JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 16, No. 2, March-April 1993

Missile Autopilot Robustness Using the Real Multiloop Stability Margin

Kevin A. Wise*
McDonnell Douglas Aerospace, St. Louis, Missouri 63166

The robustness of a longitudinal missile autopilot to uncertain aerodynamics is determined using the real multiloop stability margin. This method determines control system stability robustness to simultaneous real-parameter variations without conservatism. This paper presents an overview of the theory and a computer program developed to evaluate control system robustness to real-parameter uncertainties. The program implements a signal flow graph decomposition method to compute the robustness analysis model. A polynomial-time convex-hull algorithm is presented.

Introduction

A MISSILE flight control system must guarantee stability and performance in the face of large aerodynamic uncertainties and disturbances. This requires the feedback controller to maintain system stability and loop performance for all possible variations in the plant behavior. Determination of the degree of robustness to these aerodynamic uncertainties is the focus of this paper. A recent publication by deGaston and Safonov[1] outlines an algorithm that will exactly compute the stability margin k_m of diagonally perturbed multivariable systems without conservatism. This paper presents an overview of that theory and describes software used to analyze a bank-to-turn missile autopilot.

In the early 1980s, several papers described how multivariable feedback systems can be analyzed in the frequency domain.[2,3] Doyle[4] developed several robustness theorems that were fundamental in developing the analysis techniques used to analyze model uncertainties. These methods utilize singular-value theory as a means of measuring the size of multi-input multi-output frequency-dependent matrices. These tests modeled control system uncertainties using a full single-block matrix structure. If the uncertainties were truly structured in this form, then these tests are not conservative. If the uncertainties did enter the system in a structured way, these tests produced conservative estimates of stability robustness.

Doyle[5,6] later developed a method of incorporating the structure of the uncertainty into the uncertainty analysis. This capability reduced the conservatism of the previous singular-value techniques by utilizing a multiple-block diagonal structural model of the uncertainties. This test is called the structured singular-value (SSV) μ test.

Morton[7,8] applied the μ test to the analysis of real-parameter variations. The SSV μ software generally models the real-parameter variation as a complex variation and produces a slightly conservative bound. Jones[9] and Fan and Tits[10] have worked on reducing this conservatism, but reliable software implementing the real SSV is not readily available.

In contrast to the foregoing SSV-based tests, robustness to uncertainties is also being analyzed using polynomial methods. A myriad of papers have recently been published that utilize Kharitonov's theorem[11] and variants of it. Kharitonov's theorem analyzes the robustness question by examining the Hurwitz (stability) properties of a family of polynomials whose coefficients are based on the system's characteristic equation and uncertainties. Barmish and DeMarco[12] present an excellent review of the literature on these polynomial methods.

In Sideris[13] a polynomial-time robustness algorithm is presented that assumes the closed-loop polynomial coefficients are affine in the uncertain parameters. Polynomial time refers to the time required by an algorithm to solve a problem. For some input of size m, the time required by a polynomial-time algorithm is proportional to a polynomial in m. For exponential-time algorithms, the time would be proportional to α^m (at least), where α is any number >1.

Assume that the parameter space of the uncertain parameters is an m-dimensional hypercube. Let $\Omega(j\omega)$ denote the image of the hypercube in the complex plane, mapped through the closed-loop characteristic polynomial. The image $\Omega(j\omega)$ at a fixed frequency is a parapolygon of $2m$ sides. The Sideris algorithm is a polynomial-time algorithm because it works with the edges of $\Omega(j\omega)$, of which there are $2m$. The deGaston-Safonov algorithm uses the 2^m vertices of the hypercube. As m increases, there is an exponential explosion in the number of vertex points that have to be analyzed. This problem is addressed later in this paper.

The software presented here computes a lower bound on the real margin using an analysis model derived from a signal flow graph. Previous applications[14] used an analysis model derived with a technique from Morton.[7,8] Figure 1 shows this model in which the real-parameter variations are isolated into a diagonal Δ matrix. With no parameter variations ($\Delta=0$), the system is stable [$M(s)$ is nominally stable]. The approach developed by Morton creates a state-space triple (A_m, B_m, C_m) for $M(s)$ by isolating the n real-parameter variations into $\Delta=\text{diag}[\delta_i]$ by factoring out the variations in the closed-loop system matrix, modeled as

$$A_{\text{cl}} = A_0 + \sum_{i=1}^{n} E_i \delta_i$$

If the state-space description of the control system is linear in the uncertain parameters, then the Morton method is applicable. If the parameters multiply each other, then the method is no longer applicable. A technique is presented here that does not suffer from this restriction. The only requirement is that the control system be linear and that a signal flow graph (or block diagram) can be constructed for the system. Also, the parameter uncertainties must be independent.

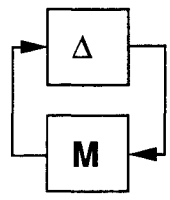

Fig. 1 Block diagonal analysis structure.

Received Oct. 1, 1990; revision received Jan. 30, 1992; accepted for publication Feb. 5, 1992.

*Staff Specialist, Advanced Guidance, Navigation, and Control, Mail Code 3064025, P.O. Box 516. Senior Member AIAA.

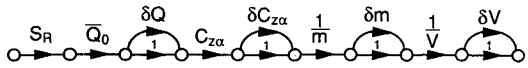

Fig. 2 Expanded stability derivative signal flow graph branch model.

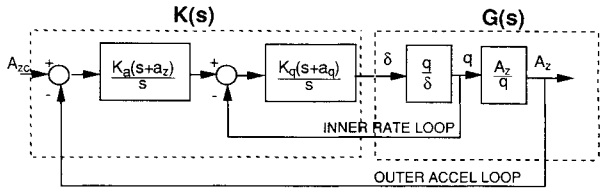

Fig. 3 Missile autopilot and dynamics.

In many problems, the coefficients of the differential equations may be complicated functions of many parameters. For example, a signal flow graph branch gain modeling the aerodynamic stability derivative Z_α (body lift due to angle of attack α) for a missile is defined as

$$Z_\alpha = \frac{\bar{Q} S_R C_{z\alpha}}{mV}$$

(Other missile aerodynamic stability derivatives modeling pitching moments and fin forces are similar functions of dynamic pressure $\bar{Q}$, mass m, velocity V, etc.) If both $\bar{Q}$ and $C_{z\alpha}$ are to be varied in an analysis, the Morton method cannot be applied. By modeling the control system in a signal flow graph, a multiparameter coefficient can be modeled as a series of branch gains with individual parameters on single branches. For example, Z_α would be represented with the branch gains shown in Fig. 2, where δQ, $\delta C_{z\alpha}$, δ_m, and δV are real-parameter variations in each of the parameters. This method decomposes each signal flow graph branch gain modeling a real parameter and isolates the variation creating a diagonal Δ. In addition to forming analysis models, this signal flow graph modeling technique is ideally suited for forming the general $\Delta - \boldsymbol{P} - \boldsymbol{K}$ H^∞ controller synthesis models (Δ uncertainties, $\boldsymbol{P}$ plant, and $\boldsymbol{K}$ controller). This is demonstrated by Wise et al.[15] where the H^∞-μ synthesis design model is formed using this method.

Robustness Theory for Real-Parameter Uncertainty

Control system sensitivity to uncertainties in dynamics has been a major focus for many years. In the past, the most widely used measure of stability robustness has been single-loop gain and phase margins derived from Bode (Nyquist) type frequency-response calculations. A new measure called the real multiloop stability margin, as defined by deGaston and Safonov,[1] is a scalar quantity interpreted as a gain margin. Its calculation gives an exact measure of control system stability robustness to real-parameter variations. This section presents the theory used to compute the real margin and is taken from deGaston and Safonov.[1]

Our missile flight control system under uncertainty is shown in Fig. 3. The uncertainties may arise from real-parameter variations, neglected/mismodeled dynamics, or combinations of both. Only real-parameter variations are addressed in this work. For analysis, the control system shown in Fig. 3 is transformed into the block diagonal perturbation structure of Fig. 1. The uncertainties in the system are isolated and placed into a diagonal matrix Δ. The transfer matrix $\boldsymbol{M}$ describes nominal system characteristics that are stabilized by a compensator. Thus, for $\Delta = 0$, the system is stable. Let

$$\Delta = \text{diag}[\delta_1, \ldots, \delta_n], \qquad \boldsymbol{M}(s) \in C^{n \times n} \tag{1}$$

The stability of the system described by Fig. 3 is implied by $\det[\boldsymbol{I} + \Delta \boldsymbol{M}] \neq 0$ (the multivarible Nyquist theorem). The stability margin k_m is defined as

$$k_m = \min_{\Delta} \{k \in [0, \infty) \mid \det[\boldsymbol{I} + k\Delta \boldsymbol{M}] = 0\} \tag{2}$$

Without loss of generality, consider each δ_i to represent an uncertain real parameter in the system. Δ is contained in the domain D, with each $\delta_i \in D_i$, where D_i is the domain of the ith parameter. If

$$(1/k_m)\delta_i \in D_i \qquad \text{for all } i \tag{3}$$

then $\Delta \boldsymbol{M}(s)$ remains stable. This defines k_m as a multiloop stability margin.

The proposed algorithm for computing k_m converges by iterating lower and upper bounds on k_m, which are determined when either the convex hulls or interior points, respectively, of certain image sets first intercept the origin. This development is made possible by the use of a mapping theorem taken from Zadeh and Desoer.[16] The multiloop stability margin is computed by finding the smallest k for which there exists $\Delta = \text{diag}[\delta_1, \ldots, \delta_n] \in D$ such that $\det[\boldsymbol{I} + k\Delta \boldsymbol{M}] = 0$.

The approach is as follows: Fix k. Map all the parameter space D into the complex plane with $\det[\boldsymbol{I} + k\Delta \boldsymbol{M}]$. If this region so mapped does not include the origin, then k is a lower bound on the stability margin k_m. Increment k positively until the origin is just included in the map. This would yield k_m exactly. However, computing the image of D under the mapping $\det[I + kDM]$ is computationally prohibitive. To circumvent this problem, the convex hull of the image of D is used. Computing the convex hull of the image is practical using a mapping theorem from Zadeh and Desoer.[16]

We begin by defining the parameter space D as

$$D = D_1 \times D_2 \times \cdots \times D_n \tag{4}$$

This parameter space describes the uncertain real parameter modeled in Fig. 1. In general, the parameter space D will be an n-dimensional polytope having 2^n vertices. By scaling the parameter uncertainties and incorporating the scaling into $\boldsymbol{M}$, a hypercube describing the parameter space may be used rather than a polytope. Define V_i as a vertex of the hypercube D, where $i = 1, \ldots, 2^n$. The vertex V_i represents a corner of the hypercube. Let

$$V = \{V_1, V_2, \ldots, V_m\}, \qquad m = 2^n \tag{5}$$

denote the set of all hypercube vertices $V \subset D$. Let Δ_{V_i} be a matrix of parameter uncertainties made up of the vertex points $v_{i,j}$ as j is varied from 1 to n. This is described as

$$\Delta_{V_i} = \text{diag}[v_{i,j},\ j = 1, \ldots, n] \tag{6}$$

Define

$$\det[I + kDM] = \{z \in C \mid z = \det[\boldsymbol{I} + k\Delta \boldsymbol{M}]\ \forall \delta_i \in D_i,$$
$$i = 1, \ldots, n, \qquad \text{with } k \text{ and } \boldsymbol{M} \text{ fixed}\} \tag{7}$$

This set is a set of points that represents the hypercube solid being mapped into the complex plane through the determinant function. It describes the entire image of D (the image of the parameter uncertainties) in the Nyquist plane. Define

$$\det[I + kVM] = \{y_i \in C \mid y_i = \det[\boldsymbol{I} + k\Delta_{V_i} \boldsymbol{M}],\ i = 1, \ldots, n\} \tag{8}$$

Equation (8) describes the set of points mapped into the complex plane by the hypercube vertices. Let $F_i = \det[\boldsymbol{I} + k\Delta_{V_i} \boldsymbol{M}]$ be the mapping of the ith vertex. F_i represents a single point in the set $\det[I + kVM]$. With these definitions, we are now ready to state the following theorem.

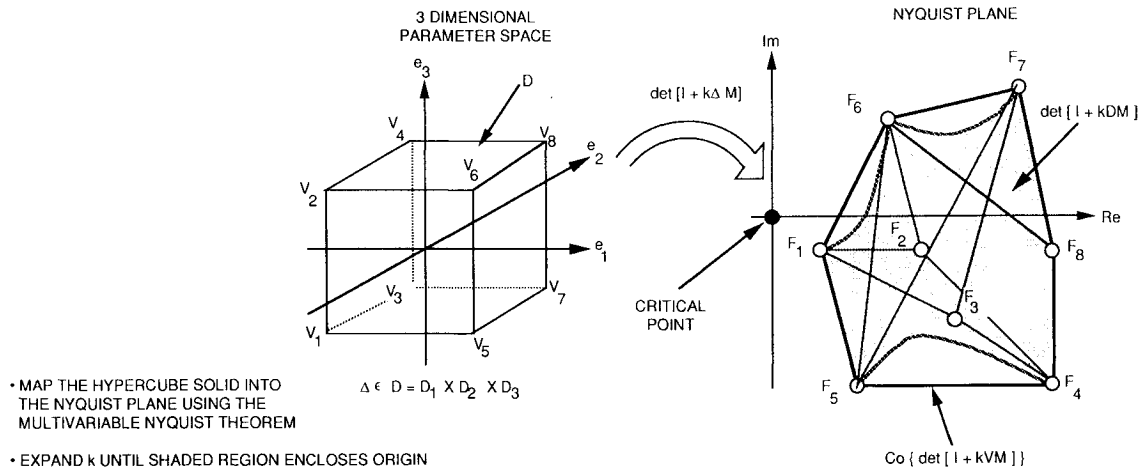

Fig. 4 Multiloop stability margin using deGaston-Safonov algorithm.

Theorem 1: Let k, M, D, $\det[I+kDM]$, $\det[I+kVM]$ be defined as previously given. Fix k. Then

$$\det[I+kDM] \subset \text{co}\{\det[I+kVM]\} \tag{9}$$

This theorem states that by mapping the 2^n vertices of D by $\det[I+kVM]$ to the complex plane and then constructing a convex hull about the 2^n points, a polygon is created that encompasses $\det[I+kDM]$. The utility of this theorem is best presented by an example.

Consider a three-dimensional uncertain parameter space. The hypercube is shown in Fig. 4 with the 2^3 vertices arbitrarily labeled. Thus, $\Delta \subset D = D_1 \times D_2 \times D_3$ with $D_i = [\delta_{i1}, \delta_{i2}]$. The parameters δ_{i1} and δ_{i2} describe the lower and upper bounds of the parameter δ_i.

Figure 4 shows the mapping of this parameter-space hypercube into the complex plane using the determinant mapping function. The shaded region depicts the true image of the cube solid mapped into the Nyquist plane. If the origin was contained in the shaded region, then the system would be unstable. Since the origin is not in the shaded region, the gain margin k used in $\det[I+kDM]$ is smaller than the true stability margin and should be increased in magnitude until the origin is included. The value of k such that the origin is just included in the shaded region is the exact multiloop stability margin k_m. In Fig. 4, the vertex points V_i are mapped into the F_i points. The convex hull containing the image of the hypercube is denoted as $\text{co}\{\det[I+kVM]\}$ and is shown as a heavy border around the $\det[I+kDM]$ image. We see from the figure that if $\text{co}\{\det[I+kVM]\}$ were used to determine k, conservatism would be present since the $\text{co}\{\det[I+kVM]\}$ contains more points than the true image of $\det[I+kdM]$. This fact is used to define a lower bound on k_m, resulting in the following lemma.

Lemma 1: Let M, D, and $\det[I+kDM]$ be defined as previously given. Then, for $k > k_0$,

$$\det[I+k_0DM] \subset \det[I+kDM] \tag{10}$$

$$\text{co}\{\det[I+k_0DM]\} \subset \text{co}\{\det[I+kDM]\} \tag{11}$$

This lemma states that the image of the hypercube solid under the determinant mapping function, for k_0, is a subset of the image mapped using a larger k. Thus, the convex hull containing $\det[I+k_0DM]$ is contained in the convex hull $\text{co}\{\det[I+kDM]\}$.

Lower Bound on the Stability Margin k_m

Application of the aforementioned lemma allows us to expand $\text{co}\{\det[I+kDM]\}$ until the origin is enclosed. We can show this graphically in Fig. 5. The solid lines represent $\text{co}\{\det[I+k_iVM]\}$ for k_1, k_2, and k_3. For all $k < k_3$, the origin is not enclosed by $\text{co}\{\det[I+kVM]\}$. Thus, $k_3 < k_m$ is a lower bound for the stability margin k_m. If k increases without $\text{co}\{\det[I+kVM]\}$ intercepting the origin, then $k_m = \infty$.

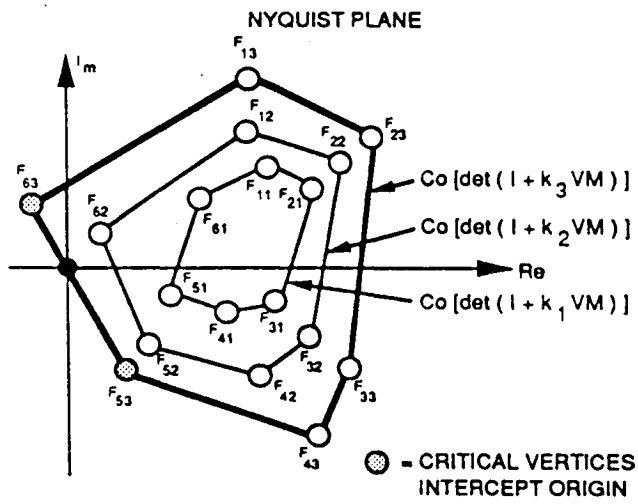

Fig. 5 Convex-hull expansion with $k_1 < k_2 < k_3$.

Upper Bound on the Stability Margin k_m

To compute the upper bound on k_m, the path between the vertices whose line segment intercepts the origin must be examined more closely. Define the following: Critical vertices: $F_i = \det[I+k\Delta_{V_i}M]$, $F_j = \det[I+k\Delta_{V_j}M]$, $i \neq j$ and $\beta \in [0,1]$ such that $(1-\beta)F_i + \beta F_j = 0$. Isolated critical vertex (ICV): F_i is isolated if $F_i \neq F_j$, $i \neq j$. Coincident critical vertex (CCV): F_i is coincident if $F_i = F_j$, $i \neq j$.

Critical vertices are defined as the two vertices whose line as segment intercepts the origin. These critical vertices are isolated if $F_i \neq F_j$. They are coincident if $F_i = F_j$. Let $m(i,j)$ be equal to the number of differing coordinates of the two vertices V_i and V_j that are mapped by $\det[I+k\Delta_{V_i}M]$ to F_i and F_j. In Fig. 5 the critical vertices are F_{53} and F_{63}. The $m(i,j)$ is the minimum number of edges on the hypercube D from vertex V_i to V_j. In Fig. 5, $m(i,j) = 1$. The following lemma will aid in the calculation of the upper bound on k_m.

Lemma 2: Any path along a single coordinate in D is mapped by $\det[I+kDM]$ to a straight line in the complex plane. For fixed M, the $\det[I+kDM]$ for $\Delta = \text{diag}[\delta_1,$

$\ldots,\delta_n] \in D$ is a polynomial in the variables δ_i and is affine with respect to each of the δ_i. This is true only for a diagonal Δ and is obtained by definition of the determinant. This affine relatoinship proves this lemma.

Lemma 2 guarantees that any point on the face of the hypercube D mapped into the Nyquist plane will be contained in the convex hull formed by the mapped vertices. This is true only for real parameters and allows us to determine stability robustness using parameter-space methods. If the parameters under variation were complex, any path along a single coordinate would trace an arc in the Nyquist plane. Thus, points contained on the face of a complex-parameter hypercube mapped into the Nyquist plane need not be contained in the convex hull formed by the hypercube vertices. This fact precludes the use of parameter-space methods in analyzing complex-parameter variations.

Define a vertex path as any path between critical vertices F_i and F_j, consisting of $m(i,j)$ straight-line segments, defined by $\det[I+k\Delta_x M]$ as x progresses from V_i to V_j along the edges of the hypercube D. The first such vertex path to touch the origin defines a point in $\det[I+kDM]$ and the associated k is an upper bound on the stabilty margin k_m. The vertex path will determine the region in the parameter space that causes instability.

Convergence to k_m

The actual stability margin is computed by an iterative algorithm. It begins by examining the vertex paths between critical vertices (ICVs or CCVs) that intercept the origin. This defines the edges of the hypercube closest to the origin. The domain D is then split along this vertex path, creating subdomains. Convex hulls around smaller and smaller subdomains are computed. As the subdomains become small, the union of all of the convex hulls for all of the subdomains gets close to the actual image of the domain D. The accuracy in the computation of k_m is then dependent on how small the subdomains are made. The following three lemmas are used to prove the convergence theorem that computes the exact multiloop stability margin.

Lemma 3: On a hypercube of dimension n with two vertices that differ by m coordinates, there are $m!$ paths between these two vertices along the edges of the hypercube. Each path between these two vertices will have $m+1$ vertices along the path (including the original vertices).

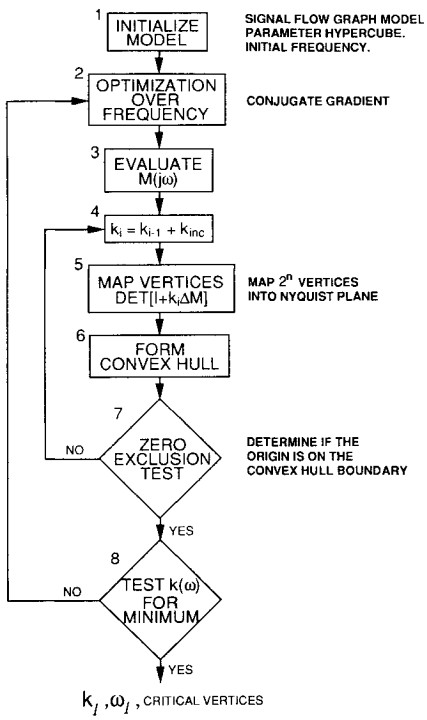

Fig. 6 ROBUSTR flowchart.

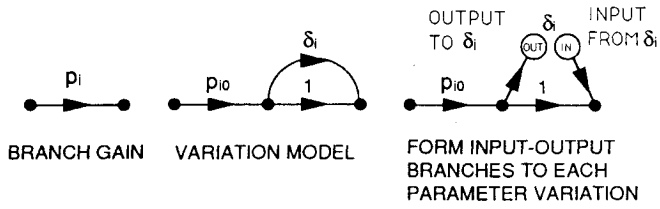

Fig. 7 Signal flow graph model for real-parameter variations.

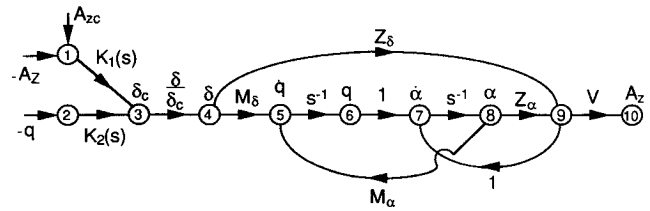

Fig. 8 Acceleration command autopilot signal flow graph.

Lemma 4: Let k, M, D, $\det[I+kDM]$, and $\det[I+kVM]$ be defined as previously given. Let F_i and F_j be isolated critical vertices with $m(i,j)\geq 2$ and F_k denote the first vertex along a vertex path emerging from F_i. Define a point along the line segment between F_i and F_k as F_x, exclusive of the end points, i.e., $F_x=(1-\beta)F_i+\beta F_k=0$, $\beta \in (0,1)$. Let V_x be the associated point on the hypercube edge defined between V_i and V_k. Cut the domain D at V_x orthogonally to this edge to create two subdomains D_1 and D_2, where $V_i \in D_1$ and $V_j \in D_2$. Then neither $\text{co}\{\det[I+kD_1M]\}$ nor $\text{co}\{\det[I+kD_2M]\}$ includes the origin.

Lemma 5: Let k, M, D, $\det[I+kDM]$, and $\det[I+kVM]$ be defined as previously given. Then there is at least one Δ_{V_i} associated with the stability margin k_m that assumes an extremal value.

Lemma 3 is used to determine the number of vertex paths between critical vertices. These vertex paths define the coordinate direction in which the parameter space domain D is split into subdomains.

Lemma 4 is the heart of the convergence theorem used to obtain k_m. It is employed when $m(i,j)\geq 2$. The utility of this lemma is best explained by an example. In Fig. 4, let vertex images F_1 and F_6 be isolated critical vertices, with k_l determined such that $\text{co}\{\det[I+k_lVM]\}$ intercepts the origin. For this case, $m(i,j)=m(1,6)=2$. The convex hull surrounding the $\det[I+kDM]$ image has a larger area than the true image of the hypercube solid (shaded area). The area contained in $\text{co}\{\det[I+k_lVM]\}$ that is not contained in $\det[I+kDM]$ makes k_l a conservative estimate, i.e., $k_l<k_m$. Lemma 4 says that if we split the parameter space into two subdomains along one of the two vertex paths $(V_1-V_2-V_6)$ or $(V_1-V_5-V_6)$, and compute convex hulls about each of the images of the two subdomains, then the origin will not be contained in either convex hull. This guarantees that we can converge to the true stability margin k_m by splitting the parameter space into subdomains. As the subdomains become smaller, we approach the true image of $\det[I+kDM]$.

Lemma 5 states that $k<k_m$ will not destabilize the system. Geometrically, this places the Δ_{V_i} on the boundary of D and guarantees a unique stability margin k_m. By using these lemmas, the convergence theorem[1] follows.

Theorem 2: Let k, M, D, $\det[I+kDM]$, and $\det[I+kVM]$ be defined as previously given; then one can construct an iterate algorithm that converges to k_m. If k_m is finite, then this procedure identifies the parameters $\delta_i \in D$ at which k_m is determined. There are three steps involved in determining k_m: 1) determine the lower bound on k_m, 2) determine the upper

bound on k_m, and 3) iterate to converge lower and upper bounds on k_m.

The actual procedure involved in each step is very problem-dependent. As one may expect, there are several special cases concerning the critical vertices that vary the algorithm. For example, let co{det$[I+kVM]$} intercept the origin between two critical vertices F_i and F_j, one or both of which are coincident. For this case, different logic is required when splitting the domain into subdomains. deGaston and Safonov[1] present an excellent exposition of these special cases. They are briefly summarized here.

Special Case 1

The co{det$[I+kVM]$} intercepts the origin at a single isolated critical vertex (ICV) $F_i = \det[I+k_l\Delta_{V_i}M]$. Then $m(i,j)=0$, $k_l=k_u=k_m$, and the algorithm stops. The parameters that cause instability are at the vertex V_i.

Special Case 2

The co{det$[I+kVM]$} intercepts the origin between two ICVs F_i and F_j, where $m(i,j)=1$. Both points F_i and F_j are contained in the mapped hypercube image det$[I+kDM]$. With Lemma 2, the line segment connecting these two vertices is also contained in the mapped hypercube image det$[I+kDM]$. Thus, $k_l=k_u=k_m$ and the algorithm stops. The Δ along this edge of the hypercube that is destabilizing is given by

$$\Delta_\beta = \text{diag}\left[(1-\beta)V_i + \beta V_j\right]$$

where

$$\beta \in (0,1) \qquad \text{such that } \det[I+k\Delta_\beta M]=0 \tag{12}$$

If either of these special cases is true, the application of step 1 determines k_m. Only if $m(i,j)\geq 2$ does the algorithm progress further.

Consider ICVs F_i and F_j with $m(i,j)\geq 2$. The upper bound on k_m is determined by examining the $m(i,j)!$ vertex paths between F_i and F_j. The upper bound k_u is determined by the largest k along one of the $m(i,j)!$ vertex paths that intercepts the origin. If k is increased and the origin is not intercepted, then $k_u=\infty$. Once the lower and upper bounds k_l and k_u have been determined, Lemmas 3–5 are used to converge to k_m.

Special Case 3

The co{det$[I+kVM]$} intercepts the origin between two critical vertices F_i and F_j in which one or both are coincident.

Special Case 3a

Consider the problem where F_i and F_j both intercept the origin, i.e., $\det[I+k\Delta_{V_i}M]=\det[I+k\Delta_{V_j}M]=0$. Then, $k_l=k_m$ and the stability margin is defined at multiple values of Δ_{V_i}.

Special Case 3b

There are several coincident vertices located at F_i and several at F_j in which m_c is defined as follows:

$$m_c = \min\{m(i,j)\} = 1$$
$$i = \{a,b,\}$$
$$j = \{s,t,\dots\} \tag{13}$$

Pick an $i\in\{a,b,\dots\}$ and $j\in\{s,t,\dots\}$. Thus, $m(i,j)=1$ and k_m is determined as in special case 2.

Special Case 3c

This is the same condition as in special case 3b, except that $m_c\geq 2$. For this case, domain splitting is used to divide the domain into subdomains. This is repeated along each of the vertex paths to each coincident critical vertex.

Let the set $\{z\}$ contain z coincident critical vertices at F_z and the set $\{y\}$ contain y critical vertices at F_y. Take the first two elements of the set $\{z\}$, say, a,b. Then $m(a,b)\geq 1$, since both a,b are vertices of the hypercube. Split the domain along the edge between these two vertices with an orthogonal cut. This creates two subdomains D_1 and D_2, each containing one of the critical vertices a and b. Continue this process z-2 times, creating z subdomains, each having an isolated critical vertex at F_z. Repeat this same process for the critical vertices in $\{y\}$. This creates zy subdomains, each having two critical vertices. Apply the procedures of the preceding special cases to each of these subdomains.

Autopilot and Missile Dynamics

The longitudinal flight control system for a bank-to-turn cruise missile is shown in Fig. 3. By using block diagram manipulations, the block diagram of Fig. 3 is transformed into the analysis structure of Fig. 1. In Fig. 3, the transfer function matrix $\boldsymbol{K}(s)$ describes only controller dynamics. The nominal rigid-body longitudinal dynamics, containing uncertain parameters, are represented by $\boldsymbol{G}(s)$. This autopilot design[17] uses proportional plus integral control elements in closing the inner pitch-rate loop and outer acceleration loop. The autopilot feedback gains are $K_a=-0.0015$, $K_q=-0.32$, $a_Z=2.0$, and $a_q=6.0$.

The longitudinal (pitch) missile dynamics form a single-input multi-output design model. The autopilot commands normal-body acceleration using tail fin control. The states modeled in the open-loop rigid-body airframe are α, q, δ, and $\dot\delta$ (angle of attack, pitch rate, fin deflection, and fin rate). The differential equations used to describe the open-loop dynamics are

$$\dot\alpha = Z_\alpha\alpha + q + Z_\delta\delta$$
$$\dot q = M_\alpha\alpha + M_\delta\delta$$
$$\ddot\delta = -2\zeta\omega\dot\delta - \omega^2\delta + \omega^2\delta_c \tag{14}$$

The measurements available are normal acceleration $A_Z = VZ_\alpha\alpha + VZ_\delta\delta$ (ft/s^2) and pitch rate q (rad/s). The scalar control input $u=\delta_c$ (rad) is the fin-angle command.

The foregoing aerodynamics have been linearized and represent a trim angle of attack of 16 deg, Mach number of 0.8, and altitude of 4000 ft. The following parameters are the nominal values of the dimensional aerodynamic stability derivatives: $Z_\alpha=-1.3046$ (1/s); $Z_\delta=-0.2142$ (1/s); $M_\alpha=47.7109$ (1/s^2); and $M_\delta=-104.8346$ (1/s^2). The sign of M_α determines the stability of the open-loop airframe. When M_α is positive, the airframe is unstable. This occurs when the aerodynamic center of pressure is forward of the center of gravity. The remaining system parameters are missile velocity $V=886.78$ (ft/s) and fin actuator damping and natural frequency $\zeta=0.6$ and $\omega=113.0$ (rad/s), respectively.

The transfer function matrix from the fin command to normal acceleration A_Z and pitch rate q is

$$\boldsymbol{G}(s) = \begin{bmatrix} \dfrac{\omega^2V(Z_\delta s^2 + Z_\alpha M_\delta - Z_\delta M_\alpha}{(s^2 - Z_\alpha s - M_\alpha)(s^2+2\zeta\omega s+\omega^2)} \\ \dfrac{\omega^2(M_\delta s + M_\alpha Z_\delta - M_\delta Z_\alpha}{(s^2 - Z_\alpha s - M_\alpha)(s^2+2\zeta\omega s+\omega^2)} \end{bmatrix} = \begin{bmatrix} \dfrac{A_Z}{\delta_c} \\ \dfrac{q}{\delta_c} \end{bmatrix} \tag{15}$$

Note that the acceleration transfer function contains a right-half plane (RHP) zero. When open loop unstable, the acceleration transfer function contains both a pole and zero in the RHP.

The autopilot design $\boldsymbol{K}(s)$ stabilizes the nominal plant model $\boldsymbol{G}(s)$ using output feedback, as shown in Fig. 3. The

analysis problem is to determine the perturbation bounds on the foregoing imprecisely known, dimensional, aerodynamic stability derivatives such that the closed-loop system remains stable. These aerodynamic parameters are modeled as follows:

$$Z_\alpha = Z_{\alpha 0}(1+\delta_1)$$

$$Z_\delta = Z_{\delta 0}(1+\delta_2)$$

$$M_\alpha = M_{\alpha 0}(1+\delta_3)$$

$$M_\delta = M_{\delta 0}(1+\delta_4) \tag{16}$$

with each $|\delta_i| < 1$.

Calculating the Real Multiloop Stability Margin

This section presents the details of a Fortran program called ROBUSTR that computes the lower bound on the real margin. ROBUSTR is a computer program that implements the signal flow graph model decomposition and computes the convex-hull lower bound on the real margin. Figure 6 presents a flowchart of ROBUSTR. The following paragraphs detail the program's content and algorithms.

Block 1: Model Initialization

In this block two important steps are performed. The first is the development of the diagonal perturbation analysis model isolating the real-parameter variations into the Δ matrix. The second step is the definition of the parameter-space hypercube modeling the parameter variations.

First, a signal flow graph model is drawn for the control system. Each parameter p_i that is to be varied is drawn isolated on an individual branch. The multiplicative parameter uncertainty model is $p_i = p_{i0}(1+\delta_i)$, where p_{i0} is the nominal value of the parameter p_i. The signal flow graph branches for p_i are shown in Fig. 7. The analysis model uses $\Delta = \text{diag}[\delta_i]$ with $M(s)$ a full matrix (shown in Fig. 1). The ΔM model is formed by breaking the δ_i branches and forming input and output nodes as shown in Fig. 7. The transfer matrix $M(s)$ is formed by computing the transfer function between each input node and each output node. This task is simplified through the use of a Fortran 77 subroutine written by Mears.[18]

This subroutine, Perseus (from Greek mythology, Perseus was the son of Zeus and Danaë who through his wit was able to slay Medusa, a Gorgon whose sight would turn a beholder into stone), will find the determinant of a matrix that has polynomial elements. Coupled with Cramer's rule from linear algebra, Perseus can be used to determine transfer functions for polynomial matrix equations of the form

$$A(s)X = B(s)U \tag{17}$$

where $A(s)$ is an $n_a \times n_a$ polynomial matrix, $B(s)$ a $n_a \times n_p$ polynomial matrix, X the output variable vector of length n_a, and U the input variable vector of length n_p. From Cramer's rule, the transfer function between output X_i and input U_j is simply the ratio of determinants

$$\frac{X_i}{U_j} = \frac{|A\sim|}{|A|} \tag{18}$$

where A is defined by Eq. (17), and $A\sim$ the matrix A with its ith column replaced by the jth column of the matrix B.

Fortunately, signal flow graph representations can be easily put into polynomial matrix form (as can block diagrams, state-space equations, and sets of simultaneous differential equations), so that Perseus is ideally suited to finding the parameter variation transfer matrix $M(s)$. Although our approach presented here uses polynomials to form the model $M(s)$, a state-space model for $M(s)$ could also be employed. This would be preferred for higher-order systems where polynomials can affect numerical accuracy.

Table 1 Hypercube vertices for $n = 4$ parameters

Vertex	Z_α	Z_δ	M_α	M_δ
V_1	−1	−1	−1	−1
V_2	−1	−1	−1	1
V_3	−1	−1	1	−1
V_4	−1	−1	1	1
V_5	−1	1	−1	−1
V_6	−1	1	−1	1
V_7	−1	1	1	−1
V_8	−1	1	1	1
V_9	1	−1	−1	−1
V_{10}	1	−1	−1	1
V_{11}	1	−1	1	−1
V_{12}	1	−1	1	1
V_{13}	1	1	−1	−1
V_{14}	1	1	−1	1
V_{15}	1	1	1	−1
V_{16}	1	1	1	1

For a system with n_p model inputs, Perseus is called $(n_p)^2 + 1$ times. The first call forms the denominator polynomial. The remaining $(n_p)^2$ calls form the numerator polynomials for each of the transfer functions.

Figure 8 is a signal flow graph model of our pitch flight control system derived from Eq. (14). The aerodynamic stability derivatives Z_α, Z_δ, M_α, and M_δ are the parameters that may vary (V is missile velocity, and ζ and ω are the actuator damping and natural frequency and are assumed to be known). In the analysis model, the M matrix is a 4×4 matrix with Δ a diagonal 4×4 matrix. Since there are $n = 4$ uncertain parameters, the parameter-space hypercube modeling the variations has $2^4 = 16$ hypercube vertices. A matrix containing the vertices of this hypercube has 2^n rows with four columns and is shown in Table 1.

Block 2: Optimization over Frequency

The real margin calculation requires finding the minimum k over frequency and using this k as the bound over all frequencies. The program ROBUSTR can compute a frequency sweep or it uses a conjugate gradient algorithm to find the minimum k. The conjugate gradient algorithm was taken from Press et al.[19]

Block 3: Evaluate $M(j\omega)$

At each frequency the transfer matrix $M(j\omega)$ must be evaluated. ROBUSTR computes this matrix by forming a frequency vector $s = [(j\omega)^p \ (j\omega)^{p-1} \ \cdots \ 1]$ and computing the dot product between this vector and each numerator polynomial. The dot product with the denominator polynomial is computed once. The matrix is formed by looping through this calculation for each numerator product, dividing by the denominator product, and placing the quotient into the proper element of the M matrix.

Block 4: $k_i = k_{i-1} + \Delta k$

This block increments the stability margin k to a larger value. The loop stops when the convex-hull boundary intercepts the origin of the complex plane. The algorithm can be adjusted to control the accuracy (number of significant digits) of the minimum $k_i = k_l$ such that the determinant $\det[I + k_i \Delta M] = 0$.

Block 5: Map Vertices $\det[I + k_i \Delta M]$

This block maps the 2^n vertices of the parameter-space hypercube into the Nyquist plane at a single frequency ω. The Δ_{V_i} matrix containing ± 1s along its diagonal is formed for each vertex V_i. If we use vertex V_{13} from Table 1, the Δ_{V13} matrix is

$$\Delta_{V_{13}} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & -1 & \\ & & & -1 \end{bmatrix} \tag{19}$$

Each vertex V_i is mapped to point F_i in the complex plane using the determinant $\det[I+k_i\Delta_{V_i}M(j\omega)]=F_i$. The mapped vertex F_i will have real and imaginary components. The Re F_i are stored in a vector x and the Im F_i in a vector y. At each increment of k_i, the determinant of an $n_p \times n_p$ matrix is computed 2^n times.

Block 6: Form Convex Hull

The lower bound on the real margin is computed by forming a convex hull covering the mapped hypercube vertices and determining if the origin of the complex plane is contained on the boundary of this convex hull. The subroutine used to compute the convex hull was written by Tang[20] as part of his thesis, but was later modified to improve its efficiency.

The convex-hull subroutine works with the x,y vectors containing the real and imaginary parts of the mapped hypercube vertices. The procedure employed is referred to as package wrapping.[21] The subroutine works as follows.

The first step is to check for coincident points. If any of the 2^n mapped vertices are coincident in the Nyquist plane, a flag is set indicating a coincident point vertex. All but one of the coincident points is deleted from the set $\{x,y\}$.

Assume that no coincident vertex points were found. There are 2^n mapped vertex points in $\{x,y\}$. Next, the origin of the complex plane is added to the set of points $\{x,y\}$, increasing the number of points to 2^n+1.

If the number of uncertain parameters is greater than 4 ($n>4$), the subroutine executes an interior-point-elimination procedure. If $n\leq 4$, interior-point-elimination is bypassed. Interior-point-elimination forms a polygon using four points known to be on the convex hull and deletes the points interior to this polygon from the set $\{x,y\}$.

The polygon is formed from the set $\{x,y\}$ by searching for the four points having the largest and smallest x and y components. An example polygon is shown in Fig. 9, with the four points highlighted. If the points found result in a polygon with less than four sides, the set $\{x,y\}$ is rotated about the origin. The rotation angle is computed such that points 3–1 become vertical. Points 2 and 4 are then found by searching for the largest and smallest x components of the rotated points.

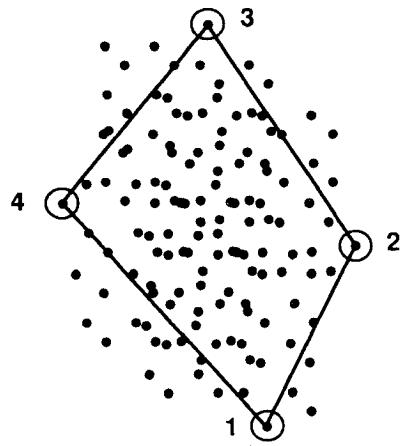

Fig. 9 Interior point elimination polygon.

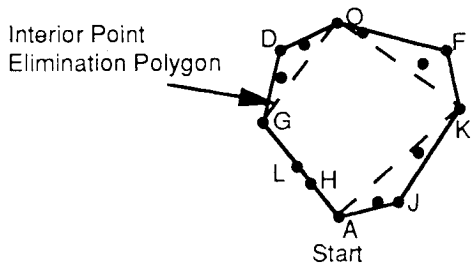

Fig. 10 Convex-hull package wrapping.

Table 2 Convex-hull boundary points

Vertex	Z_α	Z_δ	M_α	M_δ	Re(F_i)	Im(F_i)
V_1	−1	−1	−1	−1	2.32	−0.74
V_2	−1	−1	−1	1	0.40	−0.45
V_3	−1	−1	1	−1	2.45	−0.49
V_7	−1	1	1	−1	2.42	−0.46
V_{10}	1	−1	−1	1	0.10	−0.12
V_{14}	1	1	−1	1	−0.3E-5	−0.2E-4
V_{15}	1	1	1	−1	1.15	0.94
V_{16}	1	1	1	1	0.20	0.17

The algorithm tests for points that lie outside the polygon and places them in a new set $\{x_n,y_n\}$. Starting with anchor point 1, the angles from 1 to 2, denoted $\angle 12$, and from 1 to 4, denoted $\angle 14$, are computed. Any points that have an angle less than $\angle 12$ and one greater than $\angle 14$ are placed in $\{x_n,y_n\}$. After a point is placed in $\{x_n,y_n\}$, it is deleted from the set $\{x,y\}$. The next anchor point is 3. The angles from 3 to 2, denoted $\angle 32$, and from 3 to 4, denoted $\angle 34$, are computed. Any point with an angle greater than $\angle 32$ and less than $\angle 34$ is placed in the set $\{x_n,y_n\}$. The set $\{x_n,y_n\}$ contains the points 1–4 and the points outside of the polygon.

The angle function used in our algorithm does not employ any trigonometric functions or square roots. This function was taken from Sedgewick.[21] Our angle function (called θ) does not compute the true angle, but returns a more easily computed number with the same ordering properties. Since we are using an angle in sorting points, the true angle need not be computed.

After interior-point-elimination, a package-wrapping algorithm is used to determine the convex-hull boundary points. Ours is a modified version of the package-wrapping algorithm taken from Sedgewick.[21] Unlike the Sedgewick version, our algorithm will return all points on the convex-hull boundary. The Sedgewick algorithm fails when more than one point has the same θ value. Figure 10 outlines the package-wrapping procedure using an arbitrary set of points. Beginning at point A, the smallest angle θ identifies the next vertex point on the boundary of the convex hull. This process is repeated with the new convex-hull boundary point, and the convex-hull boundary is formed by progressing counterclockwise around the set of mapped points. Any vertex identified as a convex-hull boundary vertex is removed from the set $\{x_n,y_n\}$. The algorithm stops when the first point is reached again, closing the boundary. Note that the points L and H lying along a line connecting G and A would not be identified as boundary points by the Sedgewick algorithm. (This capability is used to determine if the convex-hull lower bound is exact.) The convex-hull subroutine outputs a vector of indices identifying the vertices that lie on the boundary of the convex hull.

Block 7: Zero Exclusion Test

The stability margin k_i is increased in magnitude until the origin is just included in the interior of the convex hull. This is referred to as the zero (origin) exclusion test. If k_i is less than k_l, then the array containing the indices for the convex-hull boundary vertices will contain the index of the origin that was added to $\{x,y\}$. As long as the origin (zero) is a convex-hull boundary point, the stability margin loop increments k_i to a larger value. As soon as the origin is not a boundary point, k_i is decreased by the current incremental value Δk.

Block 8: Test $k(\omega)$ for Minimum

The conjugate gradient algorithm adjusts the frequency ω to find the minimum $k(\omega)=k_l$. If a maximum number of iterations is reached, the algorithm automatically stops. The program output is the stability-margin bound k_l, the critical frequency ω_l, and the hypercube vertices whose line segment intercepts the origin. If the line segment that intercepts the origin was mapped from an edge of the hypercube, then the convex-hull bound is an exact bound.

Table 3 ROBUSTR CPU usage

Autopilot	Uncertain parameters, n	Model inputs, n_p	Node equations, n_a	CPU time to form model, s	Analysis CPU time, s	
Pitch pitch-rate command	4	4	13	0.6	43	
Pitch acceleration command	4	4	18	1.6	44	
Roll-yaw	4	4	40	57.5	38	
roll-rate	5	5	40	78.0	165	(0.04 h)
command		6	6	40	96.5	289 (0.08 h)
	7	7	40	117.5	1033	(0.28 h)
	8	8	40	136.1	3093	(0.86 h)
	9	9	40	156.4	13683	(3.80 h)

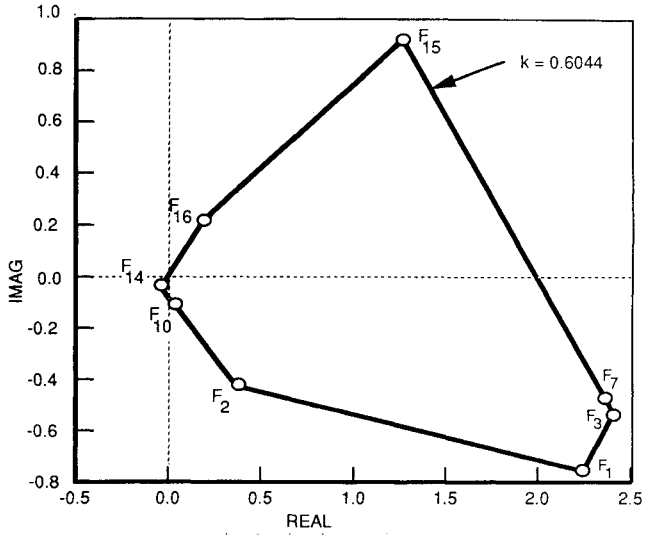

Fig. 11 Convex hull co[det($I+kVM$)].

ROBUSTNESS THEORY	% PERTURBATION
SMALL GAIN THEORM	13.8%
SSV μ (UNIT WEIGHTS)	49.0%
STABILITY HYPERSPHERE Xp = δ	0.1%
STABILITY HYPERSPHERE p= A α0+ b	20.3%
STABILITY HYPERSPHERE (LYAPUNOV – UNSCALED)	0.007%
STABILITY HYPERSPHERE (LYAPUNOV – SCALED)	0.02%
KHARITONOV'S THEOREM	15.3%
deGASTON - SAFONOV	60.44%
MONTE CARLO EIGEN ANALYSIS	60-61%

Fig. 12 Robustness test summary.

Missile Autopilot Analysis Results

The missile autopilot problem shown in Fig. 8 was analyzed by ROBUSTR. A subroutine was written to input the signal flow graph model data. These data are used by subroutine Perseus to compute the transfer function elements of $M(s)$. For this problem, there are $n=4$ uncertain parameters and $n_p=4$ model inputs.

The program calculated using the convex-hull boundary the lower bound $k_l=0.6044$ at $\omega=3.46$ rad/s. The convex-hull boundary points are listed in Table 2 and correspond to the hypercube vertices given in Table 1. Figure 11 displays the convex-hull boundary in the Nyquist plane. The vertex points on co{det$[I+kVM]$} are labeled to determine which vertices are critical ones.

The next step in computing k_m is to determine the upper bound k_u. This is computed by examining the vertex paths between critical vertices. Figure 11 shows that the line segment from F_{14}–F_{16} intercepts the origin. It is clear from Fig. 11 that these vertices are two isolated critical vertices.

The ICVs V_{14} and V_{16} are

	Z_α	Z_δ	M_α	M_δ
V_{14}	1.0	1.0	−1.0	1.0
V_{16}	1.0	1.0	1.0	1.0

These two vertices differ in the third coordinate. Thus, $m(14,16)=1$. By using Lemma 2, the line segment connecting these two vertices is contained in the mapped hypercube image det$[I+kDM]$. Thus, $k_l=k_u=k_m$ and the algorithm stops. The value $k=0.6044$ is the multiloop stability margin k_m. Applying this uniformly over each of the aero parameters produces a 60% variation bound.

The parameter values that first cause instability are determined from Eq. (12). The value of β along the line segment F_{14}–F_{16} is $\beta=6.31\times10^{-5}$. This yields

$$\Delta_\beta=\text{diag}[1.0 \quad 1.0 \quad -0.99987 \quad 1.0]$$

which is very close to the vertex V_{14}. The closed-loop characteristic polynomial for the missile flight control system has the following coefficients:

s^6: 1

s^5: $2\zeta\omega - Z_\alpha$

s^4: $\omega^2 K_q K_a V Z_\delta + \omega^2 - M_\alpha - 2\zeta\omega Z_\alpha$

s^3: $\omega^2 K_q\left[K_a V(a_Z + a_q)Z_\delta + M_\delta\right] - 2\zeta\omega M_\alpha - \omega^2 Z_\alpha$

s^2: $\omega^2 K_q\left[a_q(M_\delta + a_Z K_a V Z_\delta) + (K_a V - 1)(Z_\alpha M_\delta - Z_\delta M_\alpha)\right]$

s^1: $\omega^2 K_q(Z_\alpha M_\delta - Z_\delta M_\alpha)\left[a_q(K_a V - 1) + a_Z K_a V\right]$

s^0: $\omega^2 K_q a_Z a_q K_a V(Z_\alpha M_\delta - Z_\delta M_\alpha)$

The smallest combination of destabilizing variations occurs when Z_α, Z_δ, and M_δ are at a positive 60% variation and M_α is at a negative 60% variation ($\Delta_\beta = \text{diag}[1 \quad 1 \quad -1 \quad 1]$). When this variation occurs, the aforementioned closed-loop characteristic polynomial fails to be Hurwitz.

CPU Usage

Table 3 summarizes ROBUSTR CPU usage on a Vax station 3200 (2.7 Mips) for three missile autopilot applications. The pitch dynamics contain four uncertain aerodynamic parameters ($n = 4$); the roll-yaw dynamics contain nine uncertain aerodynamic parameters ($n = 9$).

The number of node equations n_a describes the dimension of the A matrix in Eq. (17). The signal flow graph model will have n_p inputs. Using Cramer's rule [Eq. (18)] to compute $M(s)$ requires computing the determinant of an $n_a \times n_a$ polynomial matrix [the A matrix in Eq. (18)] $(n_p)^2 + 1$ times. The roll-yaw results (Table 3) show that our application of Cramer's rule with Perseus is a polynomial-time algorithm in forming the analysis model $M(s)$. (For fixed n_a, CPU usage is proportional to n_p.)

Analysis CPU usage dramatically increases with the number n of uncertain parameters. Although ROBUSTR uses a polynomial-time algorithm in determining the convex hull,[21] ROBUSTR is an exponential-time analysis algorithm. Each time we increase k_i (in computing the lower bound k_l), the 2^n hypercube vertices are mapped into the Nyquist plane using the determinant $\det[I + k_i \Delta M]$. As n increases, the number of points computed is increased exponentially. Further work is needed to reduce the impact of this increase.

Comparison with Other Robustness Tests

Using results taken from Wise,[22-24] Fig. 12 summarizes autopilot stability robustness predictions obtained from two singular-value robustness tests, the real multiloop stability margin, a Monte Carlo eigenanalysis, and results based on several polynomial tests. The only robustness test not found to be conservative was the real multiloop stability margin calculation by deGaston and Safonov.

Conclusions

Our combination of signal flow graphs with Perseus has created an efficient procedure capable of computing robustness analysis models for independent real-parameter variations. This same procedure can be extended to form controller synthesis models, isolating uncertainties in Δ and controller transfer functions in K, by forming the interconnection plant matrix P. These types of $\Delta - P - K$ models are used in the synthesis of H_∞ optimal controllers.

Our use of interior point elimination with package wrapping has produced an efficient convex-hull routine. However, we will always have to evaluate the 2^n vertex points at least once in the algorithm. The exponential explosion in the number of vertex points will continue to be a problem with the deGaston-Safonov algorithm. Further research is needed in this area.

References

[1]DeGaston, R. R., and Safonov, M., "Exact Calculation of the Multiloop Stability Margin," *IEEE Transactions on Automatic Control,* Vol. 33, No. 2, Feb. 1988, pp. 156-171.

[2]"Multivariable Analysis and Design Techniques," AGARDograph Lecture Series 117, Oct. 1981.

[3]*IEEE Transactions on Automatic Control: Special Issue on Linear Multivariable Control Systems,* Vol. AC-26, No. 1, Jan. 1981.

[4]Doyle, J. C., "Robustness of Multiloop Linear Feedback Systems," *Proceedings of the 17th IEEE Conference on Decision and Control* (San Diego, CA), Inst. of Electrical and Electronics Engineers, New York, Dec. 1978, pp. 12-18.

[5]Doyle, J. C., "Structured Uncertainty in Control System Design," *Proceedings of the 24th IEEE Conference on Decision and Control* (Ft. Lauderdale, FL), Inst. of Electrical and Electronics Engineers, New York, Dec. 1985, pp. 260-265.

[6]Doyle, J. C., Wall, J. E., and Stein, G., "Performance and Robustness Analysis for Structured Uncertainty," *Proceedings of the 21st IEEE Conference on Decision and Control* (Orlando, FL), Inst. of Electrical and Electronics Engineers, New York, Dec. 1982, pp. 629-636.

[7]Morton, B. G., "New Applications of Mu to Real Parameter Variation Problems," *Proceedings of the 24th IEEE Conference on Decision and Control* (Ft. Lauderdale, FL), Inst. of Electrical and Electronics Engineers, New York, Dec. 1985, pp. 233-238.

[8]Morton, B. G., and McAfoos, R. M., "A Mu-Test for Robustness Analysis of a Real-Parameter Variation Problem," *Proceedings of the American Control Conference* (Boston, MA), American Automatic Control Council, June 1985, pp. 135-138.

[9]Jones, R. D., "Structured Singular Value Analysis for Real Parameter Variations," *Proceedings of the AIAA Guidance, Navigation, and Control Conference,* AIAA, Washington, DC, 1987 (AIAA Paper 87-2589).

[10]Fan, M. K. H., and Tits, A. L., "m-Form Numerical Range and the Computation of the Structured Singular Value," *IEEE Transactions on Automatic Control,* Vol. 33, No. 3, 1988, pp. 284-289.

[11]Kharitonov, V. L., "Asymptotic Stability of an Equilibrium Position of a Family of Systems of Linear Differential Equations," *Differensial Uravnen.,* Vol. 14, No. 11, 1978, pp. 2086-2088.

[12]Barmish, B. R., and DeMarco, C. L., "Criteria for Robust Stability with Structured Uncertainty: A Perspective," *Proceedings of the American Control Conference* (Minneapolis, MN), American Automatic Control Council, June 1987, pp. 476-481.

[13]Sideris, A., "A Polynomial Time Algorithm for Checking the Robust Stability of a Polytope of Polynomials," *Proceedings of the American Control Conference* (Pittsburgh, PA), American Automatic Control Council, May 1989, pp. 651-565.

[14]Wise, K. A., "A Comparison of Six Robustness Tests Evaluating Missile Autopilot Robustness to Uncertain Aerodynamics," *Proceedings of the American Control Conference* (San Diego, CA), American Automatic Control Council, June 1990, pp. 755-763.

[15]Wise, K. A., Mears, B. D., and Poolla, K., "Missile Autopilot Design Using H^∞ Optimal Control with μ-Synthesis," *Proceedings of the American Control Conference* (San Diego, CA), American Automatic Control Council, June 1990, pp. 2362-2367.

[16]Zadeh, L. A., and Desoer, C. A., *Linear System Theory,* McGraw-Hill, New York, 1963.

[17]Wise, K. A., "Optimizing Singular Value Robustness Measures in a Conventional Bank-to-Turn Missile Autopilot Design," *Proceedings of the AIAA Guidance, Navigation, and Control Conference,* AIAA, Washington, DC, 1988, pp. 296-306 (AIAA Paper 88-4089).

[18]Mears, B. C., "Open Loop Aspects of Two Wheeled Vehicle Stability Characteristics," Ph.D. Dissertation, Univ. of Illinois, Urbana, IL, Nov. 1989.

[19]Press, W., Flanney, B., Teykolsky, S., and Vetting, W., *Numerical Recipes,* Cambridge University Press, New York, 1986.

[20]Tang, C., "A General Program Evaluating Control System Robustness to Real Parameter Variations," Master's Thesis, Southern Illinois Univ., Edwardsville, IL, Dec. 1989.

[21]Sedgewick, R., *Algorithms,* Addison-Wesley, Reading, MA, 1988.

[22]Wise, K. A., "Singular Value Robustness Tests for Missile Autopilot Uncertainties," *Journal of Guidance, Control, and Dynamics* (to be published).

[23]Wise, K. A., "Flight Control System Sensitivity to Uncertain Parameter: Stability Hypersphere Radius Calculation," *Journal of Guidance, Control, and Dynamics* (to be published).

[24]Wise, K. A., "A Comparison of Six Robustness Tests Evaluating Missile Autopilot Robustness to Uncertain Aerodynamics," *Proceedings of the American Control Conference* (San Diego, CA), American Automatic Control Council, May 1990, pp. 755-763.