small body, all within the law of gravity and the solar radiation pressure. The same is true for a horizontal hover. A PatchPoint is an LTool class that denotes a space-time event with some extra information for differential correction, including a set of constraints to be satisfied by T-LDC. Given a set of PatchPoints, each with its own constraint, the T-LDC differentially corrects the entire trajectory by connecting each trajectory leg joined by PatchPoints while satisfying all specified constraints at the same time.

Vertical and horizontal hover both are needed to minimize delta-v spent for station keeping. A Python I/F to NPOPT has been written to be used from an LTool script. In vertical hovering, the spacecraft stays along the line joining the Sun and a small body. An instantaneous delta-v toward the anti-Sun direction is applied at the closest approach to the small body for station keeping. For example, the spacecraft hovers between the minimum range (2 km) point and the maximum range (2.5 km) point from the asteroid 1989ML. Horizontal hovering buys more time for a spacecraft to recover if, for any reason, a planned thrust fails, by returning almost to the initial position after some time later via a near elliptical orbit around the small body. The mapping or staging orbit may be similarly generated using T-LDC with a set of constraints. Some delta-v tables are generated for several different asteroid masses.

This work was done by Min-Kun J. Chung of Caltech 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-44452.

Efficient Model-Based Diagnosis Engine

A system as large as several thousand components can be diagnosed efficiently.

NASA's Jet Propulsion Laboratory, Pasadena, California

An efficient diagnosis engine — a combination of mathematical models and algorithms — has been developed for identifying faulty components in a possibly complex engineering system. This model-based diagnosis engine embodies a twofold approach to reducing, relative to prior model-based diagnosis engines, the amount of computation needed to perform a thorough, accurate diagnosis. The first part of the approach involves a reconstruction of the general diagnostic engine to reduce the complexity of the mathematical-model calculations and of the software needed to perform them. The second part of the approach involves algorithms for computing a minimal diagnosis (the term "minimal diagnosis" is defined below).

A somewhat lengthy background discussion is prerequisite to a meaningful summary of the innovative aspects of the present efficient model-based diagnosis engine. In model-based diagnosis, the function of each component and the relationships among all the components of the engineering system to be diagnosed are represented as a logical system denoted the system description (SD). Hence, the expected normal behavior of the engineering system is the set of logical consequences of the SD. Faulty components lead to inconsistencies between the observed behaviors of the system and the SD (see figure). Diagnosis — the task of finding faulty components — is reduced to finding those components, the abnormalities of which could explain all the inconsistencies. The solution of the diagnosis problem should be a minimal diagnosis, which is a minimal set of faulty components. A minimal diagnosis stands in contradistinction to the trivial solution, in which all components are deemed to be faulty, and which, therefore, always explains all inconsistencies.

The general diagnosis engine (GDE) is widely used in the discipline of automated diagnosis. The GDE combines a model of each component of an engineering system with observations of the actual behavior of the component to detect discrepancies and diagnose root causes. The GDE uses an inference engine to compute the consequences of observations and uses an assumption-based truth maintenance system (ATMS) to manage the assumptions underlying each computation. One of the side effects of managing the assumptions is the detection of inconsistent sets of assumptions, which leads to conflict sets used in calculating minimal diagnoses. Unfortunately the GDE has two major limitations:

- The combination of the inference engine and ATMS must be represented by software that is so complex that the use of the GDE is too difficult and impractical for many complex engineering systems.
- The calculation of a minimal diagnosis is inherently a hard problem. Using typical prior algorithms, the conversion from conflict sets to a minimal diagnosis requires amounts of computation time and memory that increase exponentially with the number of components of the engineering system.

This concludes the background discussion.

In the present efficient model-based diagnosis engine, the first-mentioned limitation of the GDE is overcome by the reconstructed general diagnostic engine (RGDE). Like the GDE, the RGDE combines a model of each component of an engineering system (represented graphically as a network) with observations of the actual behavior of the component to detect discrepancies and diagnose root causes. Also like the GDE, the RGDE performs a causal simulation by taking variable observations and using rules to compute the values of other variables in the network.

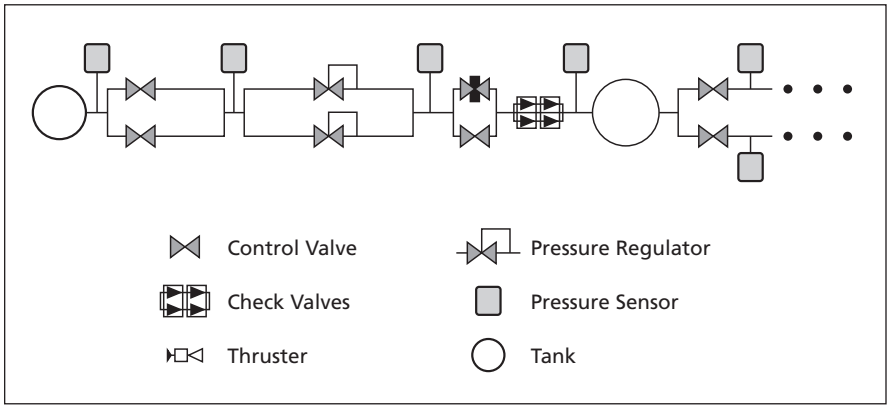
Although assumptions underly the computations in the RGDE as in the GDE, the RGDE does not include an ATMS. Instead, taking advantage of the discovery that the ATMS and the inference engine have many similarities, the RGDE combines the ATMS with the inference engine to simplify the diagnosis-engine algorithm and the software that implements it. In this approach, the value of each variable is tagged with the set of assumptions that contribute to its computation. This set of tags comprises the collective union of the tags of values that feed into the computation with a tag representing the computation itself. A discrepancy arises when two incompatible values are assigned to the same variable. In general, whenever the RGDE computes two incompatible values for the same variable, the union of the two supporting assumption sets is incompatible; that is, it is a conflict set. Typically in the course of causal simulation, no discrep-

ancies are found, but when failures occur, multiple incompatible assumption sets appear. This process continues to determine new incompatible sets until the causal simulation is completed.

The second-mentioned limitation of the GDE is overcome by a combination of two methods, embodied in algorithms that, relative to prior algorithms used for the same purpose, require less computation to arrive at a minimal diagnosis. These methods and algorithms were described in "Fast Algorithms for Model-Based Diagnosis" (NPO-30582) and "Two Methods of Efficient Solution of the Hitting-Set Problem" (NPO-30584), both published in *NASA Tech Briefs*, Vol. 29, No. 3, March 2005, pages 69 and 74, respectively. To recapitulate: One of the two improved methods is based on mapping of the diagnosis problem onto the Boolean satisfiability problem. This mapping makes it possible to utilize Boolean function theory to reduce the diagnosis problem to the prime-implicant problem (one of the problems in the theory). This, in turn, makes it possible to utilize very efficient algorithms, developed previously for the satisfiability problem, to compute the minimal diagnosis. The algorithm thus developed to solve the diagnosis problem requires an amount of computation proportional to a superpolynomial function of n (meaning that the computation time is proportional to $n^{\ln(n)}$), where n is the number of components of the engineering system.

The other improved method is based on the mapping of the diagnosis problem onto the integer-programming problem. This mapping makes it possible to utilize a variety of algorithms developed previously for integer programming to solve the diagnosis problem. In the integer-programming approach, the diagnosis problem can be formulated as a linear integer optimization problem, which can be solved by use of well-developed integer-programming algorithms. Some of these algorithms, modified to make them suitable for solving the diagnosis problem, can efficiently diagnose a system that contains as many as several thousand components.

The development of this efficient model-based diagnosis engine has been accompanied by the derivation of a deep matrix analysis of the integer programming problem that makes it possible to extract bounds for the sizes of the solutions of the optimization problem, without solving the problem explicitly. This analysis could be helpful in the development of algorithms that would be much more efficient for solving specific problems.



A **Relatively Simple Engineering System** of tanks, valves, pipes, and pressure sensors serves to illustrate the basic diagnostic principle. This system would be diagnosed by comparing actual and expected values of pressure-sensor readings as correlated with commanded openings and closings of the valves.

This work was done by Amir Fijany, Farokh Vatan, Anthony Barrett, Mark James, Ryan Mackey, and Colin Williams of Caltech for NASA's Jet Propulsion Laboratory. 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:

Innovative Technology Assets Management
JPL
Mail Stop 202-233
4800 Oak Grove Drive
Pasadena, CA 91109-8099
E-mail: iaoffice@jpl.nasa.gov

Refer to NPO-40544, volume and number of this NASA Tech Briefs *issue, and the page number.*

DSN Simulator

NASA's Jet Propulsion Laboratory, Pasadena, California

The DSN Simulator (wherein "DSN" signifies NASA's Deep Space Network) is an updated version of the software described in "DSN Array Simulator" (NPO-44506), *Software Tech Briefs* (Special supplement to *NASA Tech Briefs*), Vol. 32, No. 9 (September 2008), page 26. To recapitulate: This software is used for computational modeling of proposed DSN facilities comprising arrays of antennas and transmitting and receiving equipment for microwave communication with spacecraft on interplanetary missions. Such modeling is performed to estimate facility performance, evaluate requirements that govern facility design, and evaluate proposed improvements in hardware and/or software. The software includes a Monte Carlo simulation component that enables rapid generation of key mission-set metrics (e.g., numbers of links, data rates, and data volumes), and statistical distributions thereof as functions of time.

The prior version of the software could model only one DSN facility at a time and included hard-coded, unconfigurable metrics. The present updated version is capable of modeling the entire DSN and provides for configurable metrics, making it possible to perform loading analyses for alternative future DSN architectures and mission-set scenarios. The present version also features an improved user interface and interfaces for exchange of data with other DSN software and with a DSN mission model database.

This program was written by Ryan M. Mackey and Raffi P. Tikidjian of Caltech 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-45513.