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.

The use of trademarks or names of manufacturers in this report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.

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.
for a method applicable in later design phases when
the number of design variables is much larger. Meth-
ods that build a path in design space fit that require-
ment. 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 ob-
jective 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 de-
rivatives 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 de-
sign subspaces and the common system-level design
space. Each segment of that path results in an im-
proved design so that if one starts from a feasible state,
the feasibility in each modular design subspace is pre-
served 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. Be-
cause the system analysis is performed at the outset of
each segment of the path, the process can be termi-
nated 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 com-
mittling 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(Yr(Z,Xr)) - analysis of BB, to compute Yr for
given Z and Xr
BBOF - BB Objective Function computed in BB
BBOPT(Xr,θr,Gr) - optimization in BB, defined by
eq.(2.1/9)
BBOSA(Xr,opt,Z,Yr opt) - analysis of BB optimum for
sensitivity to parameters
BBSA(D(Yr(Z,Xr,Yr opt))) - sensitivity analysis of BB, to
compute its output derivatives w.r.t. Z, Xr, and Yr opt
D(V1,V2) - total derivative dV1/dV2
d(V1,V2) - partial derivative ∂V1/∂V2;
D(), and d() dimensionality depends on the dimension-
alities of V1 and V2:
V1 vector, V2 scalar, then D and d are scalars
V1 scalar, V2 vector, then D and d are vectors
V1 vector, V2 vector, then D and d are matrices
G0 - vector of constraints active at the constrained
minimum, length NG0
Gr - vector of the constraint functions, gr local to BBr ,
Gr ≤ 0 is a satisfied constraint
Gps - 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 Sub-
systems.
I - identity matrix.
L - vector of the Lagrange multipliers corresponding to
Go, length NG0
LP - Linear Programming
NB - the number of BBs in the system
NLP - NonLinear Programming
opt - subscript denotes optimized quantity
P - vector of parameters, pr kept constant in the proc-
ness 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,Φ) - system objective optimization defined by
eq. (2.2.3/1)
SSA(D(Y,Z,X)) - system sensitivity analysis to com-
pute sensitivity of the system response Y w.r.t. Z
and X
TOGW - take-off gross weight
T - superscript denotes transposition.
Xr - vector of the design variables xri, length NXr,
these variables are local to BBr; X without sub-
script - a vector of all concatenated Xr, length NX
XL, XU - lower and upper bounds on X, side-
constraints.
Yr - vector of behavior (state) variables output from
BB, these are the coupling variables; an element
of Yr is denoted yrj, some of yrj are routed as in-
puts to other BBs, and may also be routed as out-
put to the outside; the Yr length is NYr; Y without sub-
scripts - a vector of all concatenated Yr, length NY
Y_{ts} - vector of variables input to BB, from BB, these are the coupling variables; an element of Y_{ts} is denoted y_{ts,i}; note that by this definition Y_{ts} is a subset of Y, vector length NY_{ts}.

Z - vector of the design variables z_i 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_j - the local objective function in BB_j

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.

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,i} - a noise abatement constraint on the mission pro-

file; g_{2,i} - limit of the chordwise pressure gradient;

g_{3,i} - 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

Find: ΔX and ΔZ

Minimize: Φ(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 mini-
mization 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 BB1 so that

$$\Phi = y_{1,i}$$  \hspace{1cm} (1)

is one of the elements of Y1.

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}

$$[A] \{D(Y, x_{r,j})\} = \{d(Y, X_{r,j})\}$$  \hspace{1cm} (2)

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

$$\begin{bmatrix}
1 & A_{1,2} & A_{1,3} \\
A_{2,1} & 1 & A_{2,3} \\
A_{3,1} & A_{3,2} & 1
\end{bmatrix}$$  \hspace{1cm} (3)

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

$$A_{2,3} = -[d(Y_2, Y_3)], \text{ NY}_2x\text{NY}_3$$

$$A_{3,2} = -[d(Y_3, Y_2)], \text{ NY}_3x\text{NY}_2$$  \hspace{1cm} (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 \Phi as a function of X by the linear part of the Taylor series

$$\Phi = y_{1,i} = (y_{1,i})_0 + D(y_{1,i}, X_1)^T \Delta X_1 + D(y_{1,i}, X_2)^T \Delta X_2 + D(y_{1,i}, X_3)^T \Delta X_3$$  \hspace{1cm} (5)

where D-terms are vectors of length NX_r.

We see from eq. 5 that

$$\Delta \Phi = D(y_{1,i}, X_1)^T \Delta X_1 + D(y_{1,i}, X_2)^T \Delta X_2 + D(y_{1,i}, X_3)^T \Delta X_3$$  \hspace{1cm} (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 BB2 as an example, objective \phi_2 is

$$\phi_2 = D(y_{1,i}, X_2)^T \Delta X_2, j = 1\rightarrow NX_2$$  \hspace{1cm} (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 design variables X_r on the entire system objective function.

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

$$\phi_2 = D(y_{1,i}, X_2)^T \Delta X_2 + D(y_{1,i}, X_2)^T \Delta X_2 + ... + D(y_{1,i}, X_2)^T \Delta X_2 + ..., j = 1\rightarrow NX_2$$  \hspace{1cm} (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, where \Phi is being computed. In the aircraft example it is \Phi = y_{1,i} in BB1 according to eq. 1. However, the BB1 optimization objective is not \phi_1 = y_{1,i}. Instead, it is \phi_1 from an equation analogous to eq.8.

The local optimization problem may be stated formally for BB2

Given: \hspace{1cm} X_2, Z, and Y_{2,1}, \ Y_{2,3} \hspace{1cm} (9)

Find: \hspace{1cm} \Delta X_2; \text{ length NX}_2

Minimize: \hspace{1cm} \phi_2 = D(y_{1,i}, X_2)^T \Delta X_2

Satisfy: \hspace{1cm} G_2 \leq 0, \text{ including side-constraints}

Incidentally, we adhere to the convention which calls for minimization of the objective function. If the appli-
cation requires that function be maximized, as it does in the example of aircraft range, we convert the objective, e.g., \( \Phi = - (\text{range}) \).

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 \( \Phi \) 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 \( Z \)s 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

\[
\begin{bmatrix}
\frac{d(Y,z_{i})}{d(X,z_{i})} \\
\frac{d(Y,z_{i})}{d(X,z_{i})}
\end{bmatrix}
= \begin{bmatrix}
\frac{d(Y,z_{i})}{d(X,z_{i})} \\
\frac{d(Y,z_{i})}{d(X,z_{i})}
\end{bmatrix}
\]

termed GSE/OS for GSE/Optimized Subsystems. The GSE/OS yields a vector \( D(Y,z_{i}) \) and \( D(X,z_{i}) \), and because \( \Phi \) is one of the elements of \( Y \), \( \Phi = y_{1,i} \), we get the desired derivative \( D(\Phi,z_{i}) \). 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 \( d(Yr,Y0) \), \( d(Yr,Xr) \), and \( d(Xr,Y0) \) 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 \( d(X,z_{i}) \) and \( d(X,Y_{i}) \) 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 \( \Phi \), for the constraint changes caused by incrementing \( p_c \). For a general case of the objective \( F=F(P) \) and \( G_o=G_o(P) \), the algorithm gives the following formula for \( D(F,P) \)

\[
D(F,P)_0 = D(F,P) + L^T d(G_o,P)
\]

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

\[
D(y_{1,i},Z)_0 = (L^T d(G_o,Z))_1 + (L^T d(G_o,Z))_2 + (L^T d(G_o,Z))_3 + (L^T d(Y_o,Y))_1 + (L^T d(Y_o,Y))_2 + (L^T d(Y_o,Y))_3 + (L^T d(Y_o,Y))_4 + (L^T d(Y_o,Y))_5
\]

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 BBs
- \( L \) - obtained for BB, 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 \( D(Y,Z) \)

National Aeronautics and Space Administration
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 \( D(y_{1,i},Z) \), 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 \( \Phi_0 \)  
Find: \( \Delta Z \)  
Minimize: \( \Phi = \Phi_0 + D(y_{1,i},Z)^T \Delta Z \)  
Satisfy: \( Z_L \leq Z + \Delta Z \leq Z_U; \Delta Z_L \leq \Delta Z \leq \Delta Z_U \)

Where \( \Phi_0 \) 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 \( G_0 = 0 \) 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:

\[
G_{yz}^T = G_{yz,0}^T + (d(G_{yz,Z}) + d(G_{yz,Y})D(Y,Z))^T \Delta Z \leq 0
\]

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.

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 \( d(Y,X) \), \( d(Y_{1,i},Y_i) \), \( d(G,Z) \), and \( d(G,Y) \), and SSA, eq. (2.1/2), to get \( D(Y,X) \) and

Figure 2: BLISS/B Flowchart
Here is an opportunity for concurrent processing.

4. BBOPT for all BBs, eq. (2.1/9) using $\phi$ formulated individually for each BB (eq. (2.1/6, 7)), get $\phi_{opt}$ and $\Delta x_{opt}$, 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 $\Delta z_{opt}$ 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.

Caveat: because in BLISS/B the extrapolation of $\Phi$ 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 3: Polynomial Representation of Wing Twist](image)

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 dominant of NB X-spaces (Step #4) and the single Z-space (Step #6).

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 (BB1), aerodynamics (BB2), propulsion (BB3), and aircraft range (BB4). All the disciplines were represented by modules comprising an analysis level typical for an early conceptual design stage.

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

<table>
<thead>
<tr>
<th>var \ cycle</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range (SSA)</td>
<td>535.79</td>
<td>1581.67</td>
<td>3425.35</td>
<td>3960.56</td>
<td>3963.91</td>
</tr>
<tr>
<td>Extpl. Error</td>
<td>-535.79</td>
<td>-536.67</td>
<td>-431.63</td>
<td>-56.26</td>
<td>-3.43</td>
</tr>
<tr>
<td>X Extpl.</td>
<td>60.02</td>
<td>110.75</td>
<td>110.75</td>
<td>110.75</td>
<td>110.75</td>
</tr>
<tr>
<td>Z Extpl.</td>
<td>449.19</td>
<td>1301.30</td>
<td>559.90</td>
<td>559.90</td>
<td>559.90</td>
</tr>
<tr>
<td>Range (Extpl.)</td>
<td>1045.00</td>
<td>2939.72</td>
<td>3905.15</td>
<td>3960.56</td>
<td>3963.91</td>
</tr>
<tr>
<td>λ</td>
<td>0.25</td>
<td>0.14951</td>
<td>0.17476</td>
<td>0.25775</td>
<td>0.38757</td>
</tr>
<tr>
<td>x</td>
<td>1.00</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>C_l</td>
<td>1.00</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>T</td>
<td>0.5</td>
<td>0.1676</td>
<td>0.20703</td>
<td>0.15624</td>
<td>0.15624</td>
</tr>
<tr>
<td>h(ft)</td>
<td>45000</td>
<td>54000</td>
<td>60000</td>
<td>60000</td>
<td>60000</td>
</tr>
<tr>
<td>M</td>
<td>1.6</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
</tr>
<tr>
<td>AR</td>
<td>5.5</td>
<td>4.4</td>
<td>3.3</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td>S_{m2}(ft^2)</td>
<td>1000</td>
<td>1200</td>
<td>1400</td>
<td>1500</td>
<td>1500</td>
</tr>
</tbody>
</table>

*One cycle is one pass through the BLISS procedure*

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.

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

<table>
<thead>
<tr>
<th>var \ cycle</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range (SSA)</td>
<td>535.79</td>
<td>1581.67</td>
<td>3425.35</td>
<td>3960.56</td>
<td>3963.91</td>
</tr>
<tr>
<td>Extpl. Error</td>
<td>-535.79</td>
<td>-536.67</td>
<td>-431.63</td>
<td>-56.26</td>
<td>-3.43</td>
</tr>
<tr>
<td>X Extpl.</td>
<td>60.02</td>
<td>110.75</td>
<td>110.75</td>
<td>110.75</td>
<td>110.75</td>
</tr>
<tr>
<td>Z Extpl.</td>
<td>449.19</td>
<td>1301.30</td>
<td>559.90</td>
<td>559.90</td>
<td>559.90</td>
</tr>
<tr>
<td>Range (Extpl.)</td>
<td>1045.00</td>
<td>2939.72</td>
<td>3905.15</td>
<td>3960.56</td>
<td>3963.91</td>
</tr>
<tr>
<td>λ</td>
<td>0.25</td>
<td>0.14951</td>
<td>0.17476</td>
<td>0.25775</td>
<td>0.38757</td>
</tr>
<tr>
<td>x</td>
<td>1.00</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>C_l</td>
<td>1.00</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>T</td>
<td>0.5</td>
<td>0.1676</td>
<td>0.20703</td>
<td>0.15624</td>
<td>0.15624</td>
</tr>
<tr>
<td>h(ft)</td>
<td>45000</td>
<td>54000</td>
<td>60000</td>
<td>60000</td>
<td>60000</td>
</tr>
<tr>
<td>M</td>
<td>1.6</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
</tr>
<tr>
<td>AR</td>
<td>5.5</td>
<td>4.4</td>
<td>3.3</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td>S_{m2}(ft^2)</td>
<td>1000</td>
<td>1200</td>
<td>1400</td>
<td>1500</td>
<td>1500</td>
</tr>
</tbody>
</table>

*One cycle is one pass through the BLISS procedure*

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 BB4 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 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.

Table 4: Electronic Packaging Data

<table>
<thead>
<tr>
<th>Method</th>
<th>Case</th>
<th>Initial Design Objective</th>
<th>Final Design Objective</th>
<th>Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-in-O</td>
<td>1</td>
<td>7.79440E+01</td>
<td>6.39720E+05</td>
<td>122E-03</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6.83030E+02</td>
<td>6.39720E+05</td>
<td>122E-03</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1.51110E+03</td>
<td>6.39720E+05</td>
<td>122E-03</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.46070E+03</td>
<td>6.39720E+05</td>
<td>122E-03</td>
</tr>
<tr>
<td>BLISS/A</td>
<td>1</td>
<td>7.79440E+01</td>
<td>6.39720E+05</td>
<td>122E-03</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6.83030E+02</td>
<td>6.39720E+05</td>
<td>122E-03</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1.51110E+03</td>
<td>6.39720E+05</td>
<td>122E-03</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.46070E+03</td>
<td>6.39720E+05</td>
<td>122E-03</td>
</tr>
<tr>
<td>BLISS/B</td>
<td>1</td>
<td>7.79440E+01</td>
<td>6.39720E+05</td>
<td>122E-03</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6.83030E+02</td>
<td>6.39720E+05</td>
<td>122E-03</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1.51110E+03</td>
<td>6.39720E+05</td>
<td>122E-03</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.46070E+03</td>
<td>6.39720E+05</td>
<td>122E-03</td>
</tr>
</tbody>
</table>

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

National Aeronautics and Space Administration
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


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


**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

\[
(A1) \quad Y_{1,2} = Y_{1,2}(Z, X_2, Y_{2,1}, Y_{2,3})
\]

\[
(A2) \quad Y_{1,3} = Y_{1,3}(Z, X_3, Y_{3,1}, Y_{3,2})
\]

\[
(A3) \quad Y_{2,1} = Y_{2,1}(Z, X_1, Y_{1,2}, Y_{1,3})
\]

\[
(A4) \quad Y_{2,3} = Y_{2,3}(Z, X_3, Y_{3,1}, Y_{3,2})
\]

\[
(A5) \quad Y_{3,1} = Y_{3,1}(Z, X_1, Y_{1,2}, Y_{1,3})
\]

\[
(A6) \quad Y_{3,2} = Y_{3,2}(Z, X_2, Y_{2,1}, Y_{2,3})
\]

where the independent variables are $X$ and $Z$.

Observe that eq. A1-A6 are coupled by $Y$, 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

\[
(A7) \quad D(Y_{2,1},x_{1,j}) = d(Y_{2,1},x_{1,j}) + d(Y_{2,1},Y_{1}) D(Y_{1},x_{1,j})
\]

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 $x_{1,j}$ with $z_k$ in eq. (2.1/2) to obtain

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

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.

\[
(A9) \quad Y_{1,2} = Y_{1,2}(Z, X_2, Y_{2,1}, Y_{2,3})
\]
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

\[
D(Y_{2,2}, z_k) = d(Y_{2,2}, z_k) + d(Y_{2,2}, X_2)D(X_2, z_k) + d(Y_{2,2}, Y_1)D(Y_1, z_k)
\]

and for \( X_2 \), again as one example:

\[
D(X_{2,2}, z_k) = d(X_{2,2}, z_k) + d(X_{2,2}, Y_1)D(Y_1, z_k) + d(X_{2,2}, Y_3)D(Y_3, z_k)
\]

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 \( Y \), w.r.t. \( Y \) and \( Y \), w.r.t. \( X \), are obtained from BBSA, using any sensitivity analysis algorithm appropriate for the particular BB, (including the option of finite differencing). The derivatives of \( X \), w.r.t. \( z \), and \( X \), w.r.t. \( Y \), are produced by an analysis of optimum for sensitivity to parameters, BBOSA, 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 \( D(Y_{2,2}, z_k) \) and \( D(X_{2,2}, z_k) \) 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

\[
[M_y][D(Y,Z_k)] + [M_x][D(X,z_k)] = d(Y,z_k)
\]

The internal structure of the \( M \)-matrices in the above is

for \([M_y]\):

\[
\begin{bmatrix}
1 & -d(Y_{1,1}, Y_{2,1}) & -d(Y_{1,1}, Y_{3,1}) \\
-d(Y_{2,1}, Y_{1,1}) & 1 & -d(Y_{2,1}, Y_{3,1}) \\
-d(Y_{3,1}, Y_{1,1}) & -d(Y_{3,1}, Y_{2,1}) & 1
\end{bmatrix}
\]

for \([M_x]\):

\[
\begin{bmatrix}
-d(Y_{1,1}, X_{1,1}) & 0 & 0 \\
0 & -d(Y_{1,1}, X_{2,2}) & 0 \\
0 & 0 & -d(Y_{1,1}, X_{3,3})
\end{bmatrix}
\]

for \([M_{xy}]\):

\[
\begin{bmatrix}
0 & -d(X_{1,1}, Y_{1,1}) & -d(X_{1,1}, Y_{2,2}) \\
-d(X_{2,2}, Y_{1,1}) & 0 & -d(X_{2,2}, Y_{3,3}) \\
-d(X_{3,3}, Y_{1,1}) & -d(X_{3,3}, Y_{2,2}) & 0
\end{bmatrix}
\]

and for \([M_{yx}]\):

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

Again, in the above, all \( Y_{z_k} \) are folded into \( Y \) for compactness, and the terms are falling into the previously introduced categories as follows:

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

As in GSE, one may obtain \( D(Y_{2,2}, z_k) \) and \( D(X_{2,2}, z_k) \) for all \( z_k \), \( k = 1 \rightarrow NZ \) 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

\[(A21) \text{ Given: } P \]
\[
\text{Find: } X
\]
\[
\text{Minimize: } (X,P)
\]
\[
\text{Satisfy: } G(X,P) \leq 0, \text{ including side-}
\]
\[
\text{constraints and move limits } Z_k, \text{ and } Y_{r,s}. \text{because these quantities are kept constant in BBOPT.}
\]

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

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

1. Choose parameter \( P_k \), an element of \( Z \) or \( Y \), and increment it by \( \Delta P \)

In the BLISS application, the parameters \( P \) in BB, are

**Table A1: BB Definitions**

<table>
<thead>
<tr>
<th>BB</th>
<th>Inputs</th>
<th>Internal</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structures</td>
<td>AR, ( \Lambda ), ( t/c )</td>
<td>( t = \frac{S_{\text{REF}}}{\sqrt{S_{\text{REF}} \cdot AR}} ); ( b/2 = \frac{\sqrt{S_{\text{REF}}} \cdot AR}{2} ); ( R = \frac{1 + 2\Lambda}{3(1 + \Lambda)} ); ( \theta = )</td>
<td>( W_T, W_r, \theta )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \Delta x = \text{constant while an optimizer manipulates } X. )</td>
<td></td>
</tr>
<tr>
<td>Aerodynamics</td>
<td>M, ( \Lambda ), AR, ( \frac{1}{c} ), ( S_{\text{REF}}, W_T ), ( \theta ), ESF, ( C_{\text{D\downarrow}}, M_{\text{\downarrow}}, C_{\text{f}} )</td>
<td>if ( h &lt; 36089 \text{ft}; ) ( V = M1116.39 \sqrt{1 - (6.875 \cdot 0.06)h}, \rho = (2.377e-03)(1 - (6.875 \cdot 0.06)h)^{1.246} ); ( V = M968.1, \rho = (2.377e-03) \text{ for } h &gt; 36089 \text{ft}: ) ( \Delta x = )</td>
<td>( L, D, \frac{L}{D} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \Delta s = \frac{0.5V^2S_{\text{REF}}}{C_{\text{D\downarrow}}, M_{\text{\downarrow}}, C_f, \Delta x} ); ( \theta = )</td>
<td></td>
</tr>
<tr>
<td>Propulsion</td>
<td>M, ( \Lambda ), ( W_{\text{BE}}, T )</td>
<td>( T = 7616.66; ) Temp = ( \frac{\text{ESF} \cdot \text{M16.66}}{1.1324 + 1.5344M - (3.295 \cdot 0.06)h - (1.6379 \cdot 0.04)T}; ) ( \text{SFC} = 1.3124 + 1.5344M - (3.295 \cdot 0.06)h - (1.6379 \cdot 0.04)T ); ( +0.31623M^2 + (8.21338 - 0.06)Mh - (10.495 - 0.05)TM - (8.574 - 11)h^2 ); ( \frac{4(3.8042 - 0.09)\text{h} + (1.0600 - 0.08)\text{T}^2}{2}; ) ( W_\tau = 3W_{\text{BE}} \text{ESF} + 0.05 ); ( \text{T}_{\text{opt}} = 11484 + 10856M - 0.50802h + 3200.2M^2 - 0.29326Mh^2 + (6.8572 - 0.06)h );</td>
<td>( S_{\text{REF}}, W_T, \text{ESF} )</td>
</tr>
<tr>
<td>Range</td>
<td>M, ( \Lambda ), ( W_T, W_F )</td>
<td>( \theta = 1.6.875e-06h, ) if ( h &lt; 36089 \text{ft}; ( \theta = 0.7519 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{if } h &gt; 36089 \text{ft: } R = \frac{M(L \cdot D)661h\ln\left(\frac{W_T}{W_F}\right)}{SFC}; \text{W}<em>{\text{BE}} = 20001b; \text{W}<em>r = 250001b; \text{N}<em>z = 6g; \text{W}</em>{\text{BE}} = 43601b; C</em>{\text{D\downarrow}, M</em>{\text{\downarrow}}, C_f} = 0.01375 )</td>
<td></td>
</tr>
</tbody>
</table>

National Aeronautics and Space Administration
linearly and by Linear Programming solve

\[
\begin{align*}
\text{Find} \quad & X \\
\text{Minimize} \quad & F \\
\text{Satisfy} \quad & G_0 \leq 0 \\
\text{XL} \leq & X \leq \text{XU}; \text{where XL and XU incorporate} \\
& \text{the side constraints and the move limits;}
\end{align*}
\]

to obtain \( X_{\text{opt}} \).

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

4. Repeat from #1 for all elements of \( Z \) and \( Y \) input into \( BB_i \).

Repeated for all \( BB_i \), the above procedure yields a set of \( \Delta (X,Z) \) and \( \Delta (X,Y) \) to be entered as \( d(X,Z) \) and \( d(X,Y) \) into GSE/OS, eq. A20. Solution of eq. A20 provides \( \Delta (X,Z) \) and \( \Delta (Y,Z) \). The latter is substituted into eq. (2.2.3/2), and \( \Delta (\Phi,Z) \), 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 \( BB_i \) for the aircraft model. Polynomial functions are represented by \( 'p/(p)' \) with independent variables in the parentheses. Each polynomial function is of the form:

\[
(A22) \quad PF = A_o + A_i S^T + (1/2) S A_{ij} S^T
\]

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 \( p/(p) \) 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:

- \( \Theta \) ---&\( A_o = [1.0]; A_i = [0.3 \cdot 0.3 \cdot 0.3 \cdot 0.2]; A_{ii} = [0.4 \cdot 0.4 \cdot 0.4 \cdot 0.4] \)
- \( \Phi_1 \) ---&\( A_o = [1.0]; A_i = [6.25]; A_{ii} = [0] \)
- \( \sigma_1 \) ---&\( A_o = [1.0]; A_i = [-0.75 \cdot 0.5 \cdot -0.75 \cdot 0.5] \)

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLISS</td>
<td>16</td>
</tr>
<tr>
<td>SYSTEM_ANALYSIS</td>
<td>19</td>
</tr>
<tr>
<td>BB_WEIGHT</td>
<td>20</td>
</tr>
<tr>
<td>BB_DRAGPOLAR</td>
<td>23</td>
</tr>
<tr>
<td>BB_POWER</td>
<td>23</td>
</tr>
<tr>
<td>BB_RANGE</td>
<td>24</td>
</tr>
<tr>
<td>POLYAPPROX</td>
<td>25</td>
</tr>
<tr>
<td>BB1OPT</td>
<td>26</td>
</tr>
<tr>
<td>BB1WRAPPER</td>
<td>28</td>
</tr>
<tr>
<td>BB2OPT</td>
<td>28</td>
</tr>
<tr>
<td>BB2WRAPPER</td>
<td>29</td>
</tr>
<tr>
<td>BB3OPT</td>
<td>30</td>
</tr>
<tr>
<td>BB3WRAPPER</td>
<td>31</td>
</tr>
<tr>
<td>SYSOPT</td>
<td>32</td>
</tr>
<tr>
<td>SYSWRAPPER</td>
<td>33</td>
</tr>
<tr>
<td>INBOUNDS</td>
<td>34</td>
</tr>
<tr>
<td>FIN_DIFF</td>
<td>34</td>
</tr>
</tbody>
</table>

The electronic version of this code has been placed in custody of Dr. Jaroslaw Sobieski, NASA Langley Research Center, Hampton, VA 23681.

Author: Jeremy S. Age, NASA Langley/GWU, Spring '98

Variables:
- **A**: Coefficient matrix in GSE
- **DY_AR**: Vector of total derivatives, behavior variables w.r.t aspect ratio
- **DY_Cf**: Vector of total derivatives, behavior variables w.r.t skin friction coefficient
- **DYE1_Z**: Matrix of total derivatives, behavior variables from BB1 w.r.t Z variables
- **DYE2_Z**: Matrix of total derivatives, behavior variables from BB2 w.r.t Z variables
- **DYE3_Z**: Matrix of total derivatives, behavior variables from BB3 w.r.t Z variables
- **DY4_Z**: Vector of total derivatives, range w.r.t Z variables
- **DY_h**: Vector of total derivatives, behavior variables w.r.t altitude
- **DY_Lamda**: Vector of total derivatives, behavior variables w.r.t wing sweep
- **DY_lamda**: Vector of total derivatives, behavior variables w.r.t taper ratio
- **DY_M**: Vector of total derivatives, behavior variables w.r.t Mach number
- **DY_Sref**: Vector of total derivatives, behavior variables w.r.t wing surface area
- **DY_T**: Vector of total derivatives, behavior variables w.r.t throttle setting
- **DY_tc**: Vector of total derivatives, behavior variables w.r.t thickness/chord ratio
- **DY_x**: Vector of total derivatives, behavior variables w.r.t wingbox x-section
- **DYX_nd**: Array of non-dimensional total derivatives, behavior w.r.t. X variables

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 (OSAA). Finally, all optimized changes to design variables are used to update the model for an improved range.
% DYZ nd - Array of non-dimensional total derivatives, vary
% behavior w.r.t. Z variables
% ext_error - Difference between previous pass extrapolated 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 ft²

% 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 ----%

v1b=[.75 .75 .10 30000 1.4 2.5 40 500];
i0=[.25 11.5 0.05 45000 1.6 6.5 55 1000];
vub=[.4 1.25 1.25 1.09 60000 1.8 8.5 701500];
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,Y24,Y31,Y32,Y34,G1,G2,G3,C,Twist_initial,x_initial,L_initial,R_initial,ESF_initial,CF_initial,Lift_initial,te_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,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,te_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;

National Aeronautics and Space Administration
GRADphi_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]=SysOPT(vlb_nd,vub_nd,i0,P_var,Y4,GRADphi_Z,Z,df 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_lamda DY_lamda 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_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';
printf(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';
printf(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';
printf(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 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_initial - 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
% %
%function(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_analysis(Z,X1,X2,X3,i0)

National Aeronautics and Space Administration
%----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_initial,LC);

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

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

[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 compute 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 deg
Z(5) - Wing sweep none
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.
% Y1(4) - Total aircraft weight lb
% Y1(2) - Wing twist p.f.
% Y1(4) - Total aircraft weight lb
% Y1(4) - 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

% function[Y1,Y12,Y14,G1]=BB_weight(Z,Y21,Y31,X1,x_initial,L_initial,R_initial,Lift_initial,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=[.08];

F0=PolyApprox(S_initial2,S2,flag2,bound2);
W_wing = F0*(.0051*(Y1(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)^1*(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 = [1.1,1.1,1,1];
G1(1)=PolyApprox(S_initial3,S3,flag3,bound3); %--wing stress

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

National Aeronautics and Space Administration
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 = [4,4,1,1,1];
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 = [4,4,1,1,1];
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 polynomials.

Author : Jeremy S. Agte  NASA Langley/GWU  Spring '98

Input Variables:
%  C       - Vector of constants
%  Cf_initial - Initial coefficient of friction
%  ESF_initial- Initial engine scale factor
%  tc_initial - Initial thickness to chord ratio
%  Twist_initial- Initial wing twist
%  X2       - Coefficient of friction
%  Y12(1)   - Total aircraft weight
%  Y12(2)   - Wing twist
%  Y32      - Engine scale factor
%  Z(1)     - Thickness/chord ratio
%  Z(2)     - Altitude
%  Z(3)     - Mach number
%  Z(4)     - Aspect ratio
%  Z(5)     - Wing sweep
%  Z(6)     - Wing surface area
%  G2       - Pressure gradient
%  Y2(1)    - Lift
%  Y2(2)    - Drag
%  Y2(3)    - Lift-to-drag ratio
%  Y21      - Lift
%  Y23      - Drag
%  Y24      - Lift-to-drag ratio
%  CL       - Coefficient of lift
%  CD       - Coefficient of drag
%  CDmin    - Minimum drag coefficient
%  k        - Induced drag factor
%  rho      - Density
%  V        - Velocity

Subfunctions:
%  PolyApprox - Forms polynomial functions for desired variables

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function[Z12,Y21,Y23,Y24,G2]=BB_dragpolar(Z,Y12,Y32,X2,ESF_initial,Cf_initial,Twist_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;

National Aeronautics and Space Administration
else
   V = Z(3)*968.1;
   rho = (2.377e-03)*(2.971)*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 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: l/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: l/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,Y32,X3,M_initial,h_initial,T_initial,C)

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

Thrust = Y32(1);
Dim_Throttle = X3(1)*16168.6;  %--non-dimensionnal 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 -0.0000000008574
  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*s(2)*Z(2)^2+s(6)
  +2*Dim_Throttle*Z(3)*s(7)+s(8)*Z(2)^2+2*_dim_Throttle*Z(2)*s(9)+s(10)*_dim_Throttle^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-le_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

National Aeronautics and Space Administration
% Y24 - Lift-to-drag ratio
% Y34 - Specific fuel consumption
% Z(1) - Thickness/chord ratio
% Z(2) - Altitude
% Z(3) - Mach number
% Z(4) - Aspect ratio
% Z(5) - Wing sweep
% Z(6) - Wing surface area
% Output Variables
% Y4 - Range
% Theta - Temperature ratio
% Output Variables
% flag
% S
% S_bound
% S_new
% Input Variables
% Author : Jeremy S. Agte  NASA Langley/GWU  Spring '98
% % Input Variables
% % flag  - Indicates functional relationship b/w 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
% a
% b
% F_bound
% Mtx_shifted
% R
% Sl
% S_norm
% So
% S_shifted
% Su
% %----THIS SECTION COMPUTES THE A/C RANGE (Breguet)-----

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

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

National Aeronautics and Space Administration
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
    end
    %DETERMINE BOUNDS ON FF DEPENDING ON SLOPE-SHAPE
    a=0.1;
    b=a;
    if flag(i)==5
        %DETERMINE BOUNDS ON FF DEPENDING ON SLOPE-SHAPE
        a=0.1;
        b=a;
    end
    %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
%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);
%DETERMINE BOUNDS ON FF DEPENDING ON SLOPE-SHAPE
%CALCULATE POLYNOMIAL COEFFICIENTS
%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
% 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

function[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)=0.0001;
options(3)=0.0001;
options(4)=0.001;
options(14)=1000;
options(17)=.01;

[fstore,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);
% 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 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 ft²
%
% 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_IA - Lower allowable limit for twist constraint none
% Twist_uA - Upper allowable limit for twist constraint none
%
% Subfunctions :
% BB_weight - Calculates a/c structural weights
%
function [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_IA=.97;

g(1)=G1(1)/Sigma_uA-1;
g(2)=G1(2)/Sigma_uA-1;
g(3)=G1(3)/Sigma_uA-1;
g(4)=G1(4)/Sigma_uA-1;
g(5)=G1(5)/Sigma_uA-1;
g(6)=G1(6)/Twist_uA-1;
g(7)=Twist_IA/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 vary
% ESF_Cf - Initial engine scale factor vary
% i0 - Design variable initial values none
% P_var - Vector of current design variable values vary
% tc_initial - Initial thickness to chord ratio none
% Twist_initial - Initial wing twist p.f.
% vlbd - Non-dimensional lower bounds on design variables none
% vubd - Non-dimensional upper bounds on design variables none
% X2 - Coefficient of friction p.f.
% Y12(1) - Total aircraft weight none
% 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
% vlbd - Non-dimensional lower bounds on BB2 design variables none
% vubd - 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

function(x2,Lagrange2,fstore)=BB2OPT(vlbd,vubd,i0,P_var,Z,Y12,Y32,x2,ESF_initial,Cf_initial,Twist_initial,tc_initial,C,dy_cfd)

vlbd=vlbd(3); vubd=vubd(3); x0=[x(1)/60(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,vlbd,vubd,[],i0,P_var,Z,Y12, Y32,ESF_initial,Cf_initial,Twist_initial,tc_initial,C,dy_cfd);

[fstore,gstore,dx2]=BB2WRAPPER(x,i0,P_var,Z,Y12,Y32,ESF_initial,Cf_initial,Twist_initial,tc_initial,C,dy_cfd);
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
- 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, ft²

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
- X2=[i0(3)*(1+x(1))];
- [Y2,Y21,Y23,Y24,G2]=BB_dragpolar(Z,Y12,Y32,X2,ESF_initial,Cf_initial,Twist_initial,tc_initial,C,DY_Cf);
- Pg_uA=1.04;
- g(1)=G2(1)/Pg_uA-1;
- dX2=[X2(1)-P_var(3)];
- f=-(DY_Cf(10))*dX2;
function([X3,Lagrange3,fstore,gstore0]=BB3OPT(vlb_nd,vub_nd,i0,P_var,Z,Y23,X3,M_initial,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_initial,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,\nD Y_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 vary
% % h_initial - Initial altitude ft

National Aeronautics and Space Administration
function [f,g,dX3]=BB3WRAPPER(x,i0,P_var,Z,Y23,M_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,DY_T);

ESF_uA=1.5;
ESF_uA=5;

Temp_uA=1.02;

g(1)=G3(1)/ESF_uA-1;
g(2)=ESF_uA/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 w.r.t 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 ft²
%
% 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

function[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=[Z1(i0(5:10));Z(2)/i0(6:10):1.Z(3)/i0(7:1:10:Z(4)/i0(8:1:10:Z(5)/i0(9:1:10:Z(6)/i0(10:1):1;options(1)=1;
options(2)=0.001;
options(3)=0.001;
options(4)=0.001;
options(14)=1000;
options(17)=0.01;

[fstore,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,G2);

Subfunction SYSWRAPPER

% 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
% Z1 - Thickness/chord ratio none
% Z(2) - Altitude ft
% Z(3) - Mach number none
% Z(4) - Aspect ratio deg
% Z(5) - Wing sweep ft^2
% Z(6) - Wing surface area

% Output Variables
% dZ - Vector of optimal changes in Z variables vary
% f - Value of objective function for system optimization 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+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;
P_g_uA=1.04;
G2(1)=G2(1)/P_g_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);

------------------
% vub
% - Non-dimensional upper bounds on BB1 design variables
% x0
% - Vector of non-dimensional starting points for BB1 optimization
% Output Variables:
% vlb_nd
% - Non-dimensional lower bounds on design variables
% vub_nd
% - Non-dimensional upper bounds on design variables

------------------

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)-
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)-
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
% Cf_initial - Initial coefficient of friction
% ESF_initial - Initial engine scale factor
% G1(1) - Stress on wing
% G1(2) - Stress on wing
% G1(3) - Stress on wing
% G1(4) - Stress on wing
% G1(5) - Stress on wing

National Aeronautics and Space Administration
<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Unit(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1(6)</td>
<td>Wing twist as constraint</td>
<td>p.f.</td>
</tr>
<tr>
<td>G2</td>
<td>Pressure gradient</td>
<td>p.f.</td>
</tr>
<tr>
<td>G3(1)</td>
<td>Engine scale factor constraint</td>
<td>none</td>
</tr>
<tr>
<td>G3(2)</td>
<td>Engine temperature</td>
<td>p.f.</td>
</tr>
<tr>
<td>G3(3)</td>
<td>Throttle setting constraint</td>
<td>none</td>
</tr>
<tr>
<td>h_initial</td>
<td>Initial altitude</td>
<td>ft</td>
</tr>
<tr>
<td>L_initial</td>
<td>Initial halfspan length</td>
<td>ft</td>
</tr>
<tr>
<td>Lift_initial</td>
<td>Initial lift</td>
<td>lb</td>
</tr>
<tr>
<td>M_initial</td>
<td>Initial Mach number</td>
<td>none</td>
</tr>
<tr>
<td>R_initial</td>
<td>Initial location of lift as fraction of halfspan</td>
<td>none</td>
</tr>
<tr>
<td>tc_initial</td>
<td>Initial thickness to chord ratio</td>
<td>none</td>
</tr>
<tr>
<td>T_initial</td>
<td>Initial throttle setting</td>
<td>none</td>
</tr>
<tr>
<td>Twist_initial</td>
<td>Initial wing twist</td>
<td>p.f.</td>
</tr>
<tr>
<td>x_initial</td>
<td>Initial wingbox x-sectional thickness</td>
<td>p.f.</td>
</tr>
<tr>
<td>Y1(1)</td>
<td>Total aircraft weight</td>
<td>lb</td>
</tr>
<tr>
<td>Y1(2)</td>
<td>Fuel weight</td>
<td>lb</td>
</tr>
<tr>
<td>Y1(3)</td>
<td>Wing twist</td>
<td>p.f.</td>
</tr>
<tr>
<td>Y12(1)</td>
<td>Total aircraft weight</td>
<td>lb</td>
</tr>
<tr>
<td>Y12(2)</td>
<td>Wing twist</td>
<td>p.f.</td>
</tr>
<tr>
<td>Y14(1)</td>
<td>Total aircraft weight</td>
<td>lb</td>
</tr>
<tr>
<td>Y14(2)</td>
<td>Fuel weight</td>
<td>lb</td>
</tr>
<tr>
<td>Y2(1)</td>
<td>Lift</td>
<td>lb</td>
</tr>
<tr>
<td>Y2(2)</td>
<td>Drag</td>
<td>lb</td>
</tr>
<tr>
<td>Y2(3)</td>
<td>Lift-to-drag ratio</td>
<td>none</td>
</tr>
<tr>
<td>Y21</td>
<td>Lift</td>
<td>lb</td>
</tr>
<tr>
<td>Y23</td>
<td>Drag</td>
<td>lb</td>
</tr>
<tr>
<td>Y24</td>
<td>Lift-to-drag ratio</td>
<td>none</td>
</tr>
<tr>
<td>Y3(1)</td>
<td>Specific fuel consumption</td>
<td>1/hr</td>
</tr>
<tr>
<td>Y3(2)</td>
<td>Engine weight</td>
<td>lb</td>
</tr>
<tr>
<td>Y3(3)</td>
<td>Engine scale factor</td>
<td>none</td>
</tr>
<tr>
<td>Y31</td>
<td>Engine weight</td>
<td>lb</td>
</tr>
<tr>
<td>Y32</td>
<td>Engine scale factor</td>
<td>none</td>
</tr>
<tr>
<td>Y34</td>
<td>Specific fuel consumption</td>
<td>1/hr</td>
</tr>
<tr>
<td>Y4</td>
<td>Objective function output from BB4</td>
<td>NM</td>
</tr>
<tr>
<td>X1(1)</td>
<td>Wing taper ratio</td>
<td>none</td>
</tr>
<tr>
<td>X1(2)</td>
<td>Wingbox x-sectional area as poly. funct.</td>
<td>p.f.</td>
</tr>
<tr>
<td>X2</td>
<td>Skin friction coefficient as poly. funct.</td>
<td>p.f.</td>
</tr>
<tr>
<td>X3</td>
<td>Throttle setting</td>
<td>none</td>
</tr>
<tr>
<td>Z(1)</td>
<td>Thickness/chord ratio</td>
<td>none</td>
</tr>
<tr>
<td>Z(2)</td>
<td>Altitude</td>
<td>ft</td>
</tr>
<tr>
<td>Z(3)</td>
<td>Mach number</td>
<td>none</td>
</tr>
<tr>
<td>Z(4)</td>
<td>Aspect ratio</td>
<td>none</td>
</tr>
<tr>
<td>Z(5)</td>
<td>Wing sweep</td>
<td>deg</td>
</tr>
<tr>
<td>Z(6)</td>
<td>Wing surface area</td>
<td>ft²</td>
</tr>
</tbody>
</table>

**Output:**

- Coefficient matrix in GSE
- Vector of derivatives, BB1 constraints
  - w.r.t Z variables
- Vector of derivatives, BB2 constraints
  - w.r.t Z variables
- Vector of derivatives, BB3 constraints
  - w.r.t Z variables
- Vector of derivatives, BB1 constraints
  - w.r.t Y variables entering BB1
- Vector of derivatives, BB2 constraints
  - w.r.t Y variables entering BB2
- Vector of derivatives, BB3 constraints
  - w.r.t Y variables entering BB3
- Vector of partial derivatives, behavior
  - variables w.r.t aspect ratio
- Vector of partial derivatives, behavior
  - variables w.r.t skin friction coefficient
- Vector of partial derivatives, behavior
  - variables w.r.t altitude
- Vector of partial derivatives, behavior
  - variables w.r.t wing sweep
- Vector of partial derivatives, behavior
  - variables w.r.t taper ratio
- Vector of partial derivatives, behavior
  - variables w.r.t Mach number
- Vector of partial derivatives, behavior
  - variables w.r.t wing surface area
- Vector of partial derivatives, behavior
  - variables w.r.t throttle setting
- Vector of partial derivatives, behavior
  - variables w.r.t thickness/chord ratio
- 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 constraint output w.r.t. change in Z variables
% fdG2_Z - Calculates BB2 constraint output w.r.t. change in Z variables
% fdG3_Z - Calculates BB3 constraint 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_S ref,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_init)
%-----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 = [...

National Aeronautics and Space Administration
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 -A32(2,1) 0 0 1 0
0 0 0 0 -A32(3,1) 0 0 0 1
-A41(1,1) -A41(1,2) 0 0 0 -A42(1,1) -A43(1,1) 0 0 1;

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

[dY1_X1, dY1_X2] = 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_initial);
[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, Twist_initial, tc_initial);
[dg2_Z] = fdG2_Z(Z, Y2, Y12, Y32, X2, G2, C, Twist_initial, ESF_initial, Cf_initial, tc_initial);
[dg3_Z] = fdG3_Z(Z, Y3, Y23, X3, G3, C, M_initial, h_initial, T_initial);

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

dY_lambda = [dY1_X1; 0; 0; 0; 0; 0; 0];
dY_x = [dY1_X2; 0; 0; 0; 0; 0; 0];
dY_Cf = [0; 0; dY2_X2; 0; 0; 0];
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];

dY_T = [0; 0; 0; 0; 0; 0; dY3_X3];
dY_te = [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'];

National Aeronautics and Space Administration
Bi-Level Integrated System Synthesis (BLISS)

NASA Langley Research Center
Hampton, VA 23681-2199

National Aeronautics and Space Administration
Washington, DC 20546-0001


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.