NASA/TM-1998-208715

Bi-Level Integrated System Synthesis (BLISS)

Jaroslaw Sobieszczanski-Sobieski
Langley Research Center, Hampton, Virginia

Jeremy S. Agte and Robert R. Sandusky, Jr.
The George Washington University
Joint Institute for Advancement of Flight Sciences
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

August 1998

Acknowledgments

The authors gratefully acknowledge the contributions of Dr. Srinivas Kodiyalam of the Engineous Co. for providing test results for the Aircraft Optimization and the Electronic Package Optimization using a software package called iSIGHT.

Available from the following:

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

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 487-4650

BI-LEVEL INTEGRATED SYSTEM SYNTHESIS (BLISS)

*Jaroslaw Sobieszczanski-Sobieski**, *Jeremy S. Agte*†,
and Robert R. Sandusky, Jr‡

Abstract

BLISS is a method for optimization of engineering systems by decomposition. It separates the system level optimization, having a relatively small number of design variables, from the potentially numerous subsystem optimizations that may each have a large number of local design variables. The subsystem optimizations are autonomous and may be conducted concurrently. Subsystem and system optimizations alternate, linked by sensitivity data, producing a design improvement in each iteration. Starting from a best guess initial design, the method improves that design in iterative cycles, each cycle comprised of two steps. In step one, the system level variables are frozen and the improvement is achieved by separate, concurrent, and autonomous optimizations in the local variable subdomains. In step two, further improvement is sought in the space of the system level variables. Optimum sensitivity data link the second step to the first. The method prototype was implemented using MATLAB and iSIGHT programming software and tested on a simplified, conceptual level supersonic business jet design, and a detailed design of an electronic device. Satisfactory convergence and favorable agreement with the benchmark results were observed. Modularity of the method is intended to fit the human organization and map well on the computing technology of concurrent processing.

0. Introduction

Optimization of complicated engineering systems by decomposition is motivated by the obvious need to distribute the work over many people and computers to enable simultaneous, multidisciplinary optimization. It is important to partition the large undertaking into subtasks, each small enough to be easily understood and controlled by people responsible for it. This implies granting people in charge of a subtask a measure of authority and autonomy in the subtask execution, and allowing human intervention in the entire optimization process.

Reconciliation of the need for subtask autonomy with the system level challenge of "everything influences everything else" is difficult. Each of the leading MDO methods that have evolved to date (survey papers: Balling and Sobieszczanski-Sobieski, 1996; and Sobieszczanski-Sobieski, J., and Haftka, R. T, 1997) tries to address that difficulty in a different way. In the system optimization based on the Global Sensitivity Equations (GSE) (Sobieszczanski-Sobieski, J. 1990, Olds, J. 1992, Olds, J. 1994), the partitioning applies only in the sensitivity analysis while optimization involves all the design variables simultaneously. The Concurrent SubSpace Optimization method provides for separate optimizations within the modules (Sobieszczanski-Sobieski, J. 1988, Renaud and Gabriele, 1991, 1993, and 1994; Stelmack and S. Batill, 1998) but handles all the design variables simultaneously in the coordination problem. The Collaborative Optimization method (Braun and Kroo, 1996; Sobieski and Kroo, 1998) also enables separate optimizations within the modules, each performed to minimize a difference between the state and design variables and their target values set in a coordination problem. This problem combines the system optimization with the system analysis, therefore its dimensionality may be quite large.

Most of the above method implementations had to overcome difficulties with integration of dissimilar codes. This has stimulated use of Neural Nets and Response Surfaces as means by which subdomains in the design space may be explored off-line and still be represented to the entire system. Unfortunately, effectiveness of this approach is limited to approximately 12 to 20 independent variables, hence, it is best suited for the early design phase. Consequently, a clear need remains

* Manager, Computational AeroSciences, and Multidisciplinary Research Coordinator, NASA Langley Research Center, MS 139 Hampton VA 23681, AIAA Fellow

† Graduate Research Scholar Assistant, George Washington University, Joint Institute for Advancement of Flight Sciences, LT, United States Air Force

‡ Professor, George Washington University, Joint Institute for Advancement of Flight Sciences, AIAA Fellow

for a method applicable in later design phases when the number of design variables is much larger. Methods that build a path in design space fit that requirement. Ultimately, one needs both domain-exploring methods and path-building methods, enhanced with seamless 'gear-shifting' between the two.

Motivated by the above state of affairs, BLISS attacks the problem by performing an explicit system behavior and sensitivity analysis using the GSE, autonomous optimizations within the subsystems performed to minimize each module contribution to the system objective under the local constraints, and a coordination problem that engages only a relatively small number of the design variables that are shared by the modules. Solution of the coordination problem is guided by the derivatives of the behavior and local design variables with respect to the shared design variables. These derivatives may be computed in two different ways, giving rise to two versions of BLISS.

In either version, BLISS builds a gradient-guided path, alternating between the set of disjointed, modular design subspaces and the common system-level design space. Each segment of that path results in an improved design so that if one starts from a feasible state, the feasibility in each modular design subspace is preserved while the system objective is reduced. In case of an infeasible start, the constraint violations are reduced while the increase of the objective is minimized. Because the system analysis is performed at the outset of each segment of the path, the process can be terminated at any time, if the budget and time limitations so require, with the useful information validated by the last system analysis. In addition to enabling complete human control in the subspace optimization, BLISS allows the engineering team to exercise judgment, at any point in the procedure, to intervene before committing to the next successive pass.

BLISS has been developed in a prototype form and has been successfully demonstrated on the small-scale test cases reported herein.

1. Notation

BB - black box, a module, in the mathematical model of a system.

$BBA(Y_r,(Z,X_r))$ - analysis of BB_r to compute Y_r for given Z and X_r

$BBOF_r$ - BB Objective Function computed in BB_r

$BBOPT_r(X_r,\phi_r,G_r)$ - optimization in BB_r defined by eq.(2.1/9)

$BBOSA_r(X_{r,opt},Z,Y_{r,s})$ - analysis of BB optimum for sensitivity to parameters

$BBSA(D(Y_r,(Z,X_r,Y_{r,s}))$ - sensitivity analysis of BB_r to compute its output derivatives w.r.t. Z, X_r, and $Y_{r,s}$

D(V1,V2) - total derivative dV1/dV2

d(V1,V2) - partial derivative $\partial V1/\partial V2$;
D(), and d() dimensionality depends on the dimensionalities of V1 and V2:
V1 and V2 - are both scalars, then D and d are scalars
V1 vector, V2 scalar, then D and d are vectors
V1 scalar, V2 vector, then D and d are vectors
V1 vector, V2 vector, then D and R are matrices

G_o - vector of constraints active at the constrained minimum, length NG_o

G_r - vector of the constraint functions, $g_{r,t}$ local to BB_r, $g_{r,t} \leq 0$ is a satisfied constraint

G_{yz} - constraints in a BB that have a stronger depend ence on Y and Z, than on X

GSE - Global Sensitivity Equations (Sobieszczanski-Sobieski, 1990); GSE/OS - GSE/Optimized Subsystems.

I - identity matrix.

L - vector of the Lagrange multipliers corresponding to G_o, length NG_o

LP - Linear Programming

NB - the number of BBs in the system

NLP - NonLinear Programming

opt - subscript denotes optimized quantity

P - vector of parameters, p_i, kept constant in the process of finding the constrained minimum, length NP.

SA((P,Z,X),Y) - system analysis; a computation that outputs Y for a system defined by P, Z, and X

SOF - System Objective Function computed in one of the BBs

$SOPT(Z,\Phi)$ - system objective optimization defined by eq. (2.2.3/1)

SSA(D(Y,(Z,X)) - system sensitivity analysis to compute sensitivity of the system response Y w.r.t. Z and X

TOGW - take-off gross weight

T - superscript denotes transposition.

X_r - vector of the design variables $x_{r,j}$, length NX_r, these variables are local to BB_r; X without subscript - a vector of all concatenated X_r, length NX

XL, XU - lower and upper bounds on X, side-constraints.

Y_r - vector of behavior (state) variables output from BB_r, these are the coupling variables; an element of Y_r is denoted $y_{r,i}$; some of $y_{r,i}$ are routed as inputs to other BBs, and may also be routed as output to the outside; the Y_r length is NY_r; Y without subscripts - a vector of all concatenated Y_r, length NY

$Y_{r,s}$ - vector of variables input to BB_r from BB_s, these are the coupling variables; an element of $Y_{r,s}$ is denoted $y_{r,s,i}$; note that by this definition $Y_{r,s}$ is a subset of Y_r, vector length $NY_{r,s}$

Z - vector of the design variables z_k that are shared by two or more BBs, these are the system-level variables; length NZ

0 - subscript denotes the present state from which to extrapolate, or the optimal state.

ZL, ZU - lower and upper bounds on Z, side-constraints

Δ - increment

ΔZL, ΔZU - move limits

ϕ_r - the local objective function in BB_r

Φ - the system objective function equated to one, particular $y_{1,i}$

2. The Algorithm

In this section, the symbols defined in Notation are used in a shorthand manner without repeating their definitions.

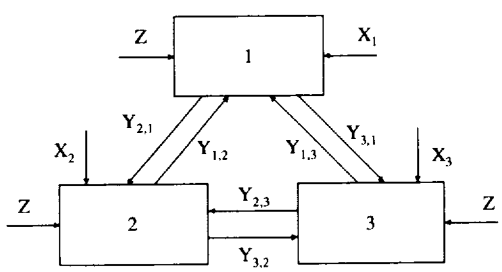

Figure 1: System of Coupled BBs

The algorithm is introduced using an example of a generic system of three BBs, as shown in Figure 1. Three is a number small enough for easy conceptual grasp and compact mathematics, yet large enough to unfold patterns that readily generalize to larger NB. Even though the system in Figure 1 is generic, it may be useful to bear a specific example in mind. Let it be an aircraft so that:

BB1 - performance analysis

BB2 - aerodynamics

BB3 - structures

Φ - maximum range for given mission characteristics

$Y_{1,2}$ - includes the aerodynamic drag; $Y_{1,3}$ - includes the structural weight; $Y_{2,1}$ - includes Mach number; $Y_{3,1}$ - includes TOGW; $Y_{2,3}$ - includes the structural deformations that alter the aerodynamic shape; $Y_{3,2}$ - includes the aerodynamic loads

$g_{1,t}$ - a noise abatement constraint on the mission profile; $g_{2,t}$ - limit of the chordwise pressure gradient; $g_{3,t}$ - allowable stress

$x_{1,j}$ - cruise altitude; $x_{2,j}$ - leading edge radius; $x_{3,j}$ – sheet metal thickness in the wing skin panel No. 138

z_1 - wing sweep angle; z_2 - wing aspect ratio; z_3 - wing airfoil maximum depth-to-chord ratio; z_4 - location of the engine on the wing

The system in Figure 1 is characterized by BB level design variables X, and by system-level design variables Z. As a reference, if an all-in-one optimization were performed, observing the system at a single level and making no distinction between the treatment of X variables and the treatment of Z variables, the problem could be stated

Given:	X and Z	(1)
Find:	ΔX and ΔZ	
Minimize:	$\Phi(X,Z,Y(X,Z))$	
Satisfy:	$G(X,Z,Y(X,Z)$	

Since BLISS approaches this optimization by means of a system decomposition, the algorithm depends on the availability of the derivatives of output with respect to input for each BB. That assumes the differentiability of the BB internal relationships to at least the first order. It is immaterial how the derivatives are computed, finite differencing may always be used, but it is expected that in most cases one will utilize one of the more efficient analytical techniques (Adelman and Haftka, 1993).

The algorithm comprises the system analysis and sensitivity analysis, local optimizations inside of the BBs (that includes the BB-internal analyses), and the system optimization. We will not elaborate on SA beyond pointing out that it is highly problem-dependent, and likely to be iterative if there are any non-linearities in the BB analyses. Each pass through the BLISS procedure improves the design in two steps: first by concurrent optimizations of the BBs using the design variables X and holding Z constant; and next, by means of a system-level optimization that utilizes variables Z. We begin with the BB-level optimization.

2.1. BB-level (discipline or subsystem) optimizations.

The basis of the algorithm is the formulation of an objective function unique for each BB such that minimization of that function in each BB results in the minimization of the system objective function. To introduce that formulation let us begin with the system objective function (SOF). The SOF is computed as a single output item in one of the BBs; without loss of generality we assume that it is BB_1 so that

$$\Phi = y_{1,i} \tag{1}$$

is one of the elements of Y_1.

Total derivatives of Y w.r.t. $x_{r,j}$, $D(Y, x_{r,j})$, are computed according to Sobieszczanski-Sobieski, 1990, by solving a set of simultaneous, algebraic equations known as Global Sensitivity Equations, GSE, (see Appendix, Section 1, for details) for a particular $x_{r,j}$

$$[A]\,\{D(Y,x_{r,j})\} = \{d(Y, x_{r,j})\} \tag{2}$$

where A is a square matrix, NYxNY, composed of submatrices forming this pattern

$$\begin{bmatrix} I & A_{1,2} & A_{1,3} \\ A_{2,1} & I & A_{2,3} \\ A_{3,1} & A_{3,2} & I \end{bmatrix} \tag{3}$$

where I stands for identity matrix, NY_rxNY_r, and $A_{r,s}$ are matrices of the derivatives that capture sensitivity of the BB_r output to input. For example

$$\begin{aligned} A_{2,3} &= -[d(Y_2,Y_3)],\ NY_2xNY_3 \\ A_{3,2} &= -[d(Y_3,Y_2)],\ NY_3xNY_2 \end{aligned} \tag{4}$$

One should note that eq. 2 can be efficiently solved for many different $x_{r,j}$ using techniques available for linear equations with many right-hand sides.

Having $D(Y,x_{r,j})$ computed from eq. 2 for all $x_{r,j}$, we can express Φ as a function of X by the linear part of the Taylor series

$$\begin{aligned} \Phi = y_{1,i} = (y_{1,i})_0 + D(y_{1,i}, X_1)^T\Delta X_1 + \\ D(y_{1,i}, X_2)^T\Delta X_2 + D(y_{1,i}, X_3)^T\Delta X_3 \end{aligned} \tag{5}$$

where D-terms are vectors of length NX_r.

We see from eq. 5 that

$$\begin{aligned} \Delta\Phi = D(y_{1,i},X_1)^T\Delta X_1 + D(y_{1,i},X_2)^T\Delta X_2 + \\ D(y_{1,i},X_3)^T\Delta X_3 \end{aligned} \tag{6}$$

the three terms showing explicitly the contributions to $\Delta\Phi$ of the local design variables from each of the three BBs.

It is apparent that to minimize $\Delta\Phi$ we need to charge each BB with the task of minimizing its own objective. Using BB_2 as an example, objective ϕ_2 is

$$\phi_2 = D(y_{1,i}, X_2)^T\Delta X_{2,j}\ ,\ j = 1{-}{-}{-}{>}NX_2 \tag{7}$$

The above equations state mathematically the fundamentally important concept that in a system optimization the contributing disciplines should not optimize themselves for a traditional, discipline-specific objective such as the minimum aerodynamic drag or minimum structural weight. They should optimize themselves for a "synthetic" objective function that measures the influence of the BB_r design variables X_r on the entire system objective function.

Another way to look at it is to observe that, in long-hand

$$\begin{aligned} \phi_2 = D(y_{1,i}, x_{2,1})^T\Delta X_{2,1} + D(y_{1,i}, x_{2,2})^T\Delta X_{2,2} + \ldots \\ + D(y_{1,i}, x_{2,j})^T\Delta X_{2,j} + \ldots,\ j = 1{-}{-}{-}{>}NX_2 \end{aligned} \tag{8}$$

so it may be regarded as a composite objective function commonly used in multiobjective optimization. One may say, therefore, that in a coupled system the local disciplinary or subsystem optimizations should be multiobjective with a composite objective function. The composite objective should be a sum of the local design variables weighted by their influence on the single objective of the whole system. It should be emphasized that this is true also in that particular BB_r where Φ is being computed. In the aircraft example it is $\Phi = y_{1,i}$ in BB_1 according to eq. 1. However, the BB_1 optimization objective is not $\phi_1 = y_{1,i}$. Instead, it is ϕ_1 from an equation analogous to eq.8.

The local optimization problem may be stated formally for BB_2

Given:	X_2, Z, and $Y_{2,1}$, $Y_{2,3}$	(9)
Find:	ΔX_2; length NX_2	
Minimize:	$\phi_2 = D(y_{1,i},X_2)^T\Delta X_2$	
Satisfy:	$G_2 \le 0$, including side-constraints	

Incidentally, we adhere to the convention which calls for minimization of the objective function. If the application requires that function be maximized, as it does in the example of aircraft range, we convert the objective, e.g., Φ = - (range).

The optimization problem for BB_1 , and BB_3 are analogous. All three problems being independent of each other may be solved concurrently. This is an opportunity for concurrent engineering and parallel processing.

By solving eq. 9 for all three BBs, we have improved the system because, according to eq. 5 and 6, we have reduced Φ by $\Delta\Phi$, while satisfying constraints in each BB.

2.2. System-level optimization.

So far we have improved the system by manipulating X in the presence of a constant Z. We can score further improvement by exploiting Zs as variables. To do so we need to know how Z influences $\Phi = y_{1,i}$. That is, we need $D(y_{1,i},Z)$.

At this point, the BLISS algorithm forks into two alternatives, termed BLISS/A and BLISS/B.

2.2.1. BLISS/A

This version of BLISS computes the derivatives of Y with respect to Z by modified GSE, eq.(2.1/2) (equations from other sections are cited in (), the section number given before the /). The GSE modification accounts for the fact that optimization of a BB turns its X into a function of Y and Z that enter that particular BB as parameters. The modification leads to a new generalization of GSE that takes the following form

$$[M]\begin{Bmatrix} D(Y,z_k) \\ D(X,z_k) \end{Bmatrix} = \begin{Bmatrix} d(Y,z_k) \\ d(X,z_k) \end{Bmatrix} \qquad (1)$$

termed GSE/OS for GSE/Optimized Subsystems. The GSE/OS yields a vector $D(Y,z_k)$ and $D(X,z_k)$, and because Φ is one of the elements of Y, $\Phi = y_{1,i}$, we get the desired derivative $D(\Phi,z_k)$. Derivation and details of the GSE/OS structure, including the definition of the matrix M, are in Section 2 of the Appendix. At this point it will suffice to say that the matrix of coefficients in GSE/OS is populated with $d(Y_r,Y_s)$, $d(Y_r,X_r)$, and $d(X_r,Y_s)$. These terms and the RHS terms of $d(Y,z_k)$ and $d(X,z_k)$ are obtained from the following sources

- $d(Y_r,Y_s)$, $d(Y_r,X_r)$, and $d(Y,z_k)$ ------ from BBSA
- $d(X_r,z_k)$, $d(X_r,Y_s)$ -------------------- from BBOSA

The terms $d(X,z_k)$ and $d(X_r,Y_s)$ are the derivatives of optimum with respect to parameters that, in principle, may be obtained by differentiation of the Kuhn-Tucker conditions, e.g., an algorithm described in Sobieszczanski-Sobieski et al, 1982. That approach, however, requires second order derivatives of behavior, too costly in most large-scale applications. Therefore, an approximate algorithm adapted from Vanderplaats and Cai, 1986, is given in Section 3 of the Appendix. In that algorithm, parameters are perturbed by a small increment, one at a time, and the BB optimization is repeated by Linear Programming (LP) starting from the optimal point. Derivatives of optimal X and Y with respect to parameters are then computed by finite differences.

2.2.2. BLISS/B

This version of BLISS avoids calculation of $d(X_r,z_k)$ and $d(X_r,Y_s)$ altogether by using an algorithm that yields $D(\Phi,P)$, where P includes both Y and Z. The algorithm, described in literature (e.g., Barthelemy and Sobieszczanski-Sobieski, 1983) is based on the well-known notion that the Lagrange multipliers may be interpreted as the prices, stated in the units of Φ, for the constraint changes caused by incrementing p_i. For a general case of the objective F=F(P) and $G_o = G_o(P)$, the algorithm gives the following formula for $D(F,P)_o$

$$D(F,P)_o = d(F,P) + L^T d(G_o,P)$$

To use the above in BLISS, consider that in P we have an independent Z but Y=Y(Z) so that the terms d() require chain-differentiation. Hence, the above general formula tranforms to

$$\begin{aligned} D(y_{1,i},Z)_o{}^T &= (L^T d(G_o,Z))_1 + (L^T d(G_o,Z))_2 + \\ &(L^T d(G_o,Z))_3 + [(L^T d(G_o,Y))_1 + (L^T d(G_o,Y))_2 \\ &+ (L^T d(G_o,Y))_3\,](D(Y,Z)) + D(y_{1,i},Z)^T \qquad (1) \end{aligned}$$

where L is the vector of Lagrange multipliers and $(\ )_1$, $(\ )_2$, and $(\ )_3$ identify the BBs 1, 2, and 3.

The terms in the above equation originate from the following sources:

- $d(G_o,Z)$ and $d(G_o,Y)$ - BBSA performed on isolated BB_r
- L - obtained for BB_r at the end of BBOPT
- D(Y,Z) - from GSE in SSA
- $D(y_{1,i},Z)$ – the column corresponding to $y_{1,i}$ in the above matrix D(Y,Z)

BLISS/B is substantially simpler in implementation than BLISS/A and it eliminates the computational cost of one LP per parameter Y and Z. Optimizers that yield L as a by-product of optimization are available for use in BBOPT, or L may be obtained as described in Haftka and Gurdal, 1992.

2.2.3. Optimization in the Z-space.

Once $D(y_{1,i},Z)$ have been computed from either eq.(2.2.1/1) or as $D(y_{1,i},Z)_o$ from eq. (2.2.2/1), we can further improve the system objective by executing the following optimization, using any suitable optimizer

Given: Z and Φ_0 (1)

Find: ΔZ

Minimize: $\Phi = \Phi_0 + D(y_{1,i},Z)^T \Delta Z$

Satisfy: $ZL \le Z + \Delta Z \le ZU$; $\Delta ZL \le \Delta Z \le \Delta ZU$

Where Φ_0 is inherited from the previous cycle SA for X and Z (initialized if it is the first cycle). It is recommended to handle the Z constraints by means of a trust-region technique, e.g., Alexandrov 1996. In the above, term $D(y_{1,i}, Z)$ is a constrained derivative that protects $G_o = 0$ in all BBs. Therefore, the optimization is unconstrained except of the side-constraints and move limits.

However, some BBs may have constraints that depend on Z and Y more strongly than on X (in the extreme case some constraints may not be functions of X at all, only of Y and Z). Such constraints, denoted G_{yz}, may be difficult (or impossible) to satisfy by manipulating only X in BBOPT. To satisfy them, one must add to the Z-space optimization in eq.1 their extrapolated values

$$G_{yz}^T = G_{yz,0}^T + (d(G_{yz},Z) + d(G_{yz},Y)D(Y,Z))^T \Delta Z \le 0 \qquad (2)$$

where $d(G_{yz},Z)$, and $d(G_{yz},Y)$ are obtained from BBSA. In this instance, the Z-space optimization becomes a constrained one.

3. Iterative Procedure

The two operations, the local optimizations in the BBs and the system-level optimization, described in Sec. 2, result in a new system, altered because of the increments on X and Z. This means that inputs to and outputs from SA, BBA, BBSA, SSA, BBOPT, BBOSA (BLISS/A), and SOPT all need to be updated, and the sequence of these operations repeated with the new values of all quantities involved, including new values of all the derivatives because they would change if there were any nonlinearities in the system (as there usually are).

In a large-scale application where execution of each BLISS cycle may require significant resources and time, the engineering team may wish to review the results before committing to the next cycle. That intervention may entail a problem reformulation, such as overriding the variable values, deleting and adding variables, constraints, and even BBs.

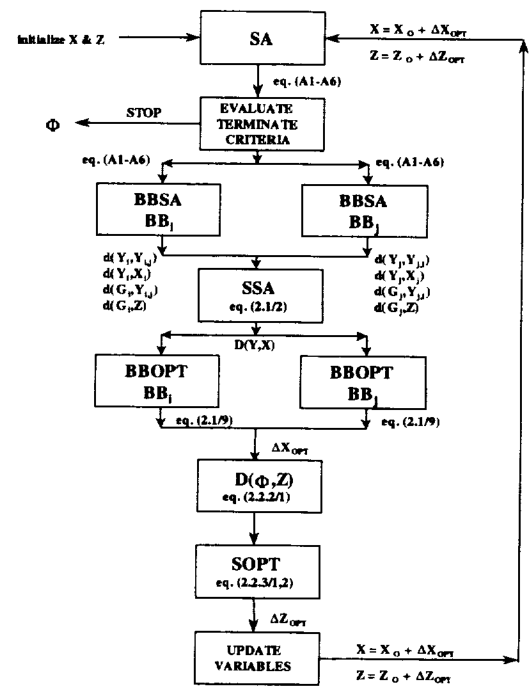

Figure 2: BLISS/B Flowchart

Thus, the following procedure emerges, illustrated also by a flowchart in Figure 2 for BLISS/B with the BLISS/A operations, if different, noted in [ ].

0. Initialize X & Z.

1. SA to get Ys and Gs; this includes BBAs for all BBs.

2. Examine TERMINATION CRITERIA, exercise judgment to override the results, modify the problem formulation, and CONTINUE or STOP.

3. BBSA to obtain $d(Y,X)$, $d(Y_{r,s},Y_s)$, $d(G,Z)$, and $d(G,Y)$, and SSA, eq. (2.1/2), to get $D(Y,X)$ [and

D(Y,Z)]; Here is an opportunity for concurrent processing.

4. BBOPT for all BBs, eq. (2.1/9) using ϕ formulatedindividually for each BB (eq. (2.1/6, 7)), get ϕ_{opt} and ΔX_{opt}; obtain L for G_o [skip L]. Here is an opportunity for concurrent processing.

5. Obtain $D(\Phi,Z)$ as in eq. (2.2.2/1). [Execute BBOSA to obtain d(X,Z) and d(X,Y), and form and solve GSE/OS (Appendix, Section 3) to generate D(Y,Z)]. Here is an opportunity for concurrent processing.

6. SOPT to get ΔZ_{opt} by eq. (2.2.3/1 and 2) herein.

7. Update all quantities, and repeat from 1.

$$X = X_0 + \Delta X_{opt};\ Z = Z + \Delta Z_{opt}$$

Note: Termination is placed as #2 after SA to ensure that the full analysis results document the final system design, as opposed to having it documented only by the extrapolated quantities. Also, at this point the engineering team may decide whether to intervene by modifying the variable values, and adding or deleting the design variables and constraints.

When started from a feasible design, the procedure will result in an improved system, while the local constraints are kept satisfied within extrapolation accuracy, even when terminated before convergence.

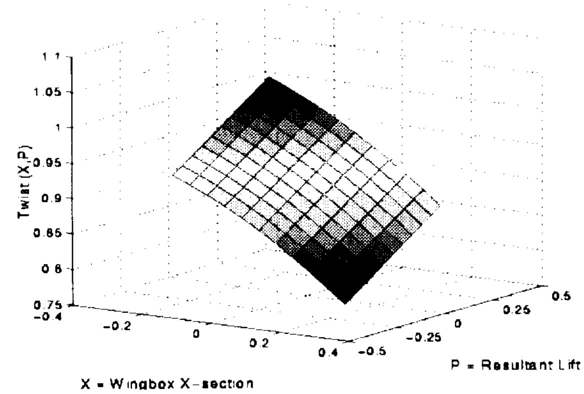

Figure 3: Polynomial Representation of Wing Twist

In case of an infeasible design start, the improvement will be in the sense of reductions in the constraint violations, while the objective may exhibit an increase, at least initially. The procedure achieves the improvement by virtue of optimization alternating between the domain of NB X-spaces (Step #4) and the single Z-space (Step #6).

Caveat: because in BLISS/B the extrapolation of Φ in eq. (2.2.3/1) is based on the Lagrange multipliers in eq. (2.2.2/1), its accuracy depends on the BBOPT yielding a feasible solution, and on the active constraints G_o remaining active for updated Z. If some constraints leave the active set G_o, or new constraints enter, a discontinuous change of the extrapolation error may result. For example, consider the wing aspect ratio AR as a Z-variable and suppose that for AR = 3 it is the stress due to the wing bending that is one of the active constraints in the structures BB. If optimization in the Z-space took the design to AR = 4, the next cycle may reveal that the stress constraint is satisfied but a flutter constraint becomes critical. Past experience (Sobieszczanski-Sobieski, 1983) shows that this discontinuity is likely to slow, but not to prevent, the process convergence, and may be controlled by adjusting the move limits.

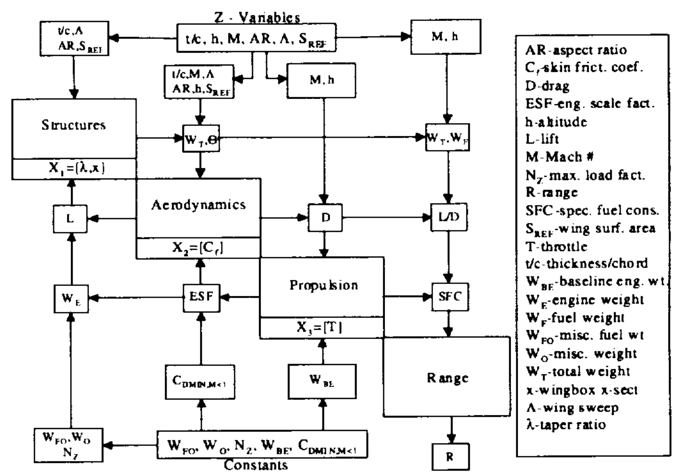

Figure 4: Data Dependencies for Range Optimization

4. Numerical Tests and Examples

BLISS/A was tested on a sample of test problems from Hock and Schittkowski, 1981, and on a design of an electronic package. BLISS/B was exercised on the latter, and also on a very simplified aircraft configuration problem. Both versions of BLISS performed as intended in all of the tests. The sole purpose of these initial numerical experiments was to test and to demonstrate the BLISS procedure logic and data flow, therefore, the BBs were merely surrogates of the numerical processes that need to be used in real applications.

4.1. Aircraft Optimization

The aircraft test was an optimum cruise segment of a supersonic business jet based on the 1995-96 AIAA Student Competition. This problem was selected because of its available data base and the availability of the black boxes written in Visual Basic in form of Excel spreadsheets. The supersonic business jet was modeled as a coupled system of structures (BB_1), aerodynamics (BB_2), propulsion (BB_3), and aircraft range (BB_4). All the disciplines were represented by modules comprising an analysis level typical for an early conceptual design stage.

var \ cycle*	1	2	3	4	5
Range (SSA)	535.79	1581.67	3425.35	3961.41	3963.98
Extpl. Error	-535.79	-536.67	-431.63	-56.26	-3.43
BB1 Extpl.	17.17	-0.16	-3.26	-0.86	0.00
BB2 Extpl.	16.85	0.00	0.00	0.00	0.00
BB3 Extpl.	26.00	110.92	-76.84	0.00	0.00
X Extpl.	60.02	110.75	-80.10	-0.86	0.00
Z Extpl.	449.19	1301.30	559.90	0.00	0.00
Range (Extpl.)	1045.00	2993.72	3905.15	3960.55	3963.98
λ	0.25	0.14951	0.17476	0.25775	0.38757
x	1	0.75	0.75	0.75	0.75
C_f	1	0.75	0.75	0.75	0.75
T	0.5	0.1676	0.20703	0.15624	0.15624
t/c	0.05	0.06	0.06	0.06	0.06
h(ft)	45000	54000	60000	60000	60000
M	1.6	1.4	1.4	1.4	1.4
AR	5.5	4.4	3.3	2.5	2.5
Λ(°)	55	66	70	70	70
$S_{ref}(ft^2)$	1000	1200	1400	1500	1500

*One cycle is one pass through the BLISS procedure

Table 1: A/C Results for 20% Move Limit

The aircraft optimization was a maximization of therange computed through the Breguet range equation. For testing purposes, additional design and state variables were introduced in BBs 1 through 3, and functional relationships not present in the original BBs were supplied to reflect what is commonly known about the typical functions involved in design. For example, stress is expected to fall as a reciprocal of the increase of the skin thickness in a wing box. Such relationships were represented by polynomial functions. One plot of such a function is shown in Figure 3, portraying the wing twist as a function of the wing box cross-sectional dimensions scale factor and the wing lift.

Section 4 of the Appendix defines the BBs in this ex ample by their input and output variables, and by the functions that link output to input. Table A1 also identifies local constraints and side constraints. Note that BB_2 contains a constraint that does not depend on its X or Y input, thus the Z-space optimization is a constrained one, per eq. (2.2.3/1 & 2). Side constraints on Z were judiciously selected to guard against conditions not accounted for in the BBAs. For example, the lower bound of 2.5 on aspect ratio stemmed from the subsonic performance considerations.

num \ den	λ	x	C_f	T	t/c
W_T	0.01146	1.71536	0.01981	-0.15744	0.12714
W_F	0	0	0	0	0.72626
Θ	-0.03342	0.19971	3.31E-15	-1.73E-14	-2.10E-14
L	0.01146	1.71536	0.01981	-0.15744	0.12714
D	-4.19E-05	0.00581	0.12457	-0.00049	0.68108
L/D	0.0115	1.7095	-0.1046	-0.15694	-0.54935
SFC	1.98E-20	-5.07E-18	-2.70E-17	0.08544	0
W_E	-4.40E-05	0.0061	0.13083	-1.03986	0.71531
ESF	-4.19E-05	0.00581	0.12457	-0.99059	0.68108
R	-0.00077	-0.12692	-0.12581	-0.07299	0.10115
num \ den	**h**	**M**	**AR**	**Sweep**	**S_{ref}**
W_T	-0.33931	0.31958	0.08208	0.2537	0.55182
W_F	0	0	-0.36043	0	1.09211
Θ	-1.93E-13	-6.15E-14	-0.10766	3.77E-14	-0.10766
L	-0.33931	0.31958	0.08208	0.2537	0.55182
D	-2.1339	2.00984	3.37E-06	-0.83983	0.99838
L/D	1.84108	-1.6507	0.08207	1.10064	-0.43675
SFC	0.12946	0.05555	2.31E-17	-1.86E-16	0
W_E	-2.24115	2.11086	3.54E-06	-0.88204	1.04857
ESF	-2.1339	2.00984	3.37E-06	-0.83983	0.99838
R	2.07616	-1.04784	-0.39618	0.82904	0.15535

Table 2: Normalized Y Derivatives w.r.t. X and Z

The BBs are coupled by the output-to-input data transfers (design structure matrix) depicted in Figure 4. Note that BB_4 is an analysis-only module and does not feedback any data to other BBs.

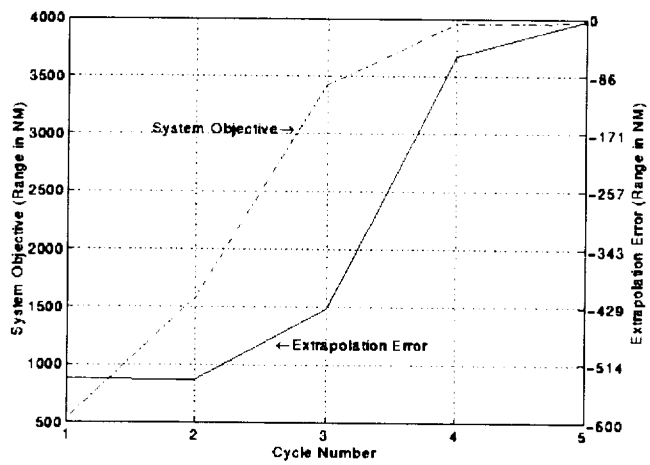

Figure 5: Range and Extrapolation Error Histogram

This test was conducted entirely using MATLAB 5 and its Optimization Toolbox. The entire MATLAB code listing for the aircraft range model may be found in Section 5 of the Appendix. To establish a benchmark, the system was first optimized using an all-in-one approach in which the MATLAB optimizer was coupled directly to SA and saw no distinction between the X and Z variables. Next, the test case was executed using

BLISS/B, starting at different infeasible initial points chosen by varying the six design variables that are not arguments in the polynomial functions. The choice of initial values for variables that are arguments of the polynomial functions was limited due to the nature of the polynomial formulation. This limitation is not a characteristic of the BLISS method itself, as the polynomial functions would not be required in a large scale optimization problem. With the move limits ranging from 10 to 70 %, the procedure convergence was satisfactory through the move limits of 60% for all initial points tested. However, in nearly all cases, no additional improvement in convergence rate was recorded for move limits greater than 20%. For instance, the objective function was advanced to within 1% of the benchmark in 5 passes for move limits 20 and 30%. Onset of an erratic behavior was observed with move limits increased past 60%, the procedure converged or diverged dependent on the starting point.

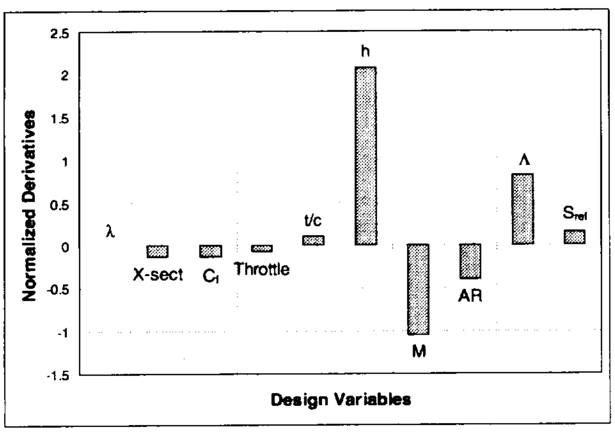

Figure 6: Range Sensitivities (1st cycle)

Table 1 displays a sample of typical results for the move limits value of 20%. It shows that the initial design range was extremely poor, only 536 nm. BLISS/B improvements advanced the range to 3964 nm. The range converged monotonically, although in some cases small amplitude oscillations were observed. Comparison of the extrapolated and actual values of the objective and constraints shows reasonable accuracy and conservatism of the extrapolations. The optimal values of the design variables reflect numerous tradeoffs typical for aircraft design. For instance, optimal t/c resulted, in part, from a trade-off between the wave drag and structural weight. Table 2 shows normalized (logarithmic) derivatives of all Ys , including the range, w.r.t. all the X and Z variables, sampled in Cycle 1 to illustrate sensitivity of the system solution to design variables.

Figure 5 illustrates the range histogram, and depicts the extrapolation error as being effectively controlled by the move limits. Range sensitivities to X and Z variables are shown in Figure 6. As expected, altitude and Mach number have the largest effect on the objective function, while taper ratio has the smallest.

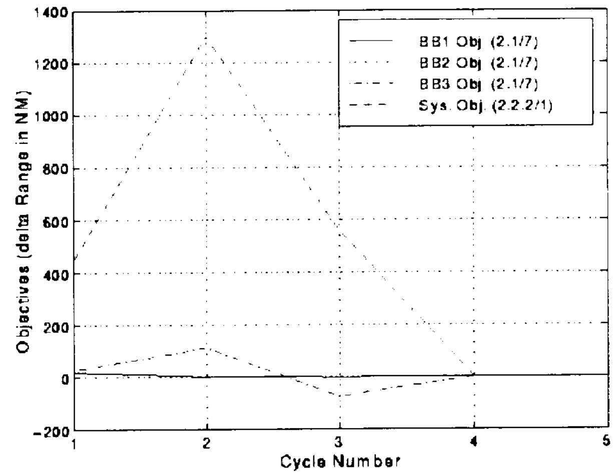

Figure 7: BB and System Contributions to Range

Figure 7 shows the individual BB and system contributions to the range objective in each cycle. Here it is observed that, in this particular case, the contribution of system level variables is significantly larger than that of the local variables in the extrapolation of range.

This test case was also implemented in a software package for system analysis and optimization called iSIGHT (iSIGHT, 1998). The iSIGHT and MATLAB results cross-check was completely satisfactory.

4.2 Electronic Package optimization

The electronic packaging was introduced as an MDO problem in Renaud, 1993. Its electrical and thermal subsystems are coupled because component resistance is influenced by operating temperatures and the temperatures depend on resistance.

The objective of the problem is to maximize the watt density for the electronic package subject to constraints. The constraints require the operation temperatures for the resistors to be below a threshold temperature and the current through the two resistors to be equal. The system diagram in Figure 8 shows the data dependencies for two BBs, representing electrical resistance analysis and thermal analysis. As Figure 8 indicates, there are no "natural" Z's in this case. Therefore, Z's were created as targets imposed on each of the Y's and the BBOPT's were required to match the Y values to those Z targets (similar as it is done in the Collaborative Optimization method). Details of the electronic packaging problem may be found in Padula, 1996.

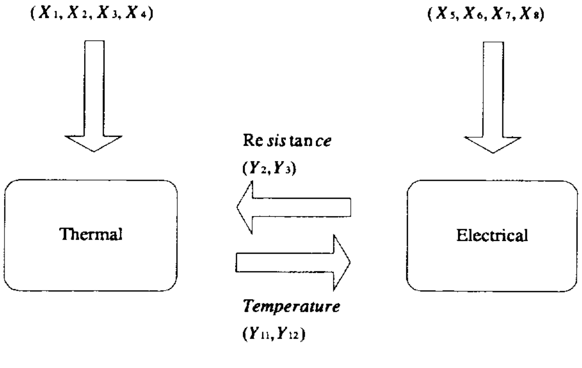

Figure 8: Electronic Packaging Data Dependencies

This test case was implemented in iSIGHT using BLISS/A and B. A benchmark result was obtained by executing an all-in-one optimization from various starting points ("A-in-O" column). BLISS/A and B were started from the same points. Table 4 displays the benchmark and the BLISS/A and B results as showing a good agreement. Table 4 also indicates a comparison of the computational labor (the "Work" column) measured by the number of BB evaluations necessary to converge the fixed-point iterations in BBAs and in SA, all repeated as needed to compute derivatives by finite-differences in a gradient-guided optimization. As Table 4 shows, the BLISS/B computational labor was substantially lower than the benchmark in all cases.

Method	Case	Initial Design Objective	Init. Des. Max Constr. Viol.	Final Design Objective	Fin. Des. Max Constr. Viol.	Work
A-in-O	1	7.79440E+01	2.16630E-08	6.39720E+05	1.22E-03	498
	2	6.83630E+03	-2.89560E-01	6.39720E+05	1.22E-03	264
	3	1.51110E+03	-4.29240E-02	6.36540E+05	1.45E-03	264
	4	1.46070E+01	-1.02490E-03	6.36940E+05	1.42E-03	175
BLISS/A	1	7.79440E+01	2.16630E-08	6.39700E+05	1.20E-03	436
	2	6.83630E+03	-2.89560E-01	6.39050E+05	1.18E-03	508
	3	1.51110E+03	-4.29240E-02	6.39050E+05	-4.89E-04	174
	4	1.46070E+01	-1.02490E-03	6.39290E+05	3.70E-04	313
BLISS/B	1	7.79440E+01	2.16630E-08	6.39720E+05	1.22E-03	365
	2	6.83630E+03	-2.89560E-01	6.39720E+05	1.22E-03	207
	3	1.51110E+03	-4.29240E-02	6.39720E+05	1.22E-03	114
	4	1.46070E+01	-1.02490E-03	6.39720E+05	1.22E-03	105

Table 4: Electronic Packaging Data

5. BLISS Status, Assessment, and Concluding Remarks

A method for engineering system optimization was developed to decompose the problem into a set of local optimizations (large number of detailed design variables) and a system-level optimization (small number of global design variables). Optimum sensitivity data link the subsystem and system level optimizations. There are two variants of the method, BLISS/A and BLISS/B, that differ by the details of that linkage. In the paper, the method algorithm was laid out in detail for a system of three subdomains (modules). Its generalization to NB subdomains is straightforward. The same algorithm may be used to decompose any of the local optimizations, hence optimization may be conducted at more than two levels.

MATLAB and iSIGHT programming languages were used to implement and test the method prototype on a simplified, conceptual level supersonic business jet design, and a detailed design of an electronic device. Dimensionality and complexity of the preliminary test cases was intentionally kept very low for an expeditious assessment of the method potential before more resources are invested in further development. Favorable agreement with the benchmark results and a satisfactory convergence observed in the above tests provided motivation for such development and future testing in larger applications.

Assessment of BLISS at the above development status is as follows. BLISS relies on linearization of a generally nonlinear optimization, therefore its effectiveness depends on the degree of nonlinearity. As any gradient-guided method, it guarantees a cycle-to-cycle improvement, but if the problem is non-convex, its convergence to the global optimum depends on the starting point and may strongly depend on the move limits. In this regard, BLISS's strong points are in the procedure being open to human intervention between the cycles and in the autonomy of the subdomain optimizations in local variables. These optimizations may be conducted by any means deemed to be most suitable by disciplinary experts, hence non-convexity, and strong nonlinearities in terms of the local variables often encountered in subdomains, e.g., the local buckling in thin-walled structures, are isolated and prevented from slowing down the system-level optimization convergence. On the other hand, the optimization robustness may be adversely affected by the local constraints leaving and entering the active constraint set. Effect of the above on BLISS/A is much less than on BLISS/B. This is probably the only reason to continue the development of BLISS/A alongside with BLISS/B, even though BLISS/B has a distinct advantage in simplicity and a much lower computational cost. Once there is more information on the relative merits and demerits of both variants, the better variant may be selected.

The demand BLISS puts on the computer storage is the same the subdomains would require for their own, stand-alone optimizations, with exception of the generation and solution of the Global Sensitivity Equations. If there is a pair of BBs that exchange large number of the $y_{r,s,i}$ - quantities, dimensionality of the corresponding matrices that store the derivatives, and computational cost of these derivatives needed to form GSE, may become prohibitive. Some relief may be provided here by application of condensation techniques and by deleting from GSE those derivative matrices that are known to have negligible effect on the system behavior.

The principal advantage of BLISS appears to lie in its separating overall system design considerations from the considerations of the detail. This makes the resulting mapping of its algorithm fit well on diverse, and potentially dispersed, human organizations. This advantage remains to be demonstrated in further development toward large-scale, complex applications.

6. References

Adelman, H. M.; and Haftka, R. T.: "Sensitivity Analysis of Discrete Systems," In *Structural Optimization: Status and Promise,* Kamat, M. P., ed., AIAA, Washington, D.C., 1993.

AIAA/UTC/Pratt & Whitney Undergraduate Individual Aircraft Design Competition, "Engine Data Package fo the Supersonic Cruise Business Jet RFP." 1995/1996.

Alexandrov, N.: "Robustness Properties of a Trust Region Framework for Managing Approximations in Engineering Optimization", Proceedings of the 6th AIAA/NASA/USAF Multidisciplinary Analysis and Optimization Symposium, AIAA-96-4102, pp.1056-1059, Bellevue, WA, Sept. 4-6, 1996.

Balling, R.J.; and Sobieszczanski-Sobieski, J.: "Optimization of Coupled Systems: A Critical Overview of Approaches," AIAA J., Vol. 34, No. 1, pp.6-17, Jan. 1996.

Barthelemy, J.-F; and Sobieszczanski- Sobieski, J.: "Optimum Sensitivity Derivatives of Objective Functions in Nonlinear Programming," *AIAA Journal*, Vol. 21, No. 6, pp. 913–915, June 1983.

Braun, R.D.; and Kroo, I.M.: "Development and Application of the Collaborative Optimization Architecture in A Multidisciplinary Design Environment," In SIAM J., Optimization 1996; Alexandrov, N., and Hussaini, M. Y., (eds.).

Hock, W.; Schittkowski, K.: "Test Examples for Nonlinear Programming Codes," Lecture Notes in Economics and Mathematical Systems (editors: Beckmann and Kunzi), 1981.

iSIGHT Designers and Developers Manual, version 3.1, Engineous Software Inc., Morrisville, North Carolina, 1998.

Haftka, R. T.; and Gurdal, Z.: "Elements of Structural Optimization," p.172; Kluwer Publishing, 1992.

MATLAB manual, the MathWorks, Inc., July 1993, version 5.0, 1997, and MATLAB Optimization Toolbox, User's Guide, 1996.

Olds, J.: "The suitability of selected multidisciplinary design and optimization techniques to conceptual aerospace vehicle design," Proc. 4th AIAA/NASA/USAF/OAI Symposium on Multidisciplinary Analysis and Optimization (held in Cleveland, OH). AIAA Paper No. 92-4791. 1992.

Olds, J.: "System sensitivity analysis applied to the conceptual design of a dual-fuel rocket SSTO," Proc. 5th AIAA/NASA/USAF/ISSMO Symp. on Multidisciplinary Analysis and Optimization (held in Panama City Beach, FL). AIAA Paper No. 94-4339. 1994.

Padula, S. L., Alexandrov, N., and Green, L. L.: "MDO Test Suite at NASA Langley Research Center." AIAA-96-4028. 1996. http://fmad-www.larc.nasa.gov/mdob/MDOB/index.html

Renaud, J. E.; Gabriele, G. A.: "Improved coordination in non-hierarchic system optimization" AIAA J. 31, 2367-2373 . 1993.

Renaud, J. E.; Gabriele, G. A.: "Approximation in non-hierarchic system optimization" AIAA J. 32, 198-205. 1994.

Renaud, J. E.; Gabriele, G. A.: "Sequential global approximation in non-hierarchic system decomposition and optimization," In: Gabriele, G. (ed.) Advances in Design Automation and Design Optimization, 19th Design Automation Conf. (held in Miami, FL), ASME Publication DE-Vol. 32-1, pp. 191-200.; 1991.

Renaud, J.E.:*An Optimization Strategy for Multidisciplinary Systems Design*, International Conference on Engineering Design, August 1993.

Sobieszczanski-Sobieski, J.; Barthelemy, J.-F. M.; and

Riley, K. M.: "Sensitivity of Optimum Solutions to Problems Parameters," *AIAA Journal*, Vol. 20, No. 9, pp. 1291–1299, September 1982.

Sobieszczanski-Sobieski, J.: "Optimization by decom position: a step from hierarchic to non-hierarchic systems" Presented at the Second NASA/Air Force Symp. on Recent Advances in Multidisciplinary Analysis and Optimization (held in Hampton, VA), NASA CP-3031, Part 1. Also NASA TM-101494. 1988.

Sobieszczanski-Sobieski, J.: "Sensitivity of Complex, Internally Coupled Systems" *AIAA Journal*, Vol. 28, No. 1, pp. 153–160, 1990.

Sobieszczanski-Sobieski, J.: "Optimization by Decomposition in Structural and Multidisciplinary Applications". Chapter in Optimization of Large Structural Systems; NATO-ASI, Sept. 1991, Berchtesgaden; publ. by Kluwer 1993.

Sobieszczanski-Sobieski, J.; and Haftka, R. T.: "Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments". Structural Optimization, pp. 1-23, Vol.14, No. 1, August 1997.

Sobieszczanski-Sobieski, J., James, B., and Dovi, A., "Stuctural Optimization by Multi-Level Optimization," AIAA *Journal*, Vol. 23, Nov. 1983.

Sobieski, I. P., and Kroo, I.M.: "Collaborative Optimization using Response Surface Estimation" AIAA-98-0915, 1998.

Stelmack, M.; and Batill, S.: "Neural Network Approximation of Mixed Continuous/Discrete Systems in Multidisciplinary Design," Univ. of Notre Dame, Notre Dame, IN. AIAA-98-0916, 1998.

Vanderplaats, G.N., and Cai, H.D.: "Alternative Methods for Calculating Sensitivity of Optimized Designs to Problem Parameters" NASA CP-2457, Proceedings of the Conference on Sensitivity Analysis in Engineering; NASA Langley Research Center, Hampton, VA, September 1986.

Appendix

This Appendix provides details of the Global Sensitivity Equations (GSE) applied to a system which optimizes BBs, the details of a technique for the BB Optimum Sensitivity Analysis, and the details of the aircraft range optimization model.

1. Global Sensitivity Equations

Derivatives of Y w.r.t. X, and Z, are obtained rigorously from the Implicit Function Theorem in Sobieszczanski-Sobieski,1990. The condensed derivation is provided below. It begins by recognizing that

(A1) $Y_{1,2} = Y_{1,2}(Z, X_{2,}, Y_{2,1}, Y_{2,3})$

(A2) $Y_{1,3} = Y_{1,3}(Z, X_3, Y_{3,1}, Y_{3,2})$

(A3) $Y_{2,1} = Y_{2,1}(Z, X_1, Y_{1,2}, Y_{1,3})$

(A4) $Y_{2,3} = Y_{2,3}(Z, X_3, Y_{3,1}, Y_{3,2})$

(A5) $Y_{3,1} = Y_{3,1}(Z, X_1, Y_{1,2}, Y_{1,3})$

(A6) $Y_{3,2} = Y_{3,2}(Z, X_2, Y_{2,1}, Y_{2,3})$

where the independent variables are X and Z.

Observe that eq. A1-A6 are coupled by $Y_{i,}$ e.g., $Y_{3,1}$ depends on $Y_{1,3}$ in eq. A5, and $Y_{1,3}$ depends on $Y_{3,1}$ in eq. A2. Consider for an example, the chain-differentiation w.r.t. $x_{1,j}$ applied to eq. A3. It yields

A7) $$D(Y_{2,1},x_{1,j}) = d(Y_{2,1},x_{1,j}) + d(Y_{2,1},Y_1)\, D(Y_1,x_{1,j}) + d(Y_{2,1},Y_3)\, D(Y_3,x_{1,j})$$

Repeating the above for the remaining equations, treating $Y_{2,1}$ as a subset of Y_1, and collecting the terms leads to eq. (2.1/2 and 3).

The derivatives of Y w.r.t. z_k are obtained by simply replacing $x_{r,j}$ with z_k in eq. (2.1/2) to obtain

A8) $[A]\ \{D(Y,z_k)\} = \{d(Y,z_k)\}$

2. GSE/Optimized Subsystems

In the preceding section both X and Z are independent variables. By virtue of BBOPT conducted for constant Z and Y inputs, X becomes dependent on Z so that derivatives of X w.r.t. exist in addition to derivatives of Y w.r.t. Z. For example, optimal X_2 depends on Z, $Y_{2,1}$, and $Y_{2,3}$, that are parameters in the optimization of BB_2. Hence, to compute the derivatives of Y and X w.r.t. Z, we begin by rewriting the functional relationships in eq. A1-A6, adding the new dependencies in all three BBs in the system,

(A9) $Y_{1,2} = Y_{1,2}(Z, X_{2,}, Y_{2,1}, Y_{2,3})$

(A10) $Y_{1,3} = Y_{1,3}(Z, X_3, Y_{3,1}, Y_{3,2})$

(A11) $Y_{2,1} = Y_{2,1}(Z, X_1, Y_{1,2}, Y_{1,3})$

(A12) $Y_{2,3} = Y_{2,3}(Z, X_3, Y_{3,1}, Y_{3,2})$

(A13) $Y_{3,1} = Y_{3,1}(Z, X_1, Y_{1,2}, Y_{1,3})$

(A14) $Y_{3,2} = Y_{3,2}(Z, X_2, Y_{2,1}, Y_{2,3})$

(A15) $X_1 = X_1(Z, Y_{1,2}, Y_{1,3})$

(A16) $X_2 = X_2(Z, Y_{2,1}, Y_{2,3})$

(A17) $X_3 = X_3(Z, Y_{3,1}, Y_{3,2})$

The same Implicit Function Theorem that is the basis of the GSE derivation may be applied to the above equations to obtain D(Y,Z). For example, by applying chain-differentiation to $Y_{2,1}$ treated as a subset of Y_2, we obtain

(A18) $$D(Y_2,z_k) = d(Y_2,z_k) + d(Y_2,X_2)D(X_2,z_k) + d(Y_2,Y_1)D(Y_1,z_k) + d(Y_2,Y_3)D(Y_3,z_k)$$

and for X_2, again as one example:

(A19) $$D(X_2, z_k) = d(X_2,z_k) + d(X_2,Y_1)D(Y_1,z_k) + d(X_2,Y_3)D(Y_3,z_k)$$

In the above, the D-terms are the total derivatives we seek, while the d-terms are the partial derivatives of two, different kinds. The derivatives of Y_r w.r.t. Y_s and Y_r w.r.t. X_r are obtained from $BBSA_r$ using any sensitivity analysis algorithm appropriate for the particular BB_r (including the option of finite differencing). The derivatives of X_r w.r.t. z_k and X_r w.r.t. Y_s are produced by an analysis of optimum for sensitivity to parameters, $BBOSA_r$, explained in later in this Appendix.

As a mathematical digression, one should mention at this point that the derivatives termed partial in the above would be called total in both BBSA and BBOSA. This is not a contradiction. It is so because the partial and total derivatives are hierarchically related in a multilevel system of parents and children. What is a total derivative in a child is partial at the parent level. In the application herein, the system of coupled three BBs is a parent, each BB is a child.

The chain-derivative expressions for Y_1, Y_3, X_1 and X_3 look similar to eq. A18 and A19, differences are only in the subscripts. When the entire set of six chain-derivative expressions is written it forms a set of simultaneous, algebraic equations in which the total derivatives such as $D(Y_2,z_k)$ and $D(X_2,z_k)$ appear as unknowns. **This is a new generalization of GSE, termed GSE/OS for GSE/Optimized Subsystems.** For the case of three-BB system, these equations may be presented in a matrix format like this

(A20) $$[M_{yy}]\{D(Y,z_k)\} + [M_{yx}]\{D(X,z_k)\} = d(Y,z_k)$$
$$[M_{xy}]\{D(Y,z_k)\} + [M_{xx}]\{D(X,z_k)\} = d(X,z_k)$$

The internal structure of the M-matrices in the above is

for $[M_{yy}]$:

$$\begin{bmatrix} I & -d(Y_1,Y_2) & -d(Y_1,Y_3) \\ -d(Y_2,Y_1) & I & -d(Y_2,Y_3) \\ -d(Y_3,Y_1) & -d(Y_3,Y_2) & I \end{bmatrix}$$

for $[M_{yx}]$:

$$\begin{bmatrix} -d(Y_1,X_1) & 0 & 0 \\ 0 & -d(Y_2,X_2) & 0 \\ 0 & 0 & -d(Y_3,X_3) \end{bmatrix}$$

for $[M_{xy}]$:

$$\begin{bmatrix} 0 & -d(X_1,Y_2) & -d(X_1,Y_3) \\ -d(X_2,Y_1) & 0 & -d(X_2,Y_3) \\ -d(X_3,Y_1) & -d(X_3,Y_2) & 0 \end{bmatrix}$$

and for $[M_{xx}]$:

$$\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}$$

Again, in the above, all $Y_{r,s}$ are folded into Y_r for compactness, and the terms are falling into the previously introduced categories as follows:

- M_{yy}, M_{yx}, and $d(Y,z_k)$ ------ from BBSA
- M_{xy} and $d(X,z_k)$ ----------- from BBOSA

As in GSE, one may obtain $D(Y_2,z_k)$ and $D(X_2,z_k)$ for all z_k, k = 1--->NZ by means of one of the efficient techniques for linear equations with many right-hand-sides.

3. Black Box Optimum Sensitivity Analysis (BBOSA)

Analysis of optimum for sensitivity to parameters (also called the post-optimum analysis) is preceded by solving a BB optimization problem

(A21) Given: P

Find: X

Minimize: (X,P)

Satisfy: G(X,P) ≤ 0, including side-constraints and move limits

where P are parameters kept constant while an optimizer manipulates X.

In the BLISS application, the parameters P in BB_r are z_k, and $Y_{r,s}$, because these quantities are kept constant in $BBOPT_r$.

After ϕ_{min}, and X_{opt} are found, one may seek sensitivity of these quantities to the change of P in form of the derivatives $D(\phi_{min},P)$ and $D(X_{opt},P)$.

Vanderplaats and Cai, 1986, review techniques, rigorous and approximate, available for calculating $D(X_{opt},P)$. The technique adapted for the BLISS/A purposes comprises the following steps executed for BB_r:

1. Choose parameter P_k, an element of Z or Y, and increment it by a ΔP
2. Use derivatives from SSA to extrapolate F and G_o

Table A1: BB Definitions

BB	Inputs	Internal	Outputs
Structures	$AR, \Lambda, {}^{t}/_{c}, S_{REF}, W_{FO}, W_O, W_E, L, N_Z, \lambda, x$	$t = \frac{{}^{t}/_{c}\, S_{REF}}{\sqrt{S_{REF}AR}}$; $b/2 = \sqrt{S_{REF}AR/2}$; $R = \frac{1+2\lambda}{3(1+\lambda)}$; $\Theta = pf(x, {}^{b}/_{2}, R, L)$; $Fo1 = pf(x)$; $W_W = (0.0051(W_T N_Z)^{0.557} S_{REF}^{0.649} AR^{0.5} ({}^{t}/_{c})^{-0.4} (1+\lambda)^{0.1} (0.1875 S_{REF})^{0.1} / \cos(\Lambda)) Fo1$; $W_{FW} = (5S_{REF}/18)({}^{2}/_{3}\, t)42.5$; $W_F = W_{FW} + W_{FO}$; $W_T = W_O + W_W + W_F + W_E$; $\sigma 1 \rightarrow \sigma 5 = pf({}^{t}/_{c}, L, x, {}^{b}/_{2}, R)$;	W_T, W_F, Θ
		Constraints: $\sigma 1 \rightarrow \sigma 5 \le 1.09$; $0.96 \le \Theta \le 1.04$	
Aero-dynamics	$M, h, \Lambda, AR, {}^{t}/_{c}, S_{REF}, W_T, \Theta, ESF, C_{Dmin,M<1}, C_f$	if $h < 36089$ft, $V = M1116.39\sqrt{1-(6.875e-06)h}$, $\rho = (2.377e-03)(1-(6.875e-06h))^{4.2561}$; $V = M968.1$, $\rho = (2.377e-03)e^{-(h-36089)\ 20806.7}$, if $h > 36089$ ft; $C_L = \frac{W_T}{0.5\rho V^2 S_{REF}}$; $Fo1 = pf(ESF, C_f)$; $C_{Dmin} = C_{Dmin,M<1} Fo1 + 3.05({}^{t}/_{c})^{5/3} \cos(\Lambda)^{3/2}$; $k = 1/(\pi 0.8AR)$; $Fo2 = pf(\Theta)$; $C_D = (C_{Dmin} + kC_L^2)Fo2$; $L = W_T$; $D = C_D 0.5\rho V^2 S_{REF}$; $dp/dx = pf({}^{t}/_{c})$	$L, D, \frac{L}{D}$
		Constraints: $dp/dx \le 1.04$, evaluated at system level	
Propulsion	M, h, D, W_{BE}, T	$\bar{T} = T * 16168.6$; $Temp = pf(M,h,T)$; $ESF = ({}^{D}/_{3})/\bar{T}$; $SFC = 1.1324 + 1.5344M - (3.2956e-05)h - (1.6379e-04)\bar{T} - 031623M^2 + (8.2138e-06)Mh - (10.496e-05)\bar{T}M - (8.574e-11)h^2 + (3.8042e-09)\bar{T}h + (1.0600e-08)\bar{T}^2$; $W_E = 3W_{BE}ESF^{1.05}$; $T_{UA} = 11484 + 10856M - 0.50802h + 3200.2M^2 - 0.29326Mh + (6.8572e-06)h^2$	SFC, W_E, ESF
		Constraints: $0.5 \le ESF \le 1.5$; $T \le T_{UA}$; $Temp \le 1.02$	
Range	$M, h, \frac{L}{D}, W_T, W_F, SFC$	$\theta = 1 - 6.875e-06 * h$, if $h < 36089$ft; $\theta = 0.7519$ if $h > 36089$ft; $R = \frac{M(L\ D)661\sqrt{\theta}}{SFC} \ln\left(\frac{W_T}{W_T - W_F}\right)$	R
Constants	$W_{FO} = 2000$lb; $W_O = 25000$lb; $N_Z = 6g$; $W_{BE} = 4360$lb; $C_{Dmin,M<1} = 0.01375$		
Side Constraints	$0.1 \le \lambda \le 0.4$; $0.75 \le x \le 1.25$; $0.75 \le C_f \le 1.25$; $0.1 \le T \le 1.0$; $0.01 \le {}^{t}/_{c} \le 0.09$; $30000 \le h \le 60000$; $1.4 \le M \le 1.8$; $2.5 \le AR \le 8.5$; $40 \le \Lambda \le 70$; $500 \le S_{REF} \le 1500$		

linearly and by Linear Programming solve

Find X
Minimize F
Satisfy $G_o \leq 0$
XL <= X <= XU; where XL and XU incorporate the side constraints and the move limits;

to obtain X_{opt}.

3. Approximate $D(X,P) = \Delta X_{opt}/\Delta P$

4. Repeat from #1 for all elements of Z and Y input into BB_r.

Repeated for all BBs, the above procedure yields a set of D(X,Z) and D(X,Y) to be entered as d(X,Z) and d(X,Y) into GSE/OS, eq. A20. Solution of eq. A20 provides D(X,Z) and D(Y,Z). The latter is substituted into eq. (2.2.3/2), and D(Φ,Z), extracted from D(Y,Z), goes into eq. (2.2.3/1).

4. A/C Range Optimization Model.

Table A1 shows the equations used in each of the BBs for the aircraft model. Polynomial functions are represented by 'pf()' with independent variables in the parentheses. Each polynomial function is of the form:

(A22) $PF = A_o + A_i*S^T + (1/2)*S*A_{ij}*S^T$

Where S is the vector of independent variables, and A_o, A_i, and A_{ij} are coefficient terms.

In calculating the polynomial functions using eq. A22, terms in the S vectors are in the same order as they appear in pf() in Table A1. The off diagonal terms of A_{ij} are random numbers between 0 and 1. For this model, they are

$$A_{ij} = \begin{bmatrix} -- & 0.3970 & 0.8152 & 0.9230 & 0.1108 \\ 0.4252 & -- & 0.6357 & 0.7435 & 0.1138 \\ 0.0329 & 0.8856 & -- & 0.3657 & 0.0019 \\ 0.0878 & 0.7248 & 0.1978 & -- & 0.0169 \\ 0.8955 & 0.4568 & 0.8075 & 0.9239 & -- \end{bmatrix}$$

The remaining coefficient are:

- Θ ---> A_o = [1.0]; A_i = [0.3 -0.3 -0.3 -0.2]; A_{ii} = [0.4 -0.4 -0.4 0];
- Fo1 ---> A_o = [1.0]; A_i = [6.25]; A_{ii} = [0];
- σ1 ---> A_o = [1.0]; A_i = [-0.75 0.5 –0.75 0.5 0.5]; A_{ii} = [-2.5 0 –2.5 0 0];
- σ2 ---> A_o = [1.0]; A_i = [-0.5 0.333 –0.5 0.333 0.333]; A_{ii} = [-1.111 0 –1.111 0 0];
- σ3---> A_o = [1.0]; A_i = [-0.375 0.25 –0.375 0.25 0.25]; A_{ii} = [-0.625 0 –0.625 0 0];
- σ4 ---> A_o = [1.0]; A_i = [-0.3 0.2 –0.3 0.2 0.2]; A_{ii} = [-0.4 0 –0.4 0 0];
- σ5 ---> A_o = [1.0]; A_i = [-0.25 0.1667 –0.25 0.1667 0.1667]; A_{ii} = [-0.2778 0 –0.2778 0 0];
- Fo2 ---> A_o = [1.0]; A_i = [0.2 0.2]; A_{ii} = [0 0];
- Fo3 ---> A_o = [1.0]; A_i = [0]; A_{ii} = [0.04];
- dp/dx ---> A_o = [1.0]; A_i = [0.2]; A_{ii} = [0];
- Temp ---> A_o = [1.0]; A_i = [0.3 –0.3 0.3]; A_{ii} = [0.4 –0.4 0.4];

Equations for SFC and the upper constraint bound on throttle setting in the Propulsion BB are polynomials representing surfaces fit to engine deck data (AIAA/UTC/Pratt & Whitney, 1995/96).

5. A/C Range MATLAB Code.

Included in the following pages is the MATLAB code for the aircraft range optimization model. The constrained optimization routine used in BB1OPT, BB2OPT, BB3OPT, and SYSOPT may be found in MATLAB's Optimization Toolbox and is based on a Sequential Quadratic Programming method. The finite differencing subfunctions in FIN_DIFF are simple one-step forward finite difference codes that use a 1 percent step increment.

```
%--------------------------------------------------------------------------------------------
%
%                          Program Listing
%
%        Name                                        Page Number
%
%        BLISS............................................... 16
%        SYSTEM_ANALYSIS..................................... 19
%        BB_WEIGHT........................................... 20
%        BB_DRAGPOLAR........................................ 23
%        BB_POWER............................................ 23
%        BB_RANGE............................................ 24
%        POLYAPPROX.......................................... 25
%        BB1OPT.............................................. 26
%        BB1WRAPPER.......................................... 28
%        BB2OPT.............................................. 28
%        BB2WRAPPER.......................................... 29
%        BB3OPT.............................................. 30
%        BB3WRAPPER.......................................... 31
%        SYSOPT.............................................. 32
%        SYSWRAPPER.......................................... 33
%        INBOUNDS............................................ 34
%        FIN_DIFF............................................ 34
%
% The electronic version of this code has been placed in custody of Dr. Jaroslaw
% Sobieski, NASA Langley Research Center, Hampton, VA 23681.
%
%--------------------------------------------------------------------------------------------


%--------------------------------------------------------------------------------------------
%
%                          Program BLISS
%
% This program calls a system analysis for an aircraft range optimization
% model, composed of the WEIGHT, DRAGPOLAR, and POWER black boxes
% (BB1, BB2, and BB3, respectively). Through black box (BBSA) and system
% sensitivity (SSA) analyses, it calculates the derivatives necessary to solve the
% Global Sensitivity Equations (Sobieszczanski_Sobieski, 1990) and solves
% them. Local optimizations are performed on each BB (BBOPT) as well as a
% system level optimization (SOPT) using a gradient guided path based on the
% Lagrange multipliers (OSAAA). Finally, all optimized changes to design
% variables are used to update the model for an improved range.
%
%     Author       : Jeremy S. Agte    NASA Langley/GWU    Spring '98
%
% Variables        :
%     A            - Coefficient matrix in GSE                              none
%     DY_AR        - Vector of total derivatives, behavior
%                    variables w.r.t aspect ratio                           vary
%     DY_Cf        - Vector of total derivatives, behavior
%                    variables w.r.t skin friction coefficient              vary
%     DYE1_Z       - Matrix of total derivatives, behavior
%                    variables from BB1 w.r.t Z variables                   vary
%     DYE2_Z       - Matrix of total derivatives, behavior
%                    variables from BB2 w.r.t Z variables                   vary
%     DYE3_Z       - Matrix of total derivatives, behavior
%                    variables from BB3 w.r.t Z variables                   vary
%     DY4_Z        - Vector of total derivatives, range w.r.t
%                    Z variables                                            vary
%     DY_h         - Vector of total derivatives, behavior
%                    variables w.r.t altitude                               vary
%     DY_Lamda     - Vector of total derivatives, behavior
%                    variables w.r.t wing sweep                             vary
%     DY_lamda     - Vector of total derivatives, behavior
%                    variables w.r.t taper ratio                            vary
%     DY_M         - Vector of total derivatives, behavior
%                    variables w.r.t Mach number                            vary
%     DY_Sref      - Vector of total derivatives, behavior
%                    variables w.r.t wing surface area                      vary
%     DY_T         - Vector of total derivatives, behavior
%                    variables w.r.t throttle setting                       vary
%     DY_tc        - Vector of total derivatives, behavior
%                    variables w.r.t thickness/chord ratio                  vary
%     DY_x         - Vector of total derivatives, behavior
%                    variables w.r.t wingbox x-section                      vary
%     DYX_nd       - Array of non-dimensional total derivatives,
%                    behavior w.r.t. X variables                            vary
```

```
%       DYZ_nd          - Array of non-dimensional total derivatives,
%                         behavior w.r.t. Z variables                  vary
%       ext_error       - Difference between previous pass extrapol-
%                         ated range and actual system analysis range  NM
%       GRADphi4_Z      - Vector of total derivatives at the optimal
%                         state, range w.r.t Z variables               vary
%       i0              - Design variable initial values               vary
%       phi_BBOPT       - Change in range due to X variables           NM
%       phi_SysOPT      - Change in range due to Z variables           NM
%       P_var           - Vector of current design variable values     vary
%       phi_X_Z         - Change in range due to X and Z variables     NM
%                         variables w.r.t taper ratio                  vary
%       vlb             - Lower bounds on design variables             vary
%       vub             - Upper bounds on design variables             vary
%       X1(1)           - Wing taper ratio                             none
%       X1(2)           - Wingbox x-sectional area as poly. funct.     p.f
%       X2              - Skin friction coefficient as poly. funct.    p.f.
%       X3              - Throttle setting                             none
%       Z(1)            - Thickness/chord ratio                        none
%       Z(2)            - Altitude                                     ft
%       Z(3)            - Mach number                                  none
%       Z(4)            - Aspect ratio                                 none
%       Z(5)            - Wing sweep                                   deg
%       Z(6)            - Wing surface area                            ft^2
%
%  Subfunctions         :
%         BB1_OPT               -Finds optimal change in X1 using MATLAB
%                                optimizer
%         BB2_OPT               -Finds optimal change in X2 using MATLAB
%                                optimizer
%         BB3_OPT               -Finds optimal change in X3 using MATLAB
%                                optimizer
%         FIN_DIFF              -Provides partial derivatives using one-
%                                step forward finite differencing
%         INbounds              -Non-dimensionalizes bounds on X and Z
%         system_analysis       -Solves for behavior variables using Gauss-
%                                Seidel iteration
%         Sys_OPT               -Finds optimal change in Z using MATLAB
%                                optimizer
%
%-----------------------------------------------------------------------------------------------

%----Initialize Variables ----%

vlb=[.1 .75 .75 .1 .01 30000 1.4 2.5 40 500];
i0=[.25 1 1 .5 .05 45000 1.6 5.5 55 1000];
vub=[.4 1.25 1.25 1 .09 60000 1.8 8.5 70 1500];
P_var=i0;
X1=i0(1:2);
X2=i0(3);
X3=i0(4);
Z=i0(5:10);
phi_X_Z = 0;

%----Begin BLISS Loop----%

for i=1:6

%----SYSTEM ANALYSIS----%

[Y1,Y2,Y3,Y4,Y12,Y14,Y21,Y23,Y24,Y31,Y32,Y34,G1,G2,G3,C,Twist_initial,x_in
itial,L_initial,R_initial,ESF_initial,Cf_initial,Lift_initial,tc_initial,M_initial,h_initial,
T_initial]=system_analysis(Z,X1,X2,X3,i0);

%----BBSA----%

[A,dY_lambda,dY_x,dY_Cf,dY_T,dY_tc,dY_h,dY_M,dY_AR,dY_Lambda,dY_Sref,
dg1_Z,dg2_Z,dg3_Z,dg1_YE1,dg2_YE2,dg3_YE3]=FIN_DIFF(Z,Y1,Y2,Y3,Y4,Y12,
Y14,Y21,Y23,Y24,Y31,Y32,Y34,X1,X2,X3,G1,G2,G3,C,Twist_initial,x_initial,L_in
itial,R_initial,ESF_initial,Cf_initial,Lift_initial,tc_initial,M_initial,h_initial,T_initial)
;

%----SSA----%

DY_lamda = A\dY_lambda;
DY_x = A\dY_x;
DY_Cf = A\dY_Cf;
DY_T = A\dY_T;
```

```
DY_tc=A\dY_tc;
DY_h=A\dY_h;
DY_M=A\dY_M;
DY_AR=A\dY_AR;
DY_Lambda=A\dY_Lambda;
DY_Sref=A\dY_Sref;

DYE1_Z=[DY_tc(4) DY_h(4) DY_M(4) DY_AR(4) DY_Lambda(4) DY_Sref(4);
DY_tc(8) DY_h(8) DY_M(8) DY_AR(8) DY_Lambda(8) DY_Sref(8)];
DYE2_Z=[DY_tc(1) DY_h(1) DY_M(1) DY_AR(1) DY_Lambda(1) DY_Sref(1);
DY_tc(3) DY_h(3) DY_M(3) DY_AR(3) DY_Lambda(3) DY_Sref(3); DY_tc(9)
DY_h(9) DY_M(9) DY_AR(9) DY_Lambda(9) DY_Sref(9)];
DYE3_Z=[DY_tc(5) DY_h(5) DY_M(5) DY_AR(5) DY_Lambda(5) DY_Sref(5)];
DY4_Z=[DY_tc(10) DY_h(10) DY_M(10) DY_AR(10) DY_Lambda(10)
DY_Sref(10)];

%----BBOPT----%

[vlb_nd,vub_nd]=INbounds(i0,vlb,vub);

[dX1,Lagrange1,phi_BB1OPT,BB1_G(i,:)]=BB1OPT(vlb_nd,vub_nd,i0,P_var,Z,Y21,
Y31,X1,x_initial,L_initial,R_initial,Lift_initial,Twist_initial,tc_initial,C,DY_lamda,
DY_x);
[dX2,Lagrange2,phi_BB2OPT]=BB2OPT(vlb_nd,vub_nd,i0,P_var,Z,Y12,Y32,X2,ES
F_initial,Cf_initial,Twist_initial,tc_initial,C,DY_Cf);
[dX3,Lagrange3,phi_BB3OPT,BB3_G(i,:)]=BB3OPT(vlb_nd,vub_nd,i0,P_var,Z,Y23,
X3,M_initial,h_initial,T_initial,C,DY_T);
phi_BBOPT = phi_BB1OPT + phi_BB2OPT + phi_BB3OPT;

%----OSAAA----%

Dphi1_Z = [Lagrange1'*dg1_Z];
Dphi2_Z = [Lagrange2'*dg2_Z];
Dphi3_Z = [Lagrange3'*dg3_Z];
Dphi1_Y = [Lagrange1'*dg1_YE1];
Dphi2_Y = [Lagrange2'*dg2_YE2];
Dphi3_Y = [Lagrange3'*dg3_YE3];

GRADphi4_Z = [Dphi1_Z + Dphi2_Z + Dphi3_Z + Dphi1_Y*DYE1_Z +
Dphi2_Y*DYE2_Z + Dphi3_Y*DYE3_Z + DY4_Z]

%----SOPT----%

[dZ,phi_SysOPT,G_sys(i,:)]=SysOPT(vlb_nd,vub_nd,i0,P_var,Y4,GRADphi4_Z,Z,dg
2_Z,G2);
phi_SysOPT=phi_SysOPT+Y4(1);
ext_error = -phi_X_Z - Y4(1);
phi_X_Z=-Y4(1)+phi_BBOPT+phi_SysOPT;

%----Non-dimensionalize Derivatives for Output Observation----%

DYX = [DY_lamda DY_x DY_Cf DY_T];
DYZ = [DY_tc DY_h DY_M DY_AR DY_Lambda DY_Sref];
[XY_init,ZY_init] = NonDim(X1,X2,X3,Y1,Y2,Y3,Y4,Z);
DYX_nd(:,:,i) = DYX.*XY_init;
DYZ_nd(:,:,i) = DYZ.*ZY_init;

%----Store Interim Results----%

Var(1:18,i)=[Y4 ext_error -phi_BB1OPT -phi_BB2OPT -phi_BB3OPT -phi_BBOPT
-phi_SysOPT -phi_X_Z X1 X2 X3 Z]';

%----Update X and Z variables----%

X1=X1+dX1;
X2=X2+dX2;
X3=X3+dX3;
Z=Z+dZ';
P_var=[X1 X2 X3 Z];

end %End BLISS loop

%----Format Output Parameters----%

RLB = 'Range_SSA ext_error dR_BB1 dR_BB2 dR_BB3 dR_X dR_Z Range_ext
TapRat WingBox Cf Thrtl t/c h M AR lambda Sref';
```

```
CLB = 'Pass_1 Pass_2 Pass_3 Pass_4 Pass_5 Pass_6 Pass_7 Pass_8 Pass_9 Pass_10
Pass_11 Pass_12 Pass_13 Pass_14 Pass_15 Pass_16 Pass_17 Pass_18 Pass_19
Pass_20 Pass_21 Pass_22 Pass_23 Pass_24 Pass_25 Pass_26 Pass_27 Pass_28
Pass_29 Pass_30 Pass_31 Pass_32 Pass_33 Pass_34 Pass_35 Pass_36 Pass_37
Pass_38 Pass_39 Pass_40';
printmat(Var,[],RLB,CLB);

RLB1 = 'Y1(1) Y1(2) Y1(3) Y2(1) Y2(2) Y2(3) Y3(1) Y3(2) Y3(3) Y4';
CLB1 = 'X1(1) X1(2) X2 X3';
%printmat(DYX_nd(:,:,1),'Non-Dimensional D(Y,X)',RLB1,CLB1);


RLB2 = 'Y1(1) Y1(2) Y1(3) Y2(1) Y2(2) Y2(3) Y3(1) Y3(2) Y3(3) Y4';
CLB2 = 'Z1 Z2 Z3 Z4 Z5 Z6';
%printmat(DYZ_nd(:,:,1),'Non_Dimensional D(Y,Z)',RLB2,CLB2);

G=[BB1_G G_sys(:,1) BB3_G];
RLB3 = 'Pass_1 Pass_2 Pass_3 Pass_4 Pass_5 Pass_6';
CLB3 = 'sig1 sig2 sig3 sig4 sig5 twist_u twist_l dp/dx ESF_u ESF_l temp Throttle';
printmat(G,'Constraints at Beginning of Pass',RLB3,CLB3);


%-----------------------------------------------------------------------------------------------
%
%                      Subfunction SYSTEM_ANALYSIS
%
%  This subfunction uses Gauss-Seidel iteration on the aircraft range optimization
%  model to compute behavior variables, given a set of design variables.  Black boxes
%  WEIGHT, DRAGPOLAR, and POWER are called.
%
%      Author        : Jeremy S. Agte     NASA Langley/GWU     Spring '98
%
%  Input Variables     :
%      i0             - Design variable initial values                  vary
%      X1(1)          - Wing taper ratio                                none
%      X1(2)          - Wingbox x-sectional area as poly. funct.        p.f.
%      X2             - Skin friction coefficient as poly. funct.       p.f.
%      X3             - Throttle setting                                none
%      Z(1)           - Thickness/chord ratio                           none
%      Z(2)           - Altitude                                        ft
%      Z(3)           - Mach number                                     none
%      Z(4)           - Aspect ratio                                    none
%      Z(5)           - Wing sweep                                      deg
%      Z(6)           - Wing surface area                               ft^2
%
%  Output Variables    :
%      C              - Vector of constants                             vary
%      Ga             - Vector of constraint values in BBa (a = 1,2,3)  vary
%      Ya             - Vector of behavior variables output from
%                       from BBa (a = 1,2,3)                            vary
%      Yab            - Vector of behavior variables output from
%                       BBa, input to BBb (a & b = 1,2,3)               vary
%      Y4             - Objective function output from BB4              NM
%      var_inititial  - preserved values for polynomial construction
%                       (var differs depending on particular poly.)     vary
%
%  Local Variables     :
%      Lu             - Test variable used in G-S iteration for
%                       convergence of lift                             lb
%      Weu            - Test variable used in G-S iteration for
%                       convergence of engine weight                    lb
%      ESFu           - Test variable used in G-S iteration for
%                       convergence of engine scale factor              none
%
%  Subfunctions        :
%          BB_weight          -Calculates a/c structural weights
%          BB_dragpolar       -Calculates aerodynamic values
%          BB_power           -Calculates propulsion values
%          BB_range           -Calculates system objective function
%
%-----------------------------------------------------------------------------------------------

func-
tion[Y1,Y2,Y3,Y4,Y12,Y14,Y21,Y23,Y24,Y31,Y32,Y34,G1,G2,G3,C,Twist_initial,
x_initial,L_initial,R_initial,ESF_initial,Cf_initial,Lift_initial,tc_initial,M_initial,h_in
itial,T_initial]=system_analyis(Z,X1,X2,X3,i0)

[Y1,Y2,Y3,Y4,Y12,Y14,Y21,Y23,Y24,Y31,Y32,Y34,C]=Y_variables;
```

```
%-----Preserve initial values for polynomial calculations-----%

Twist_initial=Y12(2);
x_initial=i0(2);
tc_initial=i0(5);
L_initial=sqrt(i0(8)*i0(10))/2;
R_initial=(1+2*i0(1))/(3*(1+i0(1)));
ESF_initial=Y32(1);
Cf_initial=i0(3);
Lift_initial=Y21(1);
M_initial=i0(7);
h_initial=i0(6);
T_initial=i0(4);

%-----Execute Gauss Seidel iteration on system to find Y variables-----%

Lu=Y21(1)+10;
Weu=Y31(1)+10;
ESFu=Y32(1)+10;
while ((abs(Lu-Y21(1))>(Y21(1)*.001)) | (abs(Weu-Y31(1))>(Y31(1)*.001)) |
(abs(ESFu-Y32(1))>(Y32(1)*.001)))
   Lu=Y21(1);
   Weu=Y31(1);
   ESFu=Y32(1);

%-----Call Black Boxes-----%

   [Y1,Y12,Y14,G1] =
BB_weight(Z,Y21,Y31,X1,x_initial,L_initial,R_initial,Lift_initial,Twist_initial,tc_ini
tial,C);

[Y2,Y21,Y23,Y24,G2]=BB_dragpolar(Z,Y12,Y32,X2,ESF_initial,Cf_initial,Twist_in
itial,tc_initial,C);
   [Y3,Y34,Y31,Y32,G3]=BB_power(Z,Y23,X3,M_initial,h_initial,T_initial,C);
   [Y4]=BB_range(Z,Y14,Y24,Y34);
end

%-----Write post-iterative variable to output file-----%

%file = 'in_out1.dat';
%write_var(Z,Y1,Y2,Y3,Y4,Y12,Y14,Y21,Y23,Y24,Y31,Y32,Y34,X1,X2,X3,C,file)
;


%-----------------------------------------------------------------------------------------------
%
%                          Subfunction BB_WEIGHT
%
%  This subfunction calculates the weight of the aircraft by structure and adds them
%  to obtain a total aircraft weight.  It calls the subfunction POLYAPPROX to com-
%  pute functions represented by polynomials.
%
%     Author          : Jeremy S. Agte     NASA Langley/GWU     Spring '98
%
%  Input Variables    :
%     C               - Vector of constants                              vary
%     L_initial       - Initial halfspan length                          ft
%     Lift_initial    - Initial lift                                     lb
%     R_initial       - Initial location of lift as fraction of halfspan none
%     tc_initial      - Initial thickness to chord ratio                 none
%     Twist_initial   - Initial wing twist                               p.f.
%     x_initial       - Initial wingbox x-sectional thickness            p.f.
%     X1(1)           - Wing taper ratio                                 none
%     X1(2)           - Wingbox x-sectional area as poly. funct.         p.f.
%     Y21             - Lift                                             lb
%     Y31             - Engine weight                                    lb
%     Z(1)            - Thickness/chord ratio                            none
%     Z(2)            - Altitude                                         ft
%     Z(3)            - Mach number                                      none
%     Z(4)            - Aspect ratio                                     none
%     Z(5)            - Wing sweep                                       deg
%     Z(6)            - Wing surface area                                ft^2
%
%  Output Variables   :
%     G1(1)           - Stress on wing                                   p.f.
%     G1(2)           - Stress on wing                                   p.f.
%     G1(3)           - Stress on wing                                   p.f.
```

```
%       G1(4)           - Stress on wing                     p.f.
%       G1(5)           - Stress on wing                     p.f.
%       G1(6)           - Wing twist as constraint           p.f.
%       Y1(1)           - Total aircraft weight              lb
%       Y1(2)           - Fuel weight                        lb
%       Y1(3)           - Wing twist                         p.f.
%       Y12(1)          - Total aircraft weight              lb
%       Y12(2)          - Wing twist                         p.f.
%       Y14(1)          - Total aircraft weight              lb
%       Y14(2)          - Fuel weight                        lb
%
%   Local Variables     :
%       L               - Halfspan                           ft
%       R               - Wing aerodynamic center            none
%       t               - Wing thickness                     ft
%       W_wing          - Weight of the wing                 lb
%       W_fuel_wing     - Wing aerodynamic center            lb
%
%   Subfunctions        :
%           PolyApprox          -Forms polynomial functions for desired variables
%
%---------------------------------------------------------------------------------------------------

func-
tion[Y1,Y12,Y14,G1]=BB_weight(Z,Y21,Y31,X1,x_initial,L_initial,R_initial,Lift_in
itial,Twist_initial,tc_initial,C)

%-----THIS SECTION COMPUTES THE TOTAL WEIGHT OF A/C-----%

t = Z(1)*Z(6)/sqrt(Z(6)*Z(4));   %--wing thickness
L=sqrt(Z(4)*Z(6))/2;   %--halfspan
R=(1+2*X1(1))/(3*(1+X1(1)));   %--wing aerodynamic center location

%-----Polynomial function calculating wing twist----%

S_initial1=[x_initial,L_initial,R_initial,Lift_initial];
S1=[X1(2),L,R,Y21(1)];
flag1 = [2,4,4,3];
bound1 = [.25,.25,.25,.25];
Y1(3) = PolyApprox(S_initial1,S1,flag1,bound1);
Y12(2) = Y1(3);

%-----Polynomial function calculating wingbox X-sectional thickness-----%

S_initial2=[x_initial];
S2=[X1(2)];
flag2=[1];
bound2=[.008];
Fo=PolyApprox(S_initial2,S2,flag2,bound2);
W_wing = Fo*(.0051*((Y21(1)*C(3))^.557)*(Z(6)^.649)*(Z(4)^.5)*(Z(1)^-.4)*((1+
X1(1))^.1)*((cos(Z(5)*pi/180))^-1)*((.1875*Z(6))^.1));

W_fuel_wing = (5*Z(6)/18)*(2/3*t)*(42.5);
Y1(2) = C(1) + W_fuel_wing;
Y1(1) = C(2) + W_wing + Y1(2) + Y31(1);
Y12(1) = Y1(1);
Y14(1) = Y1(1);
Y14(2) = Y1(2);

%-----THIS SECTION COMPUTES THE TOTAL WEIGHT OF A/C-----%

%---THIS SECTION COMPUTES CONSTRAINT POLYNOMIAL FUNCTIONS--%

S_initial3=[tc_initial,Lift_initial,x_initial,L_initial,R_initial];
S3=[Z(1),Y21(1),X1(2),L,R];
flag3 = [4,1,4,1,1];
bound3 = [.1,.1,.1,.1,.1];
G1(1)=PolyApprox(S_initial3,S3,flag3,bound3);   %--wing stress

S_initial4=[tc_initial,Lift_initial,x_initial,L_initial,R_initial];
S4=[Z(1),Y21(1),X1(2),L,R];
flag4 = [4,1,4,1,1];
bound4 = [.15,.15,.15,.15,.15];
G1(2)=PolyApprox(S_initial4,S4,flag4,bound4);   %--wing stress

S_initial5=[tc_initial,Lift_initial,x_initial,L_initial,R_initial];
S5=[Z(1),Y21(1),X1(2),L,R];
flag5 = [4,1,4,1,1];
```

```
bound5 = [.2,.2,.2,.2,.2];
G1(3)=PolyApprox(S_initial5,S5,flag5,bound5);   %--wing stress

S_initial6=[tc_initial,Lift_initial,x_initial,L_initial,R_initial];
S6=[Z(1),Y21(1),X1(2),L,R];
flag6 = [4,1,4,1,1];
bound6 = [.25,.25,.25,.25,.25];
G1(4)=PolyApprox(S_initial6,S6,flag6,bound6);   %--wing stress

S_initial7=[tc_initial,Lift_initial,x_initial,L_initial,R_initial];
S7=[Z(1),Y21(1),X1(2),L,R];
flag7 = [4,1,4,1,1];
bound7 = [.3,.3,.3,.3,.3];
G1(5)=PolyApprox(S_initial7,S7,flag7,bound7);   %--wing stress

G1(6)=Y1(3);   %--wing twist

%---THIS SECTION COMPUTES CONSTRAINT POLYNOMIAL FUNCTIONS--%


%-------------------------------------------------------------------------------------------
%
%                        Subfunction BB_DRAGPOLAR
%
%  This subfunction calculates the drag and lift-to-drag ratio of the aircraft.  It calls
%  the subfunction POLYAPPROX to compute functions represented by polynomi-
%  als.
%
%     Author          : Jeremy S. Agte     NASA Langley/GWU     Spring '98
%
%  Input Variables    :
%     C               - Vector of constants                    vary
%     Cf_initial      - Initial coefficient of friction        p.f.
%     ESF_initial     - Initial engine scale factor            none
%     tc_initial      - Initial thickness to chord ratio       none
%     Twist_initial   - Initial wing twist                     p.f.
%     X2              - Coefficient of friction                p.f.
%     Y12(1)          - Total aircraft weight                  lb
%     Y12(2)          - Wing twist                             p.f.
%     Y32             - Engine scale factor                    none
%     Z(1)            - Thickness/chord ratio                  none
%     Z(2)            - Altitude                               ft
%     Z(3)            - Mach number                            none
%     Z(4)            - Aspect ratio                           none
%     Z(5)            - Wing sweep                             deg
%     Z(6)            - Wing surface area                      ft^2
%
%  Output Variables   :
%     G2              - Pressure gradient                      p.f.
%     Y2(1)           - Lift                                   lb
%     Y2(2)           - Drag                                   lb
%     Y2(3)           - Lift-to-drag ratio                     none
%     Y21             - Lift                                   lb
%     Y23             - Drag                                   lb
%     Y24             - Lift-to-drag ratio                     none
%
%  Local Variables    :
%     CL              - Coefficient of lift                    none
%     CD              - Coefficient of drag                    none
%     CDmin           - Minimum drag coefficient               none
%     k               - Induced drag factor                    none
%     rho             - Density                                slug/ft^3
%     V               - Velocity                               ft/s
%
%  Subfunctions       :
%        PolyApprox       -Forms polynomial functions for desired variables
%
%-------------------------------------------------------------------------------------------

func-
tion[Y2,Y21,Y23,Y24,G2]=BB_dragpolar(Z,Y12,Y32,X2,ESF_initial,Cf_initial,Twis
t_initial,tc_initial,C)

%-----THIS SECTION COMPUTES THE TOTAL DRAG OF THE A/C-----%

if Z(2)<36089
   V = Z(3)*(1116.39*sqrt(1-(6.875e-06*Z(2))));
   rho = (2.377e-03)*(1-(6.875e-06*Z(2)))^4.2561;
```

```
else
    V = Z(3)*968.1;
    rho = (2.377e-03)*(.2971)*exp(-(Z(2)-36089)/20806.7);
end

CL = Y12(1)/(.5*rho*(V^2)*Z(6));   %--Lift Coefficient

%-----Polynomial function modifying CDmin for ESF and friction coefficient-----%

S_initial1=[ESF_initial,Cf_initial];
S1=[Y32(1),X2(1)];
flag1 = [1,1];
bound1 = [.25,.25];
Fo1 = PolyApprox(S_initial1,S1,flag1,bound1);
CDmin = C(5)*Fo1 + 3.05*(Z(1)^(5/3))*((cos(Z(5)*pi/180))^(3/2));

if Z(3)>= 1
    k=Z(4)*(Z(3)^2-1)*cos(Z(5)*pi/180)/(4*Z(4)*sqrt(Z(5)^2-1)-2);
else
    k=1/(pi*0.8*Z(4));
end

%-----Polynomial function modifying CD for wing twist-----%

S_initial2=[Twist_initial];
S2=[Y12(2)];
flag2=[5];
bound2=[.25];
Fo2=PolyApprox(S_initial2,S2,flag2,bound2);
CD = Fo2*(CDmin + k*(CL^2));

Y2(2) = .5*rho*(V^2)*CD*Z(6);
Y2(3) = CL/CD;
Y2(1)=Y12(1);
Y23(1)=Y2(2);
Y24(1)=Y2(3);
Y21(1)=Y2(1);

%-----THIS SECTION COMPUTES THE TOTAL DRAG OF THE A/C-----%

%-----THIS SECTION COMPUTES CONSTRAINT POLYNOMIALS-----%

S_initial3=[tc_initial];
S3=[Z(1)];
flag3=[1];
bound3=[.25];
G2(1)=PolyApprox(S_initial3,S3,flag3,bound3);   %--adverse pressure gradient

%-----THIS SECTION COMPUTES CONSTRAINT POLYNOMIALS-----%


%------------------------------------------------------------------------------------------
%
%                               Subfunction BB_POWER
%
%   This subfunction calculates fuel consumption and engine weight as well as engine
%   scale factor.  It calls the subfunction POLYAPPROX to compute functions
%   represented by polynomials.
%
%       Author          : Jeremy S. Agte    NASA Langley/GWU    Spring '98
%
%   Input Variables     :
%       C               - Vector of constants                   vary
%       h_initial       - Initial altitude                      ft
%       M_initial       - Initial Mach number                   none
%       T_initial       - Initial throttle setting              none
%       X3              - Throttle setting                      none
%       Y23             - Drag                                  lb
%       Z(1)            - Thickness/chord ratio                 none
%       Z(2)            - Altitude                              ft
%       Z(3)            - Mach number                           none
%       Z(4)            - Aspect ratio                          none
%       Z(5)            - Wing sweep                            deg
%       Z(6)            - Wing surface area                     ft^2
%
%   Output Variables    :
%       G3(1)           - Engine scale factor constraint        none
%       G3(2)           - Engine temperature                    p.f.
```

```
%       G3(3)          - Throttle setting constraint                  none
%       Y3(1)          - Specific fuel consumption                    1/hr
%       Y3(2)          - Engine weight                                lb
%       Y3(3)          - Engine scale factor                          none
%       Y31            - Engine weight                                lb
%       Y32            - Engine scale factor                          none
%       Y34            - Specific fuel consumption                    1/hr
%
%   Local Variables    :
%       Dim_Throttle   - Non-dimensional throttle setting             none
%       p              - Vector of constant coefficients for upper
%                        limit on throttle setting surface fit        none
%       s              - Vector of constant coefficients for SFC
%                        surface fit                                  none
%       Thrust         - Thrust required                              lb
%       Throttle_uA    - Upper limit on throttle setting              none
%
%   Subfunctions       :
%           PolyApprox          -Forms polynomial functions for desired variables
%
%--------------------------------------------------------------------------------------------

function[Y3,Y34,Y31,Y32,G3]=BB_power(Z,Y23,X3,M_initial,h_initial,T_initial,C)

%-----THIS SECTION COMPUTES SFC, ESF, AND ENGINE WEIGHT-----%

Thrust = Y23(1);
Dim_Throttle = X3(1)*16168.6;   %--non-diminsional throttle setting

%-----Surface fit to engine deck (obtained using least squares approx)-----%

s=[1.13238425638512 1.53436586044561 -0.00003295564466 -0.00016378694115 -
0.31623315541888 0.00000410691343 -0.00005248000590 -0.00000000008574
0.00000000190214 0.00000001059951];
Y3(1)=s(1)+s(2)*Z(3)+s(3)*Z(2)+s(4)*Dim_Throttle+s(5)*Z(3)^2+2*Z(2)*Z(3)*s(6)
+2*Dim_Throttle*Z(3)*s(7)+s(8)*Z(2)^2+2*Dim_Throttle*Z(2)*s(9)+s(10)*Dim_Th
rottle^2;

Y3(3) = (Thrust/3)/Dim_Throttle;
Y3(2) = C(4)*(Y3(3)^1.05)*3;
Y31(1) = Y3(2);
Y34(1) = Y3(1);
Y32(1) = Y3(3);

%-----THIS SECTION COMPUTES SFC, ESF, AND ENGINE WEIGHT-----%


%---THIS SECTION COMPUTES POLYNOMIAL CONSTRAINT FUNCTIONS--%

G3(1)=Y3(3);   %--engine scale factor

S_initial1=[M_initial,h_initial,T_initial];
S1=[Z(3),Z(2),X3(1)];
flag1 = [2,4,2];
bound1 = [.25,.25,.25];
G3(2) = PolyApprox(S_initial1,S1,flag1,bound1);   %--engine temperature

p=[11483.7822254806 10856.2163466548 -0.5080237941 3200.157926969 -
0.1466251679 0.0000068572];
Throt-
tle_uA=p(1)+p(2)*Z(3)+p(3)*Z(2)+p(4)*Z(3)^2+2*p(5)*Z(3)*Z(2)+p(6)*Z(2)^2;
G3(3)=Dim_Throttle/Throttle_uA-1;   %--throttle setting

%---THIS SECTION COMPUTES POLYNOMIAL CONSTRAINT FUNCTIONS--%


%--------------------------------------------------------------------------------------------
%
%                        Subfunction BB_RANGE
%
%   This subfunction calculates the system objective function, range, from the Breguet
%   range equation.
%
%       Author          : Jeremy S. Agte     NASA Langley/GWU     Spring '98
%
%   Input Variables     :
%       Y14(1)          - Total aircraft weight                        lb
%       Y14(2)          - Fuel weight                                  lb
```

```
%       Y24            - Lift-to-drag ratio                         none
%       Y34            - Specific fuel consumption                  1/hr
%       Z(1)           - Thickness/chord ratio                      none
%       Z(2)           - Altitude                                   ft
%       Z(3)           - Mach number                                none
%       Z(4)           - Aspect ratio                               none
%       Z(5)           - Wing sweep                                 deg
%       Z(6)           - Wing surface area                          ft^2
%
%  Output Variables    :
%       Y4             - Range                                      NM
%
%  Local Variables     :
%       Theta          - Temperature ratio                          none
%
%-----------------------------------------------------------------------------------------------

function[Y4]=BB_range(Z,Y14,Y24,Y34)

%-----THIS SECTION COMPUTES THE A/C RANGE (Breguet)-----%

if Z(2)<36089
   theta=1-0.000006875*Z(2);
else
   theta=.7519;
end

Y4(1) = ((Z(3)*Y24(1))*661*sqrt(theta)/Y34(1))*log(Y14(1)/(Y14(1)-Y14(2)));

%-----THIS SECTION COMPUTES THE A/C RANGE-----%


%-----------------------------------------------------------------------------------------------
%
%                         Subfunction POLYAPPROX
%
%  This subfunction calculates polynomial coefficients to characterize the behavior
%  of certain synthetic variables and function modifiers.  Move limits for each
%  polynomial are selected based on knowledge of each variable or modifier's
%  qualitative response to changes in other variables.  Possible relationships are
%  positive linear (flag = 1), negative linear (flag = 3), positive nonlinear (flag = 2),
%  negative nonlinear (flag = 4), and parabolic (flag = 5).
%
%     Author       : Jeremy S. Agte     NASA Langley/GWU     Spring '98
%
%  Input Variables     :
%     flag             - Indicates functional relationship btwn var.      none
%     S                - Vector of initial values of independent
%                        variables                                        vary
%     S_bound          - Vector of bounds used to control slope of
%                        the polynomial function (narrow = high slope)   none
%     S_new            - Vector of current values of independent
%                        variables                                        vary
%
%  Output Variables    :
%     Ai               - Vector of coefficients (2nd term)                none
%     Aij              - Matrix of coefficients (3rd term)                none
%     Ao               - Scalar coefficient (1st term)                    none
%     FF               - Value of synthetic variable or modifier          none
%
%  Local Variables     :
%     A                - Solution matrix for polynomial fitting eqns.     none
%     a                - Lower y-axis bound on polynomial                 none
%     b                - Upper y-axis bound on polynomial                 none
%     F_bound          - Bounds for dependent variable; RHS of
%                        polynomial fitting eqns.                         none
%     Mtx_shifted      - Matrix for LHS of polynomial fitting eqns.       none
%     R                - Vector of random constants used to fill off-
%                        diagonals of Aij                                 none
%     Sl               - Standard independent variable lower bound        none
%     S_norm           - Vector of current values of independent
%                        variables normalized by initial values           none
%     So               - Standard independent variable midpoint           none
%     S_shifted        - Vector of normalized values of independent
%                        variables shifted to an area near the origin     none
%     Su               - Standard independent variable upper bound        none
%
%-----------------------------------------------------------------------------------------------
```

```
function [FF, Ao, Ai, Aij] = PolyApprox(S, S_new, flag, S_bound)
  for i = 1:size(S,2)
    S_norm(i) = S_new(i)/S(i);  %normalize new S with initial S
    if S_norm(i)>1.25
      S_norm(i)=1.25;
    elseif S_norm(i)<0.75
      S_norm(i)=0.75;
    end
    S_shifted(i) = S_norm(i) - 1;  %shift S vector near origin

    %DETERMINE BOUNDS ON FF DEPENDING ON SLOPE-SHAPE
    a=0.1;
    b=a;
    if flag(i)==5

      %CALCULATE POLYNOMIAL COEFFICIENTS (S-ABOUT ORIGIN)
      So=0;
      Sl=So-S_bound(i);
      Su=So+S_bound(i);
      Mtx_shifted = [1 Sl Sl^2; 1 So So^2; 1 Su Su^2];
      F_bound = [1+(.5*a)^2; 1; 1+(.5*b)^2];
      A = Mtx_shifted\F_bound;
      Ao = A(1);
      Ai(i) = A(2);
      Aij(i,i) = A(3);
      %CALCULATE POLYNOMIAL COEFFICIENTS

    else
      switch (flag(i))
        case 0
          S_shifted(i) = 0;
        case 3
          a=-a;
          b=a;
        case 2
          b=2*a;
        case 4
          a=-a;
          b=2*a;
      end
      %DETERMINE BOUNDS ON FF DEPENDING ON SLOPE-SHAPE

      %CALCULATE POLYNOMIAL COEFFICIENTS (S-ABOUT ORIGIN)
      So=0;
      Sl=So-S_bound(i);
      Su=So+S_bound(i);
      Mtx_shifted = [1 Sl Sl^2; 1 So So^2; 1 Su Su^2];
      F_bound = [1-.5*a; 1; 1+.5*b];
      A = Mtx_shifted\F_bound;
      Ao = A(1);
      Ai(i) = A(2);
      Aij(i,i) = A(3);
      %CALCULATE POLYNOMIAL COEFFICIENTS
    end
  end

  %FILL UP OFF DIAGONALS OF Aij

  R = [...
  0.2736   0.3970   0.8152   0.9230   0.1108
  0.4252   0.4415   0.6357   0.7435   0.1138
  0.0329   0.8856   0.8390   0.3657   0.0019
  0.0878   0.7248   0.1978   0.0200   0.0169
  0.8955   0.4568   0.8075   0.9239   0.2525];

  for i = 1:size(S,2)
    for j = (i+1):size(S,2)
      Aij(i,j) = Aij(i,i)*R(i,j);
      Aij(j,i) = Aij(i,j);
    end
  end

  %CALCULATE POLYNOMIAL
  FF = Ao + Ai*(S_shifted') + (1/2)*(S_shifted)*(Aij)*(S_shifted');
```

```
%-----------------------------------------------------------------------------------------------
%
%                     Subfunction BB1OPT
%
% This subfunction serves as a shell for the Matlab 'constr' optimization routine
% performing a local optimization on the WEIGHT module.
%
%     Author          : Jeremy S. Agte    NASA Langley/GWU    Spring '98
%
% Input Variables     :
%     C               - Vector of constants                              vary
%     DY_lamda        - Vector of total derivatives, behavior            vary
%     DY_x            - Vector of total derivatives, behavior
%                       variables w.r.t wingbox x-section                vary
%     i0              - Design variable initial values                   vary
%     L_initial       - Initial halfspan length                          ft
%     Lift_initial    - Initial lift                                     lb
%     P_var           - Vector of current design variable values         vary
%     R_initial       - Initial location of lift as fraction of halfspan none
%     tc_initial      - Initial thickness to chord ratio                 none
%     Twist_initial   - Initial wing twist                               p.f.
%     vlb_nd          - Non-dimensional lower bounds on design
%                       variables                                        none
%     vub_nd          - Non-dimensional upper bounds on design
%                       variables                                        none
%     x_initial       - Initial wingbox x-sectional thickness            p.f.
%     X1(1)           - Wing taper ratio                                 none
%     X1(2)           - Wingbox x-sectional area as poly. funct.         p.f.
%     Y21             - Lift                                             lb
%     Y31             - Engine weight                                    lb
%     Z(1)            - Thickness/chord ratio                            none
%     Z(2)            - Altitude                                         ft
%     Z(3)            - Mach number                                      none
%     Z(4)            - Aspect ratio                                     none
%     Z(5)            - Wing sweep                                       deg
%     Z(6)            - Wing surface area                                ft^2
%
% Output Variables    :
%     dX1             - Vector of optimal changes in X1 variables        vary
%     fstore          - Value of objective function for BB1 optim.       NM
%     gstore0         - Vector of local constraint values at beginning
%                       of BB1 optimization                              p.f.
%     Lagrange1       - Vector of Lagrange multipliers from BB1 at
%                       optimum                                          NM
%
% Local Variables     :
%     options         - see Matlab 'constr' function
%     vlb             - Non-dimensional lower bounds on BB1 design
%                       variables                                        none
%     vub             - Non-dimensional upper bounds on BB1 design
%                       variables                                        none
%     x0              - Vector of non-dimensional starting points
%                       for BB1 optimization                             none
%
% Subfunctions        :
%         BB1WRAPPER      - Contains objective function and constraints
%                           for BB1 optimization
%         constr          - Matlab optimization routine
%
%-----------------------------------------------------------------------------------------------

func-
tion[dX1,Lagrange1,fstore,gstore0]=BB1OPT(vlb_nd,vub_nd,i0,P_var,Z,Y21,Y31,X1
,x_initial,L_initial,R_initial,Lift_initial,Twist_initial,tc_initial,C,DY_lamda,DY_x)

vlb=[vlb_nd(1) vlb_nd(2)];
vub=[vub_nd(1) vub_nd(2)];
x0=[X1(1)/i0(1)-1,X1(2)/i0(2)-1];
options(1)=1;
options(2)=.0001;
options(3)=.0001;
options(4)=.001;
options(14)=1000;
options(17)=.01;

[fstore0,gstore0,dX10]=BB1WRAPPER(x0,i0,P_var,Z,Y21,Y31,x_initial,L_initial,R_
initial,Lift_initial,Twist_initial,tc_initial,C,DY_lamda,DY_x);
```

```
[x,options,Lagrange1]=constr('BB1WRAPPER',x0,options,vlb,vub,[],i0,P_var,Z,Y21,
Y31,x_initial,L_initial,R_initial,Lift_initial,Twist_initial,tc_initial,C,DY_lamda,DY_
x);


%-----------------------------------------------------------------------------------------
%
%                         Subfunction BB1WRAPPER
%
% This subfunction computes the objective function and the constraints for the
% local optimization on the WEIGHT module.
%
%    Author          : Jeremy S. Agte     NASA Langley/GWU     Spring '98
%
% Input Variables    :
%    C               - Vector of constants                              vary
%    DY_lamda        - Vector of total derivatives, behavior            vary
%    DY_x            - Vector of total derivatives, behavior
%                       variables w.r.t wingbox x-section               vary
%    i0              - Design variable initial values                   vary
%    L_initial       - Initial halfspan length                          ft
%    Lift_initial    - Initial lift                                     lb
%    P_var           - Vector of current design variable values         vary
%    R_initial       - Initial location of lift as fraction of halfspan none
%    tc_initial      - Initial thickness to chord ratio                 none
%    Twist_initial   - Initial wing twist                               p.f.
%    x               - Vector of non-dimensional design variables
%                       for BB1 optimization                            none
%    x_initial       - Initial wingbox x-sectional thickness            p.f.
%    Y21             - Lift                                             lb
%    Y31             - Engine weight                                    lb
%    Z(1)            - Thickness/chord ratio                            none
%    Z(2)            - Altitude                                         ft
%    Z(3)            - Mach number                                      none
%    Z(4)            - Aspect ratio                                     none
%    Z(5)            - Wing sweep                                       deg
%    Z(6)            - Wing surface area                                ft^2
%
% Output Variables   :
%    dX1             - Vector of optimal changes in X1 variables        vary
%    f               - Value of objective function for BB1 optim.       NM
%    g               - Vector of local constraint values                p.f.
%
% Local Variables    :
%    Sigma_uA        - Upper allowable limit for stress constraints     none
%    Twist_lA        - Lower allowable limit for twist constraint       none
%    Twist_uA        - Upper allowable limit for twist constraint       none
%
% Subfunctions       :
%       BB_weight          -Calculates a/c structural weights
%
%-----------------------------------------------------------------------------------------

func-
tion[f,g,dX1]=BB1WRAPPER(x,i0,P_var,Z,Y21,Y31,x_initial,L_initial,R_initial,Lift
_initial,Twist_initial,tc_initial,C,DY_lamda,DY_x)

X1=[i0(1)*(1+x(1)),i0(2)*(1+x(2))];

[Y1,Y12,Y14,G1]=BB_weight(Z,Y21,Y31,X1,x_initial,L_initial,R_initial,Lift_initial
,Twist_initial,tc_initial,C);

Sigma_uA=1.05;
Twist_uA=1.03;
Twist_lA=.97;

g(1)=G1(1)/Sigma_uA-1;
g(2)=G1(2)/Sigma_uA-1;
g(3)=G1(3)/Sigma_uA-1;
g(3)=G1(4)/Sigma_uA-1;
g(5)=G1(5)/Sigma_uA-1;
g(6)=G1(6)/Twist_uA-1;
g(7)=Twist_lA/G1(6)-1;

dX1=[X1(1)-P_var(1) X1(2)-P_var(2)];
f=-([DY_lamda(10),DY_x(10)]*dX1');
```

```
%-----------------------------------------------------------------------------------------
%
%                         Subfunction BB2OPT
%
% This subfunction serves as a shell for the Matlab 'constr' optimization routine
% performing a local optimization on the DRAGPOLAR module.
%
%     Author          : Jeremy S. Agte    NASA Langley/GWU    Spring '98
%
% Input Variables     :
%     C               - Vector of constants                              vary
%     Cf_initial      - Initial coefficient of friction                  p.f.
%     DY_Cf           - Vector of total derivatives, behavior
%                         variables w.r.t skin friction coefficient      vary
%     ESF_initial     - Initial engine scale factor                      none
%     i0              - Design variable initial values                   vary
%     P_var           - Vector of current design variable values         vary
%     tc_initial      - Initial thickness to chord ratio                 none
%     Twist_initial   - Initial wing twist                               p.f.
%     vlb_nd          - Non-dimensional lower bounds on design
%                         variables                                      none
%     vub_nd          - Non-dimensional upper bounds on design
%                         variables                                      none
%     X2              - Coefficient of friction                          p.f.
%     Y12(1)          - Total aircraft weight                            lb
%     Y12(2)          - Wing twist                                       p.f.
%     Y32             - Engine scale factor                              none
%     Z(1)            - Thickness/chord ratio                            none
%     Z(2)            - Altitude                                         ft
%     Z(3)            - Mach number                                      none
%     Z(4)            - Aspect ratio                                     none
%     Z(5)            - Wing sweep                                       deg
%     Z(6)            - Wing surface area                                ft^2
%
% Output Variables    :
%     dX2             - Vector of optimal changes in X2 variables        vary
%     fstore          - Value of objective function for BB2 optim.       NM
%     Lagrange2       - Vector of Lagrange multipliers from BB2 at
%                         optimum                                        NM
%
% Local Variables     :
%     options         - see Matlab 'constr' function
%     vlb             - Non-dimensional lower bounds on BB2 design
%                         variables                                      none
%     vub             - Non-dimensional upper bounds on BB2 design
%                         variables                                      none
%     x0              - Vector of non-dimensional starting points
%                         for BB2 optimization                           none
%
% Subfunctions        :
%         BB2WRAPPER      - Contains objective function and constraints
%                             for BB2 optimization
%         constr          - Matlab optimization routine
%
%-----------------------------------------------------------------------------------------

func-
tion[dX2,Lagrange2,fstore]=BB2OPT(vlb_nd,vub_nd,i0,P_var,Z,Y12,Y32,X2,ESF_in
itial,Cf_initial,Twist_initial,tc_initial,C,DY_Cf)

vlb=[vlb_nd(3)];
vub=[vub_nd(3)];
x0=[X2(1)/i0(3)-1];
options(1)=1;
options(2)=.0001;
options(3)=.0001;
options(4)=.001;
options(14)=1000;
options(17)=.01;

[x,options,Lagrange2]=constr('BB2WRAPPER',x0,options,vlb,vub,[],i0,P_var,Z,Y12,
Y32,ESF_initial,Cf_initial,Twist_initial,tc_initial,C,DY_Cf);

[fstore,gstore,dX2]=BB2WRAPPER(x,i0,P_var,Z,Y12,Y32,ESF_initial,Cf_initial,Twi
st_initial,tc_initial,C,DY_Cf);
```

```
%-------------------------------------------------------------------------------------------------
%
%                         Subfunction BB2WRAPPER
%
% This subfunction computes the objective function and the constraints for the
% local optimization on the DRAGPOLAR module.
%
%     Author          : Jeremy S. Agte    NASA Langley/GWU    Spring '98
%
% Input Variables     :
%     C               - Vector of constants                              vary
%     Cf_initial      - Initial coefficient of friction                  p.f.
%     DY_Cf           - Vector of total derivatives, behavior
%                       variables w.r.t skin friction coefficient        vary
%     ESF_initial     - Initial engine scale factor                      none
%     i0              - Design variable initial values                   vary
%     P_var           - Vector of current design variable values         vary
%     tc_initial      - Initial thickness to chord ratio                 none
%     Twist_initial   - Initial wing twist                               p.f.
%     x               - Vector of non-dimensional design variables
%                       for BB2 optimization                             none
%     Y12(1)          - Total aircraft weight                            lb
%     Y12(2)          - Wing twist                                       p.f.
%     Y32             - Engine scale factor                              none
%     Z(1)            - Thickness/chord ratio                            none
%     Z(2)            - Altitude                                         ft
%     Z(3)            - Mach number                                      none
%     Z(4)            - Aspect ratio                                     none
%     Z(5)            - Wing sweep                                       deg
%     Z(6)            - Wing surface area                                ft^2
%
% Output Variables    :
%     dX2             - Vector of optimal changes in X2 variables        vary
%     f               - Value of objective function for BB2 optim.       NM
%     g               - Vector of local constraint values                p.f.
%
% Local Variables     :
%     Pg_uA           - Upper allowable limit on pressure gradient
%                       constraint                                       none
%
% Subfunctions        :
%          BB_dragpolar        -Calculates aerodynamic values
%
%-------------------------------------------------------------------------------------------------

func-
tion[f,g,dX2]=BB2WRAPPER(x,i0,P_var,Z,Y12,Y32,ESF_initial,Cf_initial,Twist_ini
tial,tc_initial,C,DY_Cf)

X2=[i0(3)*(1+x(1))];

[Y2,Y21,Y23,Y24,G2]=BB_dragpolar(Z,Y12,Y32,X2,ESF_initial,Cf_initial,Twist_in
itial,tc_initial,C);

Pg_uA=1.04;
g(1)=G2(1)/Pg_uA-1;


dX2=[X2(1)-P_var(3)];
f=-([DY_Cf(10)]*dX2);


%-------------------------------------------------------------------------------------------------
%
%                         Subfunction BB3OPT
%
% This subfunction serves as a shell for the Matlab 'constr' optimization routine
% performing a local optimization on the POWER module.
%
%     Author          : Jeremy S. Agte    NASA Langley/GWU    Spring '98
%
% Input Variables     :
%     C               - Vector of constants                              vary
%     DY_T            - Vector of total derivatives, behavior
%                       variables w.r.t throttle setting                 vary
%     h_initial       - Initial altitude                                 ft
%     i0              - Design variable initial values                   vary
%     M_initial       - Initial Mach number                              none
```

```
%       P_var          - Vector of current design variable values     vary
%       T_initial      - Initial throttle setting                     none
%       vlb_nd         - Non-dimensional lower bounds on design
%                        variables                                    none
%       vub_nd         - Non-dimensional upper bounds on design
%                        variables                                    none
%       X3             - Throttle setting                             none
%       Y23            - Drag                                         lb
%       Z(1)           - Thickness/chord ratio                        none
%       Z(2)           - Altitude                                     ft
%       Z(3)           - Mach number                                  none
%       Z(4)           - Aspect ratio                                 none
%       Z(5)           - Wing sweep                                   deg
%       Z(6)           - Wing surface area                            ft^2
%
%  Output Variables    :
%       dX3            - Vector of optimal changes in X3 variables    vary
%       fstore         - Value of objective function for BB3 optim.   NM
%       gstore0        - Vector of local constraint values at beginning
%                        of BB3 optimization                          p.f.
%       Lagrange3      - Vector of Lagrange multipliers from BB3 at
%                        optimum                                      NM
%
%  Local Variables     :
%       options        - see Matlab 'constr' function
%       vlb            - Non-dimensional lower bounds on BB3 design
%                        variables                                    none
%       vub            - Non-dimensional upper bounds on BB3 design
%                        variables                                    none
%       x0             - Vector of non-dimensional starting points
%                        for BB3 optimization                         none
%
%  Subfunctions        :
%           BB3WRAPPER      - Contains objective function and constraints
%                             for BB3 optimization
%           constr          - Matlab optimization routine
%
%----------------------------------------------------------------------------------------
```

```
fun-
tion[dX3,Lagrange3,fstore,gstore0]=BB3OPT(vlb_nd,vub_nd,i0,P_var,Z,Y23,X3,M_i
nitial,h_initial,T_initial,C,DY_T)

vlb=[vlb_nd(4)];
vub=[vub_nd(4)];
x0=[X3(1)/i0(4)-1];
options(1)=1;
options(2)=.0001;
options(3)=.0001;
options(4)=.001;
options(14)=1000;
options(17)=.01;

[fstore0,gstore0,dX30]=BB3WRAPPER(x0,i0,P_var,Z,Y23,M_initial,h_initial,T_initi
al,C,DY_T);


[x,options,Lagrange3]=constr('BB3WRAPPER',x0,options,vlb,vub,[],i0,P_var,Z,Y23,
M_initial,h_initial,T_initial,C,DY_T);

[fstore,gstore,dX3]=BB3WRAPPER(x,i0,P_var,Z,Y23,M_initial,h_initial,T_initial,C,
DY_T);


%----------------------------------------------------------------------------------------
%
%                           Subfunction BB3WRAPPER
%
%  This subfunction computes the objective function and the constraints for the
%  local optimization on the POWER module.
%
%       Author       : Jeremy S. Agte     NASA Langley/GWU     Spring '98
%
%  Input Variables   :
%       C            - Vector of constants                          vary
%       DY_T         - Vector of total derivatives, behavior
%                      variables w.r.t throttle setting              vary
%       h_initial    - Initial altitude                              ft
```

```
%      i0             - Design variable initial values                vary
%      M_initial      - Initial Mach number                           none
%      P_var          - Vector of current design variable values      vary
%      T_initial      - Initial throttle setting                      none
%      x              - Vector of non-dimensional design variables
%                       for BB3 optimization                          none
%      Y23            - Drag                                          lb
%      Z(1)           - Thickness/chord ratio                         none
%      Z(2)           - Altitude                                      ft
%      Z(3)           - Mach number                                   none
%      Z(4)           - Aspect ratio                                  none
%      Z(5)           - Wing sweep                                    deg
%      Z(6)           - Wing surface area                             ft^2
%
%  Output Variables   :
%      dX3            - Vector of optimal changes in X3 variables     vary
%      f              - Value of objective function for BB3 optim.    NM
%      g              - Vector of local constraint values             p.f.
%
%  Local Variables    :
%      ESF_lA         - Lower allowable limit for ESF constraint      none
%      ESF_uA         - Upper allowable limit for ESF constraint      none
%      Temp_uA        - Upper allowable limit for temp. constraint    none
%
%  Subfunctions       :
%          BB_power          -Calculates propulsion values
%
%-----------------------------------------------------------------------------------------

func-
tion[f,g,dX3]=BB3WRAPPER(x,i0,P_var,Z,Y23,M_initial,h_initial,T_initial,C,DY_
T)

X3=[i0(4)*(1+x(1))];

[Y3,Y34,Y31,Y32,G3]=BB_power(Z,Y23,X3,M_initial,h_initial,T_initial,C);

ESF_uA=1.5;
ESF_lA=.5;
Temp_uA=1.02;

g(1)=G3(1)/ESF_uA-1;
g(2)=ESF_lA/G3(1)-1;
g(3)=G3(2)/Temp_uA-1;
g(4)=G3(3);
dX3=[X3(1)-P_var(4)];
f=-([DY_T(10)]*dX3);


%-----------------------------------------------------------------------------------------
%
%                        Subfunction SYSOPT
%
%  This subfunction serves as a shell for the Matlab 'constr' optimization routine
%  performing a system optimization.
%
%      Author         : Jeremy S. Agte     NASA Langley/GWU     Spring '98
%
%  Input Variables    :
%      dg2_Z          - Derivative of BB2 constraint wrt Z            p.f.
%      G2             - Pressure gradient                             p.f.
%      GRADphi4_Z     - Vector of total derivatives at the optimal
%                       state, range w.r.t Z variables                vary
%      i0             - Design variable initial values                vary
%      P_var          - Vector of current design variable values      vary
%      vlb_nd         - Non-dimensional lower bounds on design
%                       variables                                     none
%      vub_nd         - Non-dimensional upper bounds on design
%                       variables                                     none
%      Y4             - Range                                         NM
%      Z(1)           - Thickness/chord ratio                         none
%      Z(2)           - Altitude                                      ft
%      Z(3)           - Mach number                                   none
%      Z(4)           - Aspect ratio                                  none
%      Z(5)           - Wing sweep                                    deg
%      Z(6)           - Wing surface area                             ft^2
%
%  Output Variables   :
```

```
%       dZ               - Vector of optimal changes in Z variables      vary
%       fstore           - Value of objective function for system optim. NM
%       gstore0          - Vector of constraint values at beginning
%                          of system optimization                        vary
%
%  Local Variables       :
%       options          - see Matlab 'constr' function
%       vlb              - Non-dimensional lower bounds on Z
%                          variables                                     none
%       vub              - Non-dimensional upper bounds on Z
%                          variables                                     none
%       x0               - Vector of non-dimensional starting points
%                          for system optimization                       none
%
%  Subfunctions          :
%           SysWRAPPER        - Contains objective function and constraints
%                               for system optimization
%           constr            - Matlab optimization routine
%
%-----------------------------------------------------------------------------------------

func-
tion{dZ,fstore,gstore0]=SysOPT(vlb_nd,vub_nd,i0,P_var,Y4,GRADphi4_Z,Z,dg2_Z,
G2)

vlb=[vlb_nd(5:10)];
vub=[vub_nd(5:10)];
x0=[Z(1)/i0(5)-1,Z(2)/i0(6)-1,Z(3)/i0(7)-1,Z(4)/i0(8)-1,Z(5)/i0(9)-1,Z(6)/i0(10)-1];
options(1)=1;
options(2)=.0001;
options(3)=.0001;
options(4)=.001;
options(14)=1000;
options(17)=.01;

[fstore0,gstore0,dZ0]=SysWRAPPER(x0,i0,P_var,Y4,GRADphi4_Z,dg2_Z,G2);

[x]=constr('SysWRAPPER',x0,options,vlb,vub,[],i0,P_var,Y4,GRADphi4_Z,dg2_Z,G
2);
```

```
[fstore,gstore,dZ]=SysWRAPPER(x,i0,P_var,Y4,GRADphi4_Z,dg2_Z,G2);


%-----------------------------------------------------------------------------------------
%
%                        Subfunction SYSWRAPPER
%
%  This subfunction computes the objective function and constraints for the
%  system optimization.
%
%     Author         : Jeremy S. Agte     NASA Langley/GWU     Spring '98
%
%  Input Variables     :
%     dg2_Z            - Derivative of BB2 constraint wrt Z              p.f.
%     G2               - Pressure gradient                               p.f.
%     GRADphi4_Z       - Vector of total derivatives at the optimal
%                        state, range w.r.t Z variables                  vary
%     i0               - Design variable initial values                  vary
%     P_var            - Vector of current design variable values        vary
%     x                - Vector of non-dimensional design variables
%                        for system optimization                         none
%     Y4               - Range                                           NM
%     Z(1)             - Thickness/chord ratio                           none
%     Z(2)             - Altitude                                        ft
%     Z(3)             - Mach number                                     none
%     Z(4)             - Aspect ratio                                    none
%     Z(5)             - Wing sweep                                      deg
%     Z(6)             - Wing surface area                               ft^2
%
%  Output Variables    :
%     dZ               - Vector of optimal changes in Z variables        vary
%     f                - Value of objective function for system optim.   NM
%     g                - Vector of constraint values                     vary
%
%  Local Variables     :
%     a                - Used to construct move limits                   none
%     Pg_uA            - Upper allowable limit on pressure gradient
%                        constraint                                      none
```

```
%
%-----------------------------------------------------------------------------------------

function[f,g,dZ]=SysWRAPPER(x,i0,P_var,Y4,GRADphi4_Z,dg2_Z,G2)

Z = [i0(5)*(1+x(1)),i0(6)*(1+x(2)),i0(7)*(1+x(3)),i0(8)*(1+x(4)),
i0(9)*(1+x(5)),i0(10)*(1+x(6))];

dZ = [Z(1)-P_var(5) Z(2)-P_var(6) Z(3)-P_var(7) Z(4)-P_var(8) Z(5)-P_var(9) Z(6)-
P_var(10)]';

a=.2;
Pg_uA=1.04;
G2(1)=G2(1)/Pg_uA-1;

g(1)=G2(1) + dg2_Z(1,1)*dZ(1);
g(2)=abs(dZ(1))/(a*i0(5))-1;
g(3)=abs(dZ(2))/(a*i0(6))-1;
g(4)=abs(dZ(3))/(a*i0(7))-1;
g(5)=abs(dZ(4))/(a*i0(8))-1;
g(6)=abs(dZ(5))/(a*i0(9))-1;
g(7)=abs(dZ(6))/(a*i0(10))-1;


f = -(Y4(1) + GRADphi4_Z*dZ);


%-----------------------------------------------------------------------------------------
%
%                          Subfunction INBOUNDS
%
%  This subfunction calculates the non-dimensional upper and lower bounds of the
%  design variables.
%
%       Author         : Jeremy S. Agte    NASA Langley/GWU    Spring '98
%
%  Input Variables     :
%       vlb            - Non-dimensional lower bounds on BB1 design
%                        variables                                    none
%       vub            - Non-dimensional upper bounds on BB1 design
%                        variables                                    none
%       x0             - Vector of non-dimensional starting points
%                        for BB1 optimization                         none
%
%  Output Variables    :
%       vlb_nd         - Non-dimensional lower bounds on design
%                        variables                                    none
%       vub_nd         - Non-dimensional upper bounds on design
%                        variables                                    none
%
%-----------------------------------------------------------------------------------------

function[vlb_nd,vub_nd]=INbounds(x0,vlb,vub)

vlb_nd=[vlb(1)/x0(1)-1,vlb(2)/x0(2)-1,vlb(3)/x0(3)-1,vlb(4)/x0(4)-1,vlb(5)/x0(5)-
1,vlb(6)/x0(6)-1,vlb(7)/x0(7)-1,vlb(8)/x0(8)-1,vlb(9)/x0(9)-1,vlb(10)/x0(10)-1];
vub_nd=[vub(1)/x0(1)-1,vub(2)/x0(2)-1,vub(3)/x0(3)-1,vub(4)/x0(4)-1,vub(5)/x0(5)-
1,vub(6)/x0(6)-1,vub(7)/x0(7)-1,vub(8)/x0(8)-1,vub(9)/x0(9)-1,vub(10)/x0(10)-1];


%-----------------------------------------------------------------------------------------
%
%                          Subfunction FIN_DIFF
%
%  This subfunction calls several subfunctions that use one-step forward finite
%  differencing to calculate the derivatives required by the BLISS method.
%
%       Author         : Jeremy S. Agte    NASA Langley/GWU    Spring '98
%
%  Input:
%       C              - Vector of constants                          vary
%       Cf_initial     - Initial coefficient of friction              p.f.
%       ESF_initial    - Initial engine scale factor                  none
%       G1(1)          - Stress on wing                               p.f.
%       G1(2)          - Stress on wing                               p.f.
%       G1(3)          - Stress on wing                               p.f.
%       G1(4)          - Stress on wing                               p.f.
%       G1(5)          - Stress on wing                               p.f.
```

```
%      G1(6)          - Wing twist as constraint                          p.f.
%      G2             - Pressure gradient                                 p.f.
%      G3(1)          - Engine scale factor constraint                    none
%      G3(2)          - Engine temperature                                p.f.
%      G3(3)          - Throttle setting constraint                       none
%      h_initial      - Initial altitude                                  ft
%      L_initial      - Initial halfspan length                           ft
%      Lift_initial   - Initial lift                                      lb
%      M_initial      - Initial Mach number                               none
%      R_initial      - Initial location of lift as fraction of halfspan  none
%      tc_initial     - Initial thickness to chord ratio                  none
%      T_initial      - Initial throttle setting                          none
%      Twist_initial  - Initial wing twist                                p.f.
%      x_initial      - Initial wingbox x-sectional thickness             p.f.
%      Y1(1)          - Total aircraft weight                             lb
%      Y1(2)          - Fuel weight                                       lb
%      Y1(3)          - Wing twist                                        p.f.
%      Y12(1)         - Total aircraft weight                             lb
%      Y12(2)         - Wing twist                                        p.f.
%      Y14(1)         - Total aircraft weight                             lb
%      Y14(2)         - Fuel weight                                       lb
%      Y2(1)          - Lift                                              lb
%      Y2(2)          - Drag                                              lb
%      Y2(3)          - Lift-to-drag ratio                                none
%      Y21            - Lift                                              lb
%      Y23            - Drag                                              lb
%      Y24            - Lift-to-drag ratio                                none
%      Y3(1)          - Specific fuel consumption                         1/hr
%      Y3(2)          - Engine weight                                     lb
%      Y3(3)          - Engine scale factor                               none
%      Y31            - Engine weight                                     lb
%      Y32            - Engine scale factor                               none
%      Y34            - Specific fuel consumption                         1/hr
%      Y4             - Objective function output from BB4                NM
%      X1(1)          - Wing taper ratio                                  none
%      X1(2)          - Wingbox x-sectional area as poly. funct.          p.f.
%      X2             - Skin friction coefficient as poly. funct.         p.f.
%      X3             - Throttle setting                                  none
%      Z(1)           - Thickness/chord ratio                             none
%      Z(2)           - Altitude                                          ft
%      Z(3)           - Mach number                                       none
%      Z(4)           - Aspect ratio                                      none
%      Z(5)           - Wing sweep                                        deg
%      Z(6)           - Wing surface area                                 ft^2
%
%  Output:
%      A              - Coefficient matrix in GSE                         none
%      dg1_Z          - Vector of  derivatives, BB1 constaints
%                         w.r.t Z variables                               vary
%      dg2_Z          - Vector of  derivatives, BB2 constaints
%                         w.r.t Z variables                               vary
%      dg3_Z          - Vector of  derivatives, BB3 constaints
%                         w.r.t Z variables                               vary
%      dg1_YE1        - Vector of  derivatives, BB1 constaints
%                         w.r.t Y variables entering BB1                  vary
%      dg2_YE2        - Vector of  derivatives, BB2 constaints
%                         w.r.t Y variables entering BB2                  vary
%      dg3_YE3        - Vector of  derivatives, BB3 constaints
%                         w.r.t Y variables entering BB3                  vary
%      dY_AR          - Vector of partial derivatives, behavior
%                       variables w.r.t aspect ratio                      vary
%      dY_Cf          - Vector of partial derivatives, behavior
%                       variables w.r.t skin friction coefficient         vary
%      dY_h           - Vector of partial derivatives, behavior
%                       variables w.r.t altitude                          vary
%      dY_Lamda       - Vector of partial derivatives, behavior
%                       variables w.r.t wing sweep                        vary
%      dY_lamda       - Vector of partial derivatives, behavior
%                       variables w.r.t taper ratio                       vary
%      dY_M           - Vector of partial derivatives, behavior
%                       variables w.r.t Mach number                       vary
%      dY_Sref        - Vector of partial derivatives, behavior
%                       variables w.r.t wing surface area                 vary
%      dY_T           - Vector of partial derivatives, behavior
%                       variables w.r.t throttle setting                  vary
%      dY_tc          - Vector of partial derivatives, behavior
%                       variables w.r.t thickness/chord ratio             vary
%      dY_x           - Vector of partial derivatives, behavior
```

```
%                         variables w.r.t wingbox x-section            vary
%
%  Subfunctions           :
%         fin_diff_A12          -Calculates the A12 submatrix of GSE eqns.
%         fin_diff_A13          -Calculates the A13 submatrix of GSE eqns.
%         fin_diff_A21          -Calculates the A21 submatrix of GSE eqns.
%         fin_diff_A23          -Calculates the A23 submatrix of GSE eqns.
%         fin_diff_A32          -Calculates the A32 submatrix of GSE eqns.
%         fin_diff_A41          -Calculates the A41 submatrix of GSE eqns.
%         fin_diff_A42          -Calculates the A42 submatrix of GSE eqns.
%         fin_diff_A43          -Calculates the A43 submatrix of GSE eqns.
%         fdG1_Y21              -Calculates BB1 constraints w.r.t Y variables
%                                coming into BB1 from BB2; derivative
%         fdG1_Y31              -Calculates BB1 constraints w.r.t Y variables
%                                coming into BB1 from BB3; derivative
%         fdG2_Y12              -Calculates BB2 constraints w.r.t Y variables
%                                coming into BB2 from BB1; derivative
%         fdG2_Y32              -Calculates BB2 constraints w.r.t Y variables
%                                coming into BB2 from BB3; derivative
%         fdG3_Y23              -Calculates BB3 constraints w.r.t Y variables
%                                coming into BB3 from BB2
%         fdY1_X1               -Calculates Y1 output w.r.t. change in X variables
%                                for BB1
%         fdY2_X2               -Calculates Y2 output w.r.t. change in X variables
%                                for BB2
%         fdY3_X3               -Calculates Y3 output w.r.t. change in X variables
%                                for BB3
%         fdY1_Z                -Calculates Y1 output w.r.t. change in Z variables
%         fdY2_Z                -Calculates Y2 output w.r.t. change in Z variables
%         fdY3_Z                -Calculates Y3 output w.r.t. change in Z variables
%         fdY4_Z                -Calculates Y4 output w.r.t. change in Z variables
%         fdG1_Z                -Calculates BB1 contraint output w.r.t. change in
%                                Z variables
%         fdG2_Z                -Calculates BB2 contraint output w.r.t. change in
%                                Z variables
%         fdG3_Z                -Calculates BB3 contraint output w.r.t. change in
%                                Z variables
%
%-----------------------------------------------------------------------------------------

function[A,dY_lambda,dY_x,dY_Cf,dY_T,dY_tc,dY_h,dY_M,dY_AR,dY_Lambda,dY_Sref,dg1_Z,dg2_Z,dg3_Z,dg1_YE1,dg2_YE2,dg3_YE3]=FIN_DIFF(Z,Y1,Y2,Y3,Y4,Y12,Y14,Y21,Y23,Y24,Y31,Y32,Y34,X1,X2,X3,G1,G2,G3,C,Twist_initial,x_initial,L_initial,R_initial,ESF_initial,Cf_initial,Lift_initial,tc_initial,M_initial,h_initial,T_initial)

%-----calculate Y partials-----%

[A12]=fin_diff_A12(Z,Y1,Y21,Y31,X1,C,x_initial,L_initial,R_initial,Lift_initial,Twist_initial,tc_initial);
[A13]=fin_diff_A13(Z,Y1,Y21,Y31,X1,C,x_initial,L_initial,R_initial,Lift_initial,Twist_initial,tc_initial);
[A21]=fin_diff_A21(Z,Y2,Y12,Y32,X2,C,Twist_initial,ESF_initial,Cf_initial,tc_initial);
[A23]=fin_diff_A23(Z,Y2,Y12,Y32,X2,C,Twist_initial,ESF_initial,Cf_initial,tc_initial);
[A32]=fin_diff_A32(Z,Y3,Y23,X3,C,M_initial,h_initial,T_initial);
[A41]=fin_diff_A41(Z,Y4,Y14,Y24,Y34);
[A42]=fin_diff_A42(Z,Y4,Y14,Y24,Y34);
[A43]=fin_diff_A43(Z,Y4,Y14,Y24,Y34);

[dg1_Y21]=fdG1_Y21(Z,Y1,Y21,Y31,X1,G1,C,x_initial,L_initial,R_initial,Lift_initial,Twist_initial,tc_initial);
[dg1_Y31]=fdG1_Y31(Z,Y1,Y21,Y31,X1,G1,C,x_initial,L_initial,R_initial,Lift_initial,Twist_initial,tc_initial);
dg1_YE1 = [dg1_Y21 dg1_Y31];
[dg2_Y12]=fdG2_Y12(Z,Y2,Y12,Y32,X2,G2,C,Twist_initial,ESF_initial,Cf_initial,tc_initial);
[dg2_Y32]=fdG2_Y32(Z,Y2,Y12,Y32,X2,G2,C,Twist_initial,ESF_initial,Cf_initial,tc_initial);
dg2_YE2 = [dg2_Y12 dg2_Y32];
[dg3_Y23]=fdG3_Y23(Z,Y3,Y23,X3,G3,C,M_initial,h_initial,T_initial);
dg3_YE3 = [dg3_Y23];

%-----construct A matrix-----%

A = [...
```

```
1 0 0 -A12(1,1) 0 0 0 -A13(1,1) 0 0
0 1 0 -A12(2,1) 0 0 0 -A13(2,1) 0 0
0 0 1 -A12(3,1) 0 0 0 -A13(3,1) 0 0
-A21(1,1) 0 -A21(1,2) 1 0 0 0 0 -A23(1,1) 0
-A21(2,1) 0 -A21(2,2) 0 1 0 0 0 -A23(2,1) 0
-A21(3,1) 0 -A21(3,2) 0 0 1 0 0 -A23(3,1) 0
0 0 0 0 -A32(1,1) 0 1 0 0 0
0 0 0 0 -A32(2,1) 0 0 1 0 0
0 0 0 0 -A32(3,1) 0 0 0 1 0
-A41(1,1) -A41(1,2) 0 0 0 -A42(1,1) -A43(1,1) 0 0 1];

%-----calculate X partials-----%

[dY1_X1_1,dY1_X1_2]=fdY1_X1(Z,Y1,Y21,Y31,X1,C,x_initial,L_initial,R_initial,
Lift_initial,Twist_initial,tc_initial);
[dY2_X2]=fdY2_X2(Z,Y2,Y12,Y32,X2,C,Twist_initial,ESF_initial,Cf_initial,tc_initi
al);
[dY3_X3]=fdY3_X3(Z,Y3,Y23,X3,C,M_initial,h_initial,T_initial);

%-----calculate Z partials-----%

[dY1_Z1,dY1_Z4,dY1_Z5,dY1_Z6]=fdY1_Z(Z,Y1,Y21,Y31,X1,C,x_initial,L_initial,
R_initial,Lift_initial,Twist_initial,tc_initial);
[dY2_Z1,dY2_Z2,dY2_Z3,dY2_Z4,dY2_Z5,dY2_Z6]=fdY2_Z(Z,Y2,Y12,Y32,X2,C,
Twist_initial,ESF_initial,Cf_initial,tc_initial);
[dY3_Z2,dY3_Z3]=fdY3_Z(Z,Y3,Y23,X3,C,M_initial,h_initial,T_initial);
[dY4_Z2,dY4_Z3]=fdY4_Z(Z,Y4,Y14,Y24,Y34);

[dg1_Z]=fdG1_Z(Z,Y1,Y21,Y31,X1,G1,C,x_initial,L_initial,R_initial,Lift_initial,Tw
ist_initial,tc_initial);
[dg2_Z]=fdG2_Z(Z,Y2,Y12,Y32,X2,G2,C,Twist_initial,ESF_initial,Cf_initial,tc_initi
al);
[dg3_Z]=fdG3_Z(Z,Y3,Y23,X3,G3,C,M_initial,h_initial,T_initial);

%-----construct RHS matrices-----%

dY_lambda = [dY1_X1_1; 0; 0; 0; 0; 0; 0; 0];
dY_x = [dY1_X1_2; 0; 0; 0; 0; 0; 0; 0];
dY_Cf = [0; 0; 0; dY2_X2; 0; 0; 0; 0];
dY_T = [0; 0; 0; 0; 0; 0; dY3_X3; 0];

dY_tc = [dY1_Z1' dY2_Z1' 0 0 0 0]';
dY_h = [0 0 0 dY2_Z2' dY3_Z2' dY4_Z2']';
dY_M = [0 0 0 dY2_Z3' dY3_Z3' dY4_Z3']';
dY_AR = [dY1_Z4' dY2_Z4' 0 0 0 0]';
dY_Lambda = [dY1_Z5' dY2_Z5' 0 0 0 0]';
dY_Sref = [dY1_Z6' dY2_Z6' 0 0 0 0]';
```

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1998	Technical Memorandum

4. TITLE AND SUBTITLE
Bi-Level Integrated System Synthesis (BLISS)

5. FUNDING NUMBERS
509-10-31-03

6. AUTHOR(S)
Jaroslaw Sobieszczanski-Sobieski, Jeremy S. Agte, and Robert R. Sandusky, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER
L-17778

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA/TM-1998-208715

11. SUPPLEMENTARY NOTES
Sobieski: Manager, Computational AeroSciences, and Multidisciplinary Research Coordinator, NASA Langley Research Center; Agte: Graduate Research Scholar Assistant, The George Washington University, LT, U.S. Air Force; Sandusky: Professor, The George Washington University.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category 05 Distribution: Standard
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
BLISS is a method for optimization of engineering systems by decomposition. It separates the system level optimization, having a relatively small number of design variables, from the potentially numerous subsystem optimizations that may each have a large number of local design variables. The subsystem optimizations are autonomous and may be conducted concurrently. Subsystem and system optimizations alternate, linked by sensitivity data, producing a design improvement in each iteration. Starting from a best guess initial design, the method improves that design in iterative cycles, each cycle comprised of two steps. In step one, the system level variables are frozen and the improvement is achieved by separate, concurrent, and autonomous optimizations in the local variable subdomains. In step two, further improvement is sought in the space of the system level variables. Optimum sensitivity data link the second step to the first. The method prototype was implemented using MATLAB and iSIGHT programming software and tested on a simplified, conceptual level supersonic business jet design, and a detailed design of an electronic device. Satisfactory convergence and favorable agreement with the benchmark results were observed. Modularity of the method is intended to fit the human organization and map well on the computing technology of concurrent processing.

14. SUBJECT TERMS
Parallel Processing; Optimization by Decomposition; Engineering System Design

15. NUMBER OF PAGES
42

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z-39-18
298-102