Real Automation in the Field

César Muñoz
ICASE, Hampton, Virginia

Micaela Mayero
INRIA, Le Chesnay Cedex, France

December 2001
The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA’s scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA’s institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA’s counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATIONS. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA’s mission.

Specialized services that complement the STI Program Office’s diverse offerings include creating custom thesauri, building customized data bases, organizing and publishing research results . . ., even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at http://www.sti.nasa.gov

- Email your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Telephone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320
Real Automation in the Field

César Muñoz  
ICASE, Hampton, Virginia

Micaela Mayero  
INRIA, Le Chesnay Cedex, France

ICASE
NASA Langley Research Center  
Hampton, Virginia

Operated by Universities Space Research Association

National Aeronautics and  
Space Administration

Langley Research Center  
Hampton, Virginia 23681-2199

Prepared for Langley Research Center  
under Contract NAS1-97046

December 2001
Available from the following:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 487-4650
REAL AUTOMATION IN THE FIELD*

CÉSAR MUÑOZ† AND MICAELA MAYERO‡

Abstract. We provide a package of strategies for automation of non-linear arithmetic in PVS. In particular, we describe a simplification procedure for the field of real numbers and a strategy for cancellation of common terms.

Key words. non-linear arithmetic, PVS strategies, real number automation

Subject classification. Computer Science

1. Introduction. While conducting research on the formal safety analysis of Air Traffic Management (ATM) systems at the NASA Langley Research Center [2, 3, 6], we have found the verification system PVS [7] very well suited for specifying avionics systems. Avionics systems are hybrid systems, that is, they are digital systems that interact with the continuous physical environment where they are deployed. These two levels of dynamics, discrete and continuous, can be easily modeled using PVS higher-order logic. Despite that, deductive methods, such as theorem proving, are not usually the first approach for modeling and verification of hybrid systems. Most of the literature on verification of hybrid systems concerns variations of algorithmic methods, such as model checking, on restricted sub-classes of the general problem. This can be explained by the fact that although theorem provers based on higher-order logic, e.g., PVS [7], Coq [1], Isabelle-HOL [8], integrate expressive specification languages with powerful theorem provers, they lack of adequate support for continuous mathematics.

The PVS system, for instance, incorporates a rich type system supporting predicate sub-typing and dependent types. It also comes with a set of decision procedures that solve problems in a broad range of decidable fields. Moreover, it includes in the prelude library a large collection of properties for real numbers (for example, real.props and real.axioms) that are applied automatically by the decision procedures. This is true except for non-linear ones, i.e., those involving multiplication or division of non-numeric terms. Indeed, the PVS command (GRIND :theories "real.props"), although useful in several cases, fails to discharge a simple non-linear formula such as

\[
\frac{a + 1}{b+1} - b = 1
\]

where \( a, b \) are positive real numbers. In [9], Rushby and Shankar show how a small augmentation of the PVS prelude library and a deep knowledge of the PVS rewriting mechanism can provide effective automatic arithmetic simplification. However, a lack of good documentation of PVS rewriting features, forbid the access of all these capabilities to a normal user.

Manual proofs of formulas involving non-linear arithmetic can be tedious. It usually requires the ability to recognize by heart the names of the lemmas in the prelude library. It may help that some names in this

---

*This work was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-97046 while the authors were in residence at ICASE, NASA Langley Research Center, Hampton, VA 23681-2199, USA.
†Main and contact Author: ICASE, Mail Stop 132C, NASA Langley Research Center, Hampton, VA 23681-2199, USA, e-mail: munoz@icase.edu.
‡INRIA, Domaine de Voluceau - Rocquencourt - B.P. 105, 78153 Le Chesnay Cedex, France, e-mail: Micaela.Mayero@inria.fr.
library are mnemonics. For example, to check Formula 1.1 we could use Lemma div.div1, which states
\[
\frac{x}{n0y} = \frac{x \times n0z}{n0y}
\]
where \(n0y\) and \(n0z\) denote real numbers different from zero. After applying this lemma on Formula 1.1, via (REWRITE "div.div1"), we get the formula
\[
(1.2) \quad \frac{1 + b + (a \times b + a)}{1 + a} - b = 1
\]
At this point, we may want to rearrange Formula 1.2. In general, this is done with the proof command CASE. However, in this simple example, we add \(b\) to both sides of the equality using the command BOTH-SIDES, and finish the proof with (ASSERT).

The complete proof of Formula 1.1 is a couple of lines long and none of the steps is really difficult. However, the steps become the more a more involved when checking large formulas. Consider for example Formula 1.3, taken from the verification of a conflict detection and resolution algorithm developed at ICASE - NASA Langley [6]. The first naive attempt to verify this formula, under the assumption that it is well-defined, was over a hundred of lines.

\[
(1.3) \quad \frac{v_{ix} \times \sqrt{s^2 - D^2} - v_{iz} \times D}{v_{oy} \times \sqrt{s^2 - D^2} - v_{oz} \times D} = \frac{D^2 - s^2}{(s^2 - D^2) \times v_{oy} - D \times v_{oz} \times \sqrt{s^2 - D^2}} \times s \times v_{ox} + v_{ix} / v_{ox}
\]

In [11], Di Vito presents a package of strategies and functions for manipulating arithmetic expressions in PVS. Using this package, we have developed a simplification procedure for the field of real numbers. The strategy, called FIELD, helped us to reduce the proofs of formulas 1.1, 1.2, and 1.3 to just one line. In Section 2, we describe the strategy and give some examples of usage. In addition to FIELD, we have developed strategies for cancellation of common terms, and some support strategies for arithmetic manipulation. These strategies are described in Section 3 and Section 4. Finally, in Section 5, we discuss some issues related to real automation in PVS and we summarize our work. The package is available at:
http://www.icase.edu/~munoz/Field/field.html.

Most of the strategies developed in our package act on relational formulas, that is, arithmetic formulas having the form \(e \text{ \#} f\), where \(\text{\#} \in \{=, <, \leq, >, \geq\}\). Through this paper, for the sake of readability, we have used a \texttt{\LaTeX}-style for writing PVS code.

2. Reals in the Field. FIELD is a simplification procedure for relational formulas on the field of real numbers. It was originally based on the tactic Field of Coq V7 [5], but it has been extensively adapted to cope with PVS idiosyncrasies. The tactic Field of Coq is reflexive, i.e., it uses Coq’s own reduction mechanism to avoid rewriting steps, which yield large proof terms in the system. The tactic Field first translates the formula to an intermediate representation (the reflexive step), then removes all the inverses appearing in the original formula, and finally applies Ring, another reflexive tactic that implements a decision procedure for the ring of real numbers. In contrast, the strategy FIELD of PVS largely depends on decision procedures, which implement Ring capabilities but are not reflexive. It also has been extended to handle inequalities.

2.1. Algorithm. FIELD, applied to the relational formula \(e \text{ \#} f\), is described by the algorithm:

1. Collect all the inverses \(x_1, \ldots, x_n\) appearing as denominators of sub-terms of \(e, f\). If \(n = 0\), apply ASSERT.
2. Assume \(p \neq 0\), where \(p = x_1 \times \ldots \times x_n\).
3. Let \( e' = p \times e \) and \( f' = p \times f \).
4. If \( \cong \) is the equality operator, replace the equality \( e = f \) with the equality \( e' = f' \).
5. If \( \cong \) is in \( \{<, \leq, >, \geq\} \), replace the inequality \( e \cong f \) with the two inequalities (1) \( e' \cong f' \), assuming \( p > 0 \), and (2) \( f' \cong e' \), assuming \( p < 0 \).
6. For each one of the new formulas:
   (a) Cancel inverses in \( e', f' \), and simplify with decision procedures, i.e., ASSERT and GRIND.
   (b) Call FIELD with the resulting formula.

As explained in [5], the recursive call is needed to handle the case of nested divisions. In order to remove inverses, an explicit control on the application of distributive laws is required. Unfortunately, PVS applies simplification rules, including distributive laws, after performing any rewriting step. Note, for example, that the rewrite command on Formula 1.1 yields a rearranged Formula 1.2 where multiplications have been completely distributed. In particular, the common term \((a + 1)\) is no longer available for cancellation. Below, we describe a strategy that blocks the automatic application of distribute laws by introducing new variable names.

2.2. Use and Examples. Following is the description of the strategy and some examples of usage.

The examples refer to the PVS code in Figure 2.1.

**Syntax:** (FIELD &OPTIONAL fnum[1] grind?[T])

**Effect:** Applies a simplification procedure for the field of real numbers on the formula fnum.

**Requirements:** If fnum is a formula in the consequent, it should have the form \( \forall(x_1, \ldots, x_n) : e \cong f \).

Otherwise, it should have the form \( \exists(x_1, \ldots, x_n) : e \cong f \). In both cases \( n \geq 0 \). FIELD uses GRIND to prove that denominators are different from zero. That strategy sometimes falls in an infinite loop.

Using the key grind? = NIL is safer, but less effective.

The command (FIELD) takes care of lemmas ex1 and ex2, which correspond to formulas 1.1 and 1.2, respectively. FIELD also works on inequalities, e.g., the sequence of proof commands (SKOSIMP) (FIELD) discharges Lemma ex3.

Validation of Formula 1.3 requires a little bit more work. First, we import the theory reals@sqrt, which makes part of the NASA Langley PVS library reals. This library is available at http://shemesh.larc.nasa.gov/ftp/larc/PVS-library/pvslib.html.

Finally, we write Formula 1.3 as Lemma kb3d in PVS and prove it with the sequence of commands (SKOSIMP) (FIELD 4). Only one TCC of kb3d has to be proved by hand; it can be easily discharged using the properties in the theory sqrt.

Finally, FIELD can also be used to remove inverses. For example, the sequence of proof commands (SKOLEM 1 ("nx" "ny" "x" "y")) (FLATTEN) (FIELD -1) on Lemma ex4 yields

\[
\{-1\} \ x \times nx - y \times nx = 0
\]

\[
|--------|
\]

\[
[1] \ x = y
\]

To finish this proof, we multiply formula 1 with nx, for example using Di Vito's strategy [11] MULT-BY, and then apply the proof command (ASSERT). We may also recognize that the term nx in formula \{-1\} can be cancelled. We show a strategy for cancellation of common terms in the next section.

3. Cancellation of Common Terms. FIELD is very effective removing inverses, however, by doing that, it tends to introduce common terms as in the case of Lemma ex4 above. CANCEL-BY is a simplification strategy for cancellation of a common term appearing in a relational formula. First, it divides both sides by the common term, then it distributes, and finally, it simplifies the expressions using PVS decision procedures.
a,b : VAR posreal
x,y : VAR real
nx,ny : VAR nzreal

ex1 : LEMMA \[(a+1)/((a+1)/(b+1)) - b = 1\]

ex2 : LEMMA \[(1+b+(a\times b+a))/(1+a) - b = 1\]

ex3 : LEMMA \[x > 1 \Rightarrow \frac{x}{((x-1)\times(x+1))} - 1/((x-1)\times(x+1)) \geq 1/(x+1) - 1\]

ex4 : LEMMA \[x \times nx/ny - y \times nx/ny = 0 \Rightarrow x = y\]

ex5 : LEMMA \[4\times((a+1)\times b) + ((a+1)\times 6\times(a+1)) = -(a+1)\times((x+1)\times 2) \Rightarrow 2\times b + 3\times(a+1) + x + 1 = 0\]

IMPORTING reals@sqrt

t,vix,viy,vox,voy,s : VAR real
D : VAR posreal

kb3d : LEMMA
vox > 0 \& s > D \& s \times vix \times voy - s \times viy \times vox \neq 0 \&
((s \times s - D \times D) \times voy - D \times vox \times sqrt(s \times s - D \times D)) /
(s \times (vix \times voy - vox \times viy)) \times s \times vox \neq 0 \&
voy \times sqrt(s \times s - D \times D) - D \times vox \neq 0
\Rightarrow
(viy \times sqrt(s \times s - D \times D) - vix \times D) /
(voy \times sqrt(s \times s - D \times D) - vox \times D)
= (D \times D - s \times s) /
(((s \times s - D \times D) \times voy - D \times vox \times sqrt(s \times s - D \times D)) /
(s \times (vix \times voy - vox \times viy)) \times s \times vox) +
vix / vox

Fig. 2.1. PVS code for examples of FIELD

3.1. Algorithm. CANCEL-BY, applied to a term \(t\) in a relational formula \(e \equiv f\), is described by the algorithm:

1. Assume \(t \neq 0\).
2. If \(\equiv\) is the equality operator, replace the equality \(e = f\) with the equality \(e/t = f/t\).
3. If \(\equiv\) is in \(\{<,\leq,\geq\}\), replace the inequality \(e \equiv f\) with the two inequalities (1) \(e/t \equiv f/t\), assuming \(t > 0\), and (2) \(f/t \equiv e/t\), assuming \(t < 0\).
4. For each one of the new formulas:
   (a) Distribute \(t\) on expressions \(e\) and \(f\). Let \(e'\) and \(f'\) be the new formulas.
(b) Remove inverses in $e', f'$, and simplify with decision procedures.

### 3.2. Use and Examples

Following is the description of the strategy and some examples of usage.

**Syntax:** (CANCEL-BY fnum expr)

**Effect:** Cancels the expression expr in the formula fnum.

The sequent

\[
\begin{align*}
{-1} & \quad x \times nx - y \times nx = 0 \\
{[1]} & \quad x = y
\end{align*}
\]

coming from Lemma ex4, Figure 2.1, can be discharged with the command (CANCEL-BY -1 "nx").

The commands (SKOLEM 1 ("a" "b" "x")) (FLATTEN) on Lemma ex5, Figure 2.1, yield the sequent

\[
\begin{align*}
{-1} & \quad 4 \times ((a + 1) \times b) + ((a + 1) \times 6) \times (a + 1) = -(a + 1) \times ((x + 1) \times 2) \\
{[1]} & \quad 2 \times b + 3 \times (a + 1) + x + 1 = 0
\end{align*}
\]

It can be discharged with the command (CANCEL-BY -i "(a+1)x2").

### 4. Extra-tegies

In addition to FIELD and CANCEL-BY, we have implemented some general strategies, in a package called extra-tegies, to support automation on real numbers and arithmetic manipulations. This package is complementary to the one developed by Di Vito [11].

#### 4.1. Automation Support

**Syntax:** (GRIND-REALS)

**Effect:** Applies PVS command GRIND with the theories real_props and extra_real_props. The latter one makes part of the package in [11]. GRIND-REALS does no expand definitions.

**Caveat:** Just as GRIND, GRIND-REALS may not terminate.

**Syntax:** (REALS-PROPS OPTIONAL fnums[*])

**Effect:** Applies PVS command AUTO-REWRITE, with the theories real_props and extra_real_props, on the formulas fnums.

**Tip:** If GRIND-REALS does too much, try REAL-PROPS.

**Syntax:** (NAME-DISTRIBS OPTIONAL name fnums[*])

**Effect:** Introduces new names, based on name, to block the automatic application of distribute laws in formulas fnums. If no name is given, new names are automatically generated.

**Usage:** For illustration purposes, consider the sequent

\[
\begin{align*}
{-1} & \quad x \times (x + 1) = y \\
{-2} & \quad y \times (y + 1) = x \\
{[1]} & \quad x \times (x + 2) = y \times (y + 2)
\end{align*}
\]

the command (NAME-DISTRIBS "A" 1) uses "A" to create new names:

\[
\begin{align*}
{-1}, (A1:)& \quad (y + 2) = A12_{--} \\
{-2}, (A1:)& \quad (x + 2) = A11_{--} \\
{[-3]} & \quad x \times (x + 1) = y \\
{[-4]} & \quad y \times (y + 1) = x
\end{align*}
\]
In contrast, \texttt{(NAME-DISTRIBUTS :name NIL :fnums 1)} generates new names each time the command is invoked.

\textbf{caveat:} If \texttt{name} is null, there is no guarantee that the same names are used when re-running the proof. In that case, explicit reference to expressions containing new names should be avoided. If \texttt{name} is supplied, the new names are guaranteed to be the same each time the proof is executed.

\textbf{syntax:} \texttt{(REPLACES \&OPTIONAL fnums [*] in[*] from to hide?[T] dir[LR] step[(SKIP)])}

\textbf{effect:} Iterates the PVS command \texttt{REPLACE} in the formulas \texttt{in}, rewriting with the formulas in \texttt{fnums}, respecting the order. The keys \texttt{dir} and \texttt{hide?} are like in \texttt{REPLACE}. Notice that in contrast to \texttt{REPLACE}, \texttt{true} is the default value of \texttt{hide?}. Instead of using \texttt{fnums}, rewriting formulas can be addressed via \texttt{from} and \texttt{to}. The key \texttt{step} specifies the command to be executed after all the replaces have taken place.

\textbf{usage:} Given the sequent

\begin{align*}
\{-1\} & \quad \Delta = \text{sq}(B) - 4\times A \times C \\
\{-2\} & \quad \Delta \geq 0 \\
\{-3\} & \quad x = \frac{(\sqrt{\Delta} - B)}{(2 \times A)} \\
& \hline \\
\{1\} & \quad A \times \text{sq}(x) + B \times x + C = 0
\end{align*}

the command \texttt{(REPLACES :from -1 :to -3 :in 1)} yields the sequent

\begin{align*}
\{-1\} & \quad \Delta = \text{sq}(B) - 4\times A \times C \\
\{-2\} & \quad \Delta \geq 0 \\
& \hline \\
\{1\} & \quad A \times \text{sq}((\sqrt{\Delta} - B)) + \\
& \quad B \times ((\sqrt{\Delta} - B)) + C \\
& \quad = 0
\end{align*}

Note that since \texttt{\Delta} does not appear in the original formula \{1\}, it does not get replaced in the final sequent when the replacements are applied in the order \(-1, -2, -3\). In contrast, the command \texttt{(REPLACES :from -3 :to -1 :in 1)} yields the sequent

\begin{align*}
\{-1\} & \quad \Delta \geq 0 \\
& \hline \\
\{1\} & \quad A \times \text{sq}((\sqrt{\text{sq}(B) - 4\times A \times C} - B)) + \\
& \quad B \times ((\sqrt{\text{sq}(B) - 4\times A \times C} - B)) + C \\
& \quad = 0
\end{align*}

The latter sequent is also produced with the command \texttt{(REPLACES)}.

\section*{4.2. Arithmetic and Logic Manipulations.}

\textbf{syntax:} \texttt{(NEG-FORMULA fnum)}

\textbf{effect:} Negates both sides of the relational formula \texttt{fnum}.

\textbf{usage:} \texttt{(NEG-FORMULA -1)} on sequent

\begin{align*}
\{-1\} & \quad x < y
\end{align*}
I produces

\[-2\] \(a = b\)

\[-1\] \(-x > -y\)

\[1\] \(a = b\)

\[-3\] \(-x > -y\)

\[-4\] \(x \neq a + b \land y > a - b\)

\[-5\] \(y > a - b\)

syntact: (ADD-FORMULAS fnum1 fnum2 &OPTIONAL :hide? T)
effect: Adds relational formulas fnum1 and fnum2. Hides them when key hide? is true.
usage: (ADD-FORMULAS -1 -2 :hide? NIL) on the last sequent yields

\[-1\] \(-x + a > -y + b\)

\[-2\] \(-x > -y\)

\[-3\] \(a = b\)

\[-4\] \(-x > -y\)

\[-5\] \(a = b\)

\[-6\] \(x \neq a + b \land y > a - b\)

\[-7\] \(y > a - b\)

When proving properties on large arithmetic propositions, involving logical operators, it happens that side conditions have to be proven and then kept as hypothesis. The PVS command \texttt{SPLIT} is not very effective decomposing this kind of formulas, since it symmetrically splits the propositions. We have implemented an asymmetric split strategy called \texttt{SPLASH}.

syntax: (SPLASH fnum)
effect: Asymmetrically splits a conjunction fnum in the consequent (or a disjunction in the antecedent).
usage: (SPLASH 1) on the last sequent yields the two sequents

\[-1\] \(sq(a) = sq(b)\)

\[-2\] \(-x + a > -y + b\)

\[-3\] \(-x > -y\)

\[-4\] \(a = b\)

\[-5\] \(y > a - b\)

\[-6\] \(a = b\)

\[-7\] \(x \neq a + b \land y > a - b\)
\[-1\] \( \text{sq}(a) = \text{sq}(b) \)
\[-2\] \(-x + a > -y + b\)
\[-3\] \(-x > -y\)
\[-4\] \(a = b\)
\[\{\} \ x \neq a + b\]

Note the hypothesis \{-1\} in the first sequent. Now try \text{(UNDO)}, and compare with \text{(SPLIT 1)}, where this hypothesis is missing.

5. Conclusion. Hopefully, a decision procedure for the closed field of real number, such as Quantifier Elimination by Cylindrical Algebraic Decomposition \[4\], will soon be available in PVS \[10\]. In the mean time, we may have to deal with PVS limitations to handle non-linear arithmetic. Full automation has also its limitations: complex expressions may explode the computational capabilities of the machine. In the other side, proofs by hand are often tedious due to the PVS spartan support of algebraic manipulations.

Based on our experience on verification of avionics systems, we have implemented a package of strategies that combines simple arithmetic manipulation with PVS arithmetic decision procedures. For example, the strategy \text{FIELD} is a simplification procedure for the field of real numbers that works by removing inverses. The package also provides strategies for cancellation of common terms, negation and addition of formulas, and other common logic and arithmetic manipulation strategies.

The complete package was tuned and tested on several real examples. It has been shown effective in alleviating the proof effort of large and complex formulas. Last, but not least, the set of strategies contained in the package increases the automation power of the theorem prover, without compromising the soundness of the system. Indeed, since the strategies do not make external calls from the system, they are as sound as the original PVS theorem prover.

REFERENCES


Available at http://pauillac.inria.fr/jfla/2001/.


We provide a package of strategies for automation of non-linear arithmetic in PVS. In particular, we describe a simplification procedure for the field of real numbers and a strategy for cancellation of common terms.

### Abstract

- **Non-linear arithmetic, PVS strategies, real number automation**
- **Unclassified-Unclassified**
- **Standard:** NSN 7540-01-280-5500

---

**Subject Terms:**
- Non-linear arithmetic
- PVS strategies
- Real number automation

**Distribution/Availability:**
- Unclassified-Unlimited
- Subject Category 60, 61
- Distribution: Nonstandard
- Availability: NASA-CASI (301) 621-0390

---

**Report Information:**
- **Title:** Real automation in the field
- **Authors:** César Muñoz and Micaela Mayero
- **Performing Organization:** ICASE
  - Mail Stop 132C
  - NASA Langley Research Center
  - Hampton, VA 23681-2199
- **Funding Numbers:**
  - C NAS1-97046
  - WU 505-90-52-01
- **Performing Organization Report Number:** ICASE Interim Report No. 39
- **Sponsoring/Monitoring Agency:** National Aeronautics and Space Administration
  - Langley Research Center
  - Hampton, VA 23681-2199
- **Supplementary Notes:** Langley Technical Monitor: Dennis M. Bushnell
  - Final Report
- **Distribution Code:** Unclassified-Unlimited
- **Price Code:** A03
- **Number of Pages:** 14