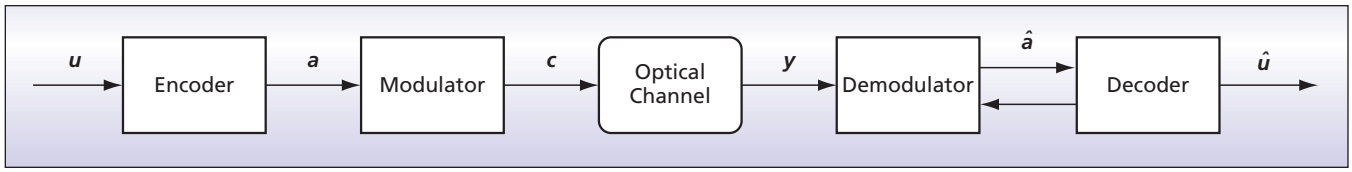

Figure 1. **Data Are Encoded**, then transmitted as a PPM optical signal. At the receiving end, the optical signal is demodulated and decoded in an iterative process.

ther the present LDPC-PPM scheme or the prior SCPPM scheme. At the transmitting terminal, the original data (u) are processed by an encoder into blocks of bits (a), and the encoded data are mapped to PPM of an optical signal (c). For the purpose of design and analysis, the optical channel in which the PPM signal propagates is modeled as a Poisson point process. At the receiving terminal, the arriving optical signal (y) is demodulated to obtain an estimate ($\hat{a}$) of the coded data, which is then processed by a decoder to obtain an estimate ($\hat{u}$) of the original data.

The demodulation and decoding subprocesses are iterated to improve the final estimates in an attempt to reconstruct the original data stream (u) exactly. The decoder implements a soft-input/soft-output (SISO) algorithm. This or any SISO decoder receives, as soft inputs, noisy versions (estimates and log-likelihoods of the estimates) of the input and output of the encoder and produces updated log-likelihoods of the estimates of the input, the output, or both. These estimates and their log-likelihoods may then be transmitted to other SISO modules in the receiver, where they are treated as noisy inputs.

In comparison with non-iterative alternatives, both the present LDPC-PPM scheme and the prior SCPPM scheme offer better performance. In comparison with iterative alternatives, both schemes afford better performance with less complexity. In comparison of these schemes with each other, each is partly advantageous and partly disadvantageous: For example, computational simulations have shown that for a block length of about 8Kb, the performance of the prior SCPPM scheme is about 0.8 dB away from the theoretical channel capacity, while the performance of the LDPC-PPM scheme is expected to be about 1.2 dB away from the theoretical channel capacity at a bit-error rate of about 2×10^{-5} (see Figure 2); in other words, the performance of the LDPC-PPM scheme is expected to be about 0.4 dB below that of the prior SCPPM scheme. On the other hand, unlike the prior SCPPM scheme, the LDPC-PPM scheme is lends itself very well to low-latency parallel processing. Either scheme could serve as the basis of design of an optical communication system, depending on requirements pertaining to the PPM order, latency, and architecture of the system.

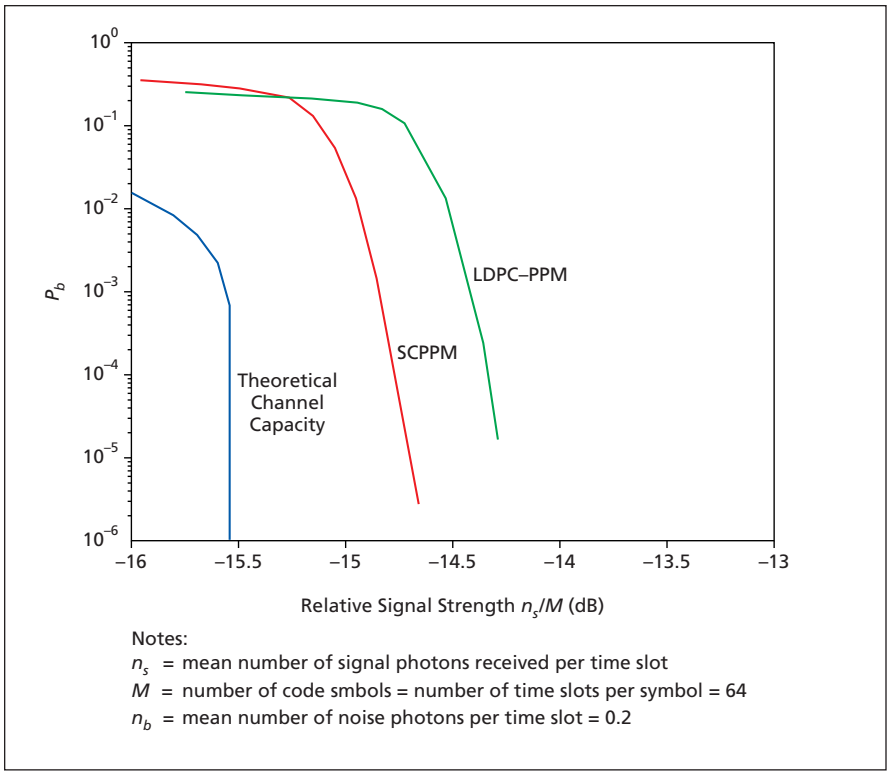

Figure 2. The **Bit-Error Rate (P_b)** was computed as a function of relative signal strength for two coding schemes and for the theoretical channel capacity for the special case of code blocks of ≈8Kb length, n_b = 0.2, and M = 64.

This work was done by Maged Barsoum, Bruce Moision, Dariush Divsalar, and Jon Hamkins of Caltech and Michael Fitz of UCLA for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1). ***NPO-44408***

Complex Event Recognition Architecture

Lyndon B. Johnson Space Center, Houston, Texas

"Complex Event Recognition Architecture" ("CERA") is the name of a computational architecture, and software that implements the architecture, for recognizing complex event patterns that may be spread across multiple streams of input data. One of the main components of CERA is an intuitive event pattern language that simplifies what would otherwise be the complex, difficult tasks of creating logical descriptions of combinations of temporal events and defining rules for combining information from different sources over time. In this language, recognition

patterns are defined in simple, declarative statements that combine point events from given input streams with those from other streams, using conjunction, disjunction, and negation. Patterns can be built on one another recursively to describe very rich, temporally extended combinations of events. Thereafter, a run-time matching algorithm in CERA efficiently matches these patterns against input data and signals when patterns are recognized.

CERA can be used to monitor complex systems and to signal operators or initiate corrective actions when anomalous conditions are recognized. CERA can be run as a stand-alone monitoring system, or it can be integrated into a larger system to automatically trigger responses to changing environments or problematic situations.

This program was written by William A. Fitzgerald and R. James Firby of I/NET, Inc. for Johnson Space Center. Further information is contained in a TSP (see page 1).

In accordance with Public Law 96;517, the contractor has elected to retain title to this invention. Inquiries concerning rights for its commercial use should be addressed to:

I/NET
P. O. Box 3338
Kalamazoo, MI 49003
Phone No.: (269) 978-6816
Fax No.: (800) 673-7352

Refer to MSC-23637-1, volume and number of this NASA Tech Briefs *issue, and the page number.*

TurboTech Technical Evaluation Automated System

Goddard Space Flight Center, Greenbelt, Maryland

TurboTech software is a Web-based process that simplifies and semiautomates technical evaluation of NASA proposals for Contracting Officer's Technical Representatives (COTRs). At the time of this reporting, there have been no set standards or systems for training new COTRs in technical evaluations. This new process provides boilerplate text in response to "interview-style" questions. This text is collected into a Microsoft Word document that can then be further edited to conform to specific cases.

By providing technical language and a structured format, TurboTech allows the COTRs to concentrate more on the actual evaluation, and less on deciding what language would be most appropriate. Since the actual word choice is one of the more time-consuming parts of a COTRs' job, this process should allow for an increase in quantity of proposals evaluated.

TurboTech is applicable to composing technical evaluations of contractor proposals, task and delivery orders, change order modifications, requests for proposals, new work modifications, task assignments, as well as any changes to existing contracts.

This work was done by Dorothy J. Tiffany of Goddard Space Flight Center. Further information is contained in a TSP (see page 1).
GSC-15554-1

Robot Vision Library

NASA's Jet Propulsion Laboratory, Pasadena, California

The JPL Robot Vision Library (JPLV) provides real-time robot vision algorithms for developers who are not vision specialists. The package includes algorithms for stereo ranging, visual odometry and unsurveyed camera calibration, and has unique support for very wide-angle lenses (as used on the Mars Exploration Rover HazCams). JPLV gathers these algorithms into one uniform, documented, and tested package with a consistent C API (application programming interface). The software is designed for real-time execution (10–20 Hz) on COTS (commercial, off-the-shelf) workstations and embedded processors.

This package incorporates algorithms developed over more than ten years of research in ground and planetary robotics for NASA, DARPA (Defense Advanced Research Projects Agency) and the Army Research Labs, and is currently being used in applications as diverse as legged vehicle navigation and large-scale urban modeling.

This work was done by Andrew B. Howard, Adnan I. Ansar, and Todd E. Litwin of Caltech and Steven B. Goldberg of Indelible Systems for NASA's Jet Propulsion Laboratory.

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-46532.

Perl Modules for Constructing Iterators

Goddard Space Flight Center, Greenbelt, Maryland

The ***Iterator*** Perl Module provides a general-purpose framework for constructing iterator objects within Perl, and a standard API for interacting with those objects. Iterators are an object-oriented design pattern where a description of a series of values is used in a constructor. Subsequent queries can request values in that series. These Perl modules build on the standard ***Iterator*** framework and provide iterators for some other types of values.

Iterator::DateTime constructs iterators from ***DateTime*** objects or ***Date::Parse*** descriptions and ICal/RFC 2445 style re-