AIAA'92

# AIAA 92-1191 Design Sheet: An Environment for Facilitating Flexible Trade Studies During Conceptual Design

M. J. Buckley, K. W. Fertig, and D. E. Smith
Rockwell International Science Center
Palo Alto Laboratory, Palo Alto, CA 94301

1992 Aerospace Design Conference
February 3-6, 1992/Irvine, California

For permission to copy or republish, contact the American Institute for Aeronautics and Astronautics
370 L'Enfant Promenade, S.W., Washington, D.C. 20024

# DESIGN SHEET: AN ENVIRONMENT FOR FACILITATING FLEXIBLE TRADE STUDIES DURING CONCEPTUAL DESIGN

M. J. Buckley, K. W. Fertig, and D. E. Smith
Rockwell International Science Center, Palo Alto Laboratory
Palo Alto, CA 94301

## Abstract

This paper summarizes the capabilities of *Design Sheet*, a software program that facilitates trade studies during conceptual design. Design Sheet permits the designer to build a model for use in conceptual design by entering a set of algebraic equations in a very flexible form. The designer can then use Design Sheet to easily change the set of independent variables in the algebraic model, and to rapidly perform trade studies, optimization, and sensitivity analyses. The basic mathematics and algorithms used in Design Sheet are outlined. The functionality of Design Sheet is illustrated first with a simple example, and then with a more complex example involving initial aircraft sizing. For realistic conceptual design problems, it is argued that Design Sheet provides the capability to perform trade studies with significantly increased flexibility and efficiency.

## I. Introduction

The development of complex systems such as aircraft requires a sequence of engineering and management decisions in which an overall design is sought which satisfies many competing requirements. Tradeoff analyses are undertaken to find design solutions that simultaneously satisfy multiple performance requirements as well as manufacturability, life-cycle cost and schedule constraints. In addition, the design team is challenged to produce a design that is robust to unavoidable variations in the manufacturing processes and to unanticipated variations in the use of the system. The overall goal is to meet or exceed customer expectations. Because of these and other issues, there has been a growing interest in Total Quality Management techniques to improve the performance of the design team (King [1989], Hauser & Clausing [1988]). While these techniques are useful, they do not address engineering analysis, which is the backbone of the product design process.

While there is an extensive industry providing CAD tools, these tools are generally not suitable for use in the conceptual design phase. The conceptual design phase is particularly important since the majority of the life-cycle costs and overall quality of the system will be determined in this phase (National Materials Advisory Board [1991]). For an extremely wide range of domains such as aircraft design, rocket engine design and communications systems design, this engineering tradeoff process during conceptual design is undertaken using fairly simple mathematical models of the underlying phenomena. These models typically take the form of sets of algebraic equations that relate the parameters of the product being designed to multiple engineering parameters, cost, etc. In the aircraft industry for example, the initial sizing of the aircraft involves several hundred variables together with a similar number of equations. In various divisions of Rockwell International, FORTRAN programs and spreadsheets have traditionally been used to perform predefined trade studies in the conceptual design of aircraft, rocket engines, and communication systems. Although these programs and spreadsheets perform important trade studies, they are very inflexible. If the design team wants to perform a different trade study, then either reprogramming is required or a new spreadsheet must be written.

This paper describes *Design Sheet*, a system that overcomes this limitation. Design Sheet allows the user to input a set of algebraic equalities and inequalities and to use these equations in almost arbitrary fashion to 1) find the values of variables given other variables, 2) perform trade studies, 3) do optimization, and 4) perform sensitivity analysis. The user is free to dynamically change which variables will be treated as independent. The system then automatically determines which variables are dependent and computes their values. The choice of independent variables is limited only by the structure and degrees of freedom of the algebraic constraints.

Section II of this paper describes the functionality of Design Sheet in the context of a simple example. Section III briefly describes the mathematics and algorithms underlying this functionality. Section IV gives a more complex example involving airplane sizing. A more detailed description of the functionality, mathematics and algorithms of Design Sheet is given in Fertig and Smith [1991]. A user's manual is also in preparation (Stubblefield and Fertig [1992]).

## II. Design Sheet Functionality

### Equations

In order to explain the workings of Design Sheet, we start with an extremely simple example using equations balancing the weight of an aircraft with its lift:

$$W_S = q\,C_L \tag{1}$$

$$q = \frac{1}{2}\rho v^2 \tag{2}$$

$$\rho = 0.00238 \tag{3}$$

These equations relate the five parameters,

| | |
|---|---|
| $W_S$ | Wing loading |
| $q$ | Dynamic pressure |
| $C_L$ | Lift coefficient |
| $v$ | Velocity |
| $\rho$ | Air density |

Copyright © 1992 by Rockwell International. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission

In using these equations in a spreadsheet, we would have to make choices about which equations to use for which variables. We have no choice for equation (3); it must be used to determine the density, which it does directly. We are left with four variables and two equations; this implies that we need to specify two independent variables and let the system of equations determine the other two. We have six possibilities:

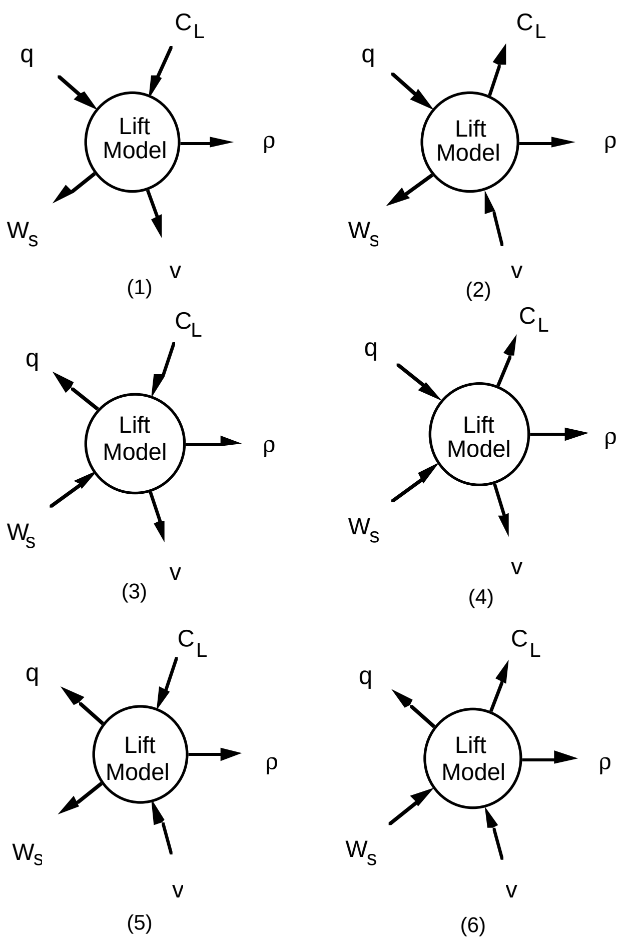


Figure II.1. Six permutations of independent variables for simple lift model.

Suppose the spreadsheet designer chooses case 5, with v and $C_L$ as independent variables. He could then use equation (2) to get q, and equation (1) to get $W_S$. If, later, he wanted to vary $W_S$, and see the effect on $C_L$, he would have to either interpolate using tables he already constructed, or re-write his spreadsheet for case (6).

Design Sheet avoids this rewrite step. It lets the user specify equality (and inequality) constraints among variables in almost arbitrary fashion, i.e.:

&lt;some algebraic expression&gt; =
&lt;some other algebraic expression&gt;,
&lt;some algebraic expression&gt; ≤
&lt;some other algebraic expression&gt;.

The user can read equations in from a file, or add them incrementally while using the system.

## Variables

Each equation relates some set of variables. In Design Sheet, variables can be in one of three states: independent, dependent, or undetermined. An independent variable is one the user is free to change the value of. A dependent variable is one that is derivable from the equation set and the independent variables. An undetermined variable is not derivable from the constraint set and the user has not specified it as independent. The system keeps track of which variables are dependent, which are independent, and which are undetermined. The user is free to change an undetermined variable to be independent at any time. Conversely, he can change an independent variable back to undetermined. Each time the user makes a change to the state of a variable, or adds or removes an equation, Design Sheet automatically determines and updates the status of all other affected variables.

For any independent variable the user can supply, change, or remove the value of that variable. When a value is changed, Design Sheet automatically updates the values of all other variables that depend on the independent variable. This calculation is not always simple, since it may require simultaneous solution of a system of equations.

## Example

As a simple example, suppose that the three equations above are entered into Design Sheet. The density, ρ, will be dependent (determined by equation (3)), while $W_S$, q, v, and $C_L$ will be undetermined. The state of the variables is therefore:

| | | | |
|---|---|---|---|
| $W_S$ | Wing loading | Undetermined | |
| q | Dynamic pressure | Undetermined | |
| $C_L$ | Lift coefficient | Undetermined | |
| v | Velocity | Undetermined | |
| ρ | Air density | Dependent | 0.00238 |

Suppose the user then specifies that the velocity, v, is independent. As soon as he does so, the system determines that it can use equation (2) to compute the dynamic pressure, q, and thus marks that variable as dependent. Wing loading, $W_S$, and lift coefficient, $C_L$, remain undetermined, and only density, ρ, has a value:

| | | | |
|---|---|---|---|
| $W_S$ | Wing loading | Undetermined | |
| q | Dynamic pressure | Dependent | |
| $C_L$ | Lift coefficient | Undetermined | |
| v | Velocity | Independent | |
| ρ | Air density | Dependent | 0.00238 |

If the user then chooses lift coefficient, $C_L$, as an independent variable, the remaining undetermined variable, $W_S$, becomes dependent:

| | | | |
|---|---|---|---|
| $W_S$ | Wing loading | Dependent | |

| | | | |
|---|---|---|---|
| q | Dynamic pressure | Dependent | |
| $C_L$ | Lift coefficient | Independent | |
| v | Velocity | Independent | |
| ρ | Air density | Dependent | 0.00238 |

If the user then supplies a value of 1.2 for $C_L$, and 50 for v, Design Sheet will compute values for the remaining two independent variables, as shown below:

| | | | |
|---|---|---|---|
| $W_s$ | Wing loading | Dependent | 10.2 |
| q | Dynamic pressure | Dependent | 8.49 |
| $C_L$ | Lift coefficient | Independent | 1.2 |
| v | Velocity | Independent | 50 |
| ρ | Air density | Dependent | 0.00238 |

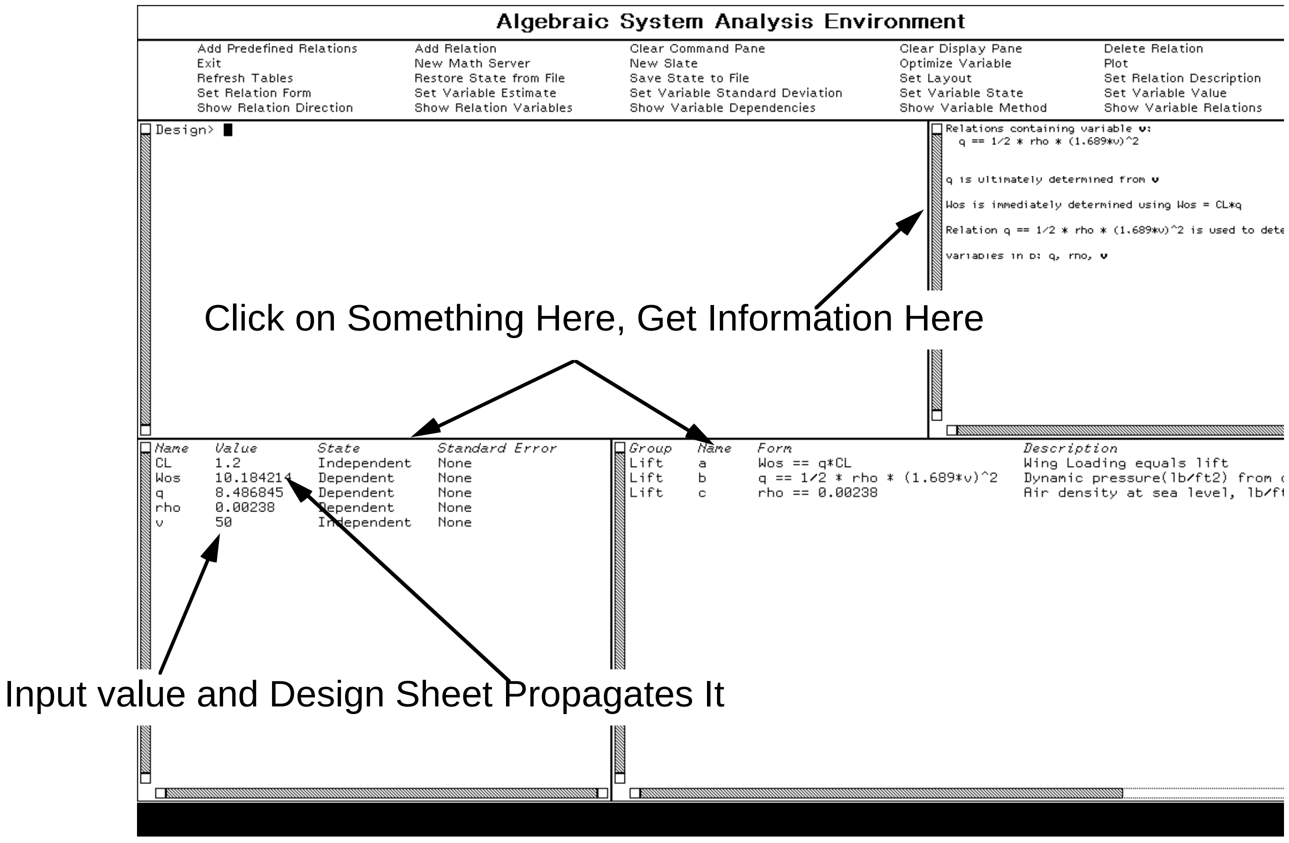
Figure II.2 is a screen snapshot of Design Sheet for this example. The equations for the example appear in the lower right pane, while a variable table similar to those used above appears in the lower left pane. Variable state changes and value changes were accomplished by simple point and click operations on the appropriate cell of the variable pane.

## Dependency Structure

With such a simple model, there is little room for confusion. When the models are much larger, the user needs help to determine *why* the system thinks a given variable is in a given state. For example, he may want to know what independent variables influence a dependent variable or which equation is being used to derive that variable. This kind of information is shown for dynamic pressure, q, in the upper right pane of Figure II.2. This information was obtained by a simple point and click operation on the variable q.

With large sets of equations it may also be difficult to determine why a given variable is not yet determined, i.e. what variables must be declared independent in order to cause that variable to become dependent. To help with this problem we have developed the concept of variable-equivalence. A set of undetermined variables are equivalent if and only if declaring any one of them independent results in the others becoming dependent. (It can be shown that this is a legitimate equivalence relation from the properties of constraint sets and the definition of dependency.) This equivalence relation partitions the undetermined variables into mutually exclusive classes so that it is easier to see what things must be specified in order for a particular variable to become determined.

To see how this concept applies, consider our simple example when all four of the variables, $W_s$, q, $C_L$, and v



```
Design>
```

```
Relations containing variable v:
  q == 1/2 * rho * (1.689*v)^2


q is ultimately determined from v

Wos is immediately determined using Wos = CL*q

Relation q == 1/2 * rho * (1.689*v)^2 is used to dete

variables in b: q, rho, v
```

```
Name    Value       State         Standard Error
CL      1.2         Independent   None
Wos     10.184214   Dependent     None
q       8.486845    Dependent     None
rho     0.00238     Dependent     None
v       50          Independent   None
```

```
Group   Name   Form                             Description
Lift    a      Wos == q*CL                      Wing Loading equals lift
Lift    b      q == 1/2 * rho * (1.689*v)^2     Dynamic pressure(lb/ft2) from 
Lift    c      rho == 0.00238                   Air density at sea level, lb/f
```

Figure II.2. Lift Model with velocity and $C_L$ declared independent. One can specify numerical values for independent variables and let Design Sheet determine the corresponding values for the dependent variables. One can "inspect/browse" through the model with simple mouse clicks to get dependency information.

are undetermined. In this case the classes are:

$$\{q, v\}\ \{C_L\}\ \{W_S\}.$$

The variables q and v occur in the same class because making either one of them independent causes the other to become determined. In contrast $C_L$ and $W_S$ appear by themselves, because no single variable can cause either of them to become determined.

Interestingly enough the structure of a set of variable equivalence classes changes once a variable is made independent. In that case, some of the remaining equivalence classes may collapse to form new, larger, equivalence classes. In our example, once v is made independent, q becomes dependent, and the remaining two equivalence classes collapse into the single class:

$$\{C_L, W_S\}.$$

This corresponds to the fact that there is only one degree of freedom remaining in the system of equations.

In large systems, having this kind of information is extremely useful for determining what variables must still be specified, and for finding bugs in the set of equations.

## Trade Studies

The major motivation behind creating the Design Sheet environment was to allow designers to more flexibly consider wider ranges of alternatives during design. This is facilitated by making it as easy as possible to perform different trade studies, where the designer wishes to see the effects of varying one or more parameters on the values of other parameters. As an example of Design Sheet's support for this activity, consider Figure II.3. There we show the user performing a simple trade study: plotting the value of the wing loading parameter versus the independent variable, velocity, for three separate values of the other independent variable, $C_L$.

In order to perform a trade study it is necessary for the user to select the independent variables that he wants to vary, supply ranges or values for those variables, and specify the dependent variables that he is interested in seeing. This information was supplied by simple point and click operations on the variable table, and by a menu driven dialog.

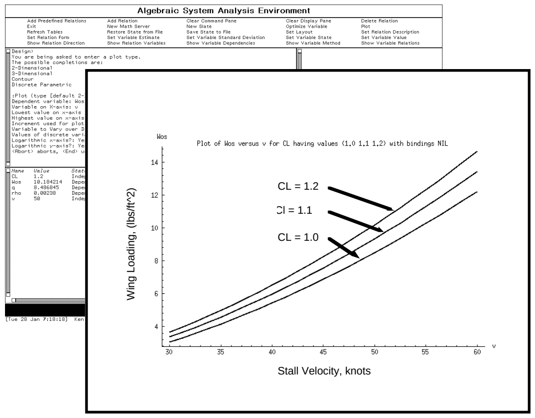
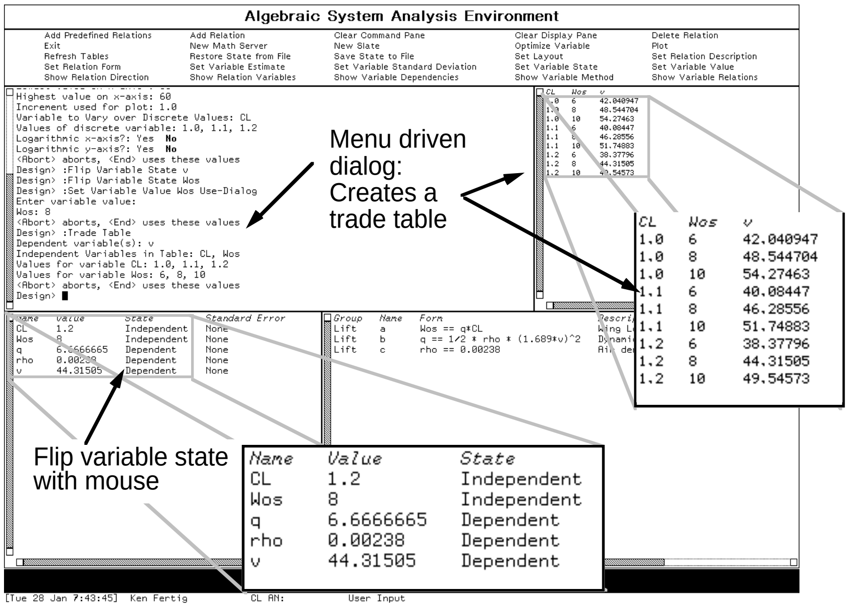


Figure II.3. Example Trade Study plotting wing loading versus velocity for various values of the lift coefficient.

The results of the trade study can be displayed in a number of ways. Figure II.4 shows a table representation of the data.

This was done for the case (3) of Figure II.1, in which $C_L$ and $W_S$ are taken as the independent variables. Three-dimensional plots and contour plots are also available. In



```
Highest value on x-axis: 60
Increment used for plot: 1.0
Variable to Vary over Discrete Values: CL
Values of discrete variable: 1.0, 1.1, 1.2
Logarithmic x-axis?: Yes  No
Logarithmic y-axis?: Yes  No
<Abort> aborts, <End> uses these values
Design> :Flip Variable State v
Design> :Flip Variable State Wos
Design> :Set Variable Value Wos Use-Dialog
Enter variable value:
Wos: 8
<Abort> aborts, <End> uses these values
Design> :Trade Table
Dependent variable(s): v
Independent Variables in Table: CL, Wos
Values for variable CL: 1.0, 1.1, 1.2
Values for variable Wos: 6, 8, 10
<Abort> aborts, <End> uses these values
Design> █
```

```
CL    Wos   v
1.0   6     42.040947
1.0   8     48.544704
1.0   10    54.27463
1.1   6     40.08447
1.1   8     46.28556
1.1   10    51.74883
1.2   6     38.37796
1.2   8     44.31505
1.2   10    49.54573
```

```
Name   Value       State         Standard Error
CL     1.2         Independent   None
Wos    8           Independent   None
q      6.6666665   Dependent     None
rho    0.00238     Dependent     None
v      44.31505    Dependent     None
```

```
Group   Name   Form                               Descri
Lift    a      Wos == q*CL                        Wing L
Lift    b      q == 1/2 * rho * (1.689*v)^2       Dynami
Lift    c      rho == 0.00238                     Air de
```

```
[Tue 28 Jan 7:43:45]  Ken Fertig        CL AN:        User Input
```

Figure II.4. Simple mouse clicks are all that are necessary to change the structure of the dependencies. Here we show case (3) of Figure II.1. We also show an example trade study presented as a table of values.

Section IV we show a more advanced example where the user displays level contours of gross takeoff weight, while superimposing the effects of inequality constraints on the same graph.

## Error Propagation

During the conceptual phases of design, the user is forced by necessity to make estimates of various parameters based on incomplete information. His models are often quite approximate. All too often the errors associated with these estimates are not explicitly determined or quantified. Design Sheet has built in techniques to overcome this difficulty.

As part of its solution of the constraint network, Design Sheet is able to determine the jacobian of the vector of dependent variables with respect to the vector of independent variables. This jacobian can then be used to perform a first order error analysis. In particular, if $\Sigma_x$ is the variance-covariance matrix of the independent variables, denoted by the vector x, and J is jacobian of the transformation from x to the dependent vector y, (i.e., $J_{ij} = \partial y_i/\partial x_j$), then the variance-covariance matrix of y is

$$\Sigma_y = J\,\Sigma_x\,J^T$$

We have implemented this in Design Sheet. Currently we only address the case where the user can assume that the errors in his current set of independent variables are uncorrelated (diagonal $\Sigma_x$). In such a case, Design Sheet correctly propagates these errors though the network, giving (correlated) errors for the dependent variables. We have plans to use this capability to implement Taguchi robust design methods for Design Sheet.

## Constrained Optimization

Optimization is a major component of the design process. The designer may be trying to maximize some performance parameter like range, he may be trying to minimize cost, maximize life, etc. He does this in the context of a set of constraints placed on the design, e.g., the plane must be able to cruise at x knots, it must be able to takeoff within y feet, etc. A limited version of constrained optimization is currently supported by Design Sheet. If a global optimum exists, the system will search through the set of inequality constraints to determine which should be the active ones at the optimum. We are currently enhancing this capability to deal with problems where there is no unconstrained optimum.

## Functional Features Summarized

We summarize the main features of Design Sheet in Table II.1.

Table II.1. Major Features of Design Sheet

| Feature | Significance |
|---|---|
| Equality and Inequality Constraints | Both inequality constraints and equality constraints are allowed by Design Sheet. It uses the equality constraints for propagation of values. It uses the inequality constraints in optimization and for display in trade studies. |
| Arbitrary Form for Constraints | The ability to input equations in any form is quite convenient. The user does not have to be concerned with the computational sequence. Rather, he can just input the equations in their "natural" form". A simple example may be a weight ratio constraint, like $W_f/W_o = 1.06(1 - W_x/W_o)$. |
| Incremental Addition and Deletion of Constraints | The user can incrementally add or delete constraints. This provides flexibility in model usage. For example, as the analysis becomes more detailed, the user inputs more refined modelling constraints. |
| Dynamic Modification of Dependencies | This is the big win with Design Sheet. The user is free to change which variables are independent and which are determined. The system automatically keeps track of the consequences of these choices and enforces consistency |
| Specify/Change Values of Independent Variables | This is the usual spreadsheet capability. Values are propagated automatically through the constraint network anytime the value of an independent variable changes. |
| Automatic Reduction and Solution of Simultaneous Systems | This, of course, is the major contribution that Design Sheet makes to the analysis process in design. We want to allow the user to think of his set of constraints as a set of relationships among his variables rather than as a computational sequence of operations. The more successful we are at that, the more the designer can think about designing rather than computation. |
| Dependency Structure | The user is given support for "browsing" through his model. He can find out which equations are being used, which independent variables influence a given dependent variable, and what equation is used to compute a given variable. |
| Trade Studies | Two- and three-dimensional plots, contour plots and general tables are all supported. |
| Error Propagation | As part of its solution of the constraint network, Design Sheet is able to propagate error information though the network, giving (correlated) errors for the dependent variables. |
| Constrained Optimization | A limited version of constrained optimization is currently supported by Design Sheet. |

Before we show Design Sheet for a more complicated example, we describe some of its inner workings in the next section.

## III. Design Sheet Technology: A Brief Description

Design Sheet is able to use the equality constraints to determine which variables are derivable from the declared set of independent variables. More than that, it is able to determine a *computational sequence* for evaluating the *values* of those determined variables, given the *values* of the independent variables. It does this by solving the non-linear set of algebraic equations which make up the constraint set. When possible, it uses symbolic methods; when this fails, it resorts to numerical methods.

The key to making this solution method possible is a set of algorithms within Design Sheet for decomposing the constraint set into tractable subsets, solving these subsets individually, and then combining these partial solutions into a complete solution.
The basic flow of Design Sheet is depicted in Figure III.1. As equations are added, the system builds up a graph relating the equations and the variables they contain.

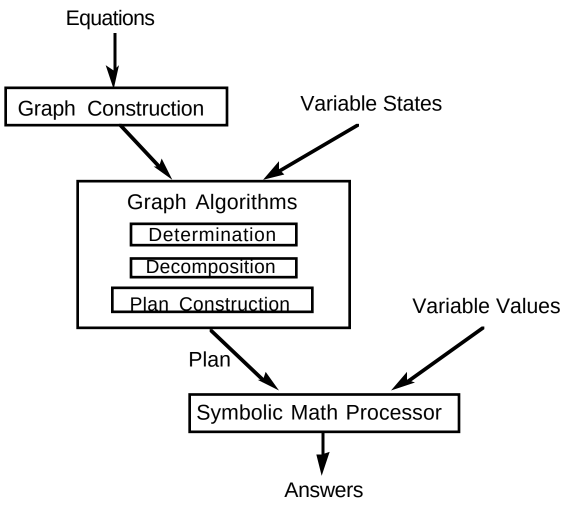


Figure III.1 The flow of Design Sheet

Graph manipulation algorithms are then used on this representation to 1) figure out which variables are determined, 2) decompose the system of equations into tractable subsets, and 3) determine the sequence of algebraic and numerical operations for deriving values of the determined variables. This computational sequence, and the values of the independent variables are then passed to a symbolic mathematics processor to derive values for dependent variables.

(a) $W_s = qC_L$
(b) $q = 1/2\ \rho v^2$
(c) $\rho = 0.00238$

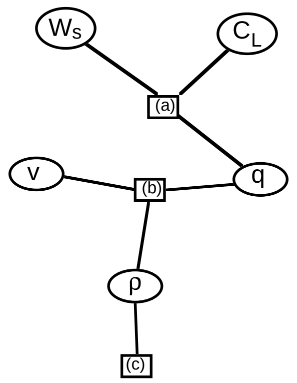


Figure III.2 Bipartite graph of equations in Lift model.

### Equation Graphs

As equations are added, Design Sheet incrementally constructs a bipartite graph (two classes of nodes, with edges only between members of different classes) relating the equations and variables. Given the three lift equations from Section 2, Design Sheet would construct the graph shown in Figure III.2.

In this graph, variable nodes are ovals, equation nodes are rectangles, and there is an edge between a variable node and an equation node if and only if the variable appears in that equation.

### Determination

When an equation is added or a variable is declared independent, Design Sheet must figure out which variables have become determined. There are two steps to this process: *labeling* and *propagation*.

A labeling is a partial directing of the graph such that:

1) Each equation is directed at exactly one variable, (making it a *committed* variable.)

2) All other edges connected to a committed variable are directed away from the variable.

3) All edges connected to an independent variable are directed away from the variable.

Intuitively a labeling is a pairing of equations and variables such that the equation could potentially be used to derive that variable. Figure III.3 shows one of the six possible labeling for the graph in Figure 3.2. (The other labeling are those where equations a and b are directed at different variables.)

(a) $W_s = qC_L$
(b) $q = 1/2\ \rho v^2$
(c) $\rho = 0.00238$

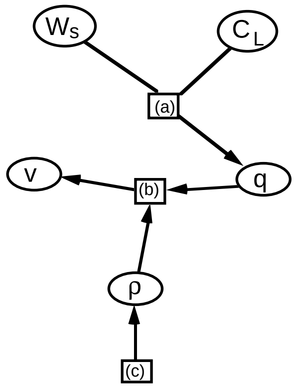


Figure III.3. One of six possible labeling of graph in Figure III.2. Equation (a) is directed towards q and (b) is directed towards v. q and v are not yet determined, however, since they have undetermined ancestors.

Each time an equation is added or a variable is declared independent, Design Sheet updates its current labeling. For example, if we declare v as an independent variable then equation (b) can no longer point at v. The only remaining possibility is to allow equation (b) to point at q. Given this fact, equation (a) can no longer point at q, and must be changed to point at either $W_S$ or $C_L$. One of these two remaining labeling is shown in Figure III.4. Labeling and re-labeling are performed using an incremental bipartite graph matching algorithm described in [Fertig & Smith].

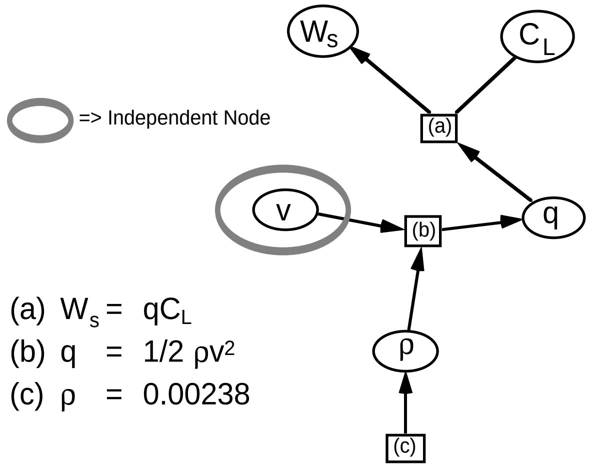


Figure III.4 Declaring v to be independent, causes a change in the labeling. Now q is determined since its ancestors are, but $W_s$ is not yet determined.

Once (re)labeling has been accomplished, a propagation technique is used to decide which variables are determined. Of course, any independent variable is determined. In addition any variable whose predecessors in the graph are determined, is also determined. For the labeling in Figure III.3, only one variable, ρ, is determined (it has no predecessors). Once v is declared independent there are three determined variables, ρ, v, and q. (q is determined since both its predecessors, ρ, and v are determined.)

Both the labeling and propagation algorithms are (worst case) linear in the number of variables and equations present. In practice, these algorithms take only a fraction of a second on systems containing over 100 equations.

## Decomposition and Plan Construction

Once labeling and propagation have taken place Design Sheet constructs a computational plan for computing the values of determined variables. For our example, if $W_S$ and $C_L$ are declared independent, the labeled graph is shown in Figure III.5

This labeling naturally decomposes the problem of solving a 3x3 system of equations into the sequential steps

Use equation (a) to get q
Use equation (c) to get ρ
Use equation (b) to get v

Design Sheet finds this decomposition and builds the resulting plan by topologically sorting the determined

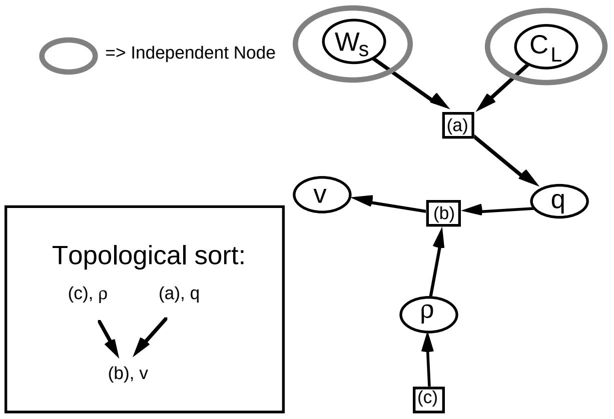


Figure III.5. Directed bipartite graph showing computational strategy when $W_S$ and $C_L$ are independent

portion of the labeled graph. It then passes this plan, along with the appropriate values for the independent variables, to a symbolic math processor to determine the values of the dependent variables. If a symbolic solution for a given equation cannot be found, Design Sheet will then resort to numerical methods.

## Strongly Connected Components

So far, we have considered only a very simple example of determination and decomposition. Figure III.6 gives a slightly more complicated set of weight equations, together with one possible labeling for the corresponding equation graph. This set of equations has the property that once R and $S_{ref}$ are declared independent, the remaining seven equations can be solved simultaneously for the remaining variables. However, it is not possible to decompose this set of equations into a sequential set of simple operations, as with our previous example.

This characteristic is manifested in the equation graph by the presence of directed cycles. More precisely, the equations and (dependent) variables in the graph are part of a *strongly connected component* (SCC). (Formally, a SCC is a maximal subset of nodes in a directed graph such that there is a path from every node in the set to every other.)

The possibility of SCCs in an equation graph has no effect on the labeling process for the graph. However, it does complicate the process of deciding which variables are determined. Operationally, this reduces to determining if, for a given variable, either:

1) the variable is independent,
2) all of the variable's predecessors are determined,

or

3) the variable is part of a SCC and the predecessors of the SCC are determined.

(a) $W_e/W_o = 2.61 * W_o^{(-0.1)} * (W_o/S_{ref})^{(-0.05)}$

(b) $W_o = W_f + W_e$

(c) $W_f/W_e = 1.06(1-W_x/W_o)$

(d) $W_{LO}/W_o = 0.97$

(e) $W_{alt}/W_{LO} = 0.985$

(f) $W_x/W_{ec} = 0.995$

(g) $W_{ec}/W_{alt} = \mathrm{Exp}[-0.00043\ R]$

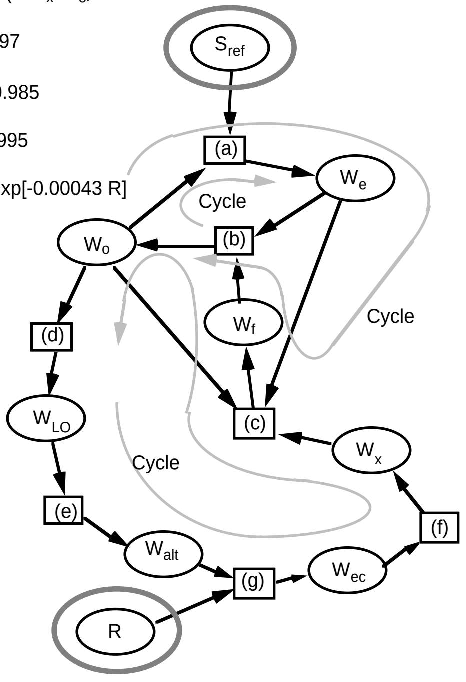


Figure III.6 Simplified set of aircraft sizing equations for the mission shown in Figure IV.1. When the range R and the reference area, $S_{ref}$, are treated as independent parameters, the seven equations are sufficient to solve for the remaining seven variables. We show one of the possible labeling. Only three of the four cycles in the graph are shown.

Here the predecessors of a SCC are those variables not in the SCC that have edges directed into the SCC. In Figure III.6, the two predecessors of the SCC are the independent variables R and $S_{ref}$. These are both determined, so the variables in the SCC are determined (by simultaneous solution of the equations in the component).

For a given variable, we can determine whether it is part of a strongly connected component in time that is (worst case) linear in the size of the graph. As a result, this addition to the determination process does not adversely affect the speed of Design Sheet.

## Component Decomposition

The presence of SCCs also has an impact on the decomposition and plan generation phases of Design Sheet. As before, the plan for an equation graph is generated by topologically sorting the labeled equation graph, however. SCCs are treated as single meta-nodes in the graph. Thus, a plan for an equation graph might look like:

Use equation 3 to get x
Solve SCC 1 to get w, u, and v
Use equation 5 to get z
Solve SCC 2 to get q and r
…

Design Sheet must also find a plan for simultaneously solving each SCC. One could ask a general purpose algebraic system solver to step in at this point and try and solve the SCC either symbolically or numerically. Since it is rare, in general, to finding symbolic solutions to nonlinear systems of equations, one is typically forced to solving them numerically. In choosing a numerical approach for Design Sheet, we did not seek the most rapid solution method, but rather the most robust. The robustness of nonlinear equation solvers is strongly affected by the dimensionality of the system being solved. Often, it is orders of magnitude easier to solve serially two problems of one equation, each in one unknown, rather than two equations in two unknowns, in parallel. As a consequence, it would be to our advantage if we could further decompose the SCCs. It turns out that this is usually possible by choosing an appropriate subset of *decomposing variables* in the SCC. We illustrate with an example. In Figure III.6, if $W_o$ were known, Design Sheet could sequentially compute values for $W_e$, $W_{LO}$, $W_{alt}$ $W_{ec}$, and $W_x$. Having both $W_e$ and $W_x$ Design Sheet could then determine $W_f$,. Using $W_f$, and $W_e$,, equation (b) could be used to get a new value for $W_o$. The computational steps in this process define a fixed point problem: $W_o = H(W_o)$, where H represents the process just outlined. We are left trying to solve one fixed point equation in one unknown rather that seven equations in seven unknowns. The numerical algorithms will be correspondingly much more robust.

Not all variables in a SCC are equally useful as decomposing variables. For example, if $W_f$, were chosen as a decomposing variable, no other variables could be determined without choosing a second decomposing variable.

Since the choice of SCC decomposing variables has enormous impact on the convergence properties of numerical iteration, Design Sheet searches to find a minimal set of such variables for each SCC. The system uses a heuristically guided branch and bound search described in [Fertig and Smith]. In general, finding the optimal decomposing variables for a simultaneous system is NP-hard. As a result, the search algorithm is exponential in the number of decomposing variables required. However, in practice, the algorithm takes less than a second to find optimal decomposing variable sets for SCCs containing more than 40 equations. We therefore expect that this process will not pose a bottleneck for large applications.

## Related Work

Bouchard[1988,e.g.] has created a design system which allows the user to input directed constraints among his design variables. The equations are solved numerically, allowing for rapid production of trade studies. No interactive facility is provided for changing which are the independent variables. Nevertheless, a factor of ten increase in turn around time is reported using this system on real design problems.

Serrano[1987] has developed a set of algorithms using bipartite matching and strong component identification for solving systems of equations. Though no effort was given to breaking down components, both logical and algebraic constraints were considered. Further, he does not directly address issues involved in *incrementally* adding and deleting constraints.

Research in constraint-based reasoning has been going on for some time. A reasonable account of this area is given by Ward[1989]. Wilhelm[1991] provides a recent example in the area of tolerance bound propagation and synthesis. Navinchandra, Fox, and Gardener[1991] consider the issues in user directed constraints. This is useful for constraints implemented, for example, as subroutines that can only compute their outputs given their inputs. Navinchandra, et al's work is closest to that discussed in this paper. Whereas they concentrate on unidirectional constraints, we concentrate on flexible changes in the dependency structure, incremental addition and deletion of constraints, and reducing the dimensionality of the simultaneous equation set to the smallest possible degree in order to have the best chance in solving the constraint network.

Each of Ward[1989], Wilhelm[1991], and Navinchandra, et al[1992] indicate that the basic ideas of constraint management can be traced to Sutherland[1963]. In fact, the CAD arena is a big driver in this research area. Barford[1987], for example, investigates the construction of systems that can solve constraints associated with solid modelers very fast, so that the designer can get real time feedback on the structure he is designing as he changes certain dimensions. Cognition, Corp. has taken this one step further by developing an integrated system that allows conceptual drafting (using variational geometry) together with mathematical modeling. The constraint engine they use for the geometric reasoning is the same one they use for the algebraic reasoning. Their system is quite sophisticated, allowing for general plotting and table generation. They, too, allow for incremental addition/deletion of constraints, and user changeability of which variables are independent and which are derived. However, the ability of the system to decompose complex systems of equations appears to be limited.

## IV. A More Complete Example

Let us now consider a slightly more complex example. We take this from Raymer[1989]. He presents a conceptual design example for a single-seat aerobatic light plane. The initial problem is to size the aircraft using constraints on takeoff distance, climb rate, turn rate, cruise speed, range, and so forth. The specific design goals are:

$V_{max} \geq 130$ kts, $V_{stall} \leq 50$ kts
Takeoff $\leq 1000$ ft over 50 ft
Rate of Climb $\geq 1500$ ft/min
Cruise Range 280 naut. miles
$V_{cruise} = 115$ kts
$W_{crew} = 200$ lbs
Turn Rate $\geq 30$ deg/sec

The set of constraints that relate performance parameters, mission requirements, and airplane characteristics can be divided into three types of equations:

- historical correlations
- definitional constraints
- simple physical models

For example, the weight balance equations typically include a historical correlation which relates the empty weight fraction to parameters such as aspect ratio, wing loading, power loading, etc. Raymer[1989] uses the following for the initial sizing effort in the current example:

$$W_e/W_o = F_A * 0.59 * Wo^{a1} * A^{a2} * (hp/W_o)^{a3} * (W_o/S_{red})^{a4} * (1.151*V_{max})^{a5}$$

where

| | | |
|---|---|---|
| $W_o$ | Gross takeoff weight | lbs |
| $W_e$ | empty weight | lbs |
| hp | engine horsepower | hp |
| $S_{ref}$ | wing area | $ft^2$ |
| $V_{max}$ | maximum velocity | knots |
| $F_A$ | adjustment factor | - |

($F_A$ is an adjustment factor to account for differences between aerobatic flying and cruising.)

An example of a definitional constraint is that the gross takeoff weight is the sum of its parts:

$$W_o = W_{crew} + W_{pay} + W_{fuel} + W_e,$$

where $W_{crew}$, $W_{pay}$, $W_{fuel}$ are the crew, payload and fuel weights respectively.

Simple approximations based on mechanistic models are also used. Thus, we have the range equation:

$$W_{ec}/W_{alt} = \text{Exp}[- 6076*R*cbhp/3600)/(550*etap*(L/D)_{cr})]$$

where R is the range, cbhp is a fuel efficiency factor, etap is a propeller efficiency, $(L/D)_{cr}$ is the lift-to-drag ratio during cruise conditions, Wec is the weight at the end of the mission, and $W_{alt}$ is the weight at altitude, just after takeoff. (See mission profile in figure below.)

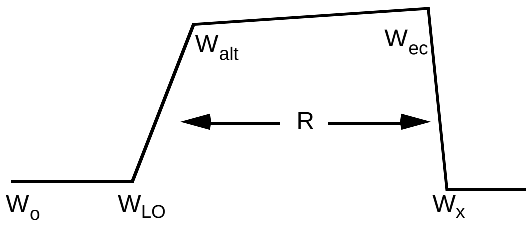


Figure IV.1 Mission profile for small plane sizing problem.

The basic set of equations for each operating constraint are derived by relating the lift, thrust, weight, and drag of the aircraft in the different operating regimes. The equations for these factors at a given operating condition, indicated by the subscript oc, are:

$$
\begin{array}{lll}
D_{oc} & = & q_{oc} * S_{ref} * CD_{oc} \\
L_{oc} & = & q_{oc} * S_{ref} * CL_{oc} \\
CD_{oc} & = & CD_{o} \;+\; K * CL_{oc}^{2} \\
\rho_{oc} & = & \rho_{sealevel} \;+\; drhodh \; * h_{oc} \\
sigma_{oc} & = & \rho_{oc}/\rho_{sealevel} \\
q_{oc} & = & 1/2 * \rho_{oc} * (1.689 * V_{oc})^{2} \\
T_{oc} & = & 550 * etap * hp_{oc}/(V_{oc} * 1.689) \\
hp_{oc} & \leq & hp_{sealevel} * [sigma_{oc} - (1 - sigma_{oc})/7.55]
\end{array}
$$

where D is the drag, L the lift, CL and CD the lift and drag coefficients, h is the altitude, drhodh is the change in density with respect to altitude, and T the thrust.

When the operating condition is cruise, for example, we have $L_{cruise} = W_{cruise} = W_{alt}$, and $D_{cruise} = T_{cruise}$. For climbing at an angle $\gamma_{climb}$, we have

$$
\begin{array}{lll}
W_{climb} & = & W_{alt} \\
T_{climb} & = & D_{climb} + W_{climb} * Sin[\gamma_{climb}] \\
L_{climb} & = & W_{climb} * Cos[\gamma_{climb}].
\end{array}
$$

Design Sheet accepts the equations in the form above and uses them to derive the consequences of various

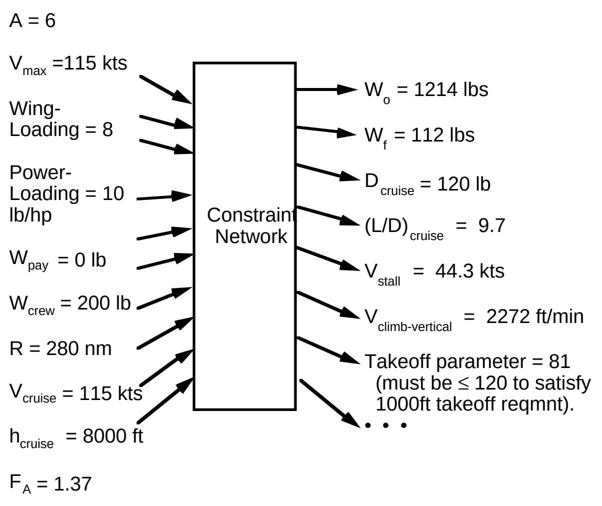
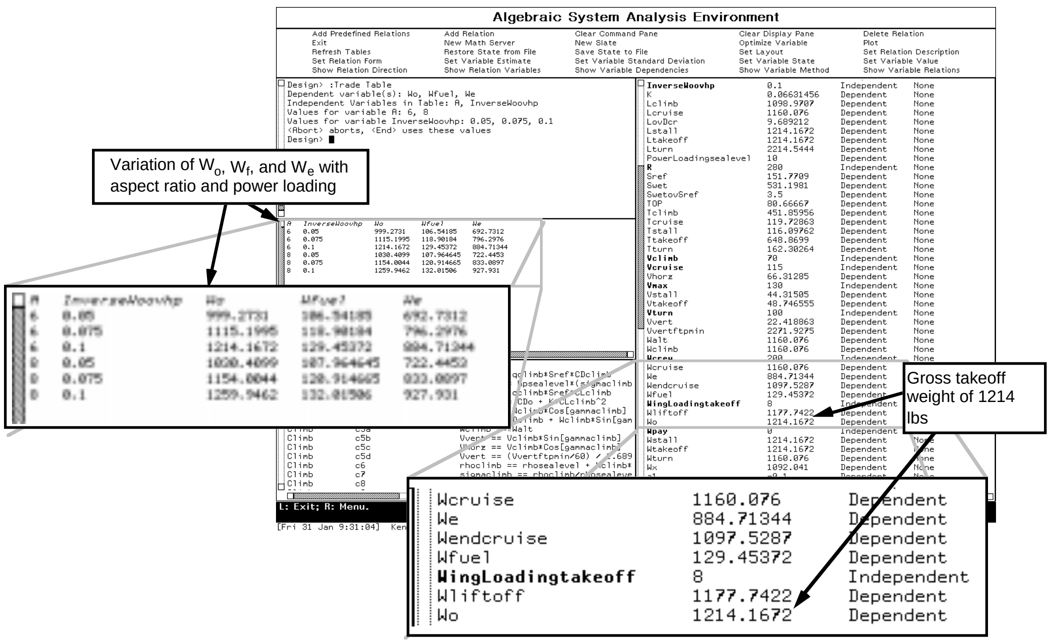


Figure IV.2 Base Design Case for Light Plane Sizing example. The constraint network involves 93 constraints among 110 variable. The solution process decomposes this network into two strong components, one of size 16, the other of size 5. The remaining 72 equations all have symbolic solutions, and are solved serially, one at a time.

constraints. The entire constraint graph involves some 93 equations among 110 variables. We do not show this network in a figure; it is too complicated. Happily, Design Sheet had little difficulty in solving the system. The base design case is shown in Figure IV.2

A screen snapshot of this situation is shown in Figure IV.3. The layout is different than that shown in Section II ad was chosen to allow for maximum viewing of the variable table. The user has multiple layout choices available to him in Design Sheet.



```
Design> :Trade Table
Dependent variable(s): Wo, Wfuel, We
Independent Variables in Table: A, InverseWoovhp
Values for variable A: 6, 8
Values for variable InverseWoovhp: 0.05, 0.075, 0.1
<Abort> aborts, <End> uses these values
Design> █
```

```
A   InverseWoovhp   Wo          Wfuel        We
6   0.05            999.2731    106.54185    692.7312
6   0.075           1115.1995   118.90184    796.2976
6   0.1             1214.1672   129.45372    884.71344
8   0.05            1030.4099   107.964645   722.4453
8   0.075           1154.0044   120.914665   833.0897
8   0.1             1259.9462   132.01506    927.931
```

```
                    qclimb*Sref*CDclimb
                    hpsealevel*(sigmaclimb
                    qclimb*Sref*CLclimb
                    CDo + K*CLclimb^2
                    Wclimb*Cos[gammaclimb]
                    Dclimb + Wclimb*Sin[gam
Climb    c5a        Wclimb == Walt
Climb    c5b        Vvert == Vclimb*Sin[gammaclimb]
Climb    c5c        Vhorz == Vclimb*Cos[gammaclimb]
Climb    c5d        Vvert == (Vvertftpmin/60) / 1.689
Climb    c6         rhoclimb == rhosealevel + hclimb*
Climb    c7         sigmaclimb == rhoclimb/rhosealeve
Climb    c8
```

```
InverseWoovhp           0.1          Independent   None
K                       0.06631456   Dependent     None
Lclimb                  1098.9707    Dependent     None
Lcruise                 1160.076     Dependent     None
LovDcr                  9.689212     Dependent     None
Lstall                  1214.1672    Dependent     None
Ltakeoff                1214.1672    Dependent     None
Lturn                   2214.5444    Dependent     None
PowerLoadingsealevel    10           Dependent     None
R                       280          Independent   None
Sref                    151.7709     Dependent     None
Swet                    531.1981     Dependent     None
SwetovSref              3.5          Dependent     None
TOP                     80.66667     Dependent     None
Tclimb                  451.85956    Dependent     None
Tcruise                 119.72863    Dependent     None
Tstall                  116.09762    Dependent     None
Ttakeoff                648.8699     Dependent     None
Tturn                   162.30264    Dependent     None
Vclimb                  70           Independent   None
Vcruise                 115          Independent   None
Vhorz                   66.31285     Dependent     None
Vmax                    130          Independent   None
Vstall                  44.31505     Dependent     None
Vtakeoff                48.746555    Dependent     None
Vturn                   100          Independent   None
Vvert                   22.418863    Dependent     None
Vvertftpmin             2271.9275    Dependent     None
Walt                    1160.076     Dependent     None
Wclimb                  1160.076     Dependent     None
Wcrew                   200          Independent   None
Wcruise                 1160.076     Dependent
We                      884.71344    Dependent
Wendcruise              1097.5287    Dependent
Wfuel                   129.45372    Dependent
WingLoadingtakeoff      8            Independent
Wliftoff                1177.7422    Dependent
Wo                      1214.1672    Dependent
Wpay                    0            Independent
Wstall                  1214.1672    Dependent     None
Wtakeoff                1214.1672    Dependent     None
Wturn                   1160.076     Dependent     None
Wx                      1092.041     Dependent     None
```

```
[Fri 31 Jan 9:31:04]   Ken
```

Figure IV.3 Screen display for small aircraft sizing problem. Equations are shown in lower left in a scrollable window. Variables are shown on right, again in a scrollable window. The user has just asked for a table producing gross take off weight, weight of fuel, and empty weight as a function of aspect ratio and hp/$W_o$.

For the base design case, we have chosen the following set of variables to be independent:

| Design Sheet Variable | Definition | Units | Initial Value |
|---|---|---|---|
| A | aspect ratio | - | 6 |
| R | range | nautical miles | 280 |
| Inverse Woovhp | ratio of hp at sea level to gross takeoff weight | hp/lb | 0.1 |
| Wing Loading takeoff | Gross takeoff weight, Wo, divided by wing reference area, Wo/Sref | $lb/ft^2$ | 8 |
| Wcrew | crew weight | lb | 200 |
| Wpay | payload | lb | 0 |
| Vcruise | cruise velocity | knots | 115 |
| hcruise | elevation at cruise | ft | 8000 |
| hstall | elevation at which to compute stall | ft | 0 |
| hturn | elevation at which to compute turn constraint | ft | 0 |
| htakeoff | elevation of airport at which takeoff is computed | ft | 0 |
| hclimb | elevation at which to compute climb | ft | 0 |
| CLmax | max lift coefficient | - | 1.2 |
| Vmax | maximum design velocity | knots | 130 |
| Vturn | velocity at which sustained turn constraint is computed | knots | 100 |
| Vclimb | speed at which to compute climb constraints | knots | 70 |
| $F_A$ | adjustment factor to weight equations to account for difference between aerobatic flying and cruising | - | 1.37 |

Table IV.1 Independent variables for light plane sizing example.

For this independence set and initial settings of the variables, Design Sheet computes the gross takeoff weight as 1214 lbs.

Basically, this set of 93 equations breaks down into one strong component of 16 variables involving the weight equations and cruise conditions (to get the lift to drag ratio for cruise conditions), another strong component of 5 equations involving climb constraints, and 72 single equations that can be solved serially. Each of the strong components has a single decomposing variable, and so can be reduced to numerically solving one equation in one unknown.

We note that for the current setting of the independent variables, all inequality constraints are satisfied. The contour plotting capability in Design Sheet allows the user to display these inequality constraint regions which define feasible design space. The boundaries of the feasible region are displayed on the contour plot along with the level curves of the dependent variable being studied. Consider Figure IV.4, for example. There the system is plotting level contours of gross takeoff weight as a function of power loading and wing loading. The boundaries of the inequality constraint regions are displayed as cross hatched lines. The user can select which constraints to show. This display follows the "sizing matrix plot" form described in Raymer[1989].

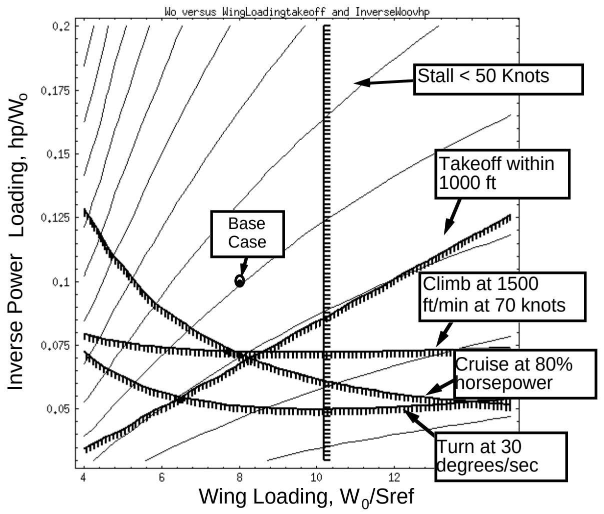
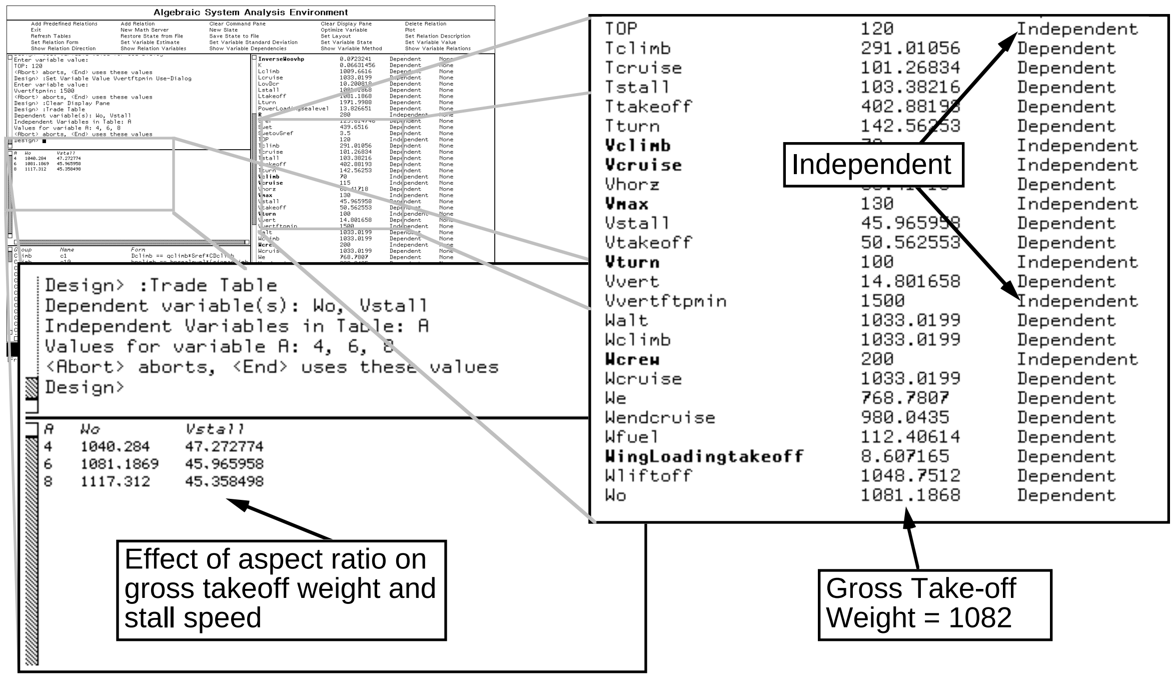


Figure IV.4 Trade study showing constant contours of gross takeoff weight as a function of wing loading and power loading. The boundaries or various constraints are superimposed on this plot. Thus, only planes to the left of the vertical line at a wing loading near 10 have a stall speed less than 50 knots, for example.

As an example of changing the dependency structure in the model, again consider Figure IV.4. We see that for this case, the climb constraint and the takeoff constraint would be active at the minimum gross takeoff weight. We may want to run a constrained optimization at this point. Alternatively, we may just want to specify that these two constraints are exactly satisfied and see what the takeoff weight is. We see from Figure IV.2 (or, if you have good eyes from Figure IV.3) that the vertical climb rate, Vvertftpermin, is currently 2272 ft/min and that the takeoff parameter is 81. We can try to adjust by hand the wing loading variable (WingLoadingtakeoff) and the inverse power loading variable (InverseWoovhp) until vertical climb is exactly 1500 ft/min and the takeoff parameter is exactly 120. (The takeoff parameter should equal 120 for the aircraft to clear 50 feet elevation within 1000 feet.) Alternatively, we can easily change the dependency structure to make the takeoff parameter and the vertical climb rate the independent parameters. We do this by first changing the wing loading variable and the inverse power loading variable to undetermined. This makes the vertical climb rate and the takeoff parameter undetermined. The user can now make these independent. Once he does so, he is free to specify their values. We show this case in Figure IV.5. We see that the gross takeoff weight is 1082 lbs. We also show the results of a small trade study in that figure, where we have varied the aspect ratio. Interestingly, in this case the 93 equations broke down into one strong



```
TOP                     120          Independent
Tclimb                  291.01056    Dependent
Tcruise                 101.26834    Dependent
Tstall                  103.38216    Dependent
Ttakeoff                402.88193    Dependent
Tturn                   142.56253    Dependent
Vclimb                  70           Independent
Vcruise                 [illegible]  Independent
Vhorz                   [illegible]  Dependent
Vmax                    130          Independent
Vstall                  45.965958    Dependent
Vtakeoff                50.562553    Dependent
Vturn                   100          Independent
Vvert                   14.801658    Dependent
Vvertftpmin             1500         Independent
Walt                    1033.0199    Dependent
Wclimb                  1033.0199    Dependent
Wcrew                   200          Independent
Wcruise                 1033.0199    Dependent
We                      768.7807     Dependent
Wendcruise              980.0435     Dependent
Wfuel                   112.40614    Dependent
WingLoadingtakeoff      8.607165     Dependent
Wliftoff                1048.7512    Dependent
Wo                      1081.1868    Dependent
```

```
Design> :Trade Table
Dependent variable(s): Wo, Vstall
Independent Variables in Table: A
Values for variable A: 4, 6, 8
<Abort> aborts, <End> uses these values
Design>
```

```
A   Wo          Vstall
4   1040.284    47.272774
6   1081.1869   45.965958
8   1117.312    45.358498
```

Algebraic System Analysis Environment
Independent
Effect of aspect ratio on gross takeoff weight and stall speed
Gross Take-off Weight = 1082

Figure IV.5 Small aircraft sizing problem with vertical climb rate and take-off parameter as independent rather than wing loading and hp/$W_O$. The user has specified a vertical climb rate of 1500 ft/min and a take-off parameter of 120 (corresponding to taking off in 1000 ft and clearing 50 ft elevation). The resulting take-off weight is 1082 lbs. A small trade study, varying aspect ratio, is shown in the pane at the left-middle.

component of 36 equations, with the remainder being solvable serially. The strong component required two decomposing variables, implying that the problem of solving the 93 equations was reduced to solving a non-linear system of two equations in two unknowns, and 91 single equations.

As mentioned in previous sections, Design Sheet has the capability of computing sensitivity derivatives and of propagating errors through the constraint network. It is interesting to demonstrate this capability for the light airplane sizing example. For the base case considered, we have the following sensitivity derivatives produced by Design Sheet:

| Independent Variable, x | $\partial W_o/\partial x$ |
|---|---|
| A | 24.8 |
| R | 0.964 |
| InverseWoovhp | 3724. |
| WingLoadingtakeoff | -47.6 |
| Wcrew | 4.21 |
| Wpay | 4.21 |
| Vcruise | 3.39 |
| hcruise | -0.0066 |
| hstall | 0 |
| hturn | 0 |
| htakeoff | 0 |
| hclimb | 0 |
| CLmax | 0 |
| Vmax | 4.86 |
| Vturn | 0 |
| Vclimb | 0 |
| $F_A$ | 2718 |

The zeros in this table reflect that fact that these independent variables played no rule in defining the actual gross takeoff weight (when wing loading and power loading are specified). The large factors are power loading (InverseWoovhp = hp/$W_O$) and the weight adjustment factor, $F_A$. The later was chosen by Raymer in his example as 1.37 to adjust the weight equation to account for differences between aerobatic flying and cruising. It turns out that small errors in this factor are amplified in the initial weight determination, using the given historical correlations. Assuming a standard error of 0.1 in this factor, and no errors in the other independent variables, Design Sheet computes a standard error of 271.8 lbs for $W_O$. See Figure IV.6.

## V. Conclusions

The name "Design Sheet" was chosen to emphasize the similarity between this tool and ordinary spreadsheets. However, Design Sheet differs considerably form typical spreadsheets. These differences are summarized in the Table II.1. The most significant distinction is that the user is freed from having to determine the computational sequence of operations to obtain values of his output variables given values of his input variables. Importantly, the technology to free him from this burden computes the relationships so fast that it can be done interactively. This also allows the designer to change which variables he wants to consider as inputs to his model and which he wants to consider as outputs, thus greatly increasing his flexibility in performing trade studies and in investigating different regions of the design space. This new capability permits the designer to explore many design options that previously could not be explored due to the time and cost

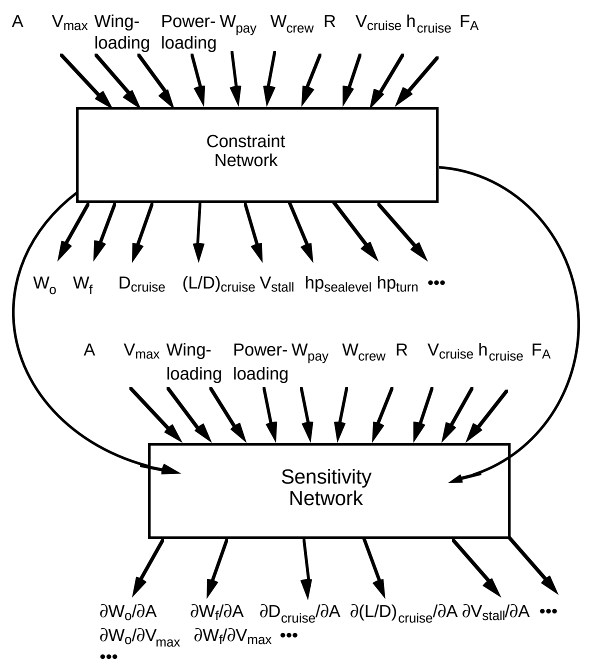

A
$V_{max}$
Wing-loading
Power-loading
$W_{pay}$
$W_{crew}$
R
$V_{cruise}$
$h_{cruise}$
$F_A$
Constraint Network
$W_o$
$W_f$
$D_{cruise}$
$(L/D)_{cruise}$
$V_{stall}$
$hp_{sealevel}$
$hp_{turn}$
•••
Sensitivity Network
$\partial W_o/\partial A$
$\partial W_f/\partial A$
$\partial D_{cruise}/\partial A$
$\partial (L/D)_{cruise}/\partial A$
$\partial V_{stall}/\partial A$
•••
$\partial W_o/\partial V_{max}$
$\partial W_f/\partial V_{max}$
•••
•••

Figure IV.6 Design Sheet can use its knowledge of the constraint network to construct a sensitivity network. This latter network involves a set of *linear* equations for the derivatives. These derivatives can then be used to propagate errors through the network. Thus,

$$\sigma^2_{W_o} = (\partial W_o/\partial A)^2\sigma^2_A + ... + (\partial W_o/\partial F_A)^2\sigma^2_{FA}.$$

In the example shown, the standard error of the weight adjustment, $F_A$, factor is amplified from below ten percent of $F_A$ to more than twenty percent of $W_O$. ($F_A = 1.37 \pm 0.1 \Rightarrow W_o = 1214 \pm 272$)

associated with developing the necessary analysis software. The ability to easily perform sensitivity analysis early in the design process should result in future designs that are more robust with respect to uncontrollable variations in both the manufacturing process and in use. While Design Sheet is still in development, it has been tried on problems involving more than 100 variables with excellent response times.

## References

Barford. Lee Alton [1987], "A Graphical, Language-Based Editor for Generic Solid Models Represented by Constraints," PhD dissertation, Cornell University. Report MRC-TM-87-02, Hewlett-Packard Laboratories, Palo Alto, CA 94304-1317.

Bouchard, E.E., G.H. Kidwall, and J. Edward Rogan, [1988], " The Application of Artificial Intelligence Technology to Aeronautical System Design," AIAA-88-4426, Presented at the AIAA Aircraft Design Systems and Operations Meeting, September 7-9, 1988, Atlanta, Georgia.

Cognition, Corporation, [1990], "Mechanical Advantage," software description manual, 755 Middlesex Turnpike, Billerica, MA 01821.1990.

Fertig, Kenneth W. and David E. Smith, "Design Sheet: An Engineers Spreadsheet," Rockwell Palo Alto Laboratory Technical Report, December, 1991.

Fertig, Kenneth W. and David E. Smith, "Using Graph Theory and Intelligent Search on Systems of Non-Linear Equations," Rockwell Palo Alto Laboratory Technical Report, in preparation.

King, Bob. 1989, Better Designs in Half the Time, GOAL/QPC, Methuen, MA.

Hauser, John R. and Don Clausing. 1988. "The House of Quality," Harvard Business Review, Vo. 66, No. 3, p. 63.

Stubblefield, P., K.W. Fertig, and D. E. Smith, "Design Sheet, A Users Manual," Rockwell Palo Alto Laboratory Technical Report, in preparation.

National Materials Advisory Board. 1991, NMAB-455, Enabling Technologies for Unified Life-Cycle Engineering of Structural Components, National Academy Press, Washington, D.C.

D. Navinchandra, Mark S. Fox, Eric S. Gardener, "On the Role of Constraints in Concurrent Design," to appear in Journal of the Institute of Industrial Engineering, 1992.

Raymer, Daniel P., [1989], Aircraft Design: A Conceptual Approach, AIAA Education Series, J.S. Przemieniecki, Editor-in-Chief, American Institute of Aeronautics and Astronautics, Inc., Washington.

Serrano, David [1987], "Constraint Management in Conceptual Design," PhD Dissertation, Massachusetts Institute of Technology, Dept. of Mechanical Engineering, October, 1987.

Sutherland, I. E., [1963], "Sketchpad - A Man-Machine Graphical Communication System," Technical Report #296, MIT Lincoln Lab., Cambridge, Massachusetts.

Ward, Allen Corlies, [1989], "A Theory of Quantitative Inference for Artifact Sets, Applied to a Mechanical Design Compiler," PhD dissertation, Massachusetts Institute of Technology, Dept. of Mechanical Engineering, January, 1989.

Wilhelm, Robert G., [1991], "Computer Methods for Tolerance Synthesis," PhD dissertation, Mechanical Engineering, University of Illinois at Urbane-Champagne.