Post-Print

LOSSLESS COMPRESSION AND RATE CONTROL FOR THE GALILEO NIMS INSTRUMENT

Robert F. Rice
Robert Mehlman

Proceedings of the Eighth Annual
AIAA/USU Conference on Small Satellites

Logan, Utah, **USA**

September 1, 1994

LOSSLESS COMPRESSION AND RATE CONTROL FOR THE GALILEO NIMS INSTRUMENT

Robert F. Rice
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Robert Mehlman
institute for Geophysics and Planetary Physics
University of California at Los Angeles
Los Angeles, California

ABSTRACT

The Galileo Near-Infrared Mapping Spectrometer (NIMS) is a sophisticated multi-spectral instrument that was developed to study the spatial/compositional aspects of the atmosphere of Jupiter and the surfaces of it's satellites, Since its original development, the communication capability of the Galileo spacecraft has been severely reduced as the result of a failure of ground controllers to open the main antenna. The data rate which will be available for all instruments at the first Jupiter encounter in 1995 has been reduced by several orders of magnitude (even after recent numerous improvements to data link efficiency,)

Because of this severe limitation, real-time return of science data is now secondary. The primary means of returning Galileo science data will be to store encounter data using the on-board tape recorder and then gradually play it back over an extended period of time, But even this time period is restricted to approximately one month since the tape recorder must be ready for additional encounters as the spacecraft orbits Jupiter.

The NIMS instrument designers built-in considerable flexibility to adjust parameters that affect spectral resolution, spatial resolution and field of view. Each of these affects the amount of data generated. As a result of the severe communication constraints, this flexibility will be enhanced even further by software changes to enable the NIMS scientists to maximize the value of their science return.

The NIMS instrument will also employ adaptive lossless data compression (by factors of 1.5:1 up to 4:1) to further improve its return capabilities. This paper describes how this is accomplished such that the flexibility to arbitrarily alter instrument parameters is not jeopardized.

The paper will also discuss a practical solution to the problem of NIMS' allocation of a fixed number of bits over a full tape load in the face of changing data characteristics and associated greatly diverse data compression factors. The approach described is a global compression rate allocation strategy, designed to achieve nearly-optimal bit usage over a complete tape load. Data history and a priori planning are the major inputs to the strategy's control structure. This mechanism then autonomously makes adjustments in specific parameters which affect data generation to ensure compliance with the overall bit constraint. By "looking ahead" the strategy results in more accurate decision making, minimizing unnecessary reductions in data quality and ensuring that unused bits will not be wasted.

I. INTRODUCTION

The Galileo Near-Infrared Mapping Spectrometer (NIMS) employs a combination of imaging and spectroscopic methods.[1-3] Simultaneous use of these two methods yields a powerful combination for the investigation of planetary geology and atmospheres. NIMS was developed as an experiment to investigate Jupiter and the Jovian satellites during a two year orbital operation period commencing in December 1995. As originally conceived, the acquisition of science data would be primarily a real-time operation. Most data would be communicated immediately through a data link with a capability of over 100 Kbits/sec. But this is no longer possible.

The Problem

The Galileo spacecraft suffered a dramatic loss in communication capability when its primary antenna failed to open. After numerous attempts to open the antenna, the loss appears to be permanent. Consequently, encounters at Jupiter will be forced to rely on an alternate low-gain antenna system, reducing downlink data rates by several orders of magnitude. Realization of this situation prompted major efforts to consider all avenues for improving the science return from orbital operations. This modified mission to Jupiter will be called here the "Low-Gain Antenna (LGA) Mission."[4]

Datalink improvements for the LGA Mission will come from: the use of much longer convolutional codes in conjunction with a more sophisticated application of Reed-Solomon codes, antenna arraying and other Deep-Space Network (DSN) improvements, But even with these changes, the orbital mission will be limited to data rates between 10 and 100 bits per second, only slightly better than when Mariner 4 first sent back images of Mars in 1964.

Other efforts to improve the science return capabilities have focussed on modifications to the way data is acquired, stored and communicated (packet telemetry). Such efforts have been constrained by the limited on-board processing resources of the Galileo spacecraft, This includes several ancient RCA 1802

microprocessors, each with a capability of only 0,066 million operations per second.*

Because of the extremely low downlink data rates, science data generated at an encounter must be accumulated on the Galileo tape recorder for later transmission, But downlink transmission time for one encounter is generally limited to "only" about 1 month before preparations for the next must take precedence.

Except for imaging data, any data placed on this recorder must be stored in raw, uncompressed form. Further, the recorder's capacity is insufficient to store all the data that could be acquired during an encounter. Thus it has become important to scrutinize what data is actually placed on the recorder.[3]

Unless the primary Galileo antenna becomes at least partially operable, there are no conditions under which the transmission time between encounters is sufficient to communicate all of the data bits stored on the recorder. It was this situation that prompted the development of software driven data compression and other enhancements to assist in the post-encounter downlink process.

Data Compression for Galileo

Images for optical navigation will be compressed by a system which achieves very high compression factors of over 100:1.5 This is accomplished by first determining where important stars or planetary/satellite limbs are located within an image and then transmitting only small regions around those locations. This concept was first demonstrated to the National Aeronautics and Space Administration (NASA) in 1981.6

Some science imaging will utilize an on-board hardware Block Adaptive Rate Controlled (BARC) image compression system.' BARC's compression factor was constrained at launch to a fixed 2.5:1, The resulting 3.24 bits/picture element guarantees near perfect quality. But to achieve desired higher compression factors a lossy cosine transform based compression algorithm will be applied to uncompressed images stored on the recorder.[8] This same algorithm will be applied to one other instrument,

Adaptive lossless compression (coding), in the form of an 1802-constrained version of the Rice Algorithms[9-12] will be applied to the data of six non-imaging instruments. The most sophisticated such application is the multi-spectral NIMS instrument,

NIMS, NIMS instrument designers originally built-in extraordinary flexibility to adjust parameters that affect spectral resolution, spatial resolution and field of view, This capability has become an important, but not sufficient, asset to the data constrained LGA Mission that NIMS scientists are now facing. Further software-based improvements to flexibility are being implemented. The application of adaptive lossless compression to NIMS data must not only achieve valuable improvements in effective data rate requirements but must also preserve this flexibility,

Outline

The sections that follow will describe these efforts to incorporate adaptive lossless compression within the new NIMS data handling environment being developed for the Galileo LGA Mission to Jupiter.

The data system perspective of the NIMS instrument is provided first, For a broader insight into the design of this instrument and its original science goals, consult Refs. 1 and 2. For more recent considerations of the LGA Mission, consult Ref. 3.

"rhe Rice Algorithms for lossless compression can be applied to an extremely broad class of data sources without the need to understand every detail about them. And this remains true for NIMS. Hence, the algorithms are only defined here in sufficient detail to support a description of their application to NIMS. Several references are available for greater detail.[9-12]

The application of efficient lossless compression to data like NIMS will generally result in data-dependent compression factors. For many earthbound applications this compression factor uncertainty is not an issue. If the compression factor is less than expected, simply take longer to transmit it. But the NIMS instrument is basically constrained to use a fixed number of bits per tape load. Using significantly more bits than expected could mean the loss of valuable data. To help alleviate this problem the global rate allocation approach is discussed. It combines data history, a priori planning and autonomous adjustments to parameters to ensure compliance with the overall bit constraint.

II. THE NfMS INSTRUMENT

A Word About Correlation

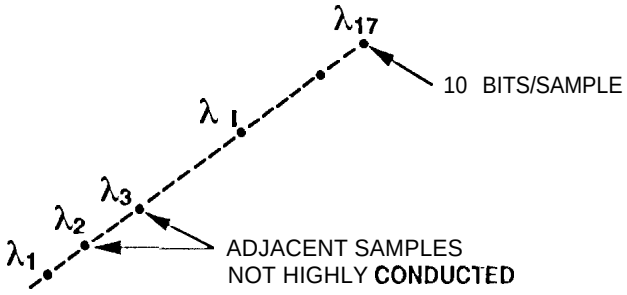
The term correlation can be defined mathematically. But here it will be used as a non-mathematical guide to data similarity and expected coding efficiency. Basically, the higher the correlation between two data samples, the better that one sample can be predicted from knowledge of the other. In later sections, better prediction will mean that a fossfess compressor will tend to perform better, achieving higher average compression factors, Thus it is important to the ultimate goal of defining an efficient lossless compressor to identify where sample-to-sample correlation exists in the NIMS data structure.

The Wavelength Dimension

The NIMS instrument has 17 detectors and a Grating. If the Grating is fixed, the wavelengths

* A much faster AMD 2900 microprocessor Is also on-board, but its availability is limited and not applicable to the considerations of this paper.

may be denoted by $\lambda_1, \lambda_2, \ldots, \lambda_{17}$ as shown in Fig, 1. If the Grating is allowed to move, as many as 23 additional wavelengths can be sampled, (The maximum number of wavelengths is 408.)



Fig, 1. Seventeen Detectors.

The y Dimension

Mirror motion within the instrument generates 20 highly correlated samples (a vector in the y direction) for each detector, resulting in a "data plane" of data as showninFig.2.The words "TOP" and "BOTTOM" appear here and in later diagrams to help maintain your orientation throughout,

The x Dimension

Spacecraft and scan platform motion creates yet another dimension (the X direction) as illustrated in Fig. 3 where X_0, X_1, . . . etc. identify the data planes generated when the instrument Grating is fixed, In instrument modes where the Grating is moving, intermediate data planes are generated.

Note that these intermediate planes are slightly offset in wavelength as illustrated in Fig. 4 where the horizontal plane of possible data for the jth component, y_j, is shown.

For a specific mirror position, say y_j, grating position x lies between X_0 and X_1, As x increases from X_0 to X_1, the observed wavelength λ_i from the ith detector increases proportionally from λ_i to λ_{i+1}. Thus, with α divisions we have

$$x_\ell = X_0 + \ell \frac{X_1 - X_0}{\alpha}, \lambda_i^\ell = \lambda_i + \ell \frac{\lambda_{i+1} - \lambda_i}{\alpha} \quad (1)$$

where ℓ = O, 1, 2, ..., a and a can be as large as 24,

Rewriting,

$$x_\ell = X_0 + \ell \Delta x, \lambda_i^\ell = \lambda_i + \ell \Delta\lambda \quad (2)$$

and

$$\Delta x = \frac{X_1 - X_0}{a}, \Delta\lambda = \frac{\lambda_{i+1} - \lambda_i}{a} \quad (3)$$

Correlation. Recall that we noted in Fig. 1 that adjacent samples between detectors at the same mirror position (y) and grating position (x) are generally not very correlated, Then correlation can be expected to be even less between samples taken at point A and point B in Fig, 4, because there is an additional change in x (and time) also.

But when there area= 24 divisions in x_ℓ and λ_i^ℓ (see Eqs. 1 -3), the corresponding samples between adjacent planes (the same vector component for the same detector) do exhibit significant correlation. Basically, the wavelengths are simply much closer together and the samples are thus more likely to be the same. The x and λ values differ only by Ax and $\Delta\lambda$ in (3),

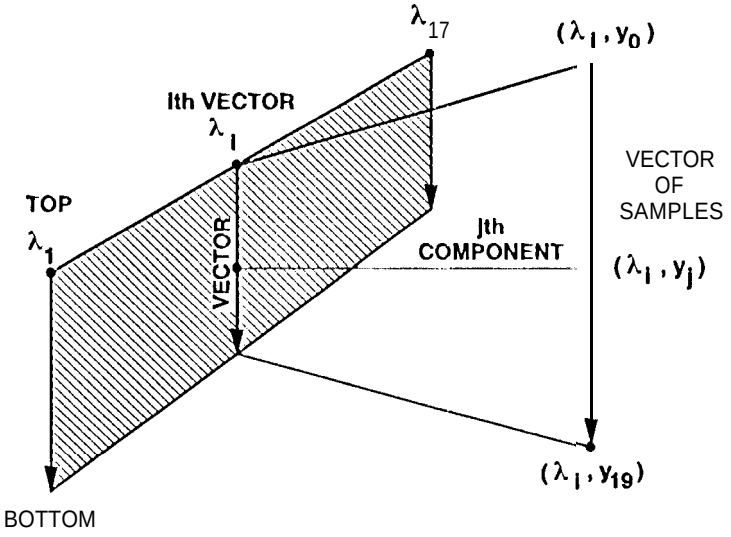


Fig. 2. Plane of Data

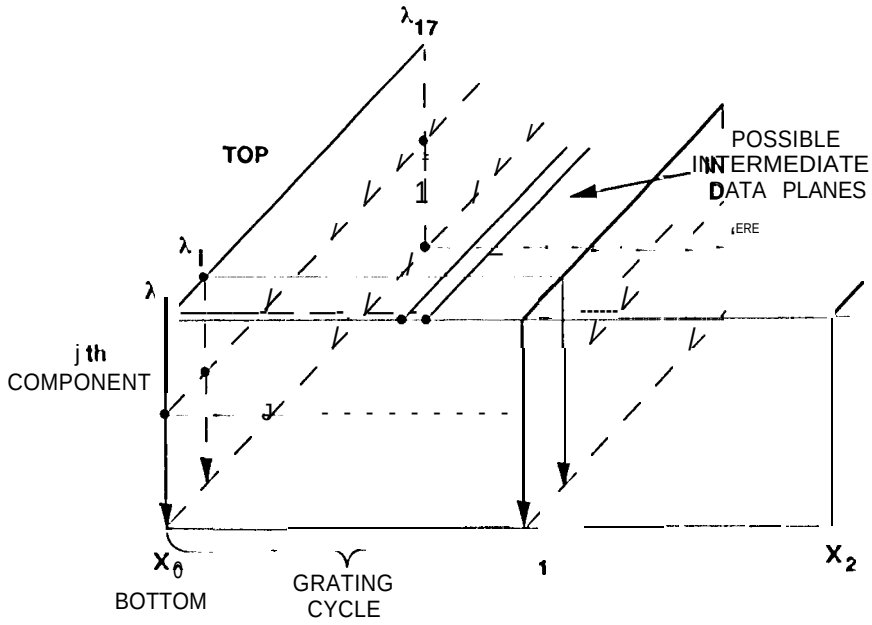


Fig. 3. The X Dimension

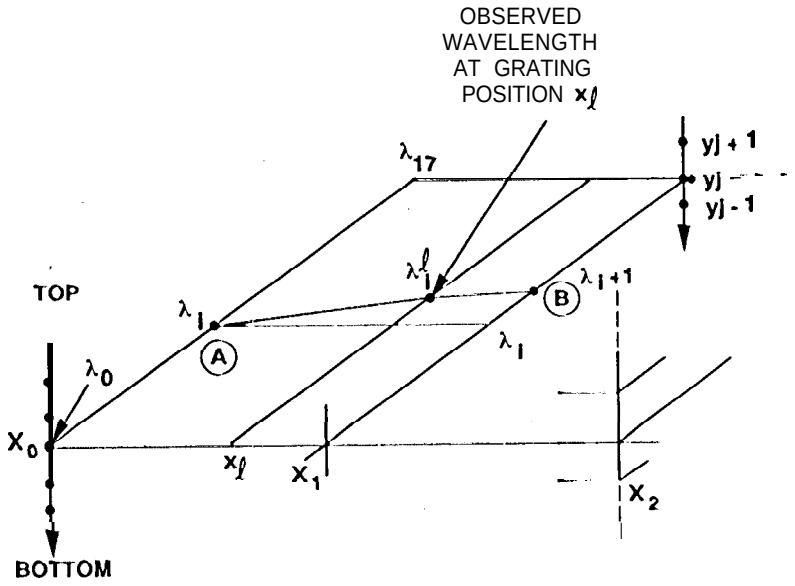


Fig. 4. Intermediate Planes

This identifies the best situation for correlation between planes. But correlation does not suddenly disappear when these conditions are not met. More generally, correlation (and ultimately coding performance) can be expected to gradually decrease from a maximum as samples are gradually separated further in x, y and λ. The lossless compression algorithms described in later sections will address these other situations by adapting to make the most of whatever correlation is present:

a) Between corresponding samples in adjacent planes when $\alpha \leq 24$;

t)) Between corresponding samples in non-adjacent planes.

Vector Masks*

"rhe ℓ parameter in Eqs. 1-3 identifies one of α-ℓ data planes that lie between Grating Cycles. Each

•An LGA Mission enhancement

plane contains seventeen 20 component vectors corresponding to each of the 17 detectors. 1 hus for each Grating Cycle, a particular vector can be identified as

$$\tilde{v}_{i,\ell} \tag{4}$$

by the parameters i and ℓ, where

$$\{1 \leq i \text{ is } 17\} = \text{detector number} \tag{5}$$

$$\{O < \ell \leq \alpha\} = \text{intermediate plane number} \tag{6}$$

"Corresponding Vectors" are vectors in different data planes with the same detector number, i. "Adjacent Planes" are planes with parameter ℓ differing only by 1.

Clearly, the storage and transmission requirements 'for NIMS data can be reduced by omitting some of the vectors which are stored and transmitted, But it is crucial to be able to perform such "editing" in an intelligent way that is in concert with the current science investigations, Hence, the capability to include or exclude any individual vector in the recorded/ transmitted data was added by software modifications, 'This is accomplished by assigning a vector mask to each possible vector.

Recall that the primary mode of interest is a real-time storage of NIMS data on the tape recorder, followed by a later processing and compression of recorded data for non-real time transmission. A mask can be applied both before recording and before processing compression. These "before recording" and "after recording*' masks are defined by

$$m_{br}(i,\ell) = \begin{cases} 1 \text{ if } \tilde{v}_{i,\ell} \text{ to be recorded} \\ O \text{ Otherwise} \end{cases} \tag{7}$$

$$m_{ar}(i,\ell) = \begin{cases} 1 \text{ if } \tilde{v}_{i,\ell} \text{ to be processed} \\ O \text{ Otherwise} \end{cases} \tag{8}$$

Clearly, if either mask for vector $\tilde{v}_{i,\ell}$ is zero, $\tilde{v}_{i,\ell}$ will not be included in the transmission of NIMS data. The resulting composite transmission mask is then

$$m_t(i,\ell) = m_{br}(i,\ell) \cdot m_{ar}(i,\ell) \tag{9}$$

where the '•' can be interpreted as either multiplication or a logical 'AND'.

Thus, vector masks provide a separate means for reducing the quantity of data recorded and transmitted. Their specification is now an important part of the LGA planning process both before and after recording, since their definition can significantly affect the science content of transmitted data.

Changing Vector Lengths*

The number of vector components actually processed for transmission has also been made adjustable for the LGA Mission. Unlike the per vector control provided by vector masks, changes to the number of vector components apply only over large spans of data. We will make use of this added flexibility in later discussions of rate control.

III. THE RICE ALGORITHMS FOR LOSSLESS COMPRESSION

I-he most general description of the Rice Algorithms that applies to the LGA Mission applications can be found in Ref. 9.** Prior to the LGA Mission considerations, software had been written by Alan Schlutsmeyer which basically duplicated the algorithm specifications in Ref. 9. This "Schlutsmeyer C Code" was then extensively used to develop and evaluate the Rice Algorithm application to NIMS and five other Galileo instruments. It will also form the basis of decompression software now being developed for the restoration of data upon return.

Following initial development, an 1802 assembler version was written by Ron Burns to demonstrate flight feasibility in speed and code size. This "Burns code" is now the basis of on-board compression software.

Basically, the Schlutsmeyer and Burns compression software provide algorithms for representing (coding) blocks of data. This section will describe these "block coders" and subsequent sections will then apply them to the NIMS data structure.

Coder for Short Blocks

Figure 5 illustrates the form of a "coding module" which applies to the representation of very short NIMS data blocks (i.e., lengths of 20 or less).

The input to this NIMS coding module, C_N, is a J sample block of n bit data

$$\tilde{X} = x_1 x_2 \ldots x_J \tag{lo}$$

and possibly another initialization sample called a Reference Sample, R. ($\tilde{X}$ here has no relationship to the $\tilde{X}_1, \tilde{X}_2 \cdots$ in earlier discussions).

* AN LGA Mission enhancement.
** For a less complete but more compact description consult Ref. 10.

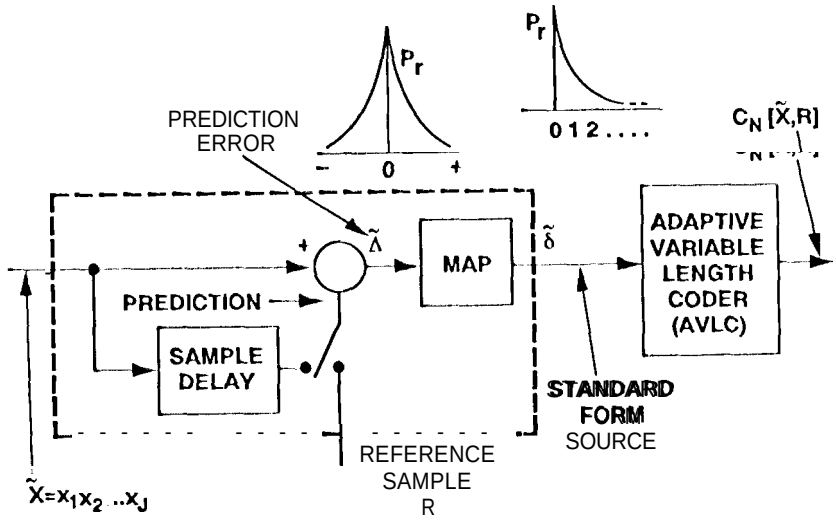


Fig. 5. Coding Module for Short Blocks, C_N

The final coded (compressed) output sequence is then

$$\tilde{\zeta} = C_N [\tilde{X}, R] \quad (11)$$

This process is "lossless" or "noiseless" so that, given R, this sequence can be "decoded" back into $\tilde{X}$

$$\tilde{X} = C_N^{-1} [\tilde{\zeta}, R] \quad (12)$$

Predictive Preprocessor. Data block $\tilde{X}$ enters a simple one-dimensional predictor which predicts that the next sample will be the same as the previous sample. When there is no prior sample to predict with (e.g., the first sample) or for other reasons, the Reference Sample, R, will be used for prediction.

The difference between an incoming sample and its prediction produces a prediction error, Block $\tilde{X}$ then results in a J sample block of prediction errors, $\tilde{\Lambda}$. Prediction errors tend to be distributed unimodally around zero, Inactive, highly correlated data will tend to produce a very peaked distribution. Similarly, more active, less correlated data will generate broader distributions. The sha p of these distributions is well modeled as Laplacian.[12]

The positive and negative prediction errors are then mapped into the non-negative integers such that (because of the unimodal error distributions) smaller integers tend to be more likely than the larger integers (O → O, +1 → +1, -1 → +2, +2 → 3, . . .). The block of J prediction errors, $\tilde{\Lambda}$, produces a block of J mapped prediction errors, $\tilde{\delta}$. Data at this point is said to have been "preprocessed" and in the form of a "Standard Source".

Adaptive Variable Length Coding. The next step, labelled "AVLC" in Fig. 1, is to represent each mapped prediction error in $\tilde{\delta}$ with a codeword from a variable length code. Here, shorter code words are assigned to smaller non-negative integers. (That is, the codeword assigned to a specific non-negative integer is never longer than the codeword assigned to a larger non-negative integer.) When smaller integers occur more frequently than larger ones, a goal of the preprocessing, this assignment assures that shorter codewords will be used more frequently than larger ones,

The latter assignment of codewords assures that a particular variable length code is properly used. It says nothing about the actual efficiency of the code to represent data. Under the assumption that the underlying distribution of preprocessed non-negative samples is fixed, it is possible to

1) Determine the "entropy" as

$$\tilde{H}_\delta = -\sum_i p_i \log_2 p_i \text{ bits/sample} \quad (13)$$

where p_i is the probability that integer i occurs, and

2) To generate an optimum variable length code for a particular distribution using the famous Huff man Algorithm.[13]

For all practical purposes, the entropy in Eq. 13 determines the best average performance for any code that is lossless. A variable length code that performs close to the entropy can be said to be efficient. Huff man Algorithm-generated codes can generally produce efficient performance in narrow ranges of entropy above about 1.5 bits/sample or so. Huffman codes have been called optimum because they will perform as well as any other single variable length

code (on data generated from the Huffman code's design distribution),

But there are several practical problems in relying on Huff man codes, In general, a pure Huffman code would require storage for all its codewords. The NIMS data quantization of 10 bits/sample would then require storage for 1024 codewords. The problem is far worse for some other Galileo instruments which have 16 bit data samples,

But equally important, relying on a single variable length code, Huffman or not, is not sufficient to assure efficient performance over all the conditions that most real data sources produce, In practice, the distribution of non-negative integers, {pi} used in Eq. 13 is not static, but varies over time, yielding entropies that may vary significantly. A single Huffman code is only "optimum" for the prescribed distribution it was designed from. It can only be expected to remain efficient over a narrow range of entropies. Figure 6 expands the AVLC box in Fig. 5 to reveal the basic concept of adaptive variable length coding usually associated with the Rice Algorithms.

Here, there is a bank of N variable length codes, labelled $c_0, c_1, \ldots, c_{N-1}$. Each of these is designed to operate efficiently over a different narrow range of entropies. A decision maker determines which of the N-1 codes will produce the shortest coded sequence when applied to the current input block, $\tilde{\delta}$.

The result of that decision is a "code identifier" represented by the integer "id" and its fixed length binary representation by

$$ID(id) \tag{14}$$

The correct coded result is then

$$c_{id}\,[id] \tag{15}$$

But to ensure that a decoder will know which decoding algorithm to use, the coded sequence must be prefaced by the code identifier. Thus the overall coded result takes the form*

$$\text{AVLC}\ [\tilde{\delta}] = ID(id) * c_{id}[\tilde{\delta}] = C_N[\tilde{X}, R] \tag{16}$$

where

$C_N[\tilde{X}, R]$ is taken from Fig, 5 to relate back to the original task of coding $\tilde{X}$.

Can the need for large lookup tables be avoided? Code c_{N-1} in Fig. 6 has been called the "backup" or "default" mode, taking the form

$$c_{N-1}\,[\tilde{\delta}] = \tilde{\delta} \tag{17}$$

Code c_1 is the "fundamental sequence code" defined for each codeword by

$$fs[i] = \overbrace{000\ldots0001}^{i\ \text{zeroes}} \qquad \text{for } i \geq 0 \tag{18}$$

The structure of this code means that each codeword can be easily generated and need not be stored.

All the other codes, starting with c_3, are equally simple. A codeword for these "Split-sample Modes" is generated by splitting off k least significant bits from a sample and applying the Fundamental Sequence Code in Eq. 18 to the remaining most significant bits. Thus again, storage for codewords is unnecessary.

Yeh recently showed that if the distribution of prediction errors (see Fig. 5) is Laplacian, a very reasonable assumption, then $c_1, c_2, \ldots, c_{N-1}$ are equivalent to Huffman codes.[12] Moreover, the entropy where these optimum codes achieve their best performance is at

$$\tilde{H}_\delta \approx k + 2 \text{ bits/sample} \tag{19}$$

for $k \geq 1$, Thus any range of entropy beyond about 1.5 bits/sample can be efficiently covered.

Coder co is a special coder developed to provide efficient performance at entropies as low as about 0.75 bits/sample. It employs a separate 8-codeword variable length code. Look for an explanation of "PSI0" in Ref. 9.

The decision on which coder to use could optimally be determined by comparing the coding results of each coder. But tests have shown that, with negligible loss in performance, decisions can be made by simply adding up the samples in $\tilde{\delta}$ and comparing the result to a set of thresholds. The latter approach is used for the Galileo coders.

Depending only on the number of code options included, average performance of such a coder will be just above the entropy as shown in Fig. 7. In instances when data characteristics vary rapidly, measurements of performance below an average entropy may be observed.**

* The asterisk,", is used here and in later expressions to denote concatenation.

** For those familiar with the contents of Ref. 9, the implemented coder for Galileo is basically PSI14,K with K = 0,1. The K=l split is done to handle 16 bidata and is done prior to preprocessing. internally, PSI14 parameter λ can equal O or 1. Block sizes for all Galileo Instruments range from 4 to 20 and sample quantization from 8 to 16 bits/sample. Formatting of the code Identifiers and split-samples is slightly changed to accomodate the limitations of the 1002 processor.

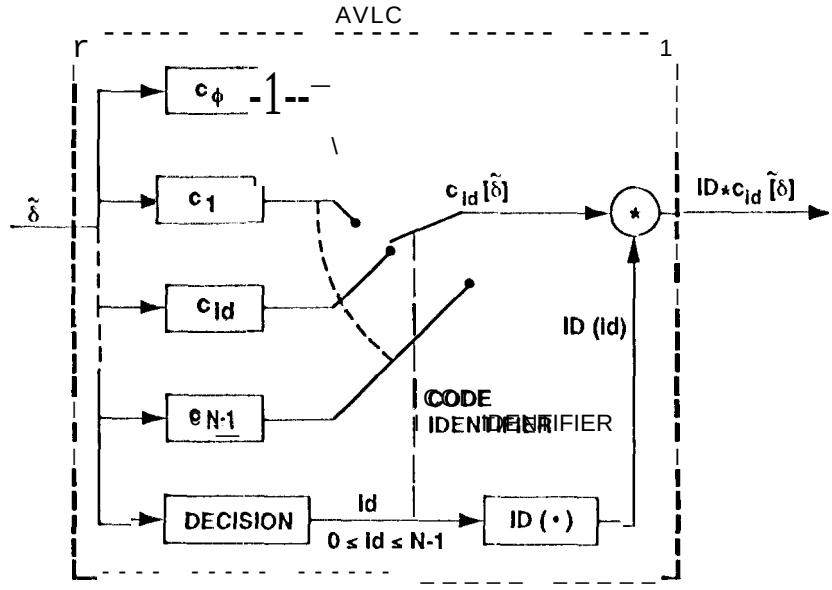


Fig. 6. Adaptive Variable Length Coder

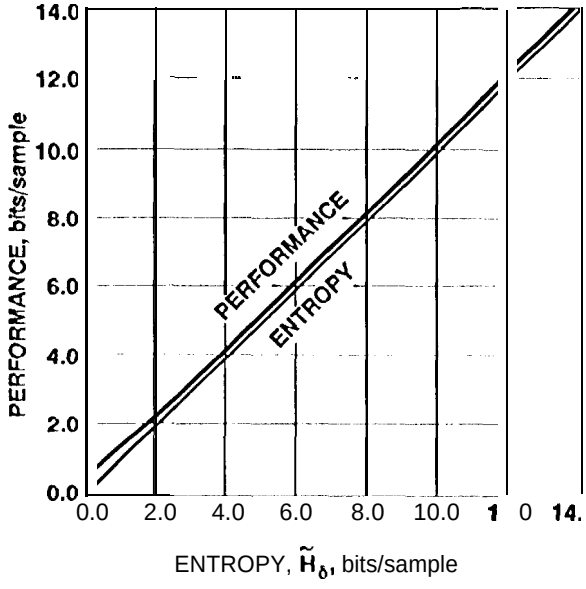


Fig. 7. Typical Performance Characteristics

Coder for Longer Blocks

Now consider the more general data sequence $\tilde{Y}$ which may be made up of one or more data blocks, Xi, which will be coded by the coding module, C_N in Fig. 5,

$$Y = y_1\ y_2\ \ldots\ y_M$$

$$= \tilde{X}_i * \tilde{X}_2 * \ldots * \tilde{X}_e \qquad (20)$$

where e ≥ 1 and the number of samples in $\tilde{X}_i$ is J_i, so that the length of $\tilde{Y}$ is

$$\mathcal{L}(\tilde{Y}) = \sum_{i=1}^{e} J_i = M \text{ samples} \qquad (21)$$

Then for example

$$\tilde{X}_i = y_1 y_2 \ldots y_{J_1} \qquad (22)$$

It is assumed that all adjacent samples in $\tilde{Y}$ are correlated. Any sample can be used as a prediction of the sample that follows it as in Fig, 5. Then, assuming that a Reference Sample, R_1, for the first block is already known, the coding of $\tilde{Y}$ takes the form

$$PSI_{NIMS}[\tilde{Y}] = C_N[XI, R_1] * C_N[\tilde{X}_2, R_2] \bullet, \ldots * C_N[\tilde{X}_e, R_e] \qquad (23)$$

where reference samples R_2, R_3, ..., R_e are provided automatically as the last sample from the preceding block (see the sample delay component in Fig. 5).

But in the case where a reference sample Is required, $\tilde{Y}$ is coded by first using y_i as a reference sample as

$$PSI^{wr}_{NIMS}[\tilde{Y}] = y_1 * C_N[\tilde{X}_1', YI] * C_N[\tilde{X}_2, R_2] * \ldots C_N[\tilde{X}_e, R_e] \qquad (24)$$

where $\tilde{X}_1'$ is the J_1 -1 sample block

$$\tilde{X}_1' = y_2\ y_3 \ldots y_{J_1} \quad (25)$$

For NIMS and other instruments, parameters were specifically chosen so that for a particular $\tilde{Y}$, the J_i will all be the same so that

$$\mathcal{L}(\tilde{Y}) = M = eJ$$

However, numerous values of $J \geq 8$ are supported.

Subsequent sections will describe how PSI_{NIM}s and PSI^{WT}_{NIMS} coders are applied to the NIMS data structure.

IV. CODING THE NIMS DATA

The ℓth plane of a NIMS Grating Cycle is shown in Fig. 8. Coding and decoding of the ith vector, $\tilde{v}_{i,\ell}$ can be specified from Eq. 23 or Eq. 24 and the vector mask in Eq. 9 as

$$NIMS_1[\tilde{v}_{i,\ell}] = \begin{cases} PSI_{NIMS}[\tilde{v}_{i,\ell}] \text{ if } m_t(i,\ell) = 1 \\ \phi = \text{empty, not transmitted} \quad \text{Otherwise} \end{cases} \quad (26)$$

or

$$NIMS_2[\tilde{v}_{i,\ell}] = \begin{cases} PSI^{WT}_{NIMS}[\tilde{v}_{i,\ell}] \text{ if } m_t(i,\ell) = 1 \\ \phi \quad \text{Otherwise} \end{cases} \quad (27)$$

In either case nothing is generated for vector $\tilde{v}_{i,\ell}$ unless the corresponding mask value is 1. Here, $\tilde{v}_{i,\ell}$ takes the place of $\tilde{Y}$ in Eq. 20.

Using $NIMS_1$ assumes that a reference sample is available to predict the first sample in $\tilde{v}_{i,\ell}$ whereas using $NIMS_2$ assures that $\tilde{v}_{i,\ell}$ can be coded/decoded independently of all others (because the first sample of $\tilde{v}_{i,\ell}$ is used as a reference).

But we will soon be interested in a situation where the use of both $NIMS_1$ and $NIMS_2$ will apply within a plane. To designate which coding approach in Eqs. 26 and 27 to use, we specify the parameter $\zeta_{i,\ell}$ as

$$\zeta_{i,\ell} \begin{cases} 1 \text{ if } NIMS_1 \text{ (in Eq. 26) is to be used for } \tilde{v}_{i,\ell} \\ 2 \text{ if } NIMS_2 \text{ (in Eq. 27) is to be used for } \tilde{v}_{i,\ell} \end{cases} \quad (28)$$

Then, the coding of the ℓth plane can be specified as

$$NIMS_{\zeta_{1,\ell}}[\tilde{v}_{1,\ell}] * NIMS_{\zeta_{2,\ell}}[\tilde{v}_{2,\ell}] * \ldots \ldots * NIMS_{\zeta_{17,\ell}}[\tilde{v}_{17,\ell}] \quad (29)$$

Predicting from Previous Planes

Again refer to Fig, 8 which shows data planes that are adjacent to the ℓth. The ith vector in each plane is highlighted, $\tilde{v}_{i,\ell-1}$, $\tilde{v}_{i,\ell}$, $\tilde{v}_{i,\ell+1}$, $\tilde{v}_{i,\ell+2}$.

Note that the mirror scan process actually results in a reversal of the order of vector components in adjacent planes.

It was noted earlier that when two data planes are close (in x of Fig. 4) there often can be useful correlation between corresponding samples (e.g., the jth component of the ith vector in each plane). Then, rather than using a "first sample" reference sample to initialize the coding of each vector, we could instead use the corresponding sample from a vector in a previous plane. Specifically, in Fig. 8, the last component of $\tilde{v}_{i,\ell}$ at point B could be used to predict the first component of $\tilde{v}_{i,\ell+1}$ at point C, and the component at point D could be used to predict the component at E, and so on.

But what about the vector masks? The component at point D might not be available to predict the component at E. Instead, if the vector $\tilde{v}_{1,\ell}$ is available, point A could be used (perhaps with slightly less effect since it is farther away in x and λ), as shown by the dashed line.

Implementation for Mask. To account for the vector mask, three registers are used to implement a previous plane predictor. Each has 17 storage positions corresponding to the 17 detectors and hence the 17 possible vectors in each data plane. They are labelled here and in Fig. 9 as TOPr, BOTTOMr and COUNTr.

TOPr and BOTTOMr hold the values to be used for prediction and COUNTr determines if these stored values will actually be used for that purpose.

When the coding of a particular plane is complete, all the top components of vectors that have been transmitted (their mask value in Eq. 9 is non-zero) are stored in the corresponding position of TOPr. Similarly, the bottom component of the same vectors are loaded into BOTTOMr.

Then, it should be clear that these registers always contain the top and bottom components of the most recently transmitted vectors,

All components of the COUNTr register are initially set to zero before any transmission occurs, as

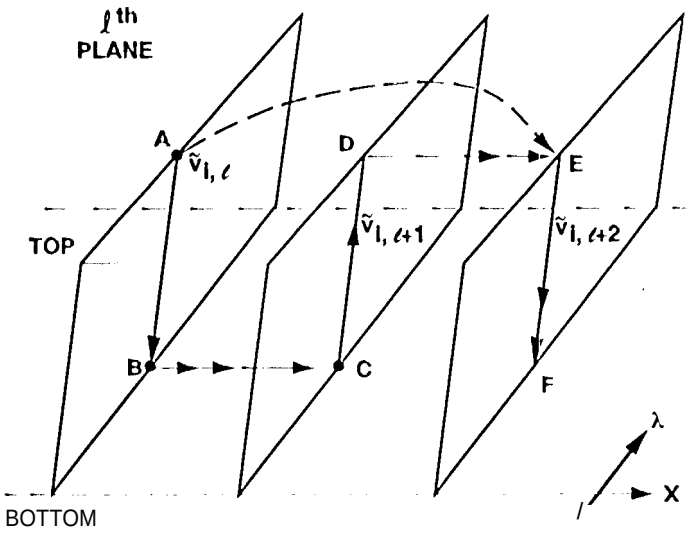


Fig. 8. Adjacent Planes

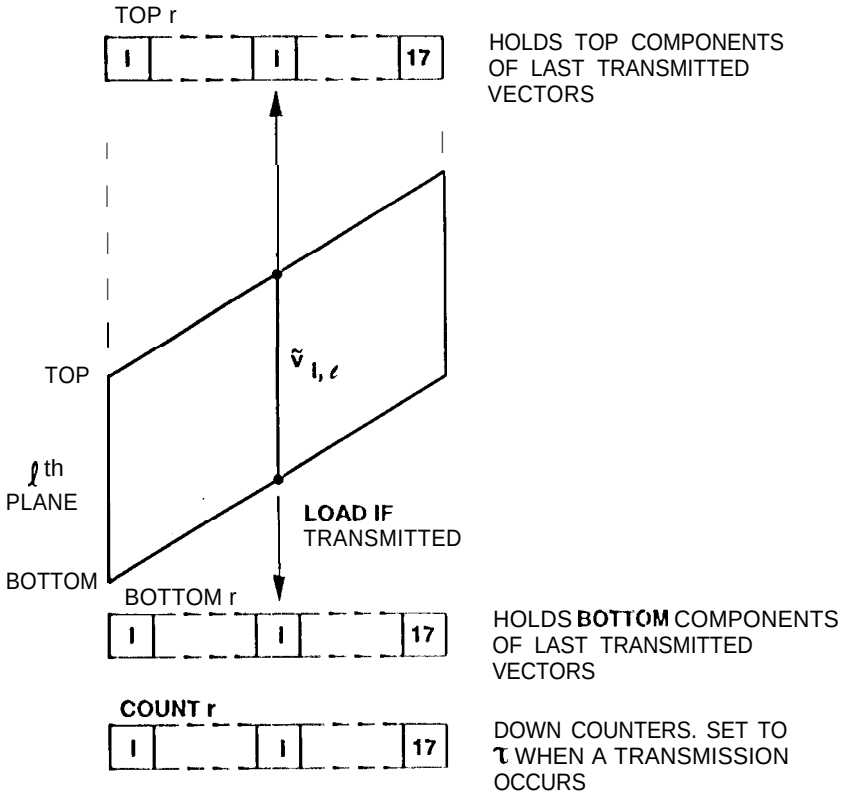


Fig. 9. Registers for Previous Plane Prediction

$$COUNTr(i) = 0 \text{ for } 1 \le i \le 17 \tag{30}$$

But when the ith vector in any plane is transmitted (its mask in non-zero), COUNTr(i) is set to a programmable count-down parameter τ

$$COUNTr(i) = \tau \tag{31}$$

If the ith vector in any plane is NOT transmitted, then

$$COUNTr(i) = \max \{COUNTr(i) - 1, O\} \tag{32}$$

Thus when the coding of a particular plane begins

$\tau + 1$-- COUNTr(i) (33)

specifies, for the ith vector, how many planes back the last transmitted vector occurred, And TOPr and BOTTOMr contain the top and bottom components of those vectors.

The coding of the ℓth plane is specified by the expression in (29) if we simply identify the parameter $\zeta_{i,\ell}$ for each i. This is obtained from COUNTr as

$$\zeta_{i,\ell} = \begin{cases} 1 \text{ if COUNTr} > 0 \\ 2 \text{ if COUNTr} = 0 \end{cases} \quad (34)$$

We see from Eqs. 28, 29 and 34 that the ith vector will be coded using coder $NIMS_2$ whenever COUNTr = O. By Eqs. 24 and 27, $NIMS_2$ is a coder that uses the first component in a vector (either the top or bottom component) as a reference sample. It does not make use of any information from other vectors. Except for initialization situations, this occurs when the last ith vector transmitted occurred more than τ planes before (see Eq. 32).

Conversely, the ith vector will be coded using $NIMS_1$ whenever COUNTr(i) ≠ O. $NIMS_1$ uses the top and bottom component of the last transmitted ith vector (stored in TOPr and BOTTOMr) to predict the first sample of the current ith vector. No reference sample is needed.

The settings for programmable countdown parameter τ are likely to vary for the various modes of operation. At this time, only a range of τ has been established. But the motivation should be clear:

Basically, τ establishes the maximum separation between data planes where correlation between samples can be expected to improve coding efficiency. If τ is set to zero, $NIMS_2$ (which uses a reference sample) will be used all the time. As τ is increased, predictions of initial vector components from as far away as τ planes is allowed (by 32 and 34). The setting for τ should be increased as long as it improves the expected coding performance. At this time such data is not available.

Packetization

The LGA Mission will employ packet telemetry to package blocks of compressed or uncompressed data from the various instruments, Slight modifications from NASA Standards were necessary.[14] For Galileo specific details on this subject, consult Ref. 15 by Kuo and Amador.

The packet telemetry format structure for NIMS can be summarized as in Fig, 10.

Packets. Starting at the top of Fig. 10, all data is placed into data packets that consist of a Primary Header which identifies the data and a data field containing data and an optional Secondary Header. The Primary Header is a fixed 7 bytes in length and includes information about the length of the data field which can be variable but must be less than 512 bytes. To reduce the penalty of packet header overhead, it is desirable to keep the packet size (for all instruments) as large as possible.

VCDU. Packets are placed into another data structure called a Virtual Channel Data Unit (VCDU). A VCDU here is a fixed 446 bytes in length, including a 4 byte header. The header contains a pointer to the start of the first packet that does not start at the beginning of the VCDU.

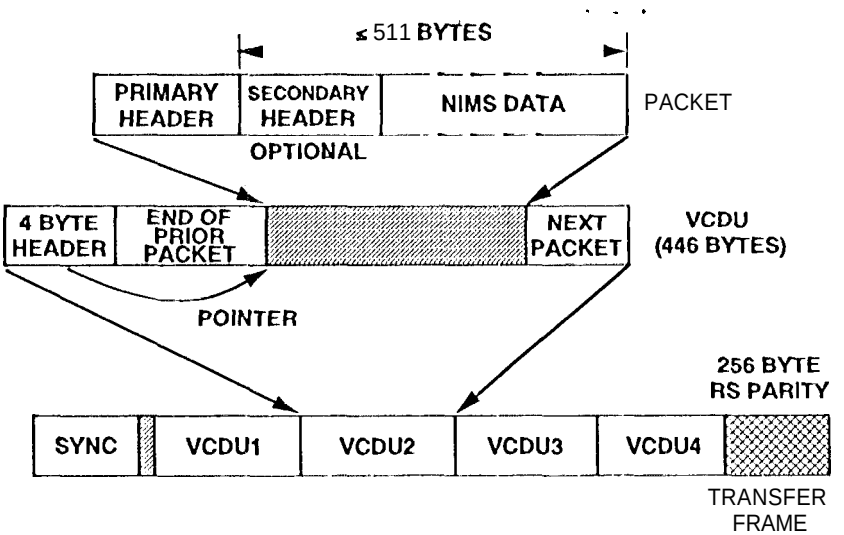


Fig. 10. Packetization

Transfer Frames, RS Coding. Four VCDUs make up a Transfer Frame that is preceeded by a Synchronization Marker. To protect each Transfer Frame from communication errors, 256 bytes of Reed-Solomon (RS) coder parity symbols follow. Unlike the J = 8, E = 16, I = 2 hardware RS coder implemented in conjunction with BARC[7], this software implemented RS coder employs higher interleave depth (1) and a variable (per codeword) error correction capability (E). This new structure maximizes the performance advantages obtained from a more powerful software replacement to the baseline Galileo K = 7, v = 2 convolutional code.[16] For additional information on NASA telemetry standards, consult Ref. 17.

Error Containment. Variable length, compressed data discussed here is very "error-sensitive". A single error in compressed data could disrupt the decompression process until it can be restarted. This is often called "error propagation". The concatenated channel noted above should produce high-performance, virtually error-free communication from spacecraft to ground. The occurrence of any errors should be only a *remote* possibility. However, there is concern for the impact of other error sources which might cause the loss of complete 1 ransfer Frames. T bus, to avoid these unlikely but catastrophic events, a method was needed to assure that each error-free data packet could be independently decoded (decompressed).

For other instruments which employ the Rice Algorithms the solution was to begin each packet with one or more reference samples, as needed. The impact of any error would therefore be confined to a packet.

For NIMS, two modes were established to deal with the error containment issue If it materializes. Under the expectation that the communication link will be error-free, the inclusion of reference samples are rarely forced on the coding process, maximizing coding efficiency. But the effect of an error within one packet could propagate into another,

If such unexpected error events do materialize to cause a problem, the NIMS system will be commanded to switch to a second mode in which every vector would be coded using $NIMS_2(\tau \rightarrow 0)$, incurring the penalty of an excess of reference samples.

Performance

Initial evaluations of coding performance using real and simulated NIMS data demonstrated that compression factors could vary over a range from 1.3:1 up to 4:1, with an expected average of around 2:1. But these results are preliminary since the complete compression system has not yet been fully tested and "real" data will have to wait until the first encounter with Jupiter.

V. LONG-TERM RATE CONTROL

Figure 10 summarizes the general operations and problems associated with a Jupiter encounter, when data is recorded, and the follow-up playback period.

Encounter instrument parameters of Grating Mode, Mirror Scan and Record Vector Masks, rMASK (see Eq. 7), are specified to determine which elements of uncompressed NIMS data are placed on the Tape Recorder,

For the playback period of about 1 month, NIMS is allowed a significant fraction of the expected average downlink data rate during the period, The result is an allocation of $\bar{A}$ bits in Fig. 10. ($\bar{A}$ may change for various reasons, such as unusual weather).

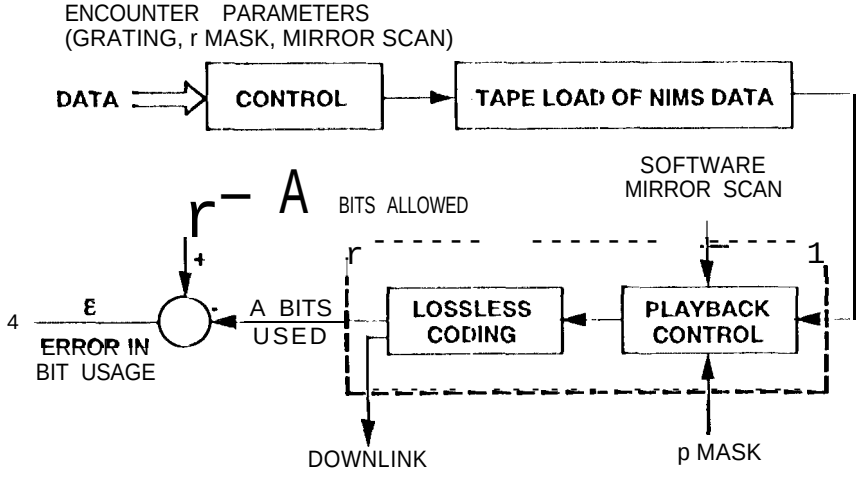


Fig. 10. Encounter/Playback

In general $\bar{A}$ is far smaller than the number of bits stored on the recorder (already a reduction from the maximum possible from the instrument). Significant further reduction must come from the combination of Playback Mask, pMASK (see Eq. 8), adjustments in the size of transmitted vectors (Software Mirror Scan) and the lossless coding.

Without lossless coding, combinations of rMASK and pMASK could be chosen so that a fixed, a priori known A bits would be presented to the downlink as required, But the expected average 2:1 or more compression factor for lossless coding means that fewer vectors need to be deleted, or more vectors can be kept at full length, This can significantly improve the science that can be derived from the returned data.

But unfortunately, the rate generated by an efficient lossless compressor will vary and its true performance characteristics when applied to real encounter data are as yet unknown. This compounds the already difficult "task of assigning pMASK and Software Mirror Scan parameters for playback -- such that science is optimized.

The uncertainty in compressor performance on various data types could result in significantly more or less than $\bar{A}$ bits being generated during the playback cycle. If more than A bits are generated, key segments towards the end of the recorder might be lost. Conversely, fewer than A bits used means that some data was unnecessarily degraded (i. e., pMASK, Software Mirror Scan).

Of course, this uncertainty in compression performance will diminish as experience with real data is obtained. But this is not sufficient to eliminate concern. Hence, this section explores a feedback control approach that uses Software Mirror Scan adjustments to ensure that $\bar{A}$ bits will be used,

Feedback Control Using Mirror Scan

We need some further notation for discussion purposes. Partition the full block of encounter data (or a subset of it - a particular observation) into N major data blocks $D_1 D_2, \ldots . D_N$ and let

$$P_{j(i)} \varepsilon \{P\} \tag{35}$$

be the set of all the combinations of Grating Mode, Vector Mask and Mirror Scan that can apply to the {D,}. Application of any $P_{j(i)}$ to data block D_i acts as a filter on D_i, modifying the number of data planes (Grating), the number of vectors (Mask) and the number of components (Mirror Scan).

The result of applying $P_{j(i)}$ to D_i results in the modified data block

$$P_{j(i)}[D_i] \tag{36}$$

which requires a known and fixed number of bits for representation (we haven't applied the lossless coding yet) given by

$$L_{j(i),i} \text{ bits} \tag{37}$$

1 hen denote by

$$C(") \tag{38}$$

a lossless (major block) coder made up of plane coders of the form in Eq. 29 that applies to any modified data block. Applying C (•) to $P_{i(i)}[D]$ yields the coded sequence

$$C(P_{j(i)}[D_i]) \tag{39}$$

requiring

$$\ell_{j(i),i} \text{ bits} \tag{40}$$

The "compression factor" for block D_i is then

$$cf_{j(i),i} = \frac{L_{j(i),i}}{\ell_{j(i),i}} \tag{41}$$

These issues and definitions are shown in Fig. 11.

But $\ell_{j(i),i}$ is not known ahead of time. It will also vary with parameter choice and data characteristics. For planning purposes, $\ell_{j(i),i}$ can be estimated as

$$\hat{\ell}_{j(i),i} = \frac{L_{j(i),i}}{\widehat{cf}_{j(i),i}} \tag{42}$$

where $\widehat{cf}_{j(i),i}$ if the best a priori estimate of compression factor (when $P_{j(i)}$ is applied to data of the type expected in D,).

The greatest uncertainty in $\hat{\ell}_{j(i),i}$ will occur before experience on real encounter data can be obtained. I-bus as time goes on $\hat{\ell}_{j(i),i}$ will become somewhat more accurate. As a result of preliminary investigations, initial planning will use

$$\widehat{cf}_{j(i),i} = 2 \tag{43}$$

for all i, j.

NIMS planners have the difficult task of choosing the $P_{j(i)}$ such that science return is maximized and

$$\sum_{i=1}^{N} \hat{\ell}_{j(i),i} = \bar{A} = \text{allocated bits} \tag{44}$$

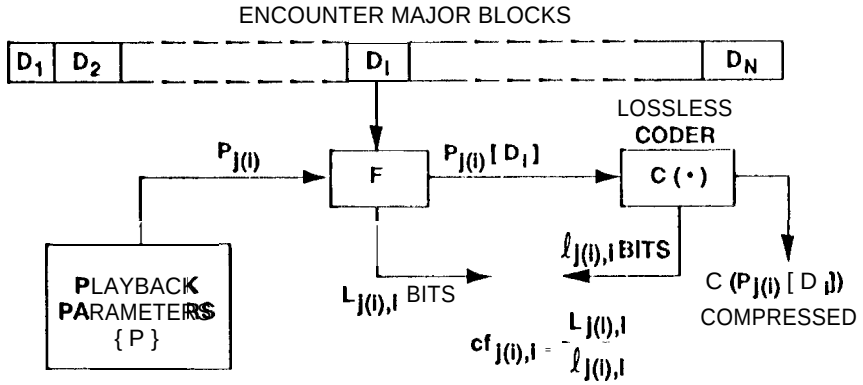


Fig. 11. Compression **at** Major Blocks

Unfortunately, the estimates on bit usage, $\hat{\ell}_{j(i),i}$, may not match the actual number of bits generated, $\ell_{j(i),i}$.

Basic Projected Bit Usage. Now assume that the parameters used for the ith block, D_i, can be altered from the planned parameters as a means of controlling rate. The altered parameters used, and the bits generated before and after compression are indicated by

$$P^*_{j(i),i}\ ,\ L^*_{j(i),i}\ ,\ \ell^*_{j(i),i} \tag{45}$$

respectively.

Then the actual accumulation of data bits through the Kth **block,** D_K, is

$$A(K) = \sum_{i=1}^{K} \ell^*_{j(i),i} = A(K-1) + \ell^*_{j(K),K} \tag{46}$$

for 1 s Ks N and A(0) = O.

The Planned accumulation of bits through the Kth block is similarly

$$\bar{A}(K) = \sum_{i=1}^{K} \hat{\ell}_{j(i),i} = \bar{A}(K-1) + \hat{\ell}_{j(i),i} \tag{47}$$

At the start of the Kth block:

$$\bar{R}(K) = \sum_{i=K}^{N} \hat{\ell}_{j(i),i}\ ,\ 1 \le K \le N = \bar{A} - \bar{A}\ (K-1) \tag{48}$$

represents the remaining number of bits planned to be used (based on planned $P_{j(i),i}$ and a priori compression estimates $\hat{\ell}_{j(i),i}$). The projected number of bits to be used at completion of the last block (if parameters are not changed) is

$$\hat{A}\ (N|K-1) = A(K-1) + R(K) \tag{49}$$

The error in this projection (at the start of the Kth block) is

$$\begin{aligned} \varepsilon(K) &= A - \bar{A}\ (N|K-1) \\ &= \bar{A}\ (K-1) - A\ (K-1) \end{aligned} \tag{50}$$

But note that E(K) is simply an accumulated error that basically assumes that all error has occurred by block D_{K-1}.

Modified Projection. Instead, use the parameters applied in the first K-1 blocks to determine achieved compression factors and use them to improve the projected accumulation when a full N blocks are completed.

Assume that adjustments are made to the parameters used on the first K--1 blocks to control rate, and let

$$\lambda\ (K-1) = \sum_{i=1}^{K-1} \hat{\ell}^*_{j(i),i} = \sum_{i=1}^{K-1} \frac{L^*_{j(i),i}}{\hat{cf}_{j(i),i}} \tag{51}$$

be the predicted bit usage up to the Kth block where $\hat{cf}_{j(i),i}$ is the a priori assumption for compression factors using the parameters actually used. But this can be written as

$$\lambda(K-1) = \frac{\sum_{i=1}^{K-1} L^*_{j(i),i}}{\hat{cf}_p} \tag{52}$$

where $\hat{cf}_p$ is an average compression factor.

The correct actual accumulation of bits is from (46)

$$A(K-1) = \frac{\sum_{i=1}^{K-1} L^{*}_{j(i),i}}{cf_a}$$

$$= \left(\frac{cf_p}{cF_a}\right) \lambda (K-1) \qquad (54)$$

We can similarly modify the projected bit usage in Eq. 49 to

$$A'(N|K-1) = A(K-1) + \frac{cf_p}{cf_a} \bar{R}(K)$$

which from Eq. 54 reduces to

$$A'(N|K-1) = A(K-1) + \frac{A(K-1)}{\lambda(K-1)} \bar{R}(K) \qquad (55)$$

The error in this projection (at the start of the Kth block) is, from Eq. 50 and Eq. 55

$$e(K) = \bar{A} - A(K-1) \left(1 + \frac{D_K}{\lambda(K-1)}\right) \qquad (56)$$

This result should be much more accurate in assessing the error in projections of accumulated bits.

Adjusting Rate. For science reasons, the only allowable change in parameters resulting from on-board autonomous control is the Software Mirror Scan.* Increasing or decreasing the Mirror Scan means to increase or decrease the number of components in all vectors of a large data block, D_i. Available values for the number of vector components are 20 (maximum), '16, 12, 8 and 4.

Basically, the rules for rate control at the start of a major block D_K are quite simple

1) If $\varepsilon(K) > a+$, adjust Mirror Scan for lower rates (57)

2) If $F(K) < a-$, adjust Mirror Scan for higher rates (58)

where thresholds α^+ and a could also be programmed for each K to reflect the expected importance of the planned segments.

Actual on-board autonomous adjustments in NIMS Mirror Scan to control rate are expected to be based on the error projection in Eq. 56 or the simpler approach in Eq. 50. But note that this same structure can be duplicated and expanded on by earth-based computers which do not suffer from the same computation/memory constraints. In this case, tables of compression factors could be automatically maintained on earth for each possible target type and instrument parameter set, making projections of bit usage progressively more accurate as the mission proceeded, Appropriate control information could be periodically communicated to the spacecraft over the (typical) one-month playback period.

* It is net desirable to vary the spectral coverage of an observation.

ACKNOWLEDGEMENTS

The authors wish to thank several individuals for their support of the efforts described herein: to William Smythe for his technical inputs on science requirements and instrument subtleties; to Alan Schlutsmeyer for his general purpose Rice Algorithm C code which formed the basis of data compression investigations for NIMS and several other Galileo instruments; to Ron Burns for his timely and compact 1802 machine code implementation of these algorithms that demonstrated flight feasibility to the Galileo project.

The Research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and the University of California at Los Angeles, under contracts with the National Aeronautics and Space Administration.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government, the Jet Propulsion Laboratory, California Institute of Technology, or the University of California at Los Angeles.

REFERENCES

1) R.W. Carlson, "Spectral Mapping of Jupiter and the Galilean Satellites in the Near Infrared", SPIE Journal on Imaging Spectroscopy, Vol. 268, p. 29-34, 1981.

2) R.W. Carlson, et al., "Near-infrared Mapping Spectrometer Experiment on Galileo", Space Science Reviews 60, pp. 457-502, 1992.

3) W.D. Smythe, et al., "Galilean Satellite Observation Plans for the Near Infrared Mapping Spectrometer Experiment on the Galileo Spacecraft," Journal of Geophysical Research, (to be published).

4) W. O'Neil, "Galileo, Challenges En Route to Jupiter," Proceedings of the 16th Annual AAS Guidance and Control Conference, Keystone, Colorado, Feb. 6-10, 1993.

5) Steve Synnott, Private Communication, Jet Propulsion Laboratory, Pasadena, California.

6) R. Rice, T. Duxbury, C. Devda, A. Schlutsmeyer, R. Weidner, "Data Compression for Optical Navigation", Interactive Graphics demonstration to NASA Headquarters (internal document), Jet Propulsion Laboratory, Pasadena, California, August 26, 1981.

7) R.F. Rice, et al., 'Block Adaptive Rate Controlled Image Data Compression," Proceedings of the 1979 National Telecommunications Conference, Washington, D. C., November 1979,

8) K.-M. Cheung, F. Pollara, and M. Shahshahani, "Integer Cosine Transform for Image Compression," The TDA Progress Report 42-105: January - March, 1991, Jet Propulsion Laboratory, Pasadena, California, May 15, 1991, pp. 45-53.

9) R.F. Rice, "Some Practical Universal Noiseless Coding Techniques, Part Ill, Module PSI14,K+," JPL Publication 97-3, Jet Propulsion Laboratory, Pasadena, California, November 15, 1991,

10) R.F. Rice, et al., "Algorithms for High-Speed Universal Noiseless Coding", Proceedings of the AIAA Computing in Aerospace 9 Conference, San Diego, California, October 19-21, 1993.

11) J. Venbrux, et al., "A VLSI Chip Set for High-Speed Lossless Data Compression," IEEE Transactions on Circuits and Systems for Video Technology, Vol. ?, No. 4, December 1992.

12) Pen-shu Yeh, et al., "On the Optimality of a Universal Noiseless Coder," Proceedings of the AIAA Computing in Aerospace 9 Conference, San Diego, California, pp. 490-498, October 19-21, 1993.

13) D. A. Huff man, "A Method for the Construction of Minimum Redundancy Codes," Proc. IRE, Vol. 40, pp. 1098-1101, 1952.

14) "Packet Telemetry," Blue Book, CCSDS 102.O-B-3, Nov. 1992.

15) N. Kuo, A. Amador, "Phase II Ground System Level 3 Requirements," (Internal document) MOS-GLL-3-200B, Appendix B, Jet Propulsion Laboratory, Pasadena, California, Nov. 5, 1993.

16) S. Dolinar and M. Belongie, "Enhanced Decoding for the Galileo S-Band Mission, " The Telecommunications and Data Acquisition Progress Report 42-1 14; April - June, 1993, Jet Propulsion Laboratory, Pasadena, California, August 15, 1993, pp. 96-111.

17) "Telemetry Channel Coding," Blue Book, CCSDS 101 ,0-B-3, May 1992,