AIAA 2000-0421

ADVANCEMENT OF BI-LEVEL INTEGRATED SYSTEM SYNTHESIS (BLISS)

Jaroslaw Sobieszczanski-Sobieski*
NASA Langley Research Center, Hampton, Virginia
j.sobieski@larc.nasa.gov

Mark S. Emiley†
George Washington University
Joint Institute for the Advancement of Flight Sciences (JIAFS)
m.s.emiley@larc.nasa.gov

Jeremy S. Agte‡
United States Air Force, San Antonio, Texas
jeremy.agte@kelly.af.mil

Robert R. Sandusky, Jr.§
George Washington University
Joint Institute for the Advancement of Flight Sciences (JIAFS)
r.r.sandusky@larc.nasa.gov

Abstract

Bi-Level Integrated System Synthesis (BLISS) is a method for optimization of an engineering system, e.g., an aerospace vehicle. BLISS consists of optimizations at the subsystem (module) and system levels to divide the overall large optimization task into sets of smaller ones that can be executed concurrently. In the initial version of BLISS that was introduced and documented in previous publications, analysis in the modules was kept at the early conceptual design level. This paper reports on the next step in the BLISS development in which the fidelity of the aerodynamic drag and structural stress and displacement analyses were upgraded while the method's satisfactory convergence rate was retained.

Introduction

BLISS, for Bi-Level Integrated System Synthesis, is an optimization method for engineering a modular system, e.g., an aerospace vehicle, in which it is beneficial to separate the design variables and constraints local to a module from those that govern the entire system. Such separation fosters development of a broad workfront of people and computers, hence it fits well the current trends for massively parallel processing in computer technology and the concurrent engineering style of the workforce organization.

The focus on dividing the optimization into the suboptimizations within each module (subsystem, also called the black box) and a coordinating optimization at the system level places BLISS in the Multidisciplinary Design Optimization (MDO) toolbox, in the company of a few other methods that have the same focus as BLISS but differ in approach. Representative examples of these methods are the Collaborative Optimization

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

† Graduate Student, NASA Langley Research Center, Multidisciplinary Optimization Branch, MS 139, Member AIAA

‡ 1st Lt United States Air Force, San Antonio Air Logistics Center, TX, Member AIAA

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

(CO) (Braun et al, 1965), the Concurrent SubSpace Optimization (CSSO) (Sobieszczanski-Sobieski, 1988, Bloebaum et al, 1992), and the Concurrent Design Optimization (Wujek et al, 1995).

The distinguishing features of BLISS are the use of the system objective (e.g., the aircraft range) as the optimization objective in each of the subsystems and at the system level, and coupling between the optimizations at the system and subsystem levels by the optimum sensitivity derivatives with respect to parameters.

The overall architecture of BLISS as a method does not depend on the fidelity of the analyses performed in each module. Consequently, in principle at least, BLISS may be used in any design phase from conceptual, through preliminary to detailed, provided that appropriate level of analysis is implemented in the modules.

The BLISS method was introduced in (Sobieszczanski-Sobieski et al, 1998a) and documented in detail in (Sobieszczanski-Sobieski et al, 1998b[*]). In the paper, that prototype is referred to as the original BLISS. In its original form BLISS modules were kept very simple corresponding to the early conceptual design phase. Satisfactory results from the initial trials of BLISS on a test case of a business jet encouraged next step in the BLISS development - upgrading its structural analysis and aerodynamic drag analysis modules - and validating on the same test case.

This paper reports on the above BLISS upgrade and results of the testing that advance the methods toward becoming a tool suitable for practical applications. The report provides a synopsis of the BLISS method, describes the salient features of the two upgraded modules, presents satisfactory convergence results, and summarizes the BLISS development status and the future development direction.

Notation

AR_{HT} – tail aspect ratio
AR_W – wing aspect ratio
BB_i – black box
C_D – coefficient of drag
C_f – skin friction coefficient
D – drag
ESF – engine scale factor
h – altitude
k – safety factor
L – lift
L/D – lift to drag ratio
L_{HT} – horizontal tail location, % mean aerodynamic chord (% MAC)
L_W – wing location, % MAC
M – Mach number
N_Z – maximum load factor
R – range
SFC – specific fuel consumption
S_{HT} – horizontal tail surface area
S_{REF} – wing surface area
T – throttle
t/c – thickness to chord ratio
t_i – wingbox sandwich face sheets thicknesses
$t_{s,i}$ – wingbox sandwich caliper thicknesses
X_i – design variables local to BB_i
XL, XU – lower and upper bounds on X, side-constraints
W_E – engine weight
W_F – fuel weight
W_T – total weight
$Y_{i,j}$ – behavior variables output from BB_i and sent as inputs BB_j
Z – system-level design variables
λ – taper ratio
Λ_{HT} – horizontal tail sweep
Λ_W – wing sweep
Θ – wing twist

Synopsis of BLISS

A synopsis of BLISS that also appeared in Agte et al, 1999 is as follows.

BLISS is a method for optimization of engineering systems that separates the system-level optimization from potentially numerous autonomous subsystem optimizations. As shown in Figure 1, it utilizes a system architecture in which design and behavior variables are split into three categories. X-variables are those design variables optimized at the local level and are unique to each particular subsystem. Behavior variables that are output from one subsystem and input to another are designated Y, and the system-level design variables are specified as Z. System-level variables are those shared by at least two subsystems.

[*] The 1998a and 1998b references are also available at http://techreports.larc.nasa.gov/ltrs/

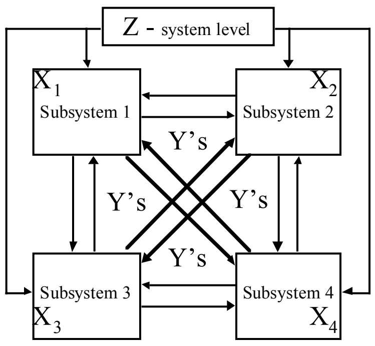


Figure 1: BLISS system structure

After a best guess initialization, the first step in the BLISS begins with the system analysis and sensitivity analysis in which Y and the derivatives of Y with respect to Z and X are computed. A linear approximation to the system objective (an element of Y) as a function of Z and X is established using the above derivatives. That approximation is adopted as the objective function in subdomain optimizations that follow next. In each subdomain (module, or black box), the Z and Y variables are frozen and an improvement in the objective function is sought by the local optimizations that use local X separately in each module. The frozen Z and Y are constant parameters in each module optimization and the module optimization is followed by computation of the derivatives of the optimum with respect to these parameters. The second step achieves improvement through the system-level variables Z and is linked to the first step by the derivatives of optimum with respect to parameters Z and Y. The derivatives are used to extrapolate each subdomain optimum as a function of Z and Y. The functional relation Y=Y(Z) is approximated by extrapolation based on the system sensitivity analysis. These steps alternate until convergence. A flowchart of the method is shown in Figure 2.

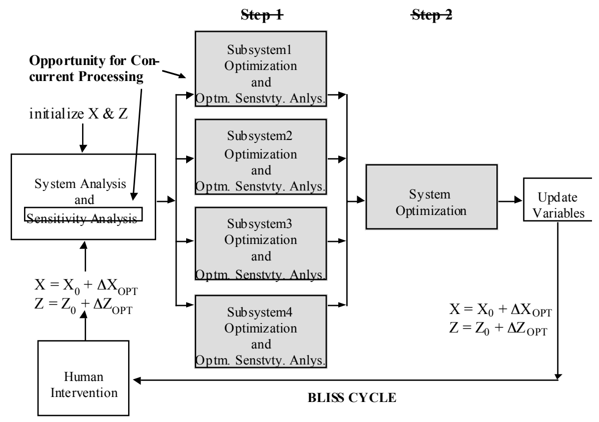

Step 1
Step 2
Opportunity for Concurrent Processing
initialize X & Z
Subsystem1 Optimization and Optm. Senstvty. Anlys.
Subsystem2 Optimization and Optm. Senstvty. Anlys.
Subsystem3 Optimization and Optm. Senstvty. Anlys.
Subsystem4 Optimization and Optm. Senstvty. Anlys.
System Analysis and Sensitivity Analysis
System Optimization
Update Variables
$X = X_0 + \Delta X_{OPT}$
$Z = Z_0 + \Delta Z_{OPT}$
$X = X_0 + \Delta X_{OPT}$
$Z = Z_0 + \Delta Z_{OPT}$
Human Intervention
BLISS CYCLE

Figure 2: BLISS Cycle

Note that the output of step 1 is an optimum change in the local design variables, ΔX_{OPT}, in the presence of constant Z, and the output of step 2 is an optimum change in system design variables, ΔZ_{OPT}.

In the original version of BLISS the modules shown generically in Fig. 2 are Propulsion, Aerodynamics, Structures, and Performance whose detailed input/output variables are identified later. The common denominator of these modules was the extreme simplicity of analyses that employed closed-form expressions for input-to-output mapping. This was so because of the need to test the overall procedure organization and the two-level algorithm convergence at the initial development stage of a new method without being encumbered by long turn-around times in the modules. The next logical step in the BLISS development is to upgrade the fidelity of the modules while holding the overall procedure organization unchanged.

Upgrades in the BLISS Structures and Aerodynamics Modules

The modularity of BLISS permits replacing or adding black boxes to refine or alter the optimization and analysis tools in each modules allowing the engineer the flexibility to exercise his judgment. Having tools of different level of fidelity in the modules enables applications of BLISS in different design phases. The advanced BLISS method incorporates two new modules that can be used in lieu of previous black boxes. The structures module now can use the Equivalent Laminated Plate Solution (ELAPS, Giles, 1986) and the aerodynamics module can use a code called AWAVE (Harris, 1964) to perform wave drag analysis.

Integration of ELAPS

In the previous application example, BLISS employed a skin-stringer representation of the internal wing box bays. This model broke the wing down into a three bay wing box whose geometry varied with the taper ratio, wing sweep, thickness to chord ratio, wingspan, and aspect ratio, all manipulated as design variables in the system-level optimization. The displacements, e.g., the wing twist, and stresses, were computed using simple, thin-walled box-beam formulas (e.g., Bruhn, 1965)

In the BLISS application shown herein, the level of accuracy in this module is raised by substituting the previous model with the Equivalent Laminated Plate

Solution (ELAPS) computer code. This code designed with preliminary design stage calculations in mind is capable of modeling aircraft wing structures with multiple trapezoidal segments. The wing structure is represented as a plate whose stiffness is set equivalent to that of the original, built-up, structural box of the wing. ELAPS employs a set of displacement functions defined over each trapezoidal segment and made compatible in regard to translations and rotations at the segment junctions. Minimization of the strain energy based on the Ritz method leads to equations from which to calculate static deflections and internal forces. The latter are then converted to stresses taking into account the details of the wing box built-up cross-section.

The accuracy of the results of ELAPS has been found to be somewhat below that of finite element codes (Giles, 1986) but the ELAPS input is much simpler and faster to develop. The computation time for an ELAPS model is more than an order of magnitude faster than that of an equivalent finite element model - an important feature for a tool to be integrated into an optimization procedure.

Integrated in BLISS, ELAPS receives its input from a pre-processor routine that generates an input file with the skin thickness, aspect ratio, taper ratio, thickness to chord ratio, sweep, reference area, and aircraft weight. The model used by ELAPS analyzes stress along the same three bay wingbox configuration used as an example in the original application of BLISS. Each wingbox consists of the top and bottom sandwich panels of different thicknesses and sandwich webs identical in the front and rear of the wingbox. The front spar of the wing box is located at 10% of the chord length and the rear spar lies at 70% of the chord length. Figure 3 depicts the configuration of the ELAPS model used by BLISS.

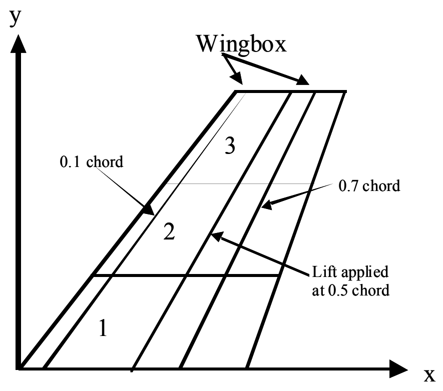


Figure 3: Wing Model

The top and bottom panels as well as the webs have the thickness of the sandwich face sheets (t) and the sandwich caliper thickness (t_s) as design variables, as depicted in Fig.4. ELAPS models such a built-up structure by representing each face and the core as separate elements linked in a common coordinate grid.

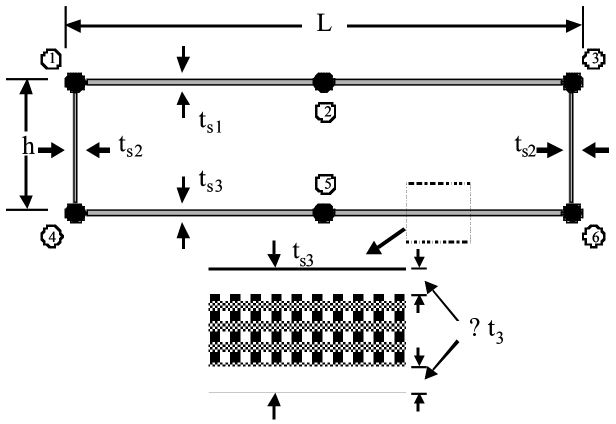


Figure 4: Wingbox Model

As it was done in the original BLISS implementation, the aerodynamic loads are being generated within the structures module in the pre-processor to structural analysis. To calculate the lift loads on the wing, the pre-processor routine averages spanwise between an elliptical lift distribution and a linear distribution that reflects the wing chord taper. The elliptical and taper ratio based lift distributions for the wing are each normalized to contain an area of unity as illustrated in Figure 5. The averaged, spanwise load distribution is multiplied by the lift required from the wing and distributed chordwise. The chordwise distribution is a typical supersonic one with the center of pressure located at 50% of the chord. The aerodynamic load distribution would be expected to be calculated by an aerodynamics module using a higher fidelity analysis, e.g., a computational fluid dynamics code. Thus, the present aerodynamic loads generation is merely a placeholder for a real aerodynamic loads analysis in a future BLISS upgrade.

In summary, the structural module employs ELAPS to calculate the stresses in the wing box for the given configuration, lift distribution, and corresponding constraints. It also outputs the wing twist and weight and the objective function for the local optimization. The aerodynamics module accepts the output and models its influence on the aerodynamic response.

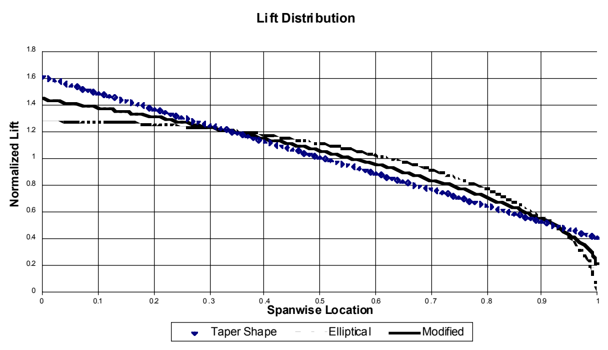


Figure 5: ELAPS Lift Distribution

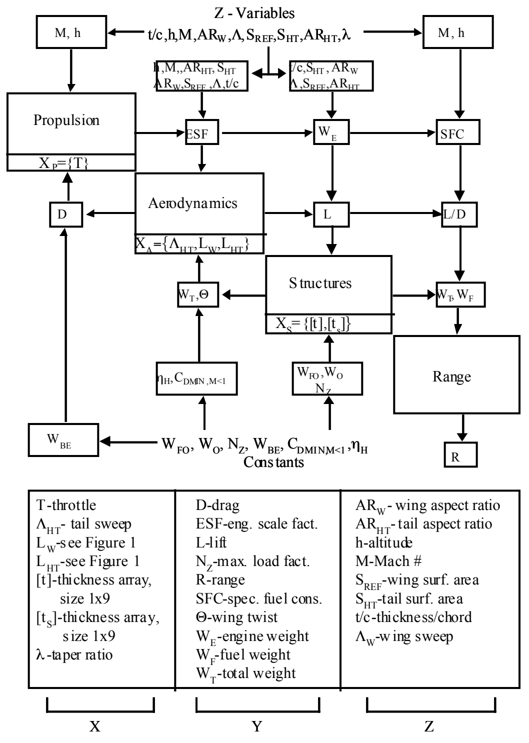
Integration of AWAVE

The cruise segment of the test case mission is supersonic. The original model (Sobieszczanski-Sobieski et al, 1998) used an approximation relying on the span efficiency factor. That approximation was replaced herein with a code, AWAVE, that is a streamlined version of the far-field wave drag program (Harris, 1964). There are two versions of the Harris wave drag program in common use at LaRC. The original version, described in the reference, treats lifting surfaces as a series of 3-dimensional solid elements. A much faster but slightly less accurate version treats lifting surfaces as 2-dimensional panels with finite thicknesses. Due to compensating errors at positive and negative roll angles of the Mach cutting plane, the panel version gives excellent results for wave drag coefficients. The objective of the last (AWAVE) effort was to develop a version of the wave drag program with the accuracy of the solid element program that is faster than the panel version. The AWAVE code implemented computes the wave drag on the basis of the aircraft cross-section distribution along the centerline, hence it requires data about the entire configuration geometry to enable the area ruling of supersonic body design.

Similarly to the integration of ELAPS, integration of AWAVE was accomplished by creating a pre-processor to generate the necessary input. The input provides the current design's aspect ratio, taper ratio, thickness to chord ratio, sweep angle, wing reference area, horizontal tail sweep angle, horizontal tail aspect ratio, and horizontal tail reference area. The pre-processor also creates and places the wing and tail airfoils according to the design configuration variables. The AWAVE output is the wave drag coefficient to be added to the other drag components whose calculation remains the same as in the original BLISS.

Numerical Implementation

Compared to the original application of BLISS to the supersonic business jet case, incorporation of ELAPS and AWAVE in BLISS required some changes to constraints and allocation of the design variables to the system and subsystem levels.


Z - Variables
t/c,h,M,AR_W,Λ,S_{REF},S_{HT},AR_{HT},λ
M, h
M, h
h,M,,AR_{HT}, S_{HT} AR_W,S_{REF} ,Λ,t/c
t/c,S_{HT} , AR_W Λ,S_{REF},AR_{HT}
Propulsion
X_P={T}
ESF
W_E
SFC
Aerodynamics
$X_A=\{\Lambda_{HT},L_W,L_{HT}\}$
D
L
L/D
Structures
$X_S= \{[t],[t_s]\}$
W_T,Θ
W_T, W_F
η_H,$C_{DMIN ,M<1}$
W_{FO},W_O N_Z
Range
R
W_{BE}
W_{FO}, W_O, N_Z, W_{BE}, $C_{DMIN,M<1}$,η_H
Constants
X
Y
Z

X	Y	Z
T-throttle	D-drag	AR_W- wing aspect ratio
Λ_{HT}- tail sweep	ESF-eng. scale fact.	AR_{HT}- tail aspect ratio
L_W-see Figure 1	L-lift	h-altitude
L_{HT}-see Figure 1	N_Z-max. load fact.	M-Mach #
[t]-thickness array, size 1x9	R-range	S_{REF}-wing surf. area
$[t_S]$-thickness array, size 1x9	SFC-spec. fuel cons.	S_{HT}-tail surf. area
λ-taper ratio	Θ-wing twist	t/c-thickness/chord
	W_E-engine weight	Λ_W-wing sweep
	W_F-fuel weight	
	W_T-total weight	

Figure 6: Data Dependencies for Business Jet Model

In the original BLISS, the taper ratio was a local variable of the structures module. With the integration of ELAPS and AWAVE, the taper ratio affects both the aerodynamics and structures module. While the aerodynamics module optimization may tend toward a taper ratio to reduce induced drag, the structures module may generate a different taper value to reduce stresses. To resolve this trade-off, the taper ratio was raised to a system variable, capable of influencing both modules. Figure 6 shows the current black box and variable interactions.

In this model there are nine system-level Z-variables, each influencing a minimum of two of the subsystems. The local variables of each subsystem are manipulated only in the optimization local to that subsystem. The propulsion module has the throttle as its sole local variable. In the present state of BLISS, the range module is an exception as it performs no optimization. It only evaluates the Breguet range formula. The aerodynamics module optimizes the local variables of the horizontal tail sweep as well as the variables that place the wing and tail along the fuselage axis. The structural subsystem optimization operates on the sandwich face sheet and caliper thicknesses for the wing cover panels and the webs of the three wingbox bays.

The ten Y-variables noted in the off-diagonal boxes in Figure 6, represent couplings of the black boxes and are computed in the system analysis.

BLISS was originally implemented in MATLAB 5.3.0 at both the system and subsystem levels. The MATLAB Optimization Toolbox was used as an optimizer in the subsystem and system optimizations. In the version of BLISS reported herein, the use of MATLAB continued as above with the exception of the structures and aerodynamics modules that incorporated ELAPS and AWAVE, both written in FORTRAN 77.

MATLAB provides a facility to invoke FORTRAN from a MATLAB code. To exploit that facility, the preprocessors to both ELAPS and AWAVE were written in FORTRAN and converted into MEX-Files using the MATLAB mex-function (Appendix). Both AWAVE and ELAPS were then directly called from within the BLISS modules. On the output side, simple post-processing generated outputs in a format acceptable to the parts of BLISS that remained being coded in MATLAB for further analysis. Because of the MATLAB ability to invoke FORTRAN codes, the BLISS upgrading process may continue by adding FORTRAN-coded modules wherever required while retaining the MATLAB core that executes the method logic illustrated by the flowchart in Fig. 1.

Results

BLISS iterations terminate when the change in the aircraft range objective varies less than ten nautical miles. This took seven passes through the flowchart in Figure 2. The system-level design variables converged within the first few passes. Further optimizations focused primarily on the local variables. Most of the changes occurred within the structures module where the new ELAPS-based optimization kept refining the variables searching for the best solution. The majority of the computational time was spent in this module. Table 1 shows the variable progression through the optimization process.

The table reflects the major trade-offs that occur between the wing sweep angle, airfoil thickness ratio, and the wing aspect ratio, all of which govern the structural weight and drag that, in turn, influence the range. Ultimately, influences of these variables on the range differ in sign, therefore, the procedure seeks a compromise. For example, the wing sweep initially increases to approximately 70 degrees and then falls to 40 while the taper ratio decreases to 0.1. The wing reference area rapidly reduces to 200 square feet as the wing aspect ratio is brought down first to 2.5 and then increased to 2.607. The wing position is briefly changed in the fourth cycle but quickly returns to its initial value. The wing configuration progression is depicted in Figure 7. The aircraft finds its optimal cruise conditions after the first cycle of Mach 2.0 at 60,000 feet.

var\cycle	1	2	3	4	5	6	7	8
Range (NM)	1051	4492	3938	3546	3399	2432	2493	2493
t_1 (inner)	2	0.617	1.326	0.944	0.643	0.192	0.192	0.192
t_1 (middle)	2	0.301	0.624	0.637	1.028	0.663	0.663	0.663
t_1 (outer)	2	0.383	0.53	0.322	0.457	0.044	0.044	0.044
t_2 (inner)	2	12	7.449	5.573	3.089	0.012	0.012	0.012
t_2 (middle)	2	12	2.803	0.909	8.864	0.012	0.012	0.012
t_2 (outer)	2	12	0.6	0.012	12	8.115	8.115	8.115
t_3 (inner)	2	0.48	0.729	0.824	0.173	1.075	1.075	1.075
t_3 (middle)	2	1.348	0.562	0.511	0.548	0.517	0.517	0.517
t_3 (outer)	2	0.115	0.267	0.248	0.265	0.012	0.012	0.012
t_{s1} (inner)	4	0.617	1.325	0.944	1.985	2.205	2.205	2.205
t_{s1} (middle)	4	0.865	0.624	0.637	1.028	0.663	0.663	0.663
t_{s1} (outer)	4	0.383	0.53	0.322	0.617	0.233	0.233	0.233
t_{s2} (inner)	4	24	24	23.17	12.46	3.466	3.466	3.466
t_{s2} (middle)	4	24	0.464	0.665	23.02	14.31	14.31	14.31
t_{s2} (outer)	4	24	2.583	1.802	24	2.229	2.229	2.229
t_{s3} (inner)	4	0.48	0.37	0.677	0.177	1.076	1.076	1.076
t_{s3} (middle)	4	1.348	0.562	0.511	0.068	0.391	0.391	0.391
t_{s3} (outer)	4	0.115	0.267	0.248	0.38	0.251	0.251	0.251
Λ_{HT} (°)	60	70	70	70	40	40	40	40
L_W (%MAC)	10	1	1	2	1	1	1	1
L_{HT} (%MAC)	250	350	350	100	350	100	100	100
T	0.35	0.319	0.236	0.241	0.255	0.281	0.31	0.31
t/c	0.05	0.058	0.058	0.058	0.058	0.058	0.058	0.058
h (ft)	55000	60000	60000	60000	60000	60000	60000	60000
M	1.8	2	2	2	2	2	2	2
AR_W	4	2.5	2.607	2.607	2.607	2.607	2.607	2.607
Λ_W (°)	55	40	40.63	40.63	40.63	40.63	40.63	40.63
S_{REF} (ft^2)	400	200	200	200	200	200	200	200
S_{HT} (ft^2)	150	150	150	150	150	150	150	150
AR_{HT}	6.5	8.5	8.5	8.5	8.5	8.5	8.5	8.5
λ_W	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Table 1: Supersonic Business Jet Results

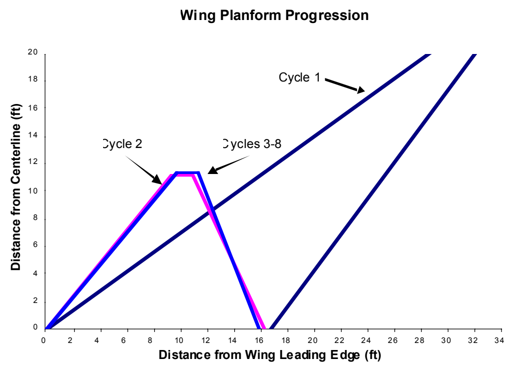


Figure 7: Wing Planform Progression

The horizontal tail position and geometry stabilize after the main wing variables reach their settling points. The tail position varies significantly but settles at a value of 100 percent of the mean aerodynamic chord. The tail sweep ends up almost matching the wing sweep but has a significantly larger aspect ratio. Further analysis of the tail may involve incorporating an ELAPS model of the tail to increase fidelity of analysis in that component.

The skin thicknesses change throughout the process seeking the maximum of the structure contribution to the range under the stress constraints for the given configuration geometry, the latter governed by the Z-variables. The resulting histogram is seen in Figure 8.

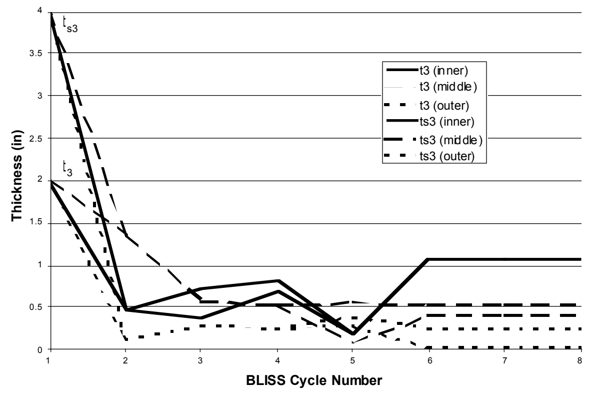


Figure 8: Plot of Skin Thickness Variation

Though the first cycle was able to converge to reasonable thicknesses, the second through fifth cycles were unable to satisfy all constraints given the system-level configuration. Then, by the sixth cycle the optimizer had found a solution that allowed all constraints to be met and in the seventh cycle it found the optimal configuration. Figure 9 shows the progression of the aircraft Take-Off Gross Weight and its components of empty weight and fuel.

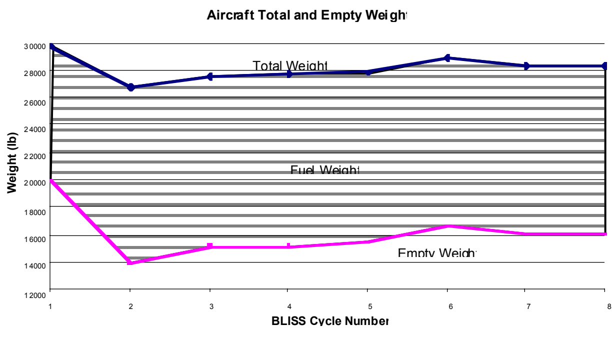


Figure 9: Aircraft Weight

Figure 10 depicts a histogram of the aircraft range. It starts off at a feasible design point. The cycles two through five did not lie within the design space, but BLISS returned to the design space and settled on a feasible design with optimized range.

The last implementation of BLISS to the supersonic business jet test case (Agte, 1999) yielded a range of 2,189 nautical miles. With the addition of AWAVE and ELAPS, the more refined analysis increased the range to 2,493 nautical miles.

Distribution of elapsed computing time over the BLISS modules is displayed in Table 2. It is evident that most of the elapsed time is spent in ELAPS but that would change drastically if a CFD-level analysis were used in the aerodynamics module. If BLISS were grown to the point where all the major modules would consume about equal amount of the elapsed time, then distributed execution on concurrently operating machines (or processors within a multiprocessor machine) would radically compress the elapsed time of the entire BLISS execution.

	BLISS	ELAPS	AWAVE
Percent of Time	8.36%	89.78%	1.87%

Table 2: Processor Time Use

The next step in the BLISS development is to incorporate additional modules to increase the analysis fidelity. The largest refinement would be expected from adding a computational fluid dynamics code to perform the aerodynamic analysis, including the loads. The propulsion data quality would benefit from replacing the current response surface fitted to a look-up table with a comprehensive engine analysis. Also, the simple Breguet formula for the aircraft range would need to be replaced by a complete performance analysis

Conclusions and Remarks

Integration of ELAPS and AWAVE into BLISS demonstrated the modular nature of the method and its ability to accommodate refinements. Used in a limited test case of a supersonic business jet design, the two-level optimization in BLISS was effective in satisfying the system-level and local constraints while attaining a system-level objective within a reasonable number of iterations. Separation of the system-level design variables from the local ones enabled optimization for a system-level objective while providing autonomy of the design decision and tool choice within disciplines represented in the modules.

The method is open to further upgrades in terms of the fidelity of analysis and optimization techniques employed in the modules. In this regard, it is up to the user to decide on the variety of tools to be integrated in BLISS as needed by the multidisciplinary optimization task at hand.

Further advancement of BLISS from its present status of a method concept demonstrator to a tool useful in actual applications calls for inserting a CFD code in the aerodynamics module, adding a comprehensive engine analysis to the propulsion module, and extending the performance analysis module to include more than just the cruise phase of a mission.

Finally, as the increased fidelity of analyses in the modules will exact its price in terms of the computing elapsed time, concurrent execution of the modular analyses and optimizations will become an attractive option.

Acknowledgement

Michael Parks of the George Washington University JIAFS program provided critical help with MATLAB refinements and FORTRAN integration. His contributions were gratefully appreciated.

References

Agte, J.; Sobieszczanski-Sobieski, J.; and Sandusky, R.: "Supersonic Business Jet Design Through Bi-Level Integrated System Synthesis," 1999 World Aviation Conference, SAE 1999-01-5622.

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

Bloebaum, C. L.; Hajela, P.; and Sobieszczanski-Sobieski, J.: "Non-Hierarchic System decomposition in Structural Optimization; Engineering Optimization," 19, pp 171-186, 1992.

Braun, R. D., and Kroo, I.M.: "Development and Applications of the Collaborative Optimization Architecture in a Multidisciplinary Design Environment," SIAM Journal on Optimization, Alexandrov, N, and Hussaini, M. Y., (eds), 1996.

Bruhn, E. F.: "Analysis and Design of Flight Vehicle Structures," Tri-State Offset Co., 1965.

Giles, G. L.: "Equivalent Plate Analysis of Aircraft Wing Box Structures with General Planform Geometry," NASA TM-87697, March 1986.

Harris, R. V.: "An Analysis and Correlation of Aircraft Wave Drag," NASA TMX-947, March 1964.

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

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

Raymer, D. P.: *Aircraft Design: A Conceptual Approach*, AIAA Education Series, 1992.

Sobieszczanski-Sobieski, J.: "Optimization by Decomposition: A Step from Hierarchic to Non-hierarchic Systems," 2nd NASA/AF Symposium on Recent Advances in Multidisciplinary Analysis and Optimization; Hampton; Virginia, Sept. 1988; NASA CP-3031, pp.51-78; also in NASA TM 101494.

Sobieszczanski-Sobieski, J.; Agte, J.; and Sandusky, R.: "Bi-Level Integrated System Synthesis," Proc. 7th AIAA/USA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA Paper No. 98-4916. Also NASA TM-1998-208715, Sept. 1998a; (to appear in AIAA J. Jan. 2000)

Sobieszczanski-Sobieski, J.; Agte, J.; and Sandusky, R.: "Bi-Level Integrated System Synthesis (BLISS)", NASA/TM-1998-208715, August 1998b.

Wujek, B. A.; Renaud, J.E.; Batill, S. M.; and Brockman, J.B.: "Concurrent Subspace Optimization using Design Variable Sharing in a Distributed Design Environment," Azarm, S. (ed), Proc. Design Engineering Technical Conf., Advances in Design Automation, ASME DE Vol. 82, pp.181-188, 1995.

Appendix

There are a few steps required for integrating a FORTRAN code into the MATLAB environment that BLISS is currently programmed in. The user must locate the place in BLISS where a call is made to analysis that is to be replaced. Then he must examine the input and output of the new and old analysis to ensure that the remainder of BLISS is capable of supplying input the new analysis requires and that the new analysis produces all the output expected. Then a pre-processor routine must be created to present the variable information to the FORTRAN code in an appropriate format. Finally the data must be harvested and returned to the MATLAB module from which the FORTRAN code is being called in a format compatible with MATLAB.

Pre-processing

In pre-processing, BLISS must pass the needed variables to a routine which will manipulate them into a form that the FORTRAN code will accept. The programmer must first ensure that the module in which he is pre-processing has access to the required variables (i.e., the subsystem must not be using X variables assigned to other subsystems).

The two FORTRAN codes integrated required formatted text input files. The easiest way to prepare these files was to use a FORTRAN subroutine to take the design variables, configure them in the way needed to represent the geometry or conditions needed by the analysis, and write the formatted input file for analysis.

In order to pass variables from the MATLAB environment to the FORTRAN pre-processing subroutine, MATLAB's mex-function was invoked. In this function, a standard gateway subroutine is added to the FORTRAN pre-processing subroutine. This new routine collects an array of variables passed in from MATLAB and assigns them to a FORTRAN array. The gateway routine sends these variables into the pre-processing subroutine.

In MATLAB, the user compiles the FORTRAN code including the gateway routine and the pre-processing routine using the mex command. This creates a mex-file which is treated as a MATLAB function requiring an input array and an output array. The user then places his variables for the pre-processing function into an array and puts this array into the new mex-file. This sends the variables to the gateway routine which assigns them to the variables used in the FORTRAN pre-processing subroutine. The input file is generated and control returns to MATLAB.

Program Insertion

Having prepared the data for analysis by the program, the programmer must locate the section of BLISS that he wishes to upgrade. The previous analysis must be removed and the code must be placed such that BLISS will have performed the new analysis and have data ready for later analysis that the user is not replacing. Having located the desired calling spot and removed the replaced analysis, the user simply calls the program from within BLISS. By previously compiling the FORTRAN code in question, the user calls the program by typing *!program_name* in the BLISS code where *program_name* is the command that runs the program from the operating system. The program then processes the prepared input file and returns to BLISS.

Data Collection

The final step that the programmer must perform in order for BLISS to carry on its optimization is harvesting the data produced by the new program. ELAPS and AWAVE both created output files with data required for BLISS. There are two basic ways to collect data produced by FORTRAN codes.

The first way is to use a post-processing technique similar to that of pre-processing. The user would create a search algorithm to locate and collect the data from the output file. This in turn would be harvested by using the mex-function to create a gateway between the FORTRAN data collection routine and the BLISS variables. The programmer would compile the mex-function gateway routine combined with his data collection routine, run the new mex-file with an array prepared to collect the output of the routine, and extract his data to the array. The programmer would then need to assign the array variables to the proper variables in the BLISS code. This technique would be best for cases where the user did not have access to the source code of the FORTRAN program.

In cases where the programmer does have the FORTRAN program's source code and a fair knowledge of how the program works, he can edit the code to output the needed results in a format compatible with MATLAB. If the user can locate the sections of code that write out the results to the output file, they can change the code to output to a file with a .m extension. Files in this format are recognized as MATLAB programs that can be called without FORTRAN interaction. By creating .m files with MATLAB variable assignments corresponding to the data that the user wishes to collect, the programmer simply runs these MATLAB files after completion of the main program call and the output is already in MATLAB format. This avoids unnecessary data file searching and reduces MATLAB-FORTRAN interaction. After the data are collected and assigned to the proper variables in BLISS, the analysis would then continue as normal.