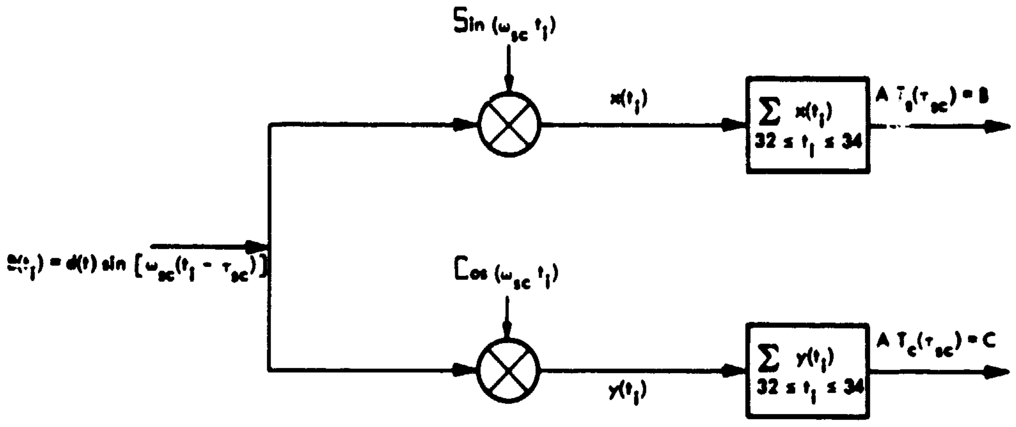
Note that verification of proper phase demodulation can be checked easily by simply averaging r(t). Since B(t) is zero mean, only the real part of r(t) should remain finite. Note that the modulation is contained in the imaginary part of r(t) and has an amplitude of $\frac{\sqrt{2}}{2}$ S.

*Note 1

S(t) is the original sample set beginning 12 seconds after initial detection, thus this is 48000 samples past i_{ref}. $\theta(t)$ is then defined at the total phase at that time. $\hat{\theta}(t)$ must be composed of the original phase due to L_A and the departures from L_A given by $\hat{\phi}(t)$ where w_A and $\dot{w}_A$ are from L_A (See page 7).

V. SUBCARRIER DEMODULATION

First note that between times 31-3/8 sec and 34-1/4 sec we have an all zero sequence. Therefore there are no transitions in symbols. (Look at the output symbol sequence of convolutional code for frame sync inputs.) Therefore we can demodulate the subcarrier and estimate the phase of subcarrier as shown in Fig. 3. In Figure 4 the output samples of the integrators in subcarrier demodulator are shown as a function of τ_{sc}.



$s(t_i) = d(t)\sin[\omega_{sc}(t_i - \tau_{sc})]$
$\sin(\omega_{sc} t_i)$
$x(t_i)$
$\sum x(t_i)$, $32 \le t_i \le 34$
$A\,T_s(\tau_{sc}) = B$
$\cos(\omega_{sc} t_i)$
$y(t_i)$
$\sum y(t_i)$, $32 \le t_i \le 34$
$A\,T_c(\tau_{sc}) = C$

Fig. 3 Subcarrier Demodulator

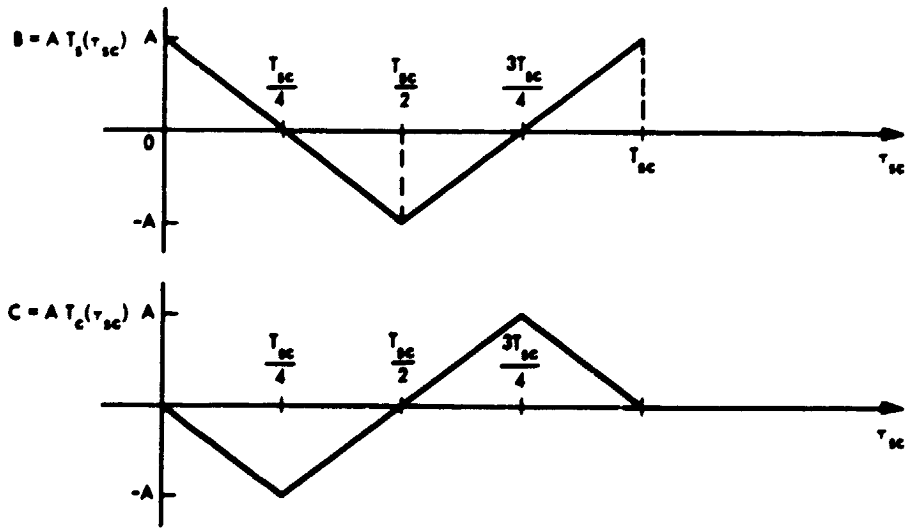


Fig. 4 Output Samples of Integrators in Subcarrier Demodulator vs. τ_{sc}

IF $B \geq 0$ and $C \leq 0$ $\qquad \tau_{SC} = \dfrac{T_{SC} \times C}{4(C-B)}$

IF $B \leq 0$ and $C \leq 0$ $\qquad \tau_{SC} = \dfrac{T_{SC}(C + 2B)}{4(C + B)}$

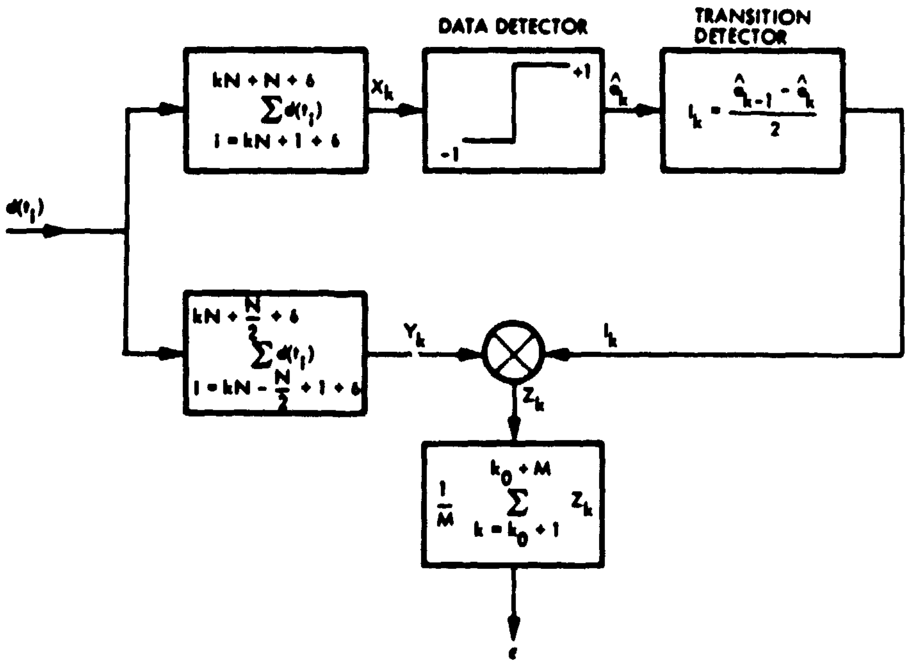
IF $B \leq 0$ and $C \geq 0$ $\qquad \tau_{SC} = \dfrac{T_{SC}(3C - 2B)}{4(C - B)}$

IF $B \geq 0$ and $C \geq 0$ $\qquad \tau_{SC} = \dfrac{T_{SC}\ (3C + 4B)}{4(C + B)}$

after finding τ_{SC} we can shift the whole subcarrier signal by τ_{SC} (delay by τ_{SC}). Then we can demodulate the whole data frame by the subcarrier to get data samples $d(t_i)$.

VI. BIT SYNCHRONIZATION

The bit synchronizer is shown in Fig. 5.



DATA DETECTOR
TRANSITION DETECTOR
$\sum_{i=kN+1+\delta}^{kN+N+\delta} d(t_i)$
X_k
+1
-1
$\hat{a}_k$
$I_k = \frac{\hat{a}_{k-1} - \hat{a}_k}{2}$
$d(t_i)$
$\sum_{i=kN-\frac{N}{2}+1+\delta}^{kN+\frac{N}{2}+\delta} d(t_i)$
Y_k
I_k
Z_k
$\frac{1}{M} \sum_{k=k_0+1}^{k_0+M} Z_k$
ϵ

Fig. 5 Bit Synchronizer

Let $d(t_i)$ be data samples. Suppose during one symbol time there are N samples $d(t_i)$. In upper branch of Fig. 5, we sum the N samples $d(t_i)$ starting at time $kN + 1 + \delta$ $(-\frac{N}{2} \leq \delta \leq \frac{N}{2})$. Next we detect the sign of symbol by passing X_k through a sign $(\cdot)$ function

$$\operatorname{sign}(x) = \begin{cases} 1 \; ; \; x > 0 \\ -1 \; ; \; x < 0 \end{cases}$$

Next we detect the transition in consecutive symbols $\hat{a}_k$ by finding one-half of difference between consecutive detected symbol signs. If $I_k = 0$ means there is no transitions, $I_k = \pm 1$ means there is a transition. If $I_k = +1$ then we have symbol transition from + 1 to - 1 and if $I_k = -1$, then we have symbol transition from - 1 to + 1.

In lower branch in Fig. 5 we sum N samples $d(t_i)$ starting at time $kN - \frac{N}{2} + 1 + \delta$. The result is sample Y_k. We multiply Y_k by I_k to get Z_k. We repeat this operation M times ($M \approx 60$), starting at $k = k_0$ where time k_0N is roughly at the beginning of (30 - 42 sec) interval. Next we find the result of the average of all M samples z_k as ε. Now if ε is positive we increase δ; if ε is negative we decrease δ. We repeat the whole operation until we get smallest ε possible.

VII. FRAME SYNCHRONIZATION

We get the smallest possible ε from bit synch. We store all detected symbols $\hat{a}_k$ in a shift register. Next we compare the pattern of symbols that we have at the output of encoder with contents of shift register. We shift this pattern until we match the pattern with the content of shift register with minimum number of discrepancies. Minimizing θ in Fig. 6 can result in frame sync, where $\hat{b}_k \in \{0,1\}$ is related to $\hat{a}_k$ by

$$\hat{b}_k = \frac{1 - \hat{a}_k}{2}$$

and $\boxed{+}$ is exclusive OR.

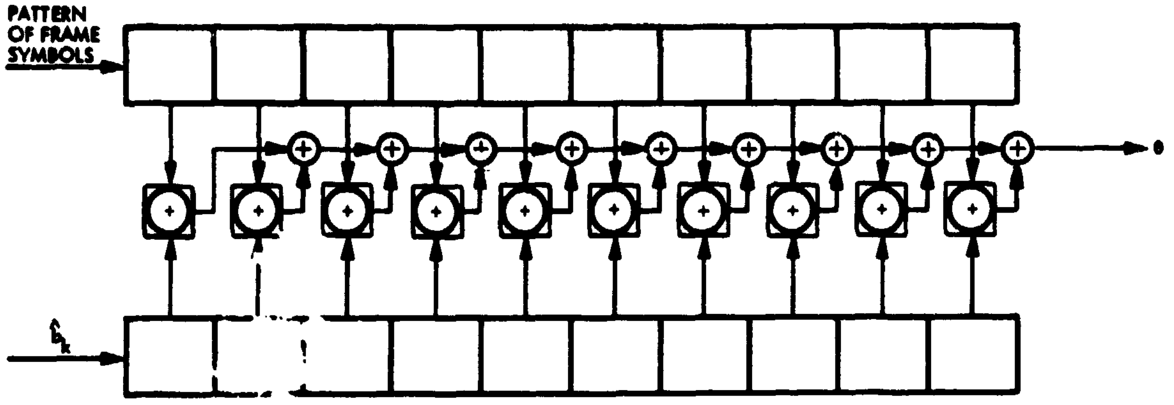


Fig. 6 Frame Synchronizer

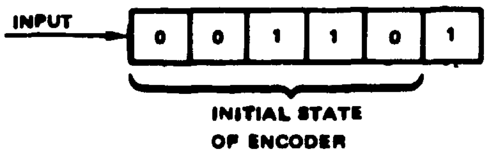
The input frame synchronization bits to the convolutional code during the time interval (30-42) seconds are

17(0), 5(1), 3(0), 2(1), 1(0), 3(1), 1(0), 1(1), 1(0), 1(1), 4(0), 1(1), 2(0), 1(1), 1(0), 2(1), 2(0).

The corresponding output pattern of frame synchronization symbols of convolutional code is

10(X), 24(0), 2(1), 2(0), 1(1), 1(0), 1(1), 5(0), 2(1), 1(0), 2(1), 1(0) 2(1), 1(0), 1(1), 1(0), 2(1), 1(0), 3(1), 1(0), 3(1), 7(0), 3(1), 3(0), 3(1), 1(0), 2(1), 1(0), 2(1), 3(0), 2(1), 1(0), 1(1).

Where the number in front of parentheses indicates the number of consecutive bits shown in parentheses, the X means unknown bit. The contents of encoder at time 42 is:



VIII. VITERBI DECODER

This section explains the Viterbi decoder algorithm for a constraint length $K = 6$, code rate $r = 1/2$ convolutional code. The code is shown to be a transparent convolutional code.[†] The weight distribution of the code has been computed by numerical analysis. The code state table, received sample quantization, metric quantization, and Viterbi algorithm using a branch indexing technique is explained in detail. The theoretical bit error performance of the code and the simulation results of this Viterbi algorithm is given.

a. Code Structure

Cons 'er a $K = 6$, $r = 1/2$ convolutional encoder which is illustrated in Figure 7.

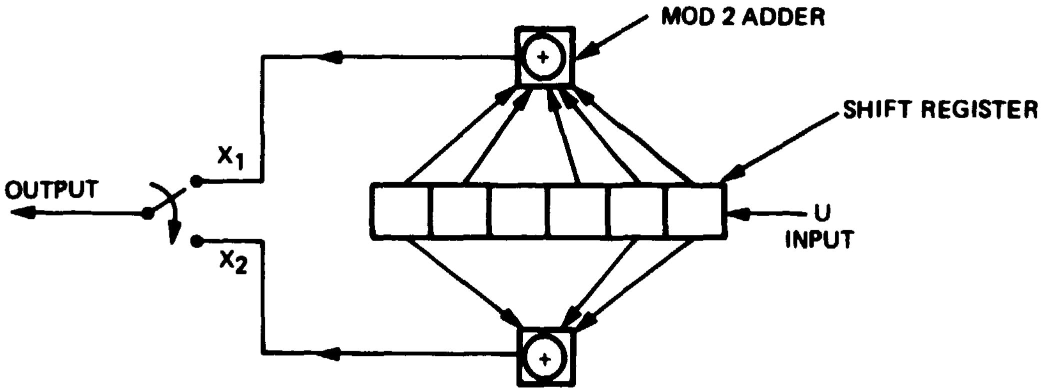


Figure 7. Code Structure

[†]The code has been provided by mission as in Fig. 7 without any further information. The description of code was not available in the literature.

This code is a transparent code, i.e., the 1's complement of the input bit sequence results in the 1's complement of the output bits. To show this, note that the output code symbols can be represented in terms of the input bits as

$$X_{1k} = U_k \oplus U_{k-1} \oplus U_{k-2} \oplus U_{k-4} \oplus U_{k-5}$$

$$X_{2k} = U_k \oplus U_{k-1} \oplus U_{k-5}$$

where $\oplus$ represents the "Exclusive OR" operation. Let U_j' denote the 1's complenent of U_j. Now if we replace U_j by U_j' and X_{ij} by X_{ij}' ; i=1,2 ; j = k, k-1,....k-5 in the above code input-output relations and if the equalities hold, then the code is transparent. This can be shown by using the following relations

$$U_j' = U_j \oplus 1 \qquad j = k, k-1, ..k-5$$

$$X_{ij}' = X_{ij} \oplus 1 \qquad i = 1,2$$

in the above code input-output relations. When the code is transparent we can use outer differential encoding and decoding for inner channel coding tJ resolve the 180° phase ambiguity.

The code weight distribution has been found by first finding the transfer function bound [2] for the code and then using a technique in [3] to get the weight distribution. The minimum distance of code is 8. The code weight distribution is shown in Table 1.

This code differs from the K = 6, r = 1/2 Odenwalder code [4], since it is transparent and the structure is different. The performance is slightly inferior to the Odenwalder code.

Table 1. Code Weight Distribution

Weight d	Number of Adversaries	Number of Bit Errors in the Adversaries a_d
8	3	6
9	0	0
10	13	60
11	0	0
12	~80	469
13	0	0
14	-	3340

b. Code State

For each input bit to the encoder we have two output bits as shown in Figure 7. The number of states NS is,

$$NS = 2^{K-1} = 32$$

The rightmost five bits of the shift register constitute the state of the code. It is clear that the input bit will be the least significant bit for the next state.

c. Quantization

We may quantize received samples into B bits. Thus we need $L = 2^B$ levels of quantization. For the binary code symbols $x \in \{0,1\}$, with the following conversion,

$$0 \rightarrow 1$$

$$1 \rightarrow -1$$

we can assign integers 0 through L-1 to the quantization levels, as shown in Figure 8.

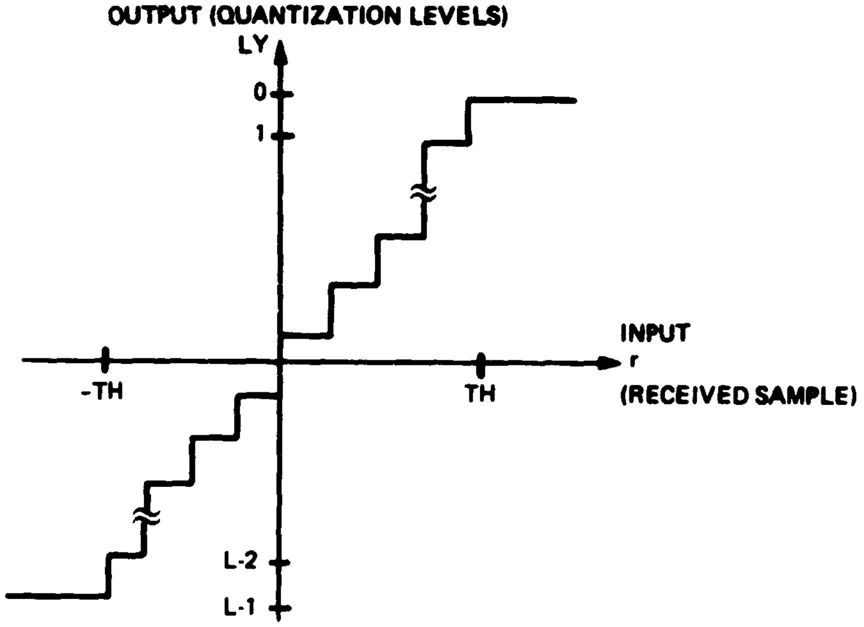


Figure 8. Uniform L-Level Quantizer Where TH is Upper Threshold of Quantization

Table 2 shows some typical values for TH assuming the noise samples are normalized to have a unit variance.

Table 2. Typical Values for TH

L	$TH = (\frac{L}{2} - 1)\Delta^*$
2	0
4	1
8	1.7
16	2.3
32	2.7
64	3.1
128	3.5
256	3.8

* Δ is threshold spacing.

d. Metric Quantization

Figure 3 shows integer code symbol metrics for L-level quantization.

		QUANTIZATION LEVELS					
	x \ LY	0	1	2	....	L-2	L-1
CODE SYMBOL	0	L-1	L-2	L-3	....	1	0
	1	0	1	2	....	L-2	L-1

Figure 9. Integer Code Symbol Metrics for L-Level Quantization

The branch metrics can be computed from Table 3.

Table 3. Branch Metrics Computation for L-Level Quantization

Branch Index IB	Branch Code Symbols x_1	x_2	Branch Metric LB
1	0	0	$2(L-1) - LY_1 - LY_2$
2	0	1	$L-1 - LY_1 + LY_2$
3	1	0	$L-1 + LY_1 - LY_2$
4	1	1	$LY_1 + LY_2$

The IB's are the branch indexes associated with the various distinct branch symbols. For example for a branch with code symbols 10, IB = 3.

e. Unquantized Metric

If we do not use quantization for the received samples, then the code symbol metrics can be computed from Figure 10, and the branch metrics are computed from Table 4.

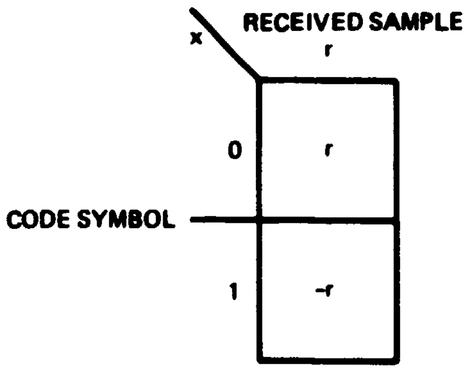


Figure 10. Code Symbol Metrics for Unquantized Samples

Table 4. Branch Metrics Computation for Unquantized Samples

Branch Index IB	Branch Code Symbols x_1	x_2	Branch Metric LB
1	0	0	$r_1 + r_2$
2	0	1	$r_1 - r_2$
3	1	0	$-r_1 + r_2$
4	1	1	$-r_1 - r_2$

f. Viterbi Algorithm

The Viterbi algorithm is essentially a maximum likelihood decoding procedure for convolutional codes. The Viterbi algorithm recursively determines a path sequence with largest metric (most likely data sequence) through the trellis of the code from the initial state to the final state. The code trellis is basically an extended version of the code state diagram in time. The algorithm at any time retains the largest metric path to each state (node) and eliminates other smaller metric paths to the state from further consideration. The retained data sequence with the largest metric to each state is called the survived data sequence and the corresponding metric is called the survived metric. To implement the procedure, we must store the survived metric and the survived data sequence for every state at any given time k. Then at time k + 1, after computing the branch metrics, we can use the existing information which has been stored at time k, to compare and select the best metrics and the path sequences for each state and store them as the new survived metrics and the survived data sequences. We can proceed with this recursion, starting at time $k = 0$ at some initial state, and ending the process at final time $k = N$.

Finally we choose the data sequence with largest metric among all states at time $k = N$ as the decoded data sequence.

In practice since N is usually large, the amount of storage needed to retain the survived data sequences is large. Therefore if N is very large, it is necessary to truncate survived sequences to some length m. However it can be shown that there is a high probability that all survived data sequences at time k will have identical data bits very far back from the present time k. This suggests that if the algorithm stores enough of the past data bits of each of the 32 survived sequences, then the oldest bits on all stored data sequences will be identical. Our simulation has shown that we only need to store $m = 36$ most recent bits of the survived data sequences; in this case the effect on performance is negligible. Then the algorithm outputs the oldest bit in the survived data sequence at state number 1, as the hard decoded bit. Another practical consideration is metric overflows. Since the survived metrics grow in time, to prevent a possible overflow, it is necessary to renormalize the survived metrics from time to time. As follows we explain the algorithm in detail.

1. Definitions:

$M_k(J)$	:	Survivor metric at state number J at time k.
$\underline{M}_k$	:	Array containing the survived metrics at time k, i.e., $\underline{M}_k=(M_k(1),M_k(2),\ldots, M_k(32))$
LB(IB)	:	Branch metric corresponding to the branch code symbols x_1 and x_2, indexed by IB
$\underline{LB}$	:	Array of branch metrics, i.e., $\underline{LB} =$ (LB(1),LB(2),LB(3),LB(4))
$S_k(J)$	:	State number J; J = 1, 2,...., 32, at time k

$\hat{\underline{u}}_{k,m}(J)$	:	Array of the m most recent bits of the survived data sequence, terminating at state J at time k
MO_k	:	36 x 32 matrix that stores all arrays $\hat{\underline{u}}_{k,36}(J)$; J = 1,2, ..., 32 at time k
MM_{k+1}	:	36 x 32 matrix that stores all arrays $\hat{\underline{u}}_{k+1,36}(J)$; J = 1,2,...., 32 at time k+1
$\underline{L0}$	:	Array of all branch indexes from state number J to state number 2J - 1, for J = 1, 2,...., 16
$\underline{L1}$	:	Array of all branch indexes from state number 16 + J to state number 2J - 1, for J = 1, 2,...., 16
$\underline{L0}'$	:	Array of all branch indexes from state number 16 + J to state number 2J, for J = 1, 2,...., 16
$\underline{L1}'$	:	Array of all branch indexes from state number J to state number 2J, for J = 1, 2,...., 16

Note that since the first and the last stages of the shift register are connected to the MOD 2 adders, then

$$\underline{L0}' = \underline{L0} \quad \text{and} \quad \underline{L1}' = \underline{L1}$$

2. State Transitions

For a given J; $1 \leq J \leq 16$, the state transitions are shown in Figure 11, where IB $\neq$ IB'. All state transitions associated with the branch indexes are shown in Table 5. Note that when we use branch indexing we need only to compute 4 metrics each time rather than computing all metric branches. Therefore branch metric computation is almost independent of number of states.

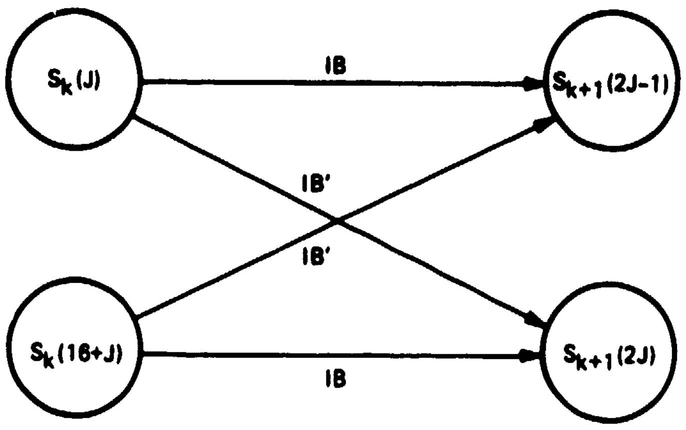


Figure 11. State Transitions Between Pair of States

Table 5. State Transitions With Associated Branch Indexes

$S_k(J)$	$S_{k+1}(2J-1)$	IB
1	1	1
2	3	4
3	5	3
4	7	2
5	9	1
6	11	4
7	13	3
8	15	2
9	17	3
10	19	2
11	21	1
12	23	4
13	25	3
14	27	2
15	29	1
16	31	4

Table 5. State Transitions With Associated Branch Indexes (Continued)

$S_k(16+J)$	$S_{k+1}(2J-1)$	IB
17	1	4
18	3	1
19	5	2
20	7	3
21	9	4
22	11	1
23	13	2
24	15	3
25	17	2
26	19	3
27	21	4
28	23	1
29	25	2
30	27	3
31	29	4
32	31	1

Table 5. State Transitions With Associated Branch Indexes (Continued)

$S_k(J)$	$S_{k+1}(2J)$	IB
1	2	4
2	4	1
3	6	2
4	8	3
5	10	4
6	12	1
7	14	2
8	16	3
9	18	2
10	20	3
11	22	4
12	24	1
13	26	2
14	28	3
15	30	4
16	32	1

Table 5. State Transitions With Associated Branch Indexes (Continued)

$S_k(16+J)$	$S_{k+1}(2J)$	IB
17	2	1
18	4	4
19	6	3
20	8	2
21	10	1
22	12	4
23	14	3
24	16	2
25	18	3
26	20	2
27	22	1
28	24	4
29	26	3
30	28	2
31	30	1
32	32	4

3. Algorithm

Step 1. Storage:

Store Branch index arrays,

LO = (1, 4, 3, 2, 1, 4, 3, 2, 3, 2, 1, 4, 3, 2, 1, 4)

L1 = (4, 1, 2, 3, 4, 1, 2, 3, 2, 3, 4, 1, 2, 3, 4, 1)

Store the survived data sequences in buffers,

$MO_k = (\hat{u}_{k,36}(1), \hat{u}_{k,36}(2), \ldots, \hat{u}_{k,36}(32))$

$MN_{k+1} = (\hat{u}_{k+1,36}(1), \hat{u}_{k+1,36}(20, \ldots, \hat{u}_{k+1,36}(32))$

Store branch metric array,

LB = (LB(1), LB(2), LB(3), LB(4))

Step 2. Initialization:

Buffers,

$MO = (\underline{0}, \underline{0}, \ldots, \underline{0})$

$MN(1, 2J-1) = 0$

$MN(1, 2J) = 1$; $J = 1, 2, \ldots, 16$

Metrics,

$M_0(13) = 0$

$M_0(I) = -2000$; $I = 1, 2, 3, \ldots, 32, I \neq 13$

Stating time, k = 0

Step 3 Recursion:

$k = k + 1$

branch metric computation,

compute branch metrics from Table 4 and store in LB

$I = 0$

Step 4 $I = I + 1$

Survived metrics,

$$M_k\ (2I-1) = \text{Max}\left\{[M_{k-1}(I) + LB(LO(I))],\ [M_{k-1}(16+I) + LB(L1(I))]\right\}$$

$$M_k\ (2I) = \text{Max}\left\{[M_{k-1}(I) + LB(L1(I))],\ [M_{k-1}(16+I) + LB(LO(I))]\right\}$$

Select the survived sequences from MO_{k-1}, add selected bit "0" or "1" to it, store in scratch buffer MN_k ,

```
If $M_k(2I-1) = M_{k-1}(I) + LB(LO(I))$
    then $MN_k\ (J+1, 2I-1) = MO_{k-1}(J, I)$ ; $J = 1, 2, ..., 35$
If $M_k(2I-1) = M_{k-1}(16+I) + LB\ (LO(I))$
    then $MN_k\ (J+1, 2I-1) = MO_{k-1}\ (J, 16+I)$ ; $J = 1, 2, ..., 35$
If $M_k(2I) = M_{k-1}(I) + LB(L1(I))$
    then $MN_k\ (J+1, 2I) = MO_{k-1}(J, I)$ ; $J = 1, 2, ..., 35$
If $M_k(2I) = M_{k-1}(I) + LB(LO(I))$
    then $MN_k\ (J+1, 2I) = MO_{k-1}(J, 16+I)$ ; $J = 1, 2, ..., 35$
If $I < 16$ go to step 4
    restore $MN_k$ into $MO_k$
        $MO_k(I,J) = MN_k(I, J)$ ; $I = 1, 2, ..., 36$
                                   $J = 1, 2, ..., 32$
```

The decoded bit at time k is

$$MO_k(36, 1)$$

Normalization of survived metrics,

Find the largest $M_k(J)$; $J = 1, 2, \ldots, 32$ and call it MXM, then

$$M_k(J) = M_k(J) - MXM \quad ; J = 1, 2, \ldots, 32$$

call node synchronization algorithm and store the result in NOD.

If $K < N$ go to step 3

Otherwise, choose the largest metric among all the survived metrics at $k = N$. The corresponding data sequence in the buffer MO is considered as the final decoded sequence.

g. Bit Error Rate Performance

From our analysis using Table 1 for weight distribution of code and then using union bound we can approximate [3] the bit error rate P_b as*

$$P_b \approx 3 \text{ erfc} \left(\sqrt{4 \frac{E_b}{N_0}} \right) + 30 \text{ erfc} \left(\sqrt{5 \frac{E_b}{N_0}} \right) + 250 \text{ erfc} \left(\sqrt{6 \frac{E_b}{N_0}} \right) + 1670 \text{ erfc} \left(\sqrt{7 \frac{E_b}{N_0}} \right)$$

where

$\frac{E_b}{N_0}$ is bit signal to noise ratio and

$$\text{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-t^2} \, dt$$

Figure 12 gives the theoretical performance of this algorithm and the simulation results. The .25 dB loss in simulation with respect to theoretical result is due to infinite bit quantization that we assumed in theoretical result, and it is due to path memory truncation and metric quantization in the software algorithm.

*Theoretical upper bound is [2]

$$P_b \leq 1/2 \text{ erfc} \left(\sqrt{4 \frac{E_b}{N_0}} \right) e^{+\frac{4E_b}{N_0}} \sum_{d=8}^{\infty} a_d \, e^{-\frac{dE_b}{2N_0}}$$

where a_d are coefficients in tr'nsfer function bound [2]. The first few are given in Table 1.

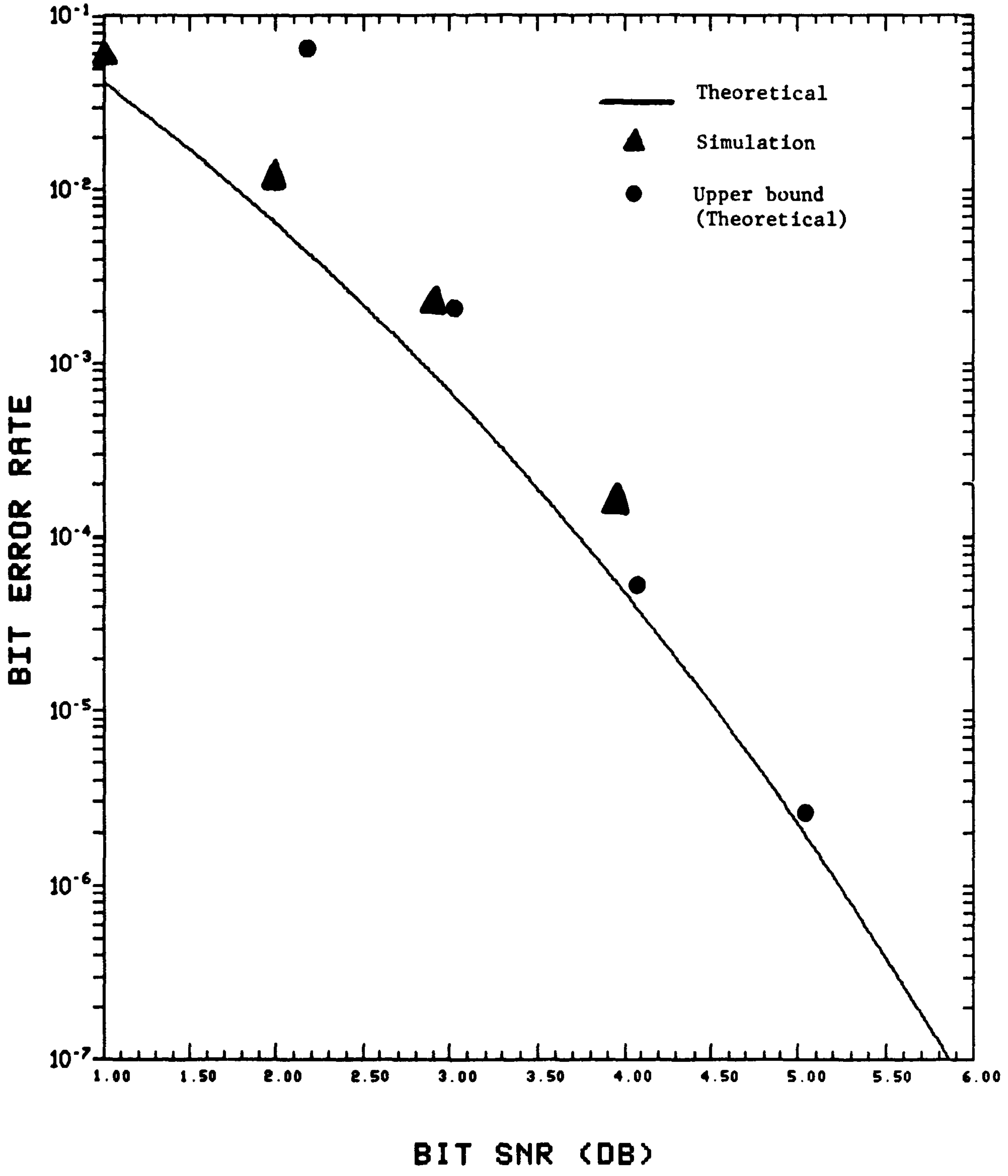


Figure 12. Bit Error Rate Performance of Viterbi Decoder

IX. CONCLUSION

The algorithms and subroutines developed in this study are suggestions for constructing an actual data processing program and may require some further optimization.

Simulations of the carrier tracking loop have indicated that the least-squares procedure for finding w and $\dot{w}$ may not converge if the starting parameters are not close enough to the actual w and $\dot{w}$, if the noise is too large, or if the phase fluctuations are so large that the 30 second coherence span is too long. Normally if the carrier extraction is successful, the subcarrier extraction, bit and frame synchronization proceed smoothly. Simulations of the decoding algorithm are in good agreement with the theoretical performance. Detailed analyses of the statistical performance of the subcarrier demodulator has not been carried out, but the techniques used for signal processing should not produce any significant degradation from the theoretical values. For this reason, the major uncertainty in the decoding is associated with the carrier extraction and the processes that could cause unmodeled phase modulation of the carrier wave.

X. REFERENCES

[1] P.A., Preston, J.H. Wilcher, and C.T. Stelzried, "The Venus Balloon Project," TDA Progress Report 48-80, pp. 195-201, October-December, 1984.

[2] Viterbi, A.J. and Omura, J.K., "Principles of Digital Communication and Coding," McGraw-Hill, 1979.

[3] Divsalar, D.. "Performance of Mismatched Receivers on Bandlimited Channels," Ph.D. dissertation, University of California, Los Angeles, 1978.

[4] Odenwalder, J.P., "Optimal Decoding of Convolutional Codes," Ph.D. dissertation, University of California, Los Angeles, 1970.

APPENDIX

```
      SUBROUTINE SETUP
C***************** SET UP FOR VITERBI DECODER ****************C
C M IS VECTOR OF STATE METRICS                                C
C LO IS INDEX VECTOR OF ALL STATE TRANSITICNS SUCH AS J->2J-1 C
C   ; J=1,2,.....,16.                                         C
C L1 IS INDEX VECTOR OF ALL STATE TRANSITIONS SUCH AS         C
C    16+J->2J-1; J=1,2,.....,16.                              C
C NOTE : INDEX VECTOR OF ALL STATE TRANSITIONS SUCH AS J->2J  C
C         IS L1.                                              C
C        INDEX VECTOR OF ALL STATE TRANSITIONS SUCH AS        C
C        16+J->2J IS LO.                                      C
C MO IS 36*32 MATRIX WHICH STORES ALL BEST PATHS TO 32 STATES,C
C    IT ONLY KEEPS PATHS WITH LENGTH 36. THE 37TH BIT CAN BE  C
C    REGARDED AS HARD DECODED BIT.                            C
C MN IS 36*32 SCRATCH MATRIX.                                 C
C*************************************************************C
      COMMON/LA1/LO(16),L1(16),M(32),MO(36,32),MN(36,32)
      DATA LO/1,4,3,2,1,4,3,2,3,2,1,4,3,2,1,4/
      DATA L1/4,1,2,3,4,1,2,3,2,3,4,1,2,3,4,1/
C INITIALIZATION
      DO 5 J=1,32
      DO 5 I=1,36
  5   MO(I,J)=0
      DO 10 I=1,16
      I1=I+I-1
      I2=I+I
      MN(1,I1)=0
  10  MN(1,I2)=1
C  ASSUME ENCODER IS IN STATE # 13
      DO 20 I=1,32
  20  M(I)=-2000
      M(13)=0
      RETURN
      END

C
C
      SUBROUTINE DECODE(KTMI,LY1,LY2,L,KTMO,KOUT)
C  THIS SUBROUTINE IS VITERBI DECODING ALGORITHM
C  KTMI IS INPUT TIME
C  KTMO IS OUTPUT TIME
C  LY1 AND LY2 ARE QUANTIZER OUTPUTS
C  L IS NUMBER OF QUANTIZATION LEVELS
C  KOUT IS DECODE BIT,IF IT IS NOT EQUAL TO 9
C  MAXN IS THE STATE NUMBER WITH LARGEST SURVIVED METRIC
      DIMENSION LB(4),MS(32)
      COMMON/LA1/LO(16),L1(16),M(32),MO(36,32),MN(36,32)
C
C COMPUTATION OF BRANCH METRICS
```

```
C
      CALL BRANCH(LY1,LY2,LB,L)
C
C SELECTING THE SURVIVED METRICS AND SURVIVED DATA SEQUENCES
      DO 10 K=1,16
      IJ=K+K-1
      IL=K+K
      IK=K+16
      MT0=M(K)+LB(L0(K))
      MT1=M(IK)+LB(L1(K))
      MT2=M(K)+LB(L1(K))
      MT3=M(IK)+LB(L0(K))
      IF(MT0.GE.MT1) GO TO 20
      MS(IJ)=MT1
      DO 30 I=1,35
      I1=I+1
   30 MN(I1,IJ)=MO(I,IK)
      GO TO 40
   20 MS(IJ)=MT0
      DO 50 I=1,35
      I1=I+1
   50 MN(I1,IJ)=MO(I,K)
   40 CONTINUE
      IF(MT2.GE.MT3) GO TO 60
      MS(IL)=MT3
      DO 70 I=1,35
      I1=I+1
   70 MN(I1,IL)=MO(I,IK)
      GO TO 10
   60 MS(IL)=MT2
      DO 80 I=1,35
      I1=I+1
   80 MN(I1,IL)=MO(I,K)
   10 CONTINUE
C
C NORMALIZE M
C
      DO 150 I=1,32
  150 M(I)=MS(I)
      CALL MAX(MXM,MAXN)
      DO 90 I=1,32
   90 M(I)=M(I)-MXM
C
      DO 100 J=1,32
      DO 100 I=1,36
  100 MO(I,J)=MN(I,J)
C  OUTPUT TIME AND DECODED BIT
      KOUT=9
      IF(KTMI.GE.36) KOUT=MO(36,1)
      IF(KTMI.GE.36) KTMO=KTMI-35
      RETURN
      END
C
C
C
      SUBROUTINE BRANCH(LY1,LY2,LB,L)
C THIS SUBROUTINE COMPUTES THE BRANCH METRICS
```

```
C LY1 AND LY2 ARE QUANTIZER OUTPUTS
C L IS NUMBER OF QUANTIZATION LEVELS
C ARRAY LB STORES 4 BRANCH METRICS
      DIMENSION LB(1)
      LB(1)=L+L-2-LY1-LY2
      LB(2)=L-1-LY1+LY2
      LB(3)=L-1+LY1-LY2
      LB(4)=LY1+LY2
      RETURN
      END
C
C
C
      SUBROUTINE MAX(MXM,MAXN)
C MXM IS LARGEST SURVIVED METRIC
C MAXN IS THE STATE NUMBER WITH LARGEST SURVIVED METRIC
      COMMON/LA1/LO(16),L1(16),M(32),MO(36,32),MN(36,32)
      MXM=M(1)
      MAXN=1
      DO 10 I=2,32
      IF(M(I).GT.MXM) GO TO 20
      GO TO 10
   20 MXM=M(I)
      MAXN=I
   10 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE QUANT(R,L,LY,TH)
C THIS SUBROUTINE IS L LEVEL QUNTIZER
C TH IS UPPER LIMIT OF QUANTIZATION
C R IS RECEIVED SAMPLE AT THE INPUT OF QUANTIZER
C LY IS OUTPUT QUANTIZATION LEVEL
      SPACE=(TH+TH)/(L-2)
      IF(R.GT.TH) LY=0
      IF(R.LE.TH) LY=L-1
      IF(ABS(R).LE.TH) GO TO 10
      GO TO 20
   10 LY=(TH-R)/SPACE+1
   20 CONTINUE
      RETURN
      END
C
C
C
```