533606

Final Report—Insensitivity of Control System Performance to Controller and System Parameters

NASA Ames Project Number NAG21311
By William S. Levine, Project Director
Department of Electrical And Computer Engineering
University of Maryland
2415 A. V. Williams Building
College park, Maryland 20742-3285
April 1, 1999 to June 24, 2001

Final Report—Insensitivity of Control System Performance to Controller and System Parameters

NASA Ames Project Number NAG21311
By William S. Levine, Project Director
June 24, 2001

Abstract:

An experimental and theoretical study of the CONDUIT sensitivity tools was conducted. The literature on sensitivity in nonlinear programming was reviewed to see in what ways it could be applied to CONDUIT. One result of this was the conclusion that the current insensitivity measure in CONDUIT is the right one. The question of scaling of the specifications in CONDUIT was also studied. Many simple examples were created, analyzed and, in most cases, solved. However, no general solution to the scaling problem was found. Instead, the appropriate scaling needs to be determined by review of a reasonable number of real aircraft design problems.

Introduction:

CONDUIT is an assistant to a control system designer. One of the most important things for an assistant to do is to communicate effectively with his or her supervisor. It is especially important for the assistant to tell its supervisor when it is having a problem and, if possible, why. CONDUIT already has very good ways to communicate progress and status to its supervisor. Sometimes CONDUIT stops making significant progress on a problem. Although it is fairly obvious when this happens, it is not necessarily obvious why. Even more importantly, it would be very helpful if CONDUIT could provide the designer/supervisor with information that would help him understand the difficulty and find means to overcome it.

The work reported here was devoted to studying a proposed way for CONDUIT to communicate the structure, in the vicinity of the current values of the design parameters, of the problem it is trying to solve. In this case, by structure, we mean the shape of the function that CONDUIT is trying to minimize. For several reasons, which will be explained in the body of this report, this structural information is entirely based on numerically computed first derivatives of the specifications. An additional, and more important objective, was to develop ways for the designer to interpret this information and use it to alter the design problem to make it solvable. Note that it is not hard to create a design problem for which there is no solution. It is important for CONDUIT to indicate this to the designer/supervisor Lastly, it is important to know how sensitive a solution is to small variations in the parameters.

Background:

At its heart CONDUIT's design approach is to solve a sequence of nonlinear programming problems. More precisely, at any iteration it is trying to minimize a function $f(\underline{x})$ subject to a collection of constraints that can be represented as $\underline{x} \varepsilon \{\underline{x}:$

$g(\underline{x}) \leq \underline{1}\}$. Here $\underline{x}$ is an n-vector of design parameters; $g(\underline{x})$ is an m-vector of functions; $\underline{1}$ is the m-vector consisting of all ones; $f(\underline{x})$ is the maximum of the active specifications [1]. Although CONDUIT can evaluate the functions $f(\underline{x})$ and $g(\underline{x})$, their functional form is not available analytically. Computing $f(\underline{x})$ or $g(\underline{x})$ involves first running the Simulink or Matrix-X model of the aircraft and then using the results of the simulation to compute the values of the specifications. These specifications then give the values of $f(\underline{x})$ and $g(\underline{x})$. This means that the user of CONDUIT knows very little about the form of the functions. For example, he or she does not know if they are convex or concave, nor how smooth they are. The relative size of the functions is also unknown. All of these possible properties are important determinants of the difficulty of solving the nonlinear programming problem. For example, there is a collection of duality theorems in nonlinear programming that upper bound the optimal value and guarantee existence of an optimal solution [2]. These results require smoothness (i.e. the existence of at least one continuous derivative) at least and, for the strongest results, convexity of $f(\underline{x})$ and concavity of $g(\underline{x})$.

In fact, most of the theory of nonlinear programming assumes the functions involved, $f(\underline{x})$ and $g(\underline{x})$ in our notation, have at least one continuous derivative. We will demonstrate in the next section that the nonlinear programming problems that CONDUIT is trying to solve are often less smooth than this. That is, the effective $f(\underline{x})$ in many CONDUIT problems is only continuous. Its first derivative is often discontinuous at the value of $\underline{x}$ that is returned at the end of an iteration. Nonetheless it is useful to review the standard results on the sensitivity of nonlinear programming problems.

First, it is instructive to understand the limitations of the theory. It is essentially impossible to prove that there is a solution to a nonlinear programming problem if one does not have any information about the functions $f(\underline{x})$ and $g(\underline{x})$. This reflects the mathematical reality. For example, a golf course has at least 18 local minima. Think of the surface of the course as the function $f(\underline{x})$ with $\underline{x}$ two-dimensional. The function is typically smooth except at the edges of the holes. Imagine that the holes can be as deep as you like. If you did not know it was a golf course and all you could do was determine the value of $f(\underline{x})$ at individual values of $\underline{x}$, then the probability of finding even one of the holes would be negligibly small. Knowing it was a golf course would not help you find the local minima.

Fortunately, CONDUIT does not even need to find a local minimum. All that is needed is a solution that satisfies all of the specifications and some indication that this solution is insensitive to small changes in the design parameters and specifications. The classic control theoretic measure of sensitivity is due to Bode [3]. He defined the sensitivity of performance of a control system to the scalar parameter x at the point x_0 to be

$$S = (\partial f(x_0)/\partial x)/(f(x_0)/x_0) \qquad (1)$$

where f(x) is some measure of performance. Note that S is a dimensionless quantity. The natural definition for insensitivity is then just 1/S.

It is instructive to apply this definition the simplest situation that might appear in CONDUIT. Specifically, suppose that CONDUIT has found an unconstrained local minimum. This is one of the cases where CONDUIT indicates that it has solved the design problem. Suppose also that a single objective is dominant at the optimal $\underline{x}_0$ and that objective is once differentiable. Note that $\underline{x}_0$ is a vector. It is well known that

$$\partial f(\underline{x}_0)/\partial \underline{x} = 0 \tag{2}$$

at such an $\underline{x}_0$. Thus, the sensitivity, S, with respect to any of the components of the vector $\underline{x}_0$ must be zero. The insensitivity must be infinite. This is completely satisfactory.

Unfortunately, the next simplest situation that occurs in CONDUIT is not so neat. Because CONDUIT is generally solving a multi-objective problem, it often happens that a local minimum occurs at a point where two objectives compete. Again, we assume the minimum is unconstrained. Now, the conditions that must be satisfied are

$$\begin{aligned} \partial f_i(x)/\partial x < 0 \qquad & x > x_0 \\ \partial f_j(x)/\partial x > 0 \qquad & x < x_0 \end{aligned} \tag{3}$$

Note that the partial derivative does not exist at x_0. The scalar case is shown. The vector case is more complicated. The conditions (3) must hold in every direction. Clearly, conditions (3) do not depend on the size of the partial derivatives; they depend only on the sign. These facts have two implications for the insensitivity. First, Bode's definition does not apply at $\underline{x}_0$ because the required derivative does not exist. Second, the magnitude of a satisfactory insensitivity need not be infinite.

What happens when CONDUIT finds a solution to a design problem but the solution is on the boundary of one of the constraints? This is fundamentally the same as the previous case. Conditions (3) apply. The only difference is that one or more of the functions is a constraint $g_i(\underline{x})$ rather than an objective.

To summarize, in most cases where CONDUIT finds an optimal solution the local behavior of $f(\underline{x})$ near the solution satisfies (3). Thus, the first derivatives of the active specifications are available but there is a discontinuity in the actual performance metric. As a result, there is no clear and precise scaling rule. Instead, the proper scaling is, like the proper specifications, going to be the result of experience with a variety of problems.

The sensitivity question does not only arise at a solution. In fact, in most cases CONDUIT does not actually compute an optimal value for the design parameters. The designer most often stops CONDUIT at a satisfactory design after CONDUIT has performed several iterations without significant improvement in the overall performance. Clearly, a designer who stops CONDUIT needs to know if his or her design is insensitive. If it is not, it is probably worthwhile to allow CONDUIT to continue, even though the rate of improvement is small. Furthermore, it is often possible to use the insensitivity data as the basis for a modification of the problem that improves the rate of

improvement. Lastly, CONDUIT sometimes stops making significant progress towards a solution even though it is still in phase 1 or 2. In essence, it bogs down before it achieves a usable design. Again, it is possible to use the insensitivity information to modify the problem, thereby enabling CONUIT to progress at a reasonable rate towards an acceptable design.

Theoretical insight into all of these situations can be obtained from one of the core results in the theory of nonlinear programming. The result of interest is used in the proof of the Kuhn-Tucker theorem. The theorem is of interest in its own right. The context is as follows:

Given the problem, maximize $f(\underline{x})$=minimize $-f(\underline{x})$
Subject to $g(x)\leq\underline{0}$

Where $f(\underline{x})$ and $g(\underline{x})$ are as before and assumed to be differentiable. The only substantive change is the maximization instead of minimization. As shown, this amounts to an added minus sign. The statement of the results is nicer for maximization problems.

Kuhn-Tucker Theorem [4]: Let x* be an optimal solution to the problem stated above. Suppose the Constraint Qualification (CQ) is satisfied at x*. Then there exists $\lambda_i^*\geq 0$, for all i such that $g_i(\underline{x}^*)=0$, such that

$$\partial f(\underline{x}^*)/\partial \underline{x} = \sum_i \lambda_i^* \partial g_i(\underline{x}^*)/\partial \underline{x} \qquad (4)$$

Note that the summation is over only those i for which $g_i(\underline{x}^*)=0$. Those i for $g_i(\underline{x}^*)<0$ play no role. The condition that CQ holds is interesting and important for CONDUIT. Roughly, CQ means that the admissible directions—that is, the directions in which $\underline{x}^*$ could be perturbed and the new x would still satisfy the constraints—are given by the gradients of the $g_i(\underline{x})$. There are many versions of CQ [2,4] but it is usually impractical to verify them.

To understand the Kuhn-Tucker Theorem it is helpful to know Farkas' Lemma [see 2 for proof].

Farkas' Lemma: Let A_i, $1\leq i\leq k$, be n-dimensional row vectors. Let c be an n-dimensional row vector. The following statements are equivalent:

(i) For all $\underline{x}\epsilon R^n$, $A_i\underline{x}\leq 0$ for $1\leq i\leq k$ implies $c\underline{x}\leq 0$.

(ii) There exist $\lambda_i\geq 0$, $1\leq i\leq k$, such that

$$c = \sum_{i=1}^{k} \lambda_i A_i$$

Clearly, the Kuhn-Tucker Theorem uses (ii) in Farkas' lemma to express (i). Condition (i) as used in (4) says that the objective, $f(\underline{x})$, is worse for small changes in any admissible direction. CQ implies that the gradients correctly give the admissible directions.

The important point for CONDUIT users is actually that (i) holds rather than that (4) is true. To understand this, remember first that for any function, say $h(\underline{x})$, the gradient $\partial h(\underline{x})/\partial \underline{x}$ gives the direction of steepest increase. Thus, (4) says that the direction of steepest increase in $f(\underline{x})$, as one moves a short distance from $\underline{x}^*$, is in the cone formed by taking positive linear combinations of the directions of steepest ascent of all the active constraints. But, (i) says—once we replace the A_i by $\partial g_i(\underline{x})/\partial \underline{x}$ and c by $\partial f(\underline{x})/\partial \underline{x}$—that perturbing $\underline{x}^*$ in any direction that is admissible when the constraint boundaries are replaced by their tangents results in an increase in $f(\underline{x})$.

The logic of the Kuhn-Tucker Theorem implies that the opposite of (i) holds when constraints are active and the current value of $\underline{x}$ is not optimal. Notice that the only information used is either the functions or their gradients.

Lastly, the theory of nonlinear programming includes a collection of sensitivity results. Sensitivity questions are the main issues addressed in the duality theory. However, most of the results require strong assumptions that are not satisfied in the typical CONDUIT problem. Specifically, the true sensitivity results require that $f(\underline{x})$ be concave (for a problem in which $f(\underline{x})$ is to be maximized) and the $g_i(\underline{x})$ must be convex. Thus, we have decided not to include a discussion of these results here. They can be found in [2,4].

Results:

The brief review of the theory of nonlinear programming shows that the decision to make the CONDUIT sensitivity measures depend only on gradients was exactly correct. Although the original rationale was primarily that second derivatives would be too inaccurate to be useful, the theory shows that the second derivatives often do not exist. More importantly, it is the gradients that really matter.

The question of how to scale the sensitivity measures in CONDUIT is much trickier. Basically, although there is a large literature on scaling (see, e.g. [5]) there do not seem to be any general methods. Mathematically, the question appears to be too imprecise. In fact, in CONSOLE-OPTCAD, the predecessor to CONDUIT as well as the optimization engine inside CONDUIT, the developers assumed the user/designer would routinely rescale the specifications on order to help the computer solve any given design problem. Their argument was that it is impossible to know in advance what the scaling of any given specification might be. Hence, it would be necessary to modify the specs—rescale—in the course of solving a problem.

However, in a limited class of problems, such as the design of stability and control augmentation systems (SCAS) for aircraft, the range of parameter values and of specifications will certainly be narrower than mathematicians would expect. In part this is because designers will normally do some scaling, often unconsciously. After all, both designers and specification writers expect the numbers they get to be reasonable. Furthermore, the choice of units is a form of scaling. The dimensions of the wing of a large aircraft are not given in millimeters because the resulting number would be too

large. Thus, we support the decision not to allow users to rescale the specs routinely even though this is a feature of CONSOLE-OPTCAD.

By the reasoning given above it is clear that the scaling question needs to be addressed by examining the scaling issue in a large collection of typical aircraft problems. Academic examples will simply not work; they are not representative of the numerical values found in real applications. Thus, we could not improve on the scaling proposed by Chad Frost [6, pp. 28-33].

We have created and solved many examples in CONDUIT to study and illustrate the sensitivity question. Many of these were done using earlier versions of CONDUIT and have thus been superseded. We have included a small but representative sample, redone using CONDUIT 3.0 below. These are all versions of the same simplified XV-15 problem in XVLab3 in the CONDUIT course notes [7]. The block diagram for this problem, modified to include a redundant design parameter (dpp_kphired), is shown below in Fig. 1. This is followed by Fig. 2 in which the corresponding specifications are displayed.

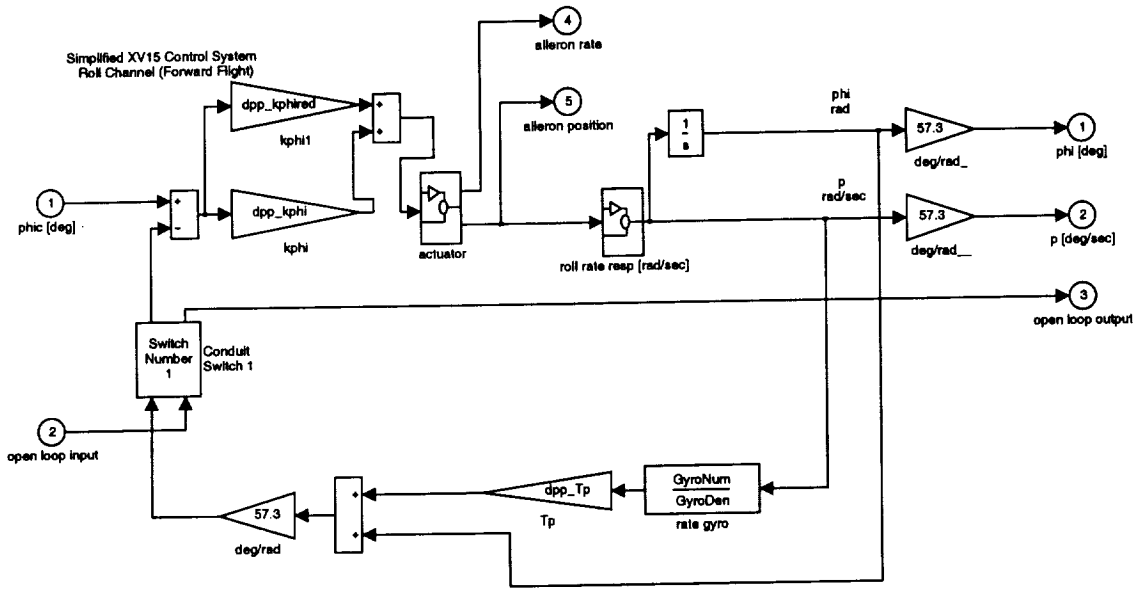

Figure 1 Block diagram of XV-15 with redundant design parameter

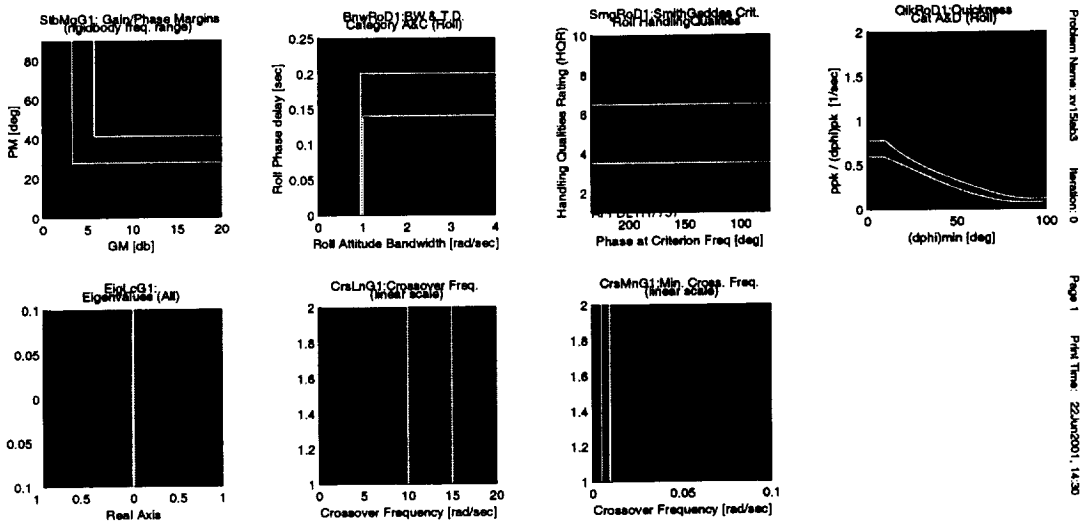

Figure 2 Handling qualities window for XV-15

This problem has been deliberately designed to have a redundant design parameter. Starting with the initial values dpp_kphi=1/2, dpp_kphired=1/2, and dpp_Tp=1 we obtained the sensitivity display shown in Fig. 3. Note that the sensitivities of the duplicate design parameters are identical, as expected. The conditional correlations (not shown) show that the two redundant design parameters (DPs) have conditional correlation equal to one, as they should. For reference purposes, the conditional correlation between dpp_Tp and dpp_kphi is -.8875. Of course, the pseudo Hessian matrix is not invertible so no Cramer-Rao bounds are computed. Surprisingly, CONDUIT has no real difficulty in solving this problem. The redundant design parameter has no noticeable affect. Of course, it does slow down the computations but in an essentially academic problem this is buried in networking and other computer overhead.

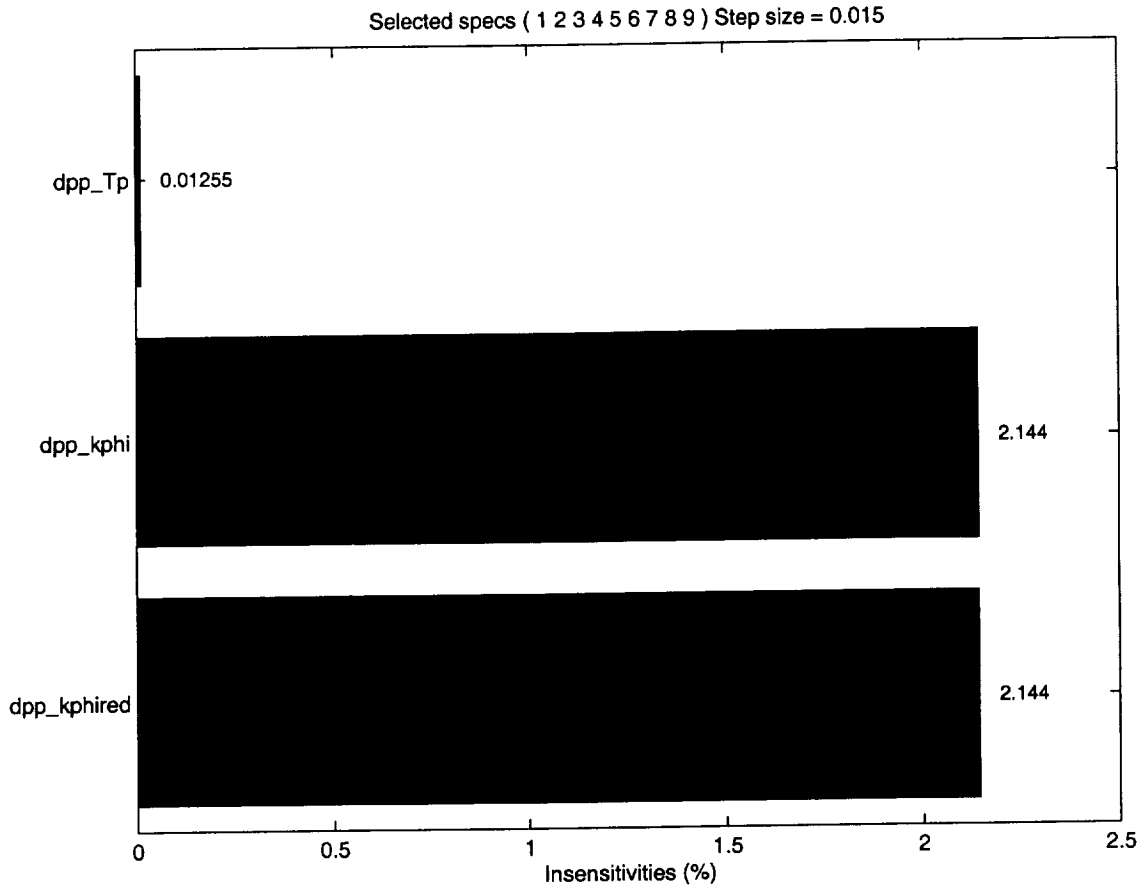

Figure 3 Sensitivity for XV-15 with redundant design parameter

Stopping after several iterations results in a reasonable solution to the design problem. All the specifications are satisfied at level 1. The DPs have values dpp_kphi=.5352, dpp_kphired=.5352, and dpp_Tp=1.05. The corresponding sensitivities are shown in Fig. 4. Of course, the conditional correlation between the redundant parameters is still one. The conditional correlation between dpp_Tp and dpp_kphi has increased in magnitude to -.9493.

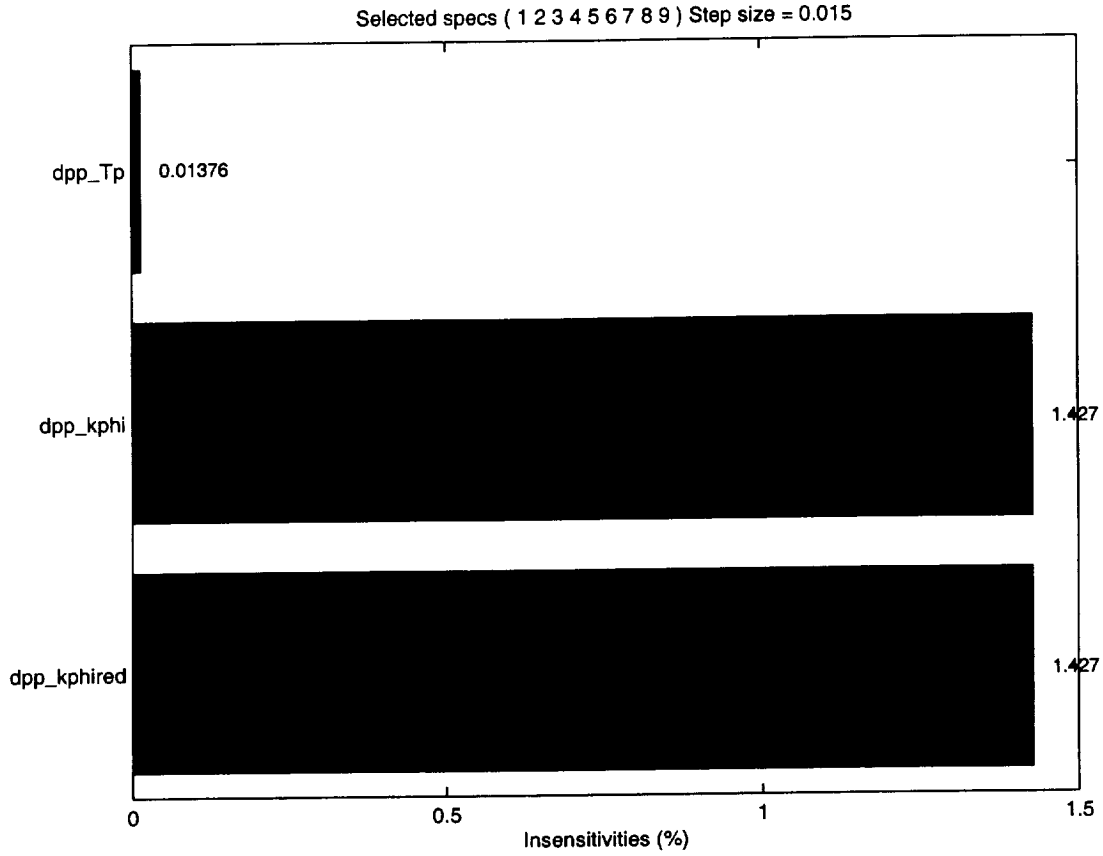

Figure 4 Sensitivity at reasonable solution

The next pair of examples was developed to study scaling directly. The redundant DP was eliminated. Instead, a gain was placed in series with dpp_kphi as shown in Fig. 5.

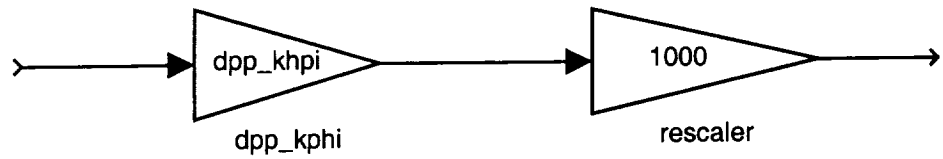

Figure 5 The new structure of dpp_kphi

The resulting insensitivities when the DPs have values dpp_kphi=.0001 and dpp_Tp=1 are shown in Fig. 6. Note that the value for dpp_kphi was chosen to produce exactly the effective value one, identical to the original problem without the rescaler.

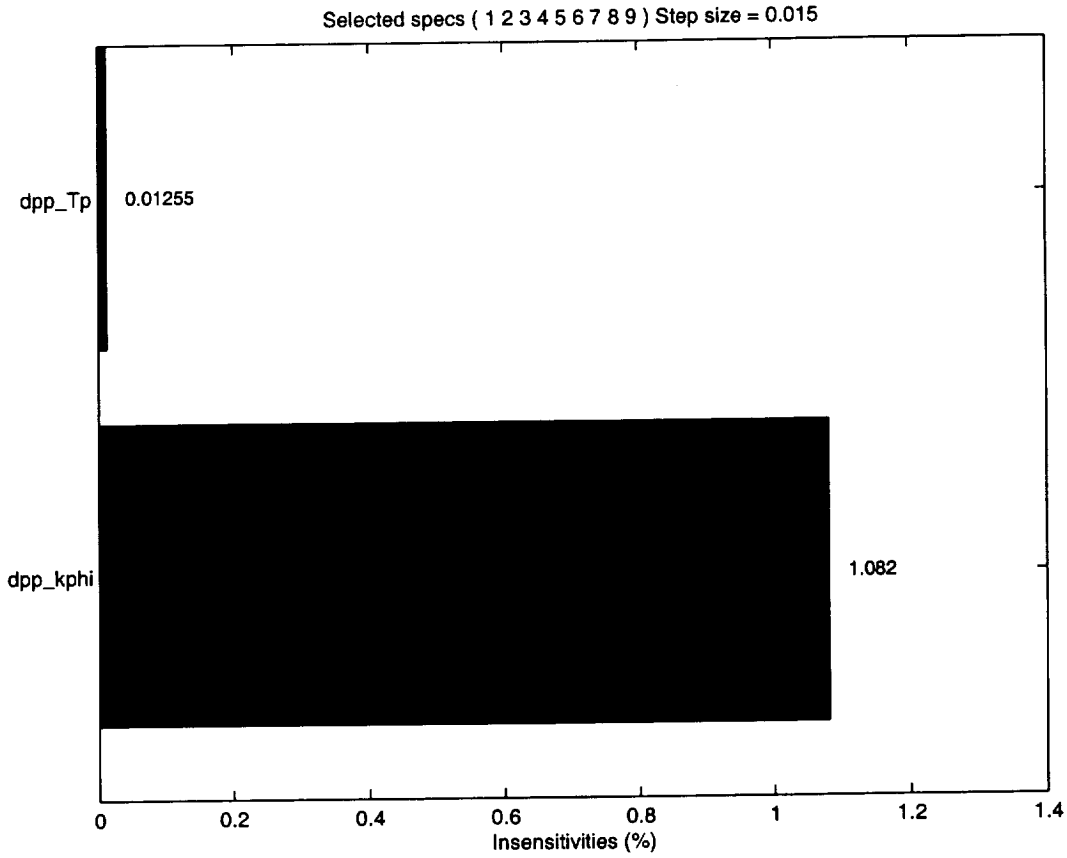

Figure 6 The insensitivities rescaled by 1000

Note that the insensitivity of dpp_kphi is different from its value in Fig. 3. This is surprising because the effective value of the design parameter is unchanged from the first example. Some aspect of the scaling scheme in CONDUIT has presumably changed the computed insensitivity. In any case, it is the rough magnitude that matters and that is unchanged. In this case, the pseudo Hessian is invertible so Cramer-Rao bounds are computed. They are shown in Fig. 7.

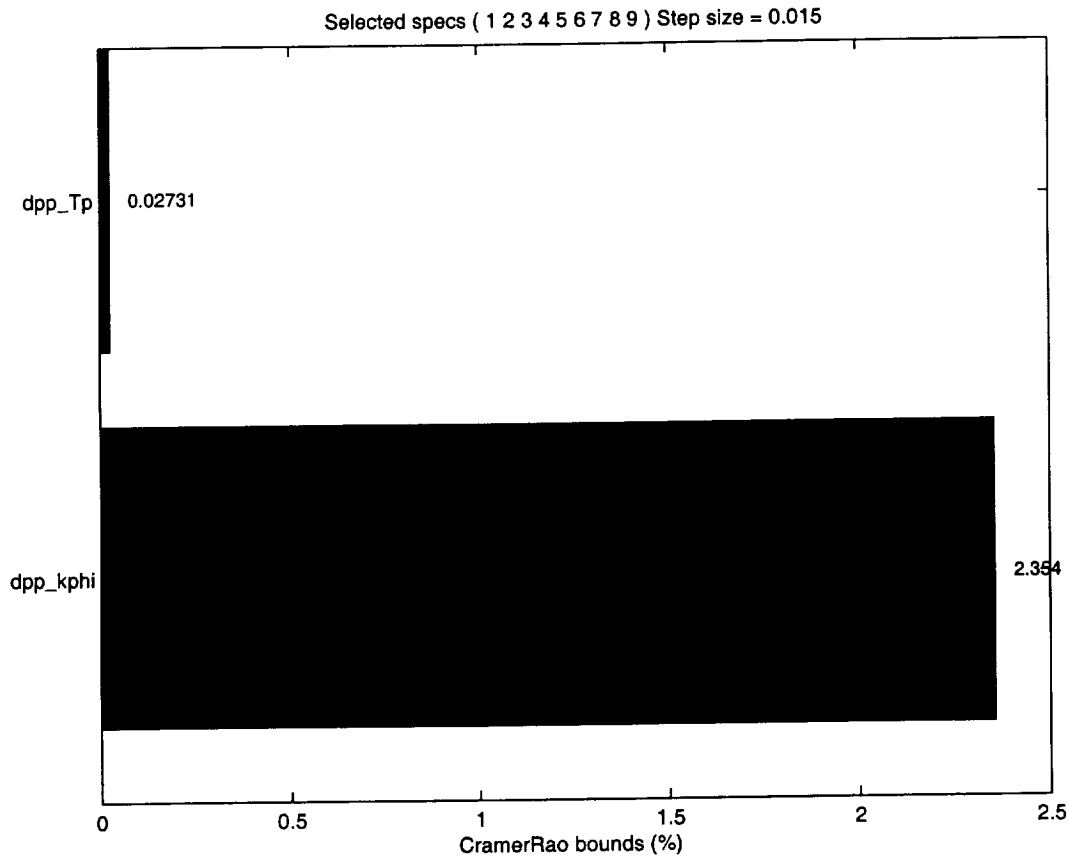

Figure 7 Cramer-Rao bounds for the rescaled by 1000 problem

Next, the gain in the rescaler, shown in Fig.5, was changed from 1000 to .001. The initial value of dpp_kphi was set to 1000. Again, the effective value of the DP is unchanged from the previous examples. The resulting insensitivities are shown in Fig. 8. However, the insensitivity computed for dpp_kphi is identical to the one computed when the rescaler is set to 1000.

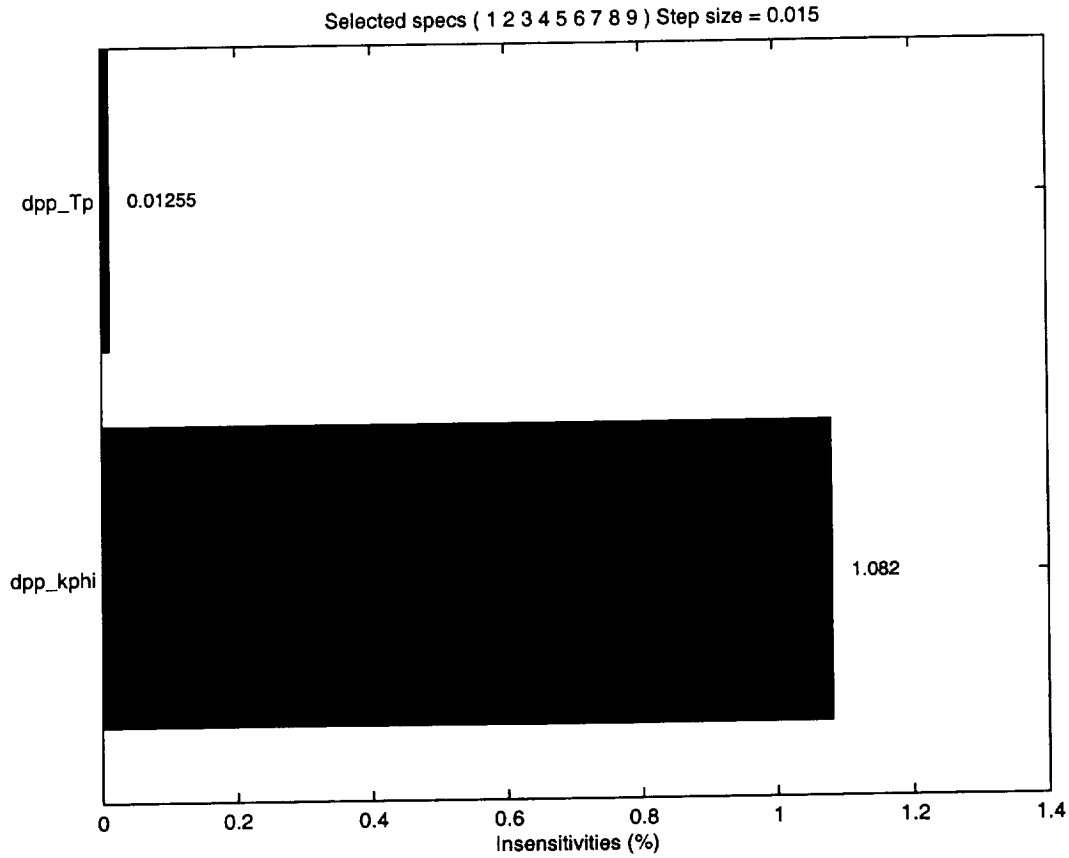

Figure 8 The insensitivities when the rescaler is .001

In this case the pseudo Hessian takes the value $\begin{bmatrix} 6.35e+07 & -654 \\ -654 & 0.00854 \end{bmatrix}$. This is too ill conditioned to be invertible. Thus, no Cramer-Rao bounds are computed.

The next example was obtained by replacing the rescaler by a DP as shown in Fig. 9. The resulting correlation between these two redundant DPs is geometric rather than linear. Of course, the calculation of the conditional correlation is based on the assumption that the DPs are linearly correlated. The resulting insensitivity array is shown in Fig. 10.

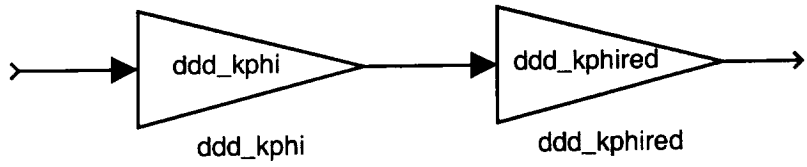

Figure 9 The rescaler has been replaced by a DP in series

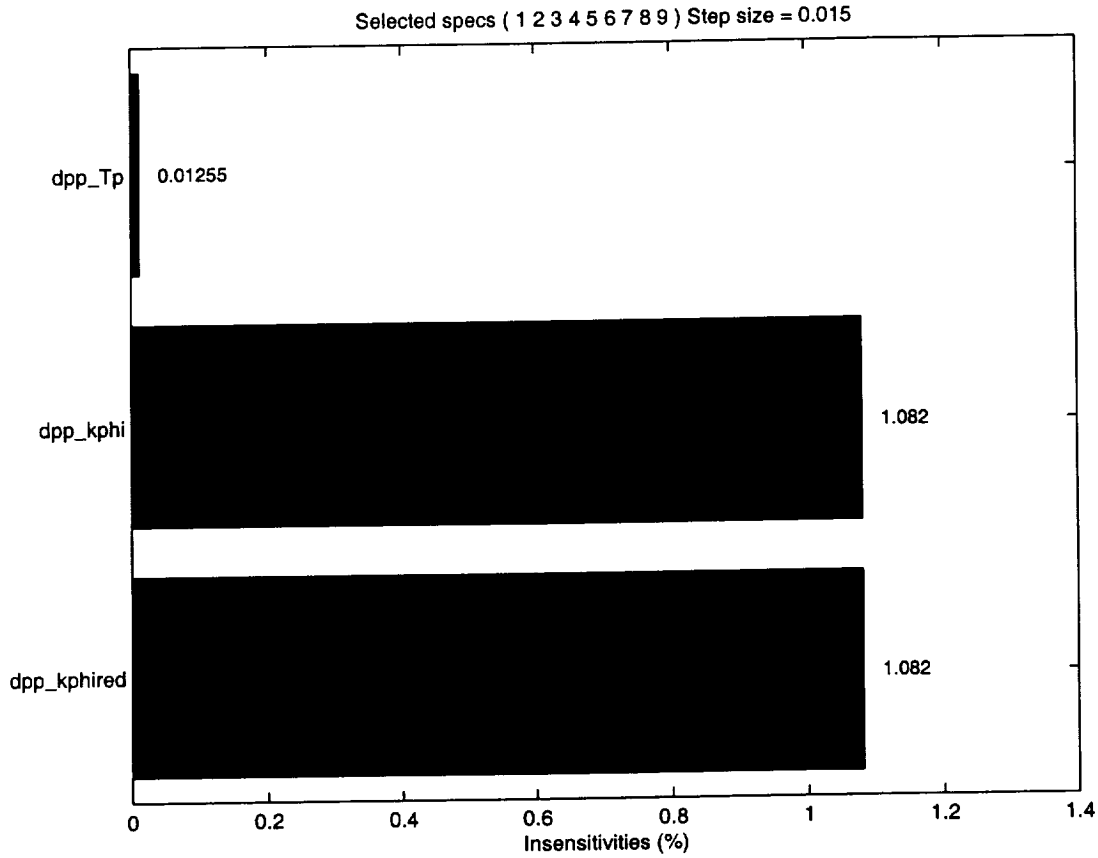

Figure 10 The insensitivity array when the redundant DPs are in series

Notice that the insensitivity of dpp_kphi is identical to that of dpp_kphired, as expected. The pseudo Hessian is not invertible so we do not get a conditional correlation. Curiously enough, CONDUIT also solved this problem without any noticeable difficulty.

We also tried to start this problem from different initial values of the DPs. Specifically, with dpp_kphi=100, dpp_kphired=.01 and dpp_Tp=1 we obtained the sensitivity array shown in Fig. 11.

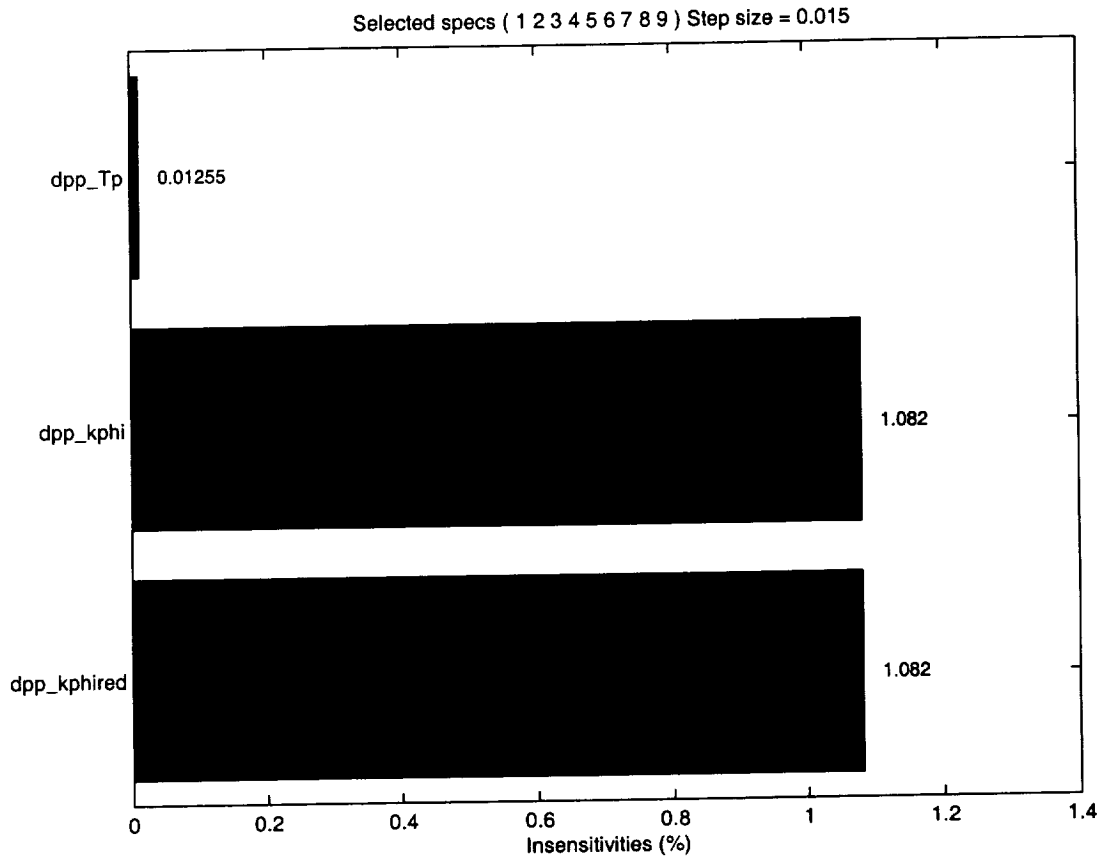

Figure 11 Insensitivities when the redundant DPs have different values

This is again somewhat of a surprise. The pseudo Hessians corresponding to figures 10 and 11 are quite different, as expected. The scaling algorithm used in CONDUIT is probably the cause of the unchanging insensitivities in the face of a changing Hessian. However, this scaling algorithm is not described in any of the publications we have and is unknown to us.

As was explained earlier, many other examples have been run although almost all of these use an earlier version of CONDUIT. Some of these are available in [8]. Perhaps the most significant aspect of these examples is that very few of them created significant difficulties for the optimization routines in CONDUIT. This is in striking contrast to what happens with real design problems, many of which have required the use of the sensitivity tools to overcome poor scaling, redundant DPs, and specs.

Conclusions:

Because the gradients of the specifications with respect to the DPs are the most important elements in solving CONDUIT problems, it is clearly true that the gradients should be the basis of the sensitivity metrics. Computing the pseudo Hessian matrix and using it to infer conditional correlations is harder to justify based on theoretical issues alone. One could argue that greater emphasis should be placed on the gradients themselves, rather than on what is effectively their squares. However, the success of the

CONDUIT sensitivity metrics in several real design problems combined with their similarity to the CIFER metrics provide compelling arguments that they are the best choice.

The scaling question cannot be answered theoretically. The theoretical range of possible scalings is much too large. It is much larger than the real range. Thus, it is argued that the proper scaling should be determined experimentally. However, the experiments must be real ones. Academic and almost academic examples simply do not have either the parameter range or the size to give a reasonable picture of the scaling problem.

Finally, we were unable to generate a significant group of simple examples that really needed the sensitivity tools. Poor scaling and redundant parameters usually did not cause CONDUIT any serious difficulties. This is in strong contrast to real applications. This is yet another argument for determining scaling rules from real problems. However, the sensitivity tools worked very well, even though they were not needed. In every case where DPs were redundant, the CONDUIT sensitivity tools indicated this was the case. Even though in some cases the numerical values were unexpected, the order of magnitude was always correct.

One question that might merit further research is why CONDUIT had so little trouble with redundant DPs and poorly scaled specs. It is conjectured that the two issues that really matter are:
(a) the size of the problem—i.e. the number of DPs, and
(b) the conditional correlations between DPs when they are close to one even though the DPs are quite different.
In our examples, the highly correlated DPs were essentially the same.

References:

[1] Moldoveanu, V. & W. S. Levine (1999). Final Report—further development, support, and enhancement of CONDUIT, Final Report, NASA Ames Project # NAG 21122.
[2] Mangasarian, O. L. (1969), *Nonlinear Programming*, McGraw-Hill
[3] Bode, H. W. (1945). *Network Analysis and Feedback Amplifier Design*, Van Nostrand, New York.
[4] Varaiya, P. P. (1972), *Notes on Optimization*, Van Nostrand Reinhold Co.
[5] Gill, P. E., Murray, W. & M. H. Wright (1981). *Practical Optimization*. Academic Press.
[6] Frost, C. R. (1999). Design and Optimization of a Rotorcraft Flight Control System Using the Control Designer's Unified Interface (CONDUIT), M. S. Thesis, California Polytechnic State University.
[7] Tischler, M. B., Colbourne, J. D., Cheung, K. K., Frost, C. R., Levine, W. S. & V. Moldoveanu (1998). CONDUIT "Control Designer's Unified Interface" Course Notes, Aeroflightdynamics Directorate, U. S. Army ATCOM, Ames Research Center, Moffett Field, CA.

[8] Chiang, S.-C. (1999). Sensitivity tools in CONDUIT (Control Designer's Unified Interface), MS Report, ISR, University of Maryland, College Park.