110058

110058

N95- 16473

Electronic Engineering Notebook: A Software Environment For Research Execution, Documentation, and Dissemination

by Dan Moerder

The Electronic Engineering Notebook (EEN) is a byproduct of several years of collaborative work between LaRC and Martin Marietta Astronautics Group. The EEN consists of a free-form research notebook, implemented in a commercial package for distributed hypermedia, which includes utilities for graphics capture, formatting and display of LaTex constructs, and interfaces to the host operating system. The latter capability consists of an informal Computer-Aided Software Engineering (CASE) tool, and a means to associate executable scripts with source objects. The EEN runs on Sun and HP workstations.

The EEN, in day-to-day use, can be used in much the same manner as the sort of research notes most of us keep during development of our projects. Graphs can be pasted in, equations can be entered via LaTex, and so on. In addition, the fact that the notebook is hypermedia permits easy management of "context": e.g. derivations and data can contain easily formed links to other supporting derivations and data. The CASE tool also permits development and maintenance of source code directly in the notebook, with access to its derivations and data.

The EEN is currently in day-to-day use in the Guidance Group of the Guidance and Control Branch, and at Martin Marietta Astronautics Group.

Electronic Engineering Notebook

LaRC CSTC Workshop
16 June 1994

Dan Moerder
moerder@moerder.larc

• @Parent

Purpose of Briefing

Describe capabilities and demonstrated applications of the Notebook

Illustrate advantages and disadvantages of Notebook-based documentation over traditional approaches to research knowledge capture

Discuss availability and status of the Notebook

• @Parent

What is the Notebook?

Distributed hypermedia toolset for managing results of a research team's work...

- Structured for "unruly" information
- On the fly hypertext links between small "chunks" of information
- Equation editing and display via LaTeX and symbol palette
- Capture and pasting of X images, e.g. Matlab plots
- "Informal," "researchy" CASE tool

446

Hypermedia Document Example

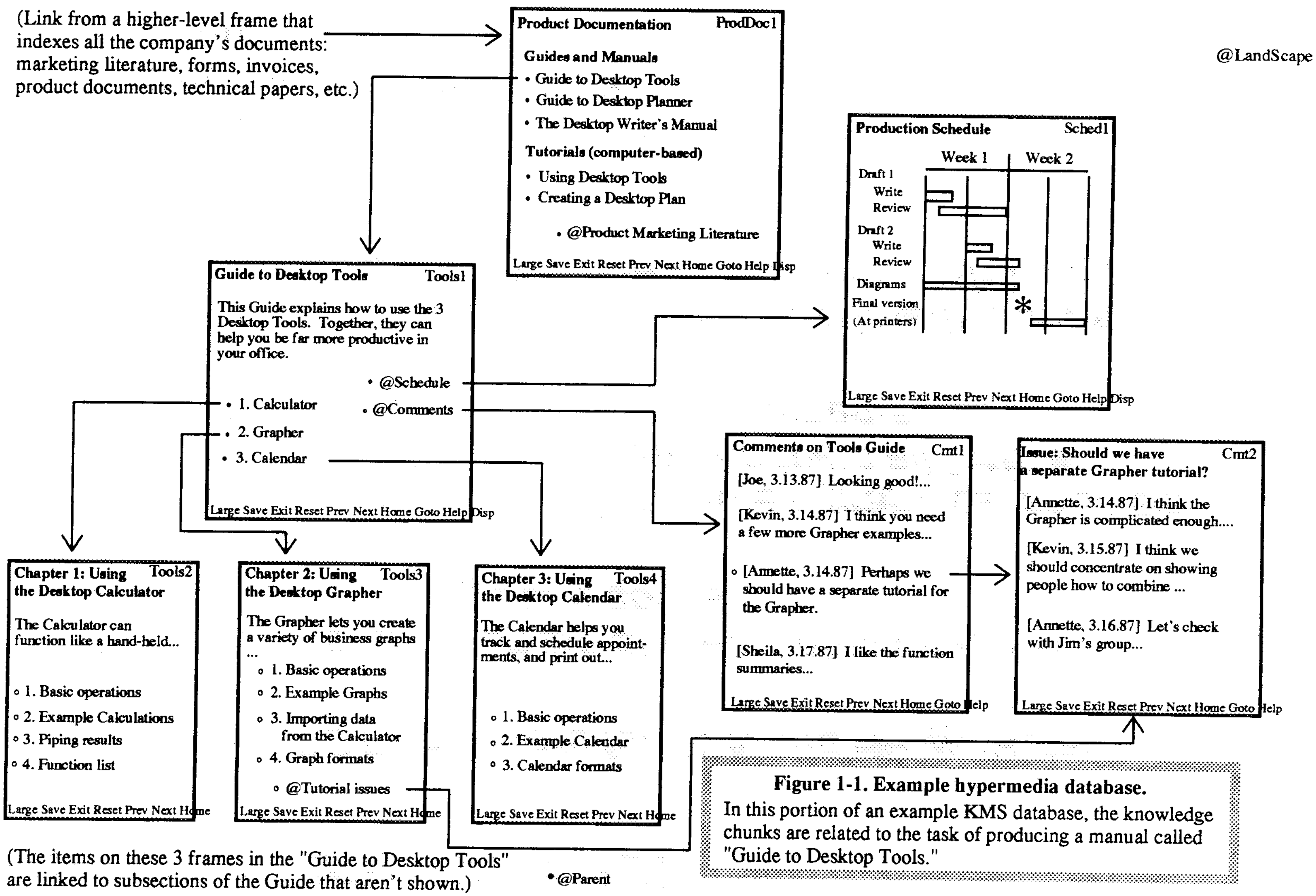


Figure 1-1. Example hypermedia database.
In this portion of an example KMS database, the knowledge chunks are related to the task of producing a manual called "Guide to Desktop Tools."

• @Parent

Notebook Implementation

- Based on COTS software for SUN and HP workstations
- Distributed databases are shared, enhancing team access to knowledge and codes.
- Knowledge, includes working notes, graphics, analysis codes, and machinery for running them all form a "document" with executable components....
- Components have been assembled, primarily, under Martin Marietta CRAD and IRAD support, with Langley involvement.
- Notebook is a component of a Martin reusable engineering system - the Process Management Environment.

• @Parent

Research Tasks in the EEN

- *Recording Derivations*
- *Doodling*
- *Writing Code*
- *Setting Up Problems for Solution*
- *Managing Data*
- *Doing Experiments*
- *Writing A Paper*

• @Parent

Case History

Task: Development of a trajectory optimization code for generating reference trajectories

- *Define simple class of optimal control problems to be solved*
- *Establish notation for representation in parameter optimization software*
- *Realize representation in FORTRAN*
- *Develop test cases*
- *Extend to more complicated optimal control structures*
- *Write a Paper*

Statement of single-phase optimal control problem

$$\left.\begin{array}{c} x^0 \in \mathcal{R}^n \\ x^1 \in \mathcal{R}^n \\ u(t) \in \mathcal{R}^m \quad t \in [0,1] \\ p \in \mathcal{R}^r \end{array}\right\} \tag{1}$$

$$\mathcal{J} = \phi(x^0, x^1, p) \tag{2}$$

$$x^0 = x(0) \qquad x^1 = x(1) \tag{3}$$

$$\dot{x} = f(x, u, p) \tag{4}$$

$$\psi(x^0, x^1, p) = 0 \tag{5}$$

$$(g_{min})_i \le g_i(x, u, p) \le (g_{max})_i \quad 0 \le t \le 1 \tag{6}$$

$$(h_{min})_j \le h_j(x, p) \le (h_{max})_j \quad 0 \le t \le 1 \tag{7}$$

$$\dot{x} = \tilde{f}(x, u, \tau) = \tau f(x, u) \tag{8}$$

@LaTeX
@UTILS DIR
@MATLAB
@XWD
@I MAGE MGR
@FONT

• @Parent

◦ M-File Tool ◦ FORTRAN Tool ◦ Get Back to Title

This frame states the problem that we're interested in solving. Determine

(1)
```
\Large{\begin{equation}
\left\begin{array}{c}
x^0\in {\cal R}^n \\
x^1 \in {\cal R}^n \\
u(t)\in {\cal R}^m --- t \in [0,1] \\
p\in {\cal R}^r
\end{array}\right\}
\end{equation}}
```

to minimize

(2)
```
\Large{\begin{equation}
{\cal J}=\phi(x^0,x^1,p)
\end{equation}}
```

where

(3)
```
\Large{\begin{equation}
x^0=x(0) ------ x^1=x(1)
\end{equation}}
```

subject to

(4)
```
\Large{\begin{equation}
\dot{x}=f(x,u,p)
\end{equation}}
```

(5)
```
\Large{\begin{equation}
\psi(x^0,x^1,p)=0
\end{equation}}
```

and the state/control inequality constraints

(6)
```
\Large{\begin{equation}
\left(g_{min}\right)_i \leq g_i(x,u,p)\leq \left(g_{max}\right)_i --- 0\leq t\leq 1
\end{equation}
```

for i=1,...,n_g, and state inequality constraints

(7)
```
\Large{\begin{equation} \left(h_{min}\right)_j \leq h_j(x,p)\leq
\left(h_{max}\right)_j --- 0\leq t\leq 1
\end{equation}}
```

for inequality constraints j=1,...,n_h. This splitting up of the constraints is due to the fact that we'll be using a state discretization in which state derivatives are calculated at the interior of discretization intervals. The controls affect state derivatives, while the state-only trajectory constraints are imposed at the discretization points.

Note that the problem is posed in Meyer form. The obvious measure for treating integral cost functions is to establish the cost function as a state. Note also that the problem is posed with unity duration. This can be generalized to free time by treating the trajectory duration as an element of the ***p*** vector, and scaling the plant ODES, e.g.

(8)
```
\Large{\begin{equation}
\dot{x}=\tilde{f}(x,u,\tau)=\tau f(x,u)
\end{equation}}
```

Note that (6,7) are expressed as they are because of the representation of constraints in NPSOL.

@landscape @FORMAT @STYLES ◦ @Type:FormattedDoc

◦ @Type:psimage

$$(h_{min})_j \leq H_j \leq (h_{max})_j \quad j = 1, \ldots, n_h \tag{7}$$

$$\psi(x_1, x_{N+1}, p) = 0 \tag{8}$$

$$Z = \begin{bmatrix} z \\ \psi \end{bmatrix} = 0 \tag{9}$$

$$Y = \begin{bmatrix} (g_{min})_1 & \leq & G_1 & \leq & (g_{max})_1 \\ \vdots & & \vdots & & \vdots \\ (g_{min})_{n_g} & \leq & G_{n_g} & \leq & (g_{max})_{n_g} \\ (h_{min})_1 & \leq & H_1 & \leq & (h_{max})_1 \\ \vdots & & \vdots & & \vdots \\ (h_{min})_{n_h} & \leq & H_{n_h} & \leq & (h_{max})_{n_h} \end{bmatrix} \tag{10}$$

$$\mathcal{J} = \phi(x_1, x_{N+1}, p) \tag{11}$$

$$X^T = \left[x_1^T \;\; u_1^T \;\; x_2^T \;\; u_2^T \;\; \cdots \;\; x_N^T \;\; u_N^T \;\; x_{N+1}^T \;\; p^T \right] \tag{12}$$


```
\setcounter{equation}{6}
```

Similarly, the state inequality constraints are arranged as

(7)
```
\Large{\begin{equation}
\left(h_{min}\right)_j\leq H_j\leq\left(h_{max}\right)_j
~~~j=1,\ldots,n_h \end{equation}}
```

Note that we are not posing the problem in such a way that the upper and lower bounds for G_i and H_j are functions of the x's u's or p. We are not totally comfortable with the notion of using the upper and lower bounds aggressively, since they are defined outside the logic for calculating the G's and H's.

The boundary conditions are restated as

(8)
```
\Large{\begin{equation}
\psi(x_1,x_{N+1},p)=0
\end{equation}}
```

and concatenated with the system dynamics (4) to form the equality constraints

(9)
```
\Large{\begin{equation}
Z=\left[\begin{array}{c}
z \\ \psi \end{array}\right]=0
\end{equation}}
```

The inequality constraints (6,7) are concatenated to form

(10)
```
\Large{\begin{equation}
Y=\left[\begin{array}{ccccc}
(g_{min})_1 &\leq& G_1 &\leq& (g_{max})_1 \\
\vdots && \vdots && \vdots \\
(g_{min})_{n_g} &\leq& G_{n_g} &\leq& (g_{max})_{n_g} \\
(h_{min})_1 &\leq& H_1 &\leq& (h_{max})_1 \\
\vdots && \vdots && \vdots \\
(h_{min})_{n_h} &\leq& H_{n_h} &\leq& (h_{max})_{n_h}
\end{array}\right]
\end{equation}}
```

Note that (10) is $N^*n_g+(N+1)^*n_h$ constraints.

As a final note, the cost is expressed as

(11)
```
\Large{\begin{equation}
{\cal J}=\phi(x_1,x_{N+1},p)
\end{equation}}
```

The free variables in this code are arranged as

(12)
```
\Large{\begin{equation}
X^T=\left[
x_1^T ~ u_1^T ~ x_2^T ~ u_2^T ~ \cdots ~ x_N^T ~ u_N^T ~ x_{N+1}^T ~ p^T
\right]
\end{equation}}
```

todinotes31

This frame summarizes the details needed to run the single-phase direct NLP shooting code. The code uses NPSOL to minimize a cost function subject to plant dynamics, boundary conditions, and miscellaneous user-specified inequality constraints. Note that, for this version of the code, no interior boundary conditions, e.g. staging, are permitted. In addition, the problem is assumed to be cast in Meyer form, with unity duration. The problem statement is

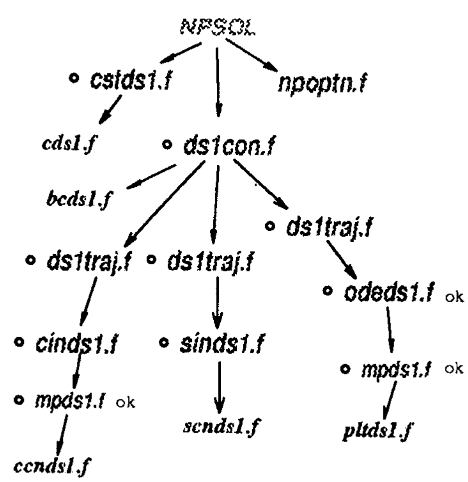
∘ *Problem Statement Here*

and its representation in the code is laid out here:

∘ *Derivation Here...*

In order to run the code, the user supplies a main routine, and five subroutines: cost, boundary conditions, RHS of the plant ODE's, state inequality constraints, and control inequatlity constraints. Templates for these routines are given below:

- ∘ ***Template for the main routine, mads1.f***
- ∘ ***Template for cds1.f (Cost Function)***
- ∘ ***Template for bcds1.f (Boundary Conditions)***
- ∘ ***Template for pltds1.f (RHS of plant ODEs)***
- ∘ ***Template for scnds1.f (State Inequality Constraints)***
- ∘ ***Template for ccnds1.f (State/Control Inequality Constraints)***



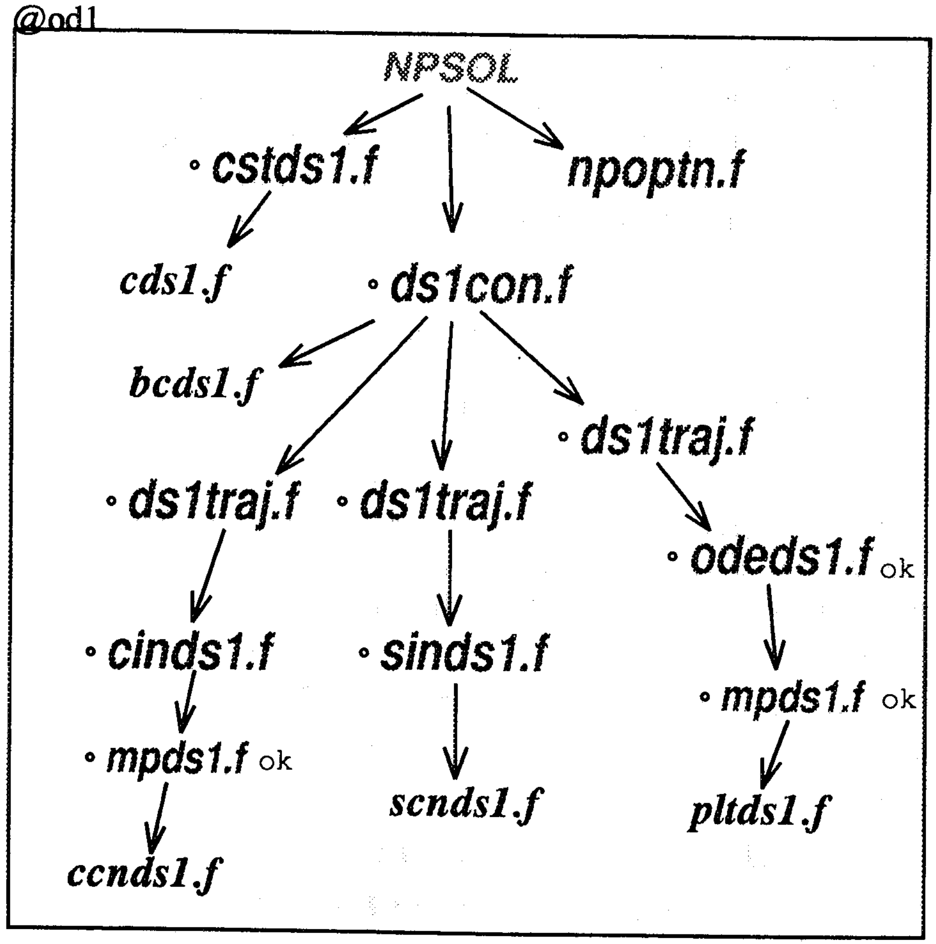
@LaTeX
@UTILS DIR
@MATLAB
@XWD
@I MAGE MGR
@FONT

∘ NPSO

• @Parent

∘ M-File Tool ∘ FORTRAN Tool ∘ Get Back to Title **@MORE**

Set up the code structure



@LaTeX
@UTILS DIR
@MATLAB
@XWD
@I MAGE MGR
@FONT

∘ M-File Tool ∘ FORTRAN Tool ∘ Get Back to Title • @Parent **@MORE**

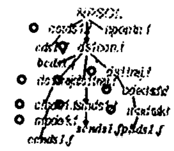
@Top of File

PARAMETERS | TOGGLE | FREEZE | EXPORT | EXECUTE | ◦ @EXECUTE SCRIPT

@

This is a template for mads.f, which calls NPSOL for doing single-phase discretized direct NLP. Here's a map of the code:

```
c
      program mads1
      implicit double precision(a-h,o-z)
```



◦ C***Declarations and User-Defined Parameters

@ NPLNTP NUP

@ NROWA NROWJ NROWR

@ maxpval maxwk nu nplnt nbc nparms

@ MAXWKP ndint nis niu

```
◦ C***User Defines Logic for getting plant
  C    parameters and initial state and
  C    control guess
```

@ NCNLN NCNLIN NCON N LWORK LIWORK

@ NROWA NROWJ NROW MAXWKP

@ maxpval maxwk nu nplnt nbc nparms

@ ndint nis niu

@ parms x

@ parms=plant parameters
x=initial guess for state,control history and parameters.

cds1.f
bcds1.f
pltds1.f
ccnds1.f
scnds1.f

◦ C***Execute NPSOL

◦ **makefile**

@ x=solution?
c=final value of constraint vector.

◦ C***User Defines logic to save results

```
C
      stop
      end
```

◦ Clone this frameset!

• @Parent

C***Set NPSOL execution parameters

PARAMETERS | TOGGLE | FREEZE | EXPORT | EXECUTE

° @EXECUTE SCRIPT

@Calculate derivatives numerically

```
c
      call npoptn('derivative level                  0')
      call npoptn('difference interval        .0000001')


c***Diagnostic
      call npoptn('print level         21')
```

@Hardwired interval for numerical differentiation. If this isn't used, NPSOL will waste time figuring this out on a case-by-case basis.

HELP

• @Parent

Last file export on: 28 January 94 at 10:49:57, current version

- ◦ Info • **Top frame of tree to write:** todi0001a2
- ◦ Info • **File:** /moerder/usr1/moerder/TODI-II/ds1/mads1.f
- ◦ Info • **Add blank line between items: yes**
- ◦ Info • **Follow tree items linking to other framesets: no**
- ◦ Info • **Follow annotation items linking within the frameset: no**
- ◦ Info • **Time version number:**
- ◦ Info • **Preserve relative indentation of items:** no
- ◦ Info • **Template frame:**
- ◦ Info • **Remove 1st character of each line during export: no**
- ◦ Info • **Program to execute script: shell script**
- ◦ Info • **Toggle text 1, family: Times**
- ◦ Info • **Toggle text 1, size: 16**
- ◦ Info • **Toggle text 2, family: Courier**
- ◦ Info • **Toggle text 2, size: 14**

◦ ***This script initially cloned from 'shoot0019a' 27 December 93 10:33:32***

• @Parent

@Top of startup script

cd /moerder/usr1/moerder/TODI-II/fighter

```
f77 -O2 -o mads1 mads1.f  cds1.o bcds1.o pltds1.o ccnds1.o scnds1.o \
getprm.o ficof.o \
../ds1/ds1con.o \
../ds1/cstds1.o \
../ds1/ds1traj.o \
../ds1/odeds1.o \
../ds1/cinds1.o \
../ds1/sinds1.o \
../ds1/mpds1.o \
        -lnpsol -llinpackd
```

• @Parent

Fighter from Hans' Thesis

This note lays out the aircraft model paraphased from Hans Seywald's thesis . The equations of motion are

$$\dot{v} = (T - D)\frac{\cos\gamma}{m} - g\sin\gamma \tag{1}$$

```
\Large{\begin{equation}
\dot{v}=(T-D)\frac{\cos\gamma}{m}-g\sin\gamma
\end{equation}}
```

$$\dot{h} = v\sin\gamma \tag{2}$$

```
\Large{\begin{equation}
\dot{h}=v\sin\gamma
\end{equation}}
```

$$\dot{\gamma} = \frac{g}{v}(n - \cos\gamma) \tag{3}$$

```
\Large{\begin{equation}
\dot{\gamma}=\frac{g}{v}\left(n-\cos\gamma\right)
\end{equation}}
```

$$\dot{x} = v\cos\gamma \tag{4}$$

```
\Large{\begin{equation}
\dot{x}=v\cos\gamma
\end{equation}}
```

where atmospheric density is

$$\rho(h) = \frac{1.225}{g}e^{y} \tag{5}$$

```
\Large{\begin{equation}
\rho(h)=\frac{1.225}{g}e^y
\end{equation}}
```

$$y = (ay)_1 + (ay)_2h + (ay)_3(e^{z(h)}) \tag{6}$$

```
\Large{\begin{equation}
y=(ay)_1+(ay)_2h+(ay)_3(e^{z(h)})
\end{equation}}
```

$$z(h) = \sum_{j=1}^{4}(az)_jh^j \tag{7}$$

```
\Large{\begin{equation}
z(h)=\sum_{j=1}^4(az)_jh^j
\end{equation}}
```

Speed of sound is

$$a(h) = 20.0468\sqrt{\theta} \tag{8}$$

```
\Large{\begin{equation}
a(h)=20.0468\sqrt\theta
\end{equation}}
```

where the temperature θ is given by

$$\theta(h) = \sum_{j=1}^{4}(a\theta)_jh^{j-1} \tag{9}$$

```
\Large{\begin{equation}
\theta(h)=\sum_{j=1}^4(a\theta)_jh^{j-1}
\end{equation}}
```

Dynamic pressure, q, is given by the usual

$$q = \rho v^2/2 \tag{10}$$

```
\Large{\begin{equation}
q=\rho v^2/2
\end{equation}}
```

◦ **@MORE**

@LaTeX
@UTILS DIR
@MATLAB
@XWD
@I MAGE MGR
@FONT

• *Get from X Windows clipboard* • *Attach to X Windows clipboard* • ~~@Parent~~

◦ **M-File Tool** ◦ **FORTRAN Tool** ◦ **Get Back to Title**

@landscape @FORMAT @STYLES ◦ @Type:FormattedDoc

◦ @Type:psImage

@Top of File

PARAMETERS TOGGLE FREEZE EXPORT EXECUTE

∘ @EXECUTE SCRIPT

∘ Miscellaneous Constants

∘ ay

@These are constants for the Hans F15 model

∘ az

∘ aθ

∘ acd0

∘ bcd0

∘ ak

∘ bk

∘ **Drag Model**

∘ ae

• @Parent

PARAMETERS TOGGLE FREEZE EXPORT EXECUTE ∘ @EXECUTE SCRIPT

~~@+7.29821847445e-1~~
~~-3.25219000620~~
~~+5.72789877344~~
~~-4.57116286752~~
~~+1.37368651246~~

```
+1.37368651246
-4.57116286752
+5.72789877344
-3.25219000620
+7.29821847445e-1
```

HELP

• @Parent

Flight Envelope Calculations

$$
\left.\begin{array}{rcl}
v^+(h) & = & \arg\max_h v \\
v^-(h) & = & \arg\min_h v
\end{array}\right\} \tag{1}
$$

$$
T(h,v) - D(h,v) = 0 \tag{2}
$$

$$
\left.\begin{array}{rcl}
\gamma & = & 0 \\
n & = & 1
\end{array}\right\} \tag{3}
$$

This note lays out the calculations for generating a flight envelope for the fighter aircraft model. In setting this up, I'm assuming that the envelope is convex. Pursuant to this assumption, we get roughly half of the envelope by solving

```
\Large{\begin{equation}
\left.\begin{array}{rcl}
v^+(h)&=&{\rm arg}\max_h v \\
v^-(h)&=&{\rm arg}\min_h v
\end{array}\right\}
\end{equation}}
```

subject to

```
\Large{\begin{equation}
T(h,v)-D(h,v)=0
\end{equation}}
```

for a given γ and load factor, in this case,

```
\Large{\begin{equation}
\left.\begin{array}{rcl}
\gamma&=&0 \\
n&=&1
\end{array}\right\}
\end{equation}}
```

◦ ~~*Code Using NPSOL and the other machinery for this example*~~

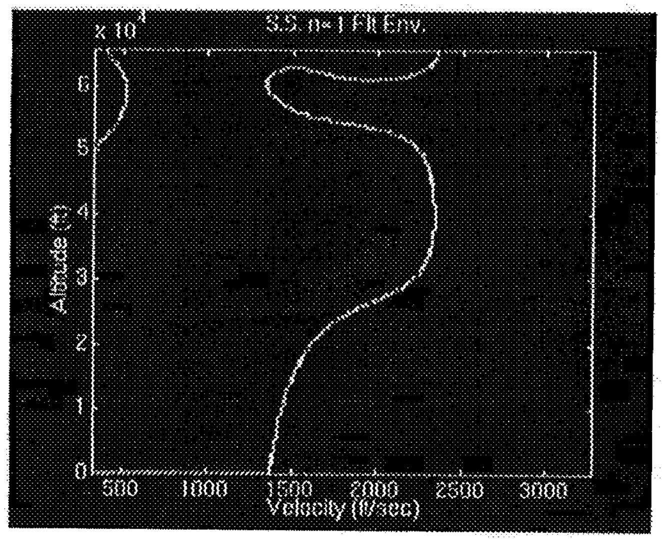
This doesn't seem to work out. Here's a check of the thrust and drag models.

◦ *Code to display this stuff* ◦ *Code to Check T-D over flight regime*

Naturally, The NPSOL business hasn't converged except in a very narrow range of conditions. I'm now checking against Hans' model to determine where our numbers diverge.

◦ *Code for Checking against Hans*

The flight envelope to the left corresponds to the case where all conflicts with Hans' model have be resolved, with the exception of the Drag expression.



@LaTeX
@UTILS DIR
@MATLAB
@XWD
@I MAGE MGR
@FONT

• *Get from X Windows clipboard* • *Attach to X Windows clipboard*

• ~~@Parent~~

◦ **M-File Tool** ◦ **FORTRAN Tool** ◦ **Get Back to Title** ***@MORE***

@landscape **@FORMAT** **@STYLES** ◦ @Type:FormattedDoc

◦ **@Type:psimage**

costappii76

∘ ***Diary for this problem (and) How to find Dan's F-15 solution !!!***

Do each of these things just in case GEORGE has dropped a disk !!

- ∘ ***(1) Matlab script to find the "slopy" F-15 data and save.***
- ∘ ***(2) Do some FORTRAN and create the necessary jacobians.***
- ∘ ***(3) Costate approx function for slopy F-15 -- super easy...***
- ***(4) Use approx costates as init guess in shooting...***

∘ Min-time-to-climb with one control segment

@LaTeX
@UTILS DIR
@MATLAB
@XWD
@I MAGE MGR
@FONT

• *Get from X Windows clipboard* • *Attach to X Windows clipboard*

• @Parent

∘ **@M-File Tool** ∘ **@FORTRAN Tool** ∘ **@Get Back to Title** ***@MORE***

@landscape @FORMAT @STYLES ∘ @Type:FormattedDoc

costappii77

Dans' solution to the F-15 mintimetoclimb problem with one control seg can be found in /moerder/usr1/moerder/TODI-II/slop_fighter/e2000_38000_21.dat !!!

This file has 21 individual problems (each with a different terminal energy). Each run has 20 nodes, 6 boundary conditions, 5 states, 1 (phoney) control, and 3 free params (that define the actual control found in x(4)) thus each solution has 128 npsol params.

20 nodes * 5 states + 19 (ctrls appear at midpoints) + 6 bc's + 3 free params = 128

Anyway the data we want is found in (2561:2688); that is the last 128 lines in the above file !!!

It took forever to figure this out !!!!!

Next we must remember that Dan's data does not have the control at the midpoints but at the node points. I will use midpoint_filter.m to correct this.

Grrrrr.....

464

Last but not least Dan's data does not have the time state as needed by my code. I will correct for this in the usual way.

Grrrrr..... Irritation -- like a rash that won't go away...

All of the above is taken care of in the (1) matlab script on the previous frame... O.K. I did this... and the result is stored in xu_slop.dat in directory -- TODI-II/fighter

---

Next the trajectory jacobians are calculated in fortran !!! O.K. I did this... and the resulting files are stored in xu_slop_fdjac.dat and xu_slop_f.dat in TODI-II/fighter

@Top of File

PARAMETERS TOGGLE FREEZE EXPORT EXECUTE • @EXECUTE SCRIPT

• %Header

• %Introduction

• %Single-Impulse Problem Statement @FIG 4

• %Results

@FIG 9

• %Two-Impulse Problem Statement

• %Figures and Tables

• % Bibliography

\end{document}

• @Parent

%Single-Impulse Problem Statement

PARAMETERS | TOGGLE | FREEZE | EXPORT | EXECUTE | ° @EXECUTE SCRIPT

\section{Vehicle Model and Mission Description}

° % Vehicle Model

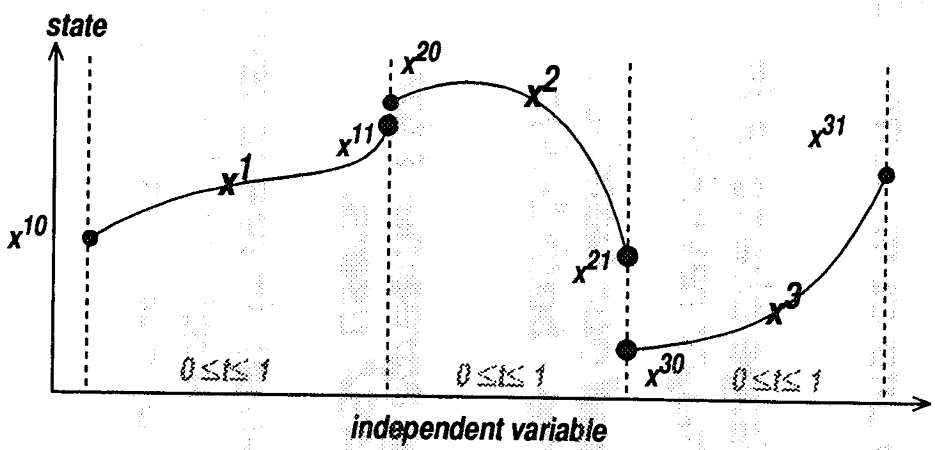
° % Mission Parameters

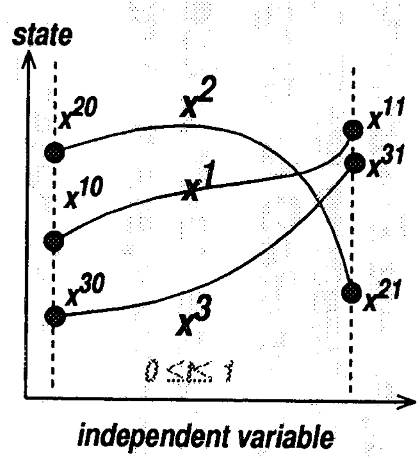
° % Constraints

° % Optimization Formulation

HELP

• @Parent


state
x^{20}
x^2
x^{11}
x^1
x^{31}
x^{10}
x^{21}
x^3
x^{30}
$0 \leq t \leq 1$
$0 \leq t \leq 1$
$0 \leq t \leq 1$
independent variable


state
x^{20}
x^2
x^{11}
x^{31}
x^{10}
x^1
x^{30}
x^3
x^{21}
$0 \leq t \leq 1$
independent variable

Current Status

- Notebook system has been (stoically) tested and commented on by GCB's Guidance Group for almost a year.
  - Testing and comments have resulted in large changes in interface; notebook is going into its 4th major revision.
- Martin Process Management Environment (and our Notebook) suffered from very troublesome learning curve - both being "dumbed down" for usability under IRAD funding.
  - Martin has developed a no-cost licensing agreement for distribution of the new system to non-NASA users.
- The host software vendor (Knowledge Systems), in response to loud and persistent suggestions from Martin and Langley will distribute read-only licenses free of charge, and full licenses free to academic institutions.
  - This is favorable for technology interchange...

• @Parent

Summary

- *The Notebook does capture and organize the work of research teams.*
  - *It is in daily use by 5 researchers at LaRC*
  - *And by roughly 40 engineers at Martin (in its PME form).*
  - *And, being baselined in a proposed university multidisciplinary design curriculum.*
- *The Notebook is very useful in its current form, and has fewer irritating features with each revision.*
- *Difficulties with distribution of notebook-capture knowledge are being resolved, e.g. licenses.*
- *The Notebook is currently available to Langley researchers with Sun or HP workstations. Moerder will gladly discuss this in more detail offline, set up live demos, etc.*

IDEAS^2 Computer Aided Engineering Software

by Pat Troutman

IDEAS^2 is a multidisciplinary Computer Aided Engineering (CAE) software tool that was developed for systems engineering and integration analysis of spacecraft. The name IDEAS^2 was derived from the two software packages that were integrated to form the tool. Interactive Design and Evaluation of Advanced Spacecraft (IDEAS) was a NASA spacecraft-specific analysis software tool that was combined with a commercially available product called Integrated Design Engineering Analysis Software (I-DEAS). I-DEAS is a Structural Dynamics Research Corporation (SDRC) product that provided capabilities lacking in NASA IDEAS such as solid and finite element modeling, thermal analysis and advanced graphics.

IDEAS^2 utilizes a common database structure which facilitates the integrated flow of data between the various analysis modules. All analysis is based on information derived from a three dimensional solid math model that is created in the commercial solid modeling program. The combination facilitates traceability and ensures all analysis is based on the same information. Once the model has been generated and stored in the common database, a wide range of analysis can be performed. IDEAS^2 has several orbital dynamics modules that can simulate/analyze spacecraft characteristics such as controllability in the presence of dynamic operations (solar array articulation, robotic arms, etc.), orbit lifetime/reboost requirements and micro gravity environment. Structural analysis capabilities are also available ranging from finite element modeling to forced response analysis. The impact of the local spacecraft environment can also be evaluated by utilizing the IDEAS^2 thermal and plume impingement analysis capabilities.

The common database and integrated analysis environment allow IDEAS^2 to be used both for high level short term studies and large program systems integration. Several NASA centers utilize the software for advanced concept analysis dealing with space platforms or Lunar/Mars exploration. The Space Station Freedom program has established IDEAS^2 as its primary Level II integration software package. IDEAS^2 models are commonly used to disseminate the latest Freedom element weights and configuration updates.IDEAS^2 has recently been upgraded to allow the entire software package to be ported to a UNIX workstation along with a new graphical user interface. This will allow smaller organizations to utilize the IDEAS^2 capability without a significant investment in computer hardware.

IDEAS^2 was initially developed from 1985 to 1986 and has continuously been enhanced to include the most up to date analysis tools and graphics interfaces

IDEAS^2

Computer Aided Engineering

Software

Pat Troutman
Spacecraft & Sensors Branch

Outline

- IDEAS^2 Background
- Current Capabilities
- Lessons Learned / Future Directions
- IDEAS^2 Analysis Simulation Video

NASA
IDEAS

I **nteractive**
D **esign and**
E **valuation of**
A **dvanced**
S **pacecraft**

+

SDRC*
I–DEAS

I **ntegrated**
D **esign**
E **ngineering**
A **nalysis**
S **oftware**

= IDEAS²

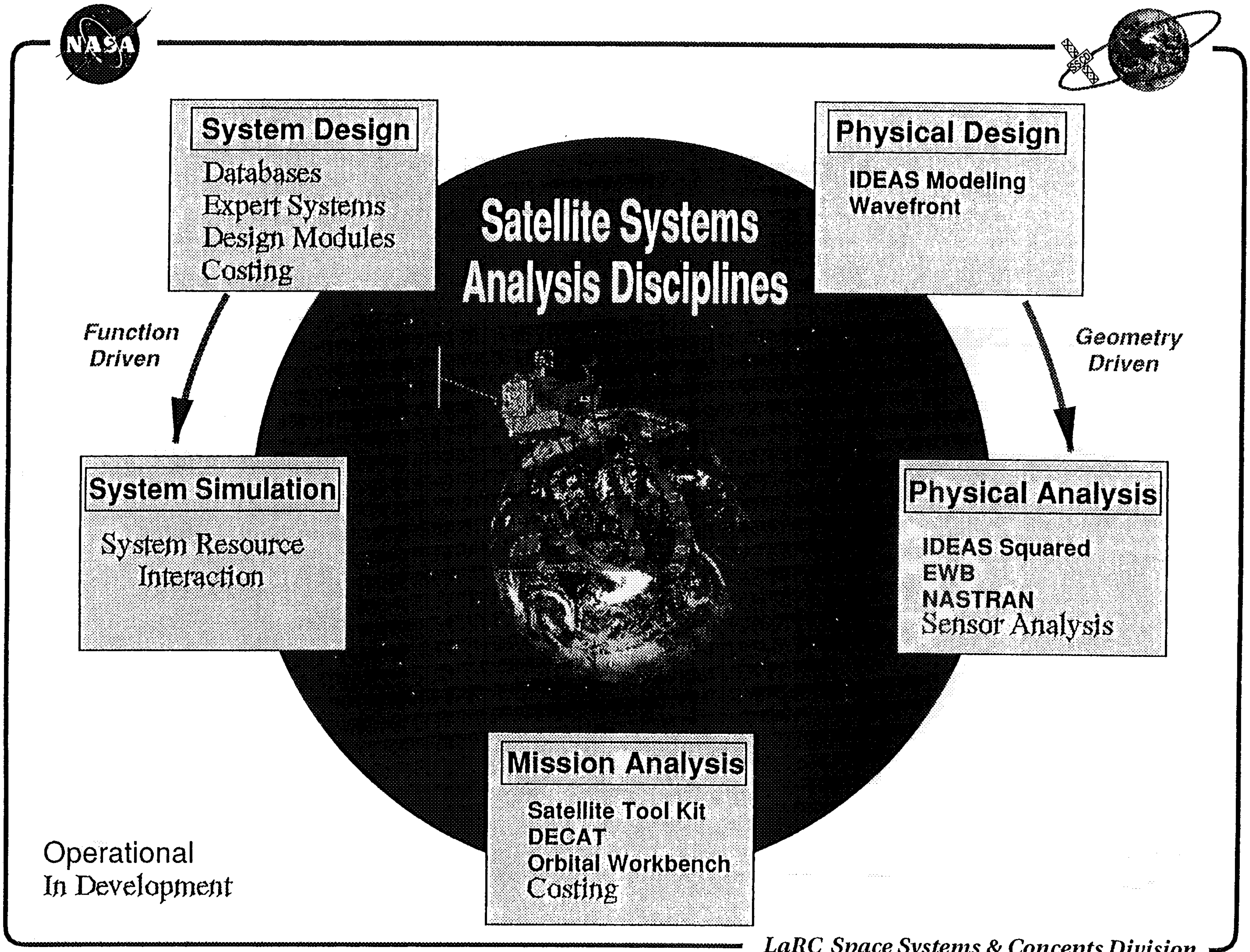
* *Structural Dynamics Research Corporation*

What is IDEAS^2

- A multidisciplinary software tool that was developed for spacecraft analysis

- All analysis is based on information derived from a three dimensional solid math model that is stored in a common database structure

- Analysis capabilities include orbital dynamics simulation, structural modeling and analysis, thermal analysis, plume impingement and orbital debris impact analysis


Satellite Systems
Analysis Disciplines
System Design
Databases
Expert Systems
Design Modules
Costing
Function
Driven
System Simulation
System Resource
Interaction
Physical Design
IDEAS Modeling
Wavefront
Geometry
Driven
Physical Analysis
IDEAS Squared
EWB
NASTRAN
Sensor Analysis
Mission Analysis
Satellite Tool Kit
DECAT
Orbital Workbench
Costing
Operational
In Development
LaRC Space Systems & Concepts Division

IDEAS^2 History

- Initially developed for the Space Station program from 1985 to 1986

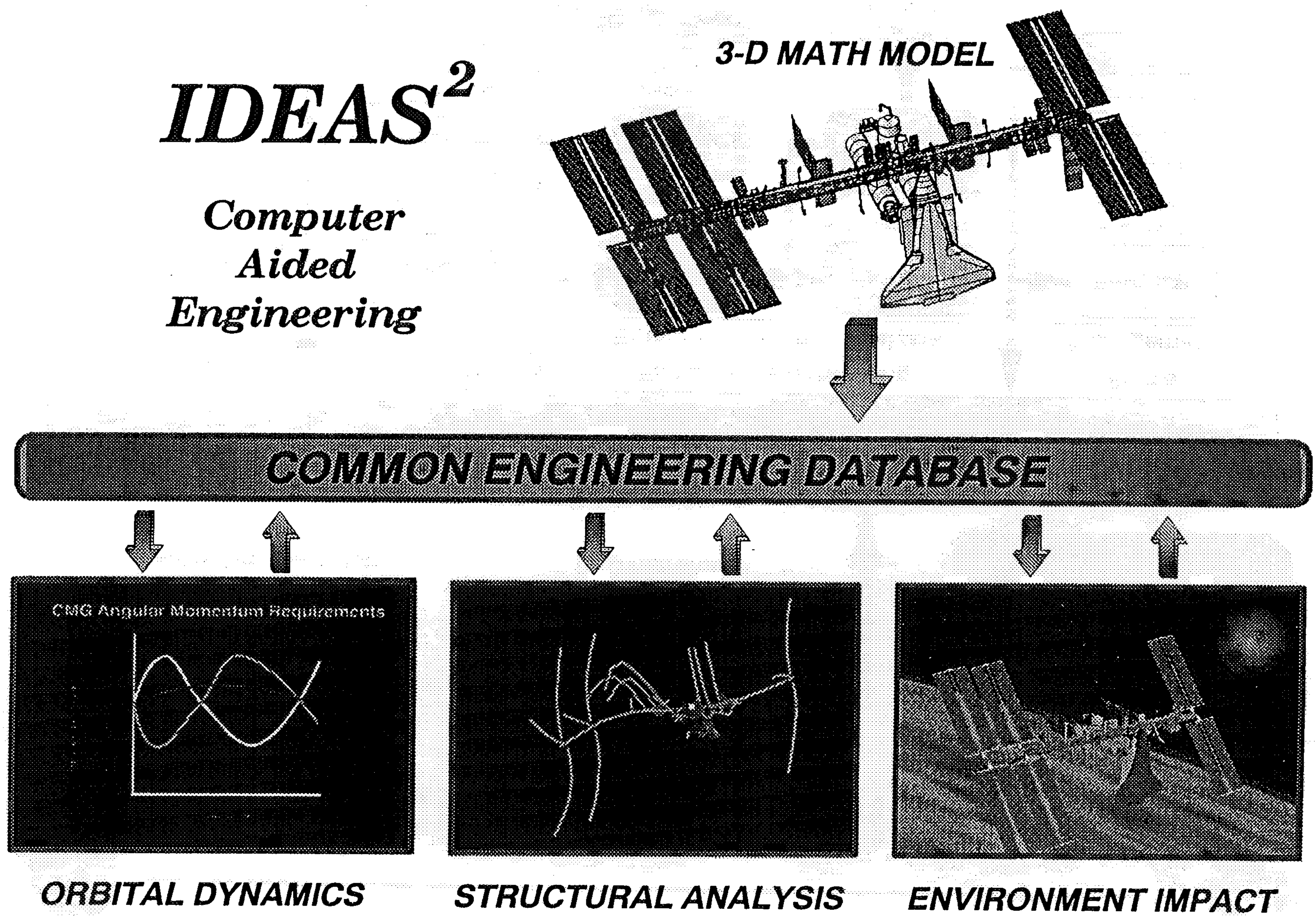
- Selected as Space Station level II integration package in 1987

- Used by LaRC, Johnson Spaceflight Center and the University of Colorado

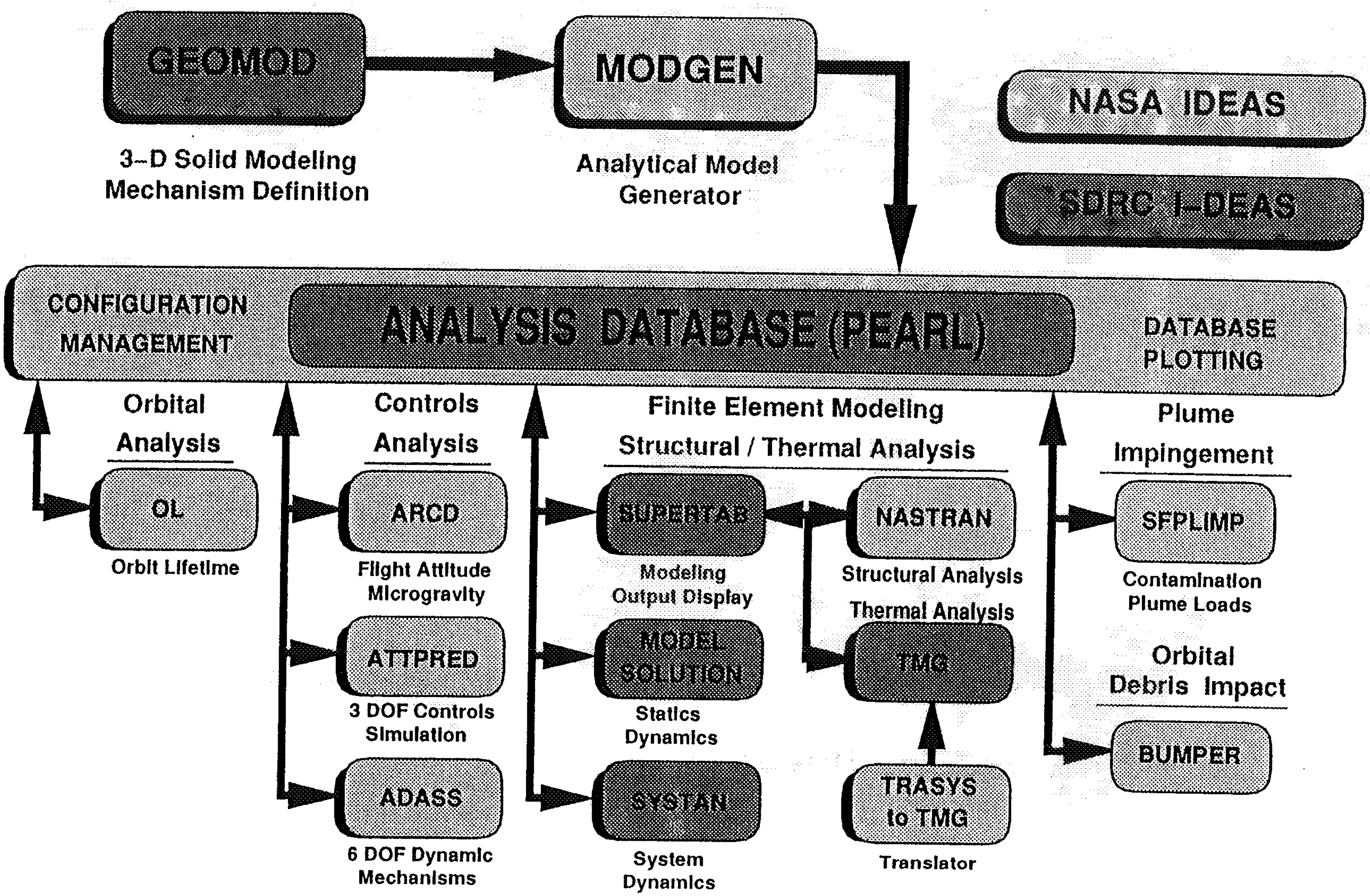
- Updated in 1993 to run on a Silicon Graphics workstation with a X window graphical user interface

IDEAS^2

Computer Aided Engineering



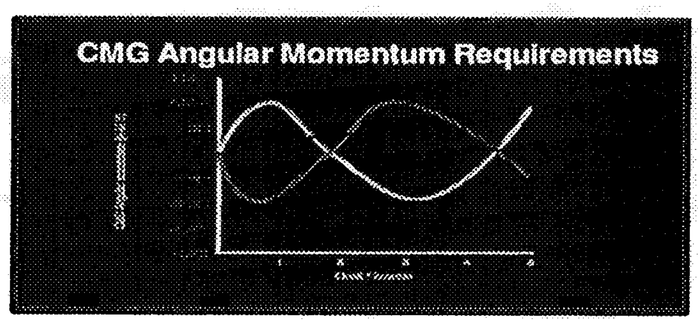
IDEAS^2 Capability



LaRC Space Systems & Concepts Division

Spacecraft Dynamics and Control Capabilities

- 3 and 6 DOF analysis programs, passive and active control
- Atitude Control Law Simulation Capabilities
  - *CMG (attitude or momentum emphasis; momentum management)*
  - *Reaction Control System (RCS Jets)*
  - *Reaction Control Wheel (with suplemental magnetic torque rods)*
  - *Passive Magnetic Dampers*
- Microgravity Environment Determination
- Optimal Attitude Determination Capability
- Orbit Lifetime Analysis
- Reboost Guidance/Optimization Simulation Capability
- Robotic Dynamic Simulation
- Station Keeping Fuel Estimation
- ACRV Escape Trajectory clearance determination



NASA

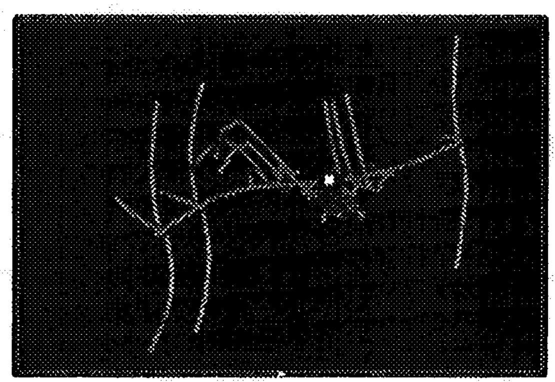
Structures/Mechanisms Analysis Capabilities

I-deas - Solid model generation (GEOMOD), finite element model generation (SUPERTAB), post processing (SUPERTAB), transient response analysis (Model Solution/SYSTAN).

ADAMS/ADASS - Mechanism/deployment analysis

NASCON - NASTRAN to SUPERTAB conversion

NASTRAN - Calculation of modes and frequencies.



Thermal / Plume Impingement / Orbital Debris Analysis Capabilities

Thermal Analysis:

I-deas - Extensive interactive geometric, finite element & finite difference modeling. Interface to "standard" thermal and structural analysis tools. ***TMG*** - Solar and planetary heat flux calculations.

Plume Impingement Analysis:

SFPLIMP - Designed to provide an assessment of the effects of jet firing on nearby space structures. The software allows the user to determine the instantaneous pressure loads, heating rates and contamination rates on surfaces due to jet exhaust.

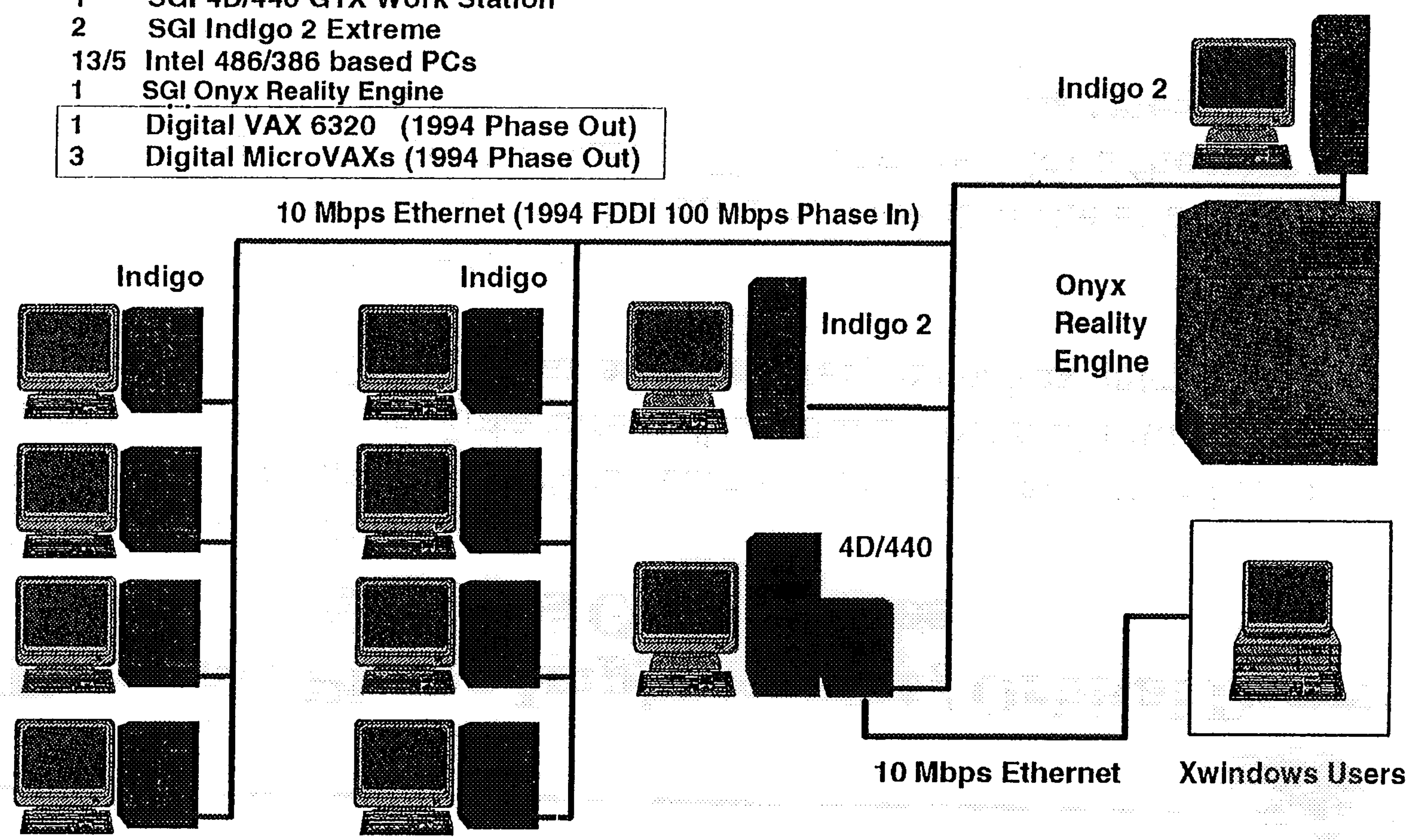
Orbital Debris Analysis:

Bumper II - Predicts the probability of no penetration or no impact for spacecraft subject to man-made orbital debris or meteoroid impact. The code accounts for varying impact velocity, impact angle, wall configurations, and the effects of spacecraft geometry and orientation.

S&SB Computer System Hardware

SYSTEMS:

8	SGI R4000 INDIGO Work Stations
1	SGI 4D/440 GTX Work Station
2	SGI Indigo 2 Extreme
13/5	Intel 486/386 based PCs
1	SGI Onyx Reality Engine
1	Digital VAX 6320 (1994 Phase Out)
3	Digital MicroVAXs (1994 Phase Out)



Lessons Learned in Developing & Maintaining IDEAS^2

Integration vs Interfacing:

IDEAS squared currently has a high degree of integration between the commercial software, the database and the NASA developed analysis codes. Changes in one area can ripple through to the others causing a maintenance backlog.

Analytical Module Development:

Engineers are hampered during analytical software development by complex database and GUI interfacing

Sample IDEAS^2 Analysis Video

Analysis Task: Verify that the Shuttle RMS can rotate the the an early stage of the Space Station through two 90 degree rotations in one orbit's time.

Geomod – Used to build model of shuttle/station configurations

ARCD – Used to establish initial and final configuration flight attitudes

ATTPRED – Used to establish initial and final configuration stability characteristics

ADASS – Used to perform free drift/robotics simulation