A GENERIC FRAMEWORK FOR MULTI-PARAMETER OPTIMIZATION OF FLIGHT TRAJECTORIES

Kenneth Chircop*, Matthew Xuereb*, David Zammit-Mangion, Ernest Cachia***
***University of Malta, **Cranfield University**

Keywords: *trajectory optimization, multidisciplinary design optimization, integration framework, optimization framework*

Abstract

This paper presents the requirements and design concept of a multi-parameter optimization tool to be used on flight trajectories. The tool, referred to as GATAC, is being developed as part of the EU-funded Clean Sky Joint Technology Initiative (JTI) programme, and a preliminary version is discussed in this paper. The tool has been evaluated and the results obtained confirm the validity of the tool, opening the way for further development. The paper also addresses the architectural design and a number of key features of the tool.

1 Introduction

Air transport currently depends entirely on fossil fuels and mainly on gas turbine technology for propulsive means. This implies that every flight contributes to the usage of a finite resource, to the emission of green-house gases and secondary emissions (mainly NOx) and also to noise pollution.

Several initiatives are being undertaken by the aviation industry to reduce the impact of air transport on the environment and on fuel resources. The European Commission (EC) is at the forefront of directing research into this area by bringing together the aviation industry through the JTI Clean Sky to address this problem. Clean Sky is a JTI that is developing breakthrough technologies to significantly reduce the impact of air transport on the environment [1].

Clean Sky is addressing six technological domains that contribute to meeting its objectives through Integrated Technology Demonstrators (ITDs). The six ITDs are Smart Fixed Wing Aircraft (SFWA), Green Regional Aircraft (GRA), Green Rotorcraft (GRC), Sustainable and Green Engines (SAGE), Systems for Green Operations (SGO) and Eco-Design (ED).

The SGO ITD will generate technologies for improved aircraft operation through two major activities, namely the management of aircraft energy (MAE) and the management of trajectory and mission (MTM). The MAE activity encompasses all aspects of on-board energy provision, storage, distribution and consumption. The MTM activity comprises the management of aircraft trajectories based on more precise, reliable and predictable three-dimensional flight paths optimized for minimum noise impact and low emission under operational constraints. Specifically, it deals with aircraft missions and with the management of new climb, cruise and descent profiles. Within MTM, a trajectory optimization tool that is referred to as GATAC is being developed. This will be used with state-of-the-art optimizers and simulation models to perform multi-objective optimization of flight trajectories under Air Traffic Management (ATM) constraints in order to reduce the environmental impacts caused by aircraft in all flight phases, with particular emphasis on the reduction of fuel consumption, carbon dioxide (CO_2), nitrogen oxides (NOx) emissions and perceived external noise. These achievements will contribute towards the environmental goals set by the Advisory Council for Aeronautics Research in Europe (ACARE), which include, amongst others, reductions of CO_2 emissions by 50% and those of NOx emissions by 80% by 2020 [2].

2 Framework Requirements

Celis et al. [3] classify the trajectory optimization problem as constrained, dynamic, optimal control, nonlinear, real-valued, and multi-objective. In addition, the problem can also be defined as multi-modal, as the space is unknown but it is assumed that there are several local minima (or maxima). It can also be defined as multi-dimensional, since several parameters will be involved during the optimization process. The trajectory optimization problem therefore requires complex, computationally-intensive mathematical operations requiring the contribution of a number of discipline specialists to solve a particular problem.

Conventionally, the optimization problem is solved by merging the inter-disciplinary simulation codes into a single computer program customized for a specific problem. This approach has two major disadvantages, the first being the lack of flexibility when attempting to modify the software to solve a different problem. Significant coding effort is required to simply modify the objective function, with even more effort being required if a completely different problem is to be formulated. The second disadvantage is in the merging of the code due to the fact that different specialists program in different ways, and moreover, program in different languages. Therefore, the merging of the code sourced from different disciplines into a single computer program normally requires major code translation and adaptation. A more flexible approach allowing inter-discipline codes to be integrated together with minimum effort while giving equivalent or better optimization results, would be very advantageous to the engineering community.

The development of Multi-Disciplinary Optimization (MDO) frameworks or problem solving environments offers the capability to meet these needs via the use of sophisticated computational procedures combined with state-of-the-art optimization or design improvement techniques [4]. As a minimum requirement, a functional MDO framework must support the integration of disciplinary analysis codes and tools used for facilitating inter-disciplinary tradeoffs [5]. The requirements for such a framework, as presented by Salas and Townsend [4], can be divided into four sections; architectural design, problem formulation, problem execution and information access. The authors are of the opinion that an additional section needs to be included to list the requirements of the programming language of choice for the development of the framework. This should be done to ensure that the programming language has the essential capabilities to develop the required framework.

A detailed but non-exhaustive list of framework requirements is presented in the following sections, using as a basis, the work presented by Salas and Townsend [4]. Extensive material can be found in the literature on requirements for MDO frameworks. In this paper, only the requirements that are relevant to the development of GATAC are presented.

2.1 Architectural Design Requirements

The major architectural design requirements are listed below. The GATAC framework:

1. shall provide an intuitive GUI that enables the user to perform all tool operations and functionalities without the need of any low-level programming.

2. shall provide an easily implementable flow diagram that facilitates the visualization of the data flow between models/modules in the system. Ideally, a complex problem formulation designed from scratch should not take more than one working day to setup, as suggested in [6].

3. shall be extensible to support the integration of new models/modules into the system.

4. should not impose a significant amount of overhead on the optimization execution time.

5. shall be able to handle large complex problems with a large number of variables and constraints.

6. shall provide flexible parsing tools to handle different input and output file formats enabling the ease of integration of simulation models.

7. should provide array processing capability to enable simpler flow diagrams when the problem formulation is divided into a number of phases/segments. A single dimensional array can then be associated to a variable/constraint with the individual array elements associated to the phase or segment.

8. shall provide robust performance and error handling capability.

9. shall be modular in nature, with modules designed to be problem independent allowing the tool to be used for any MDO formulation.

10. should provide intuitive tools for integrating simulation codes into the framework. These tools are intended to ease the creation of wrappers around simulation models and enable model specific configuration of input and output file parsers.

11. shall provide for efficient transfer, storage and data access in line with reference [7].

2.2 Problem Formulation

The major requirements associated with problem formulation are listed in this section. The GATAC framework:

1. shall not require the user to be an optimization specialist, allowing the user to focus on the problem at hand.

2. shall not require the user to have specific IT skills beyond the usual aeronautics engineer's level of software literacy.

3. shall allow the user to configure complex branching and iterative MDO problem formulations easily without low-level programming.

4. shall allow the formulation of multi-parameter, single or multi-objective optimization problems.

5. shall allow users to store, retrieve and easily reconfigure existing MDO problem formulations.

6. shall support the user in incorporating legacy codes (written in a variety of languages), proprietary codes (where the source is not available), commercial off-the-shelf (COTS) software, spreadsheets and databases stored locally or remotely accessible via a Local Area Network (LAN) into the MDO formulation.

7. shall allow easy integration of user-defined optimization techniques (gradient-based, simulated annealing, genetic algorithms and hybrid methods amongst others).

8. shall be operating system independent, allowing the software to run on Windows and Unix-derived operating systems.

2.3 Problem Execution

The major requirements associated with problem execution are listed in this section. The GATAC framework:

1. shall automatically execute processes, invoke models and enable the movement of data.

2. shall be able to execute multiple simulation models concurrently on multi-core machines and/or over a LAN of heterogeneous computers to speed up the optimization process.

3. should be able to apply intelligent load-balancing across the available processing resources, both at a multi-core machine level and on distributed hosts over the network.

4. should support user steering during the design cycle, enabling the user to modify the problem definition during the optimization process to aid convergence.

5. shall allow the user to retrieve results from previous executions and use them to provide a good initial condition for a new run.

6. shall allow the user to launch a case as a batch job from the command line.

2.4 Information Access

The major requirements associated with information access are listed in this section. The GATAC framework:

1. shall provide effective support for database management through a Structured Query Language (SQL) interface for data storage, access and manipulation.

2. shall maintain a historical database of problem formulations with corresponding results.

3. shall provide the capability to post-process results.

4. shall provide the capability to visualize intermediate, final and post-processed optimization results (design variables, objectives, constraints, response surfaces, Pareto curves and surfaces).

5. shall store the optimization problem definition and case related setup in eXtensible Markup Language (XML) files.

6. shall provide a monitoring capability for viewing the status of an execution.

7. should provide debugging information at preset levels of detail stored in log files and/or displayed on screen.

2.5 Programming Language

The requirements of the programming language chosen are the following. The language:

1. shall support object-oriented programming to benefit from ease of implementation and extension of software in a modular fashion as suggested in reference [8].

2. shall support accurate mathematical modeling.

3. shall provide database connectivity.

4. shall allow ethernet connectivity and remote procedural calls for distributed computing.

5. shall support visualization technologies.

6. shall support the creation of windows-based graphical user interfaces.

7. shall support the creation of wrappers for simulation code written in other programming languages.

8. shall be platform independent.

3 GATAC Framework Design and Implementation

The GATAC framework was designed specifically to satisfy all the requirements listed in Section 2. A top-down approach was followed in the design phase and the resulting optimization and integration framework design is shown in Fig. 1.

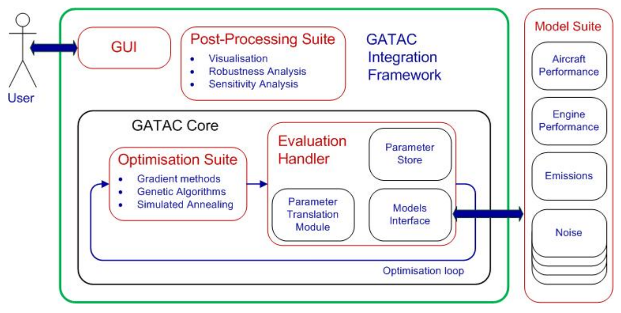


Fig. 1. GATAC Integration Framework Architecture

The architecture is made up of four system-level components, namely, the GATAC Core, the GUI, the Post-Processing Suite and the Model Suite. From a user perspective, the GATAC Core is where the optimization process takes place. The Model Suite services the GATAC Core on request by executing models and transferring data to the GATAC Core. The Post-Processing Suite post-processes the results of the optimization. Finally, the GUI provides a graphical user interface for setting up and using the other three components.

Three object-oriented languages were short-listed as candidates for the development of GATAC, namely Python, C# and JAVA. Although all three languages are powerful and widely used languages, shortfalls that might compromise the quality, performance or objectives of GATAC were identified in both Python and C#. On the one hand, Python is a language which helps the developer program quickly, but it is not strongly typed. GATAC is a large project, and a strongly typed language would be more suitable for the task. Another drawback of using Python is that the code is less readable and hence less maintainable. C# is only portable on machines running the .NET environment. This would mean that the GATAC Integration Framework written in C# would have to run exclusively on workstations with an underlying Microsoft Windows operating system. As GATAC is intended to be platform independent, C# could not be adopted. As a result, the only remaining candidate language is JAVA which is a fast, portable language satisfying all the requirements listed in Section 2.5. Hence, JAVA was chosen for the entire development of GATAC.

3.1 GATAC Core

The GATAC Core is the heart of the optimization tool. It is made up of an Optimization Suite and an Evaluation Handler.

The Optimization Suite is the module that defines the values of the variable parameters and analyses the resulting constraints and criteria values. The suite is further broken down into an Optimization Technique Suite, an Objective Handling Module and a Constraints Handling Module.

The Optimization Technique Suite hosts the optimization algorithms, allowing the user to select particular optimization techniques for specific MDO problems at hand. The Optimization Core provides a generic interface that allows the integration of optimization algorithms seamlessly using software reflection techniques. The optimization algorithms that can be incorporated vary from modern evolutionary-based algorithms to classical numerical techniques.

The algorithms making up the Optimization Technique Suite must be implemented such that, for any problem formulation, the user can select any algorithm from the suite and run the optimization process without modifying the problem formulation. To cater for this need, the Optimization Technique Suite processes only normalized variable parameters. The normalized parameters are then denormalized by a software component inside the Optimization Core for use in the Evaluation Module. Similarly, data flowing from the Evaluation Module to the Optimization Core must be normalized. During the problem formulation phase, the user must program the lower and upper limits for all the variable parameters to allow the normalization and denormalization processes to operate effectively.

The data flow between the Optimization Core and the Evaluation Handler takes place in the form of data structures called Model Parameter Carriers (MPCs). In an optimization process, the optimizer generates variable parameters that are input to simulation models, from which the resultant value of the objective function can be calculated. In a single optimization iteration, a classical numerical technique would provide a set of variable parameters which are used to populate a single MPC. This MPC is then processed by the Evaluation Module and the resultant data is sent back to the Optimization Core in another MPC. In modern evolutionary techniques such as genetic algorithms, a whole population of chromosomes is created in a single optimizer iteration. In this case, the Optimization Core transfers a list of

MPCs to the Evaluation Handler (Fig. 1). If all the models required for the computation of the MPCs are available on a number of host computers in a networked system, the Evaluation Handler is able to request the execution of a number of MPCs concurrently on different machines. Alternatively, the execution of a single MPC can be distributed according to the available resources.

The Objective Handling Module and the Constraints Handling Module evaluate the performance of the solution for the defined objectives and assess the defined constraints to ensure they are met respectively. The interaction between these two modules and the optimization algorithm drive the direction of the optimization process.

The Evaluation Handler is the unit that handles the models and controls the direct data transfer between the models in the Model Suite, the Optimization Suite and other modules in the GATAC Integration Framework. This module consists of a Parameter Translation Module, a Parameter Store and a Models Interface. The Parameter Translation Module performs conversion of parameters from the format used by the user or the models into a format required by the Optimization Core and vice-versa. The Parameter Store provides a mechanism for storing optimization data with the associated problem formulation data in a database for later retrieval. The Models Interface provides the ability to communicate with the Model Suite, which can be distributed on a number of computers.

The Evaluation Handler has three functions within GATAC. Firstly, it controls the data flow of a problem between its internal sub-modules and the Optimization Core during the optimization phase. Secondly, it performs simple mathematical calculations on the model data to create composite parameters as defined in the problem formulation. Lastly, it manages and has full authority on the simulation models residing in the Model Suite. A model version control system is implemented and this provides the mechanism for managing the different models both from a functional and an implementation perspective in the Model Suite. On request from the Optimization Core, the Evaluation Handler invokes the relevant model through the Models Interface to obtain the required data. The task of invoking models is done in an intelligent manner implementing load balancing techniques across the processing power available to the framework. Initially, invocation of models that are available on a single busy host are delayed until the host is ready to process the jobs. If a model is replicated on a number of hosts, it is invoked on the highest performance idle host. If a number of models are required to run on a single host, priorities are allocated specifying which runs first. Then, multi-core processing power on machines with this capability is exploited by invoking a number of models concurrently according to the number of cores available.

3.2 Model Suite

The GATAC software is designed to run either on a single stand-alone machine or a distributed system with multiple computers. In its simplest setup, a stand-alone system runs on a single machine that hosts both the GATAC Integration Framework and the Model Suite. When set up on a distributed system, the GATAC Integration Framework resides on a machine henceforth called the Central Server. The Model Suite is then distributed on one or more further machines acting as hosts (Fig. 2).

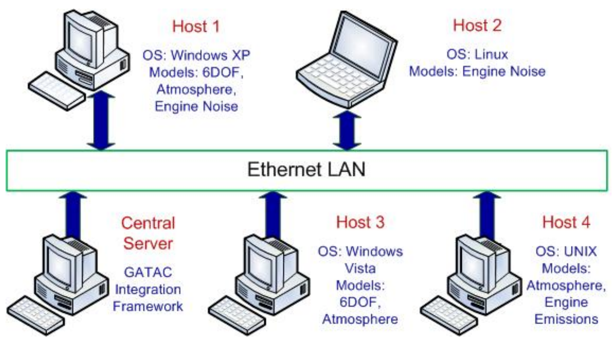


Fig. 2. Distributed System

The Model Suite is made up of the models that, in a distributed system, are distributed across the ethernet LAN available to the GATAC Integration Framework. The models being hosted by computers other than the Central Server cannot be invoked directly by the GATAC Integration Framework. For this purpose, a daemon, which is a small computer program running in the background, is installed and run on every host.

The daemon provides a common model backend to the Models Interface and invokes models as instructed by the Evaluation Module. It facilitates data transmission between the Models Interface and the Model Suite by translating data output from the models to a common structure that is transmitted on the LAN and vice-versa. The daemon is designed to support a wide range of different model implementations, including standalone programs (executables, dynamic linked libraries, etc), databases, spreadsheets and text file models. Moreover, the models can be written in a large number of different programming languages, such as, Fortran, C, Ada and can have a range of interfaces (XML, IATA SCAP, model specific text files).

In order to provide this functionality, the daemon provides a user-programmable wrapper for every model plugged to the system (Fig. 3). The daemon receives MPCs from the Evaluation Handler, disassembles them and feeds them as arguments to the models. Similarly, the data output from the models is assembled into MPCs before it is transmitted over the LAN to the Evaluation Handler. The connection between the model wrapper and the Models Interface is implemented using an Ethernet LAN through Remote Method Invocation (RMI).

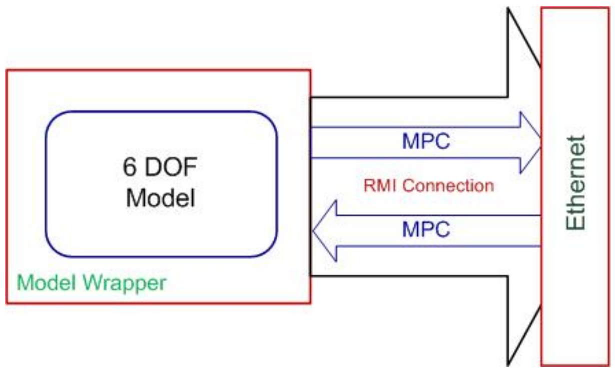


Fig. 3. Model Interface

3.3 Post-Processing Suite

The Post-Processing Suite enables the designer to analyze the solutions obtained from the optimization process. A number of tools, some of which are specific to trajectory optimization, will populate the solution analysis suite. Nevertheless, the user is given the flexibility of introducing additional tools to the suite without significant effort due to the extensible design feature of the suite. Some of the tools that will be provided with the software package include visualization, robustness and sensitivity analysis. Visualization capability facilitates the illustration of the cost function, convergence criteria, constraints as well as control and state time histories in the form of multiple 2D plots. The graphical display set up may be easily customized by the user.

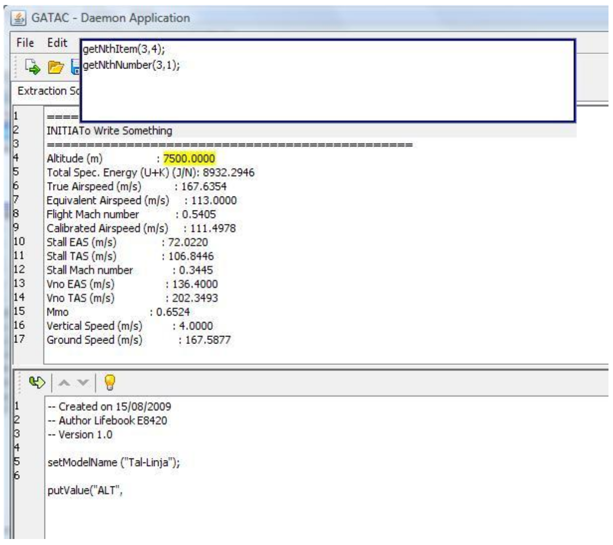
3.4 GUI

The GATAC Integration Framework is a powerful and complex software package, and thus it requires substantial time and effort for a new user to become proficient in using and achieving results with it. This steep learning curve can be alleviated with an intuitive user-friendly human-machine interface. Consequently, such a GUI is being designed with the additional benefit that it incorporates all the functionality of the tool. The GUI is divided into two parts, the first being the Model Suite GUI, the other being the Integration Framework GUI.

3.4.1 Model Suite GUI

The Model Suite GUI is used to either integrate or remove from the system. Models that are integrated into the Model Suite are then available to the user during the problem formulation in the Integration Framework GUI. A model that is to be integrated in the suite must fall under one of the following categories: executable (both binary and executable jar files), database, excel file, text file or XML file.

The interfacing with an executable model is achieved through file handling, the console or both. If the input of a model is through the console, the user only needs to program which arguments the model takes as input when it is invoked. An input through a text file, however, involves the definition of the location of each and every parameter within the file. For a complex and large input file, this can be a long and error-prone task. To help the user, a graphical tool (illustrated in Fig. 4) has been developed to allow the user to open a template of an input file (typically this can be obtained from an example supplied with the model), highlight, assign and implement simple mathematical (such as unit conversion) and logical operations on the input variables. The mathematical and logical operations are programmed using a Domain Specific Language (DSL) developed for the purpose and requiring minimal software skills to use. The resultant script from this task is used by the Models Suite to wrap the model. A similar approach is taken to create an extraction script for the output of the model.



```
getNthItem(3,4);
getNthNumber(3,1);
```

```
====
INITIATo Write Something
==============================================
Altitude (m)              : 7500.0000
Total Spec. Energy (U+K) (J/N): 8932.2946
True Airspeed (m/s)        : 167.6354
Equivalent Airspeed (m/s)    : 113.0000
Flight Mach number          : 0.5405
Calibrated Airspeed (m/s)    : 111.4978
Stall EAS (m/s)            : 72.0220
Stall TAS (m/s)            : 106.8446
Stall Mach number           : 0.3445
Vno EAS (m/s)              : 136.4000
Vno TAS (m/s)              : 202.3493
Mmo                   : 0.6524
Vertical Speed (m/s)        : 4.0000
Ground Speed (m/s)          : 167.5877
```

```
-- Created on 15/08/2009
-- Author Lifebook E8420
-- Version 1.0

setModelName ("Tal-Linja");

putValue("ALT",
```

Fig. 4. Model Suite GUI

The integration of a database involves a simple process. The user first specifies the physical location and name of the database. The tool then connects to the database and the user selects the input and output parameters from a drop-down list. Integration of excel, text or XML files make use of the DSL in a similar user-friendly manner.

3.4.2 Integration Framework

The Integration Framework GUI is used for defining, executing and post-processing optimization problems. A screenshot of the interface is illustrated in Fig. 5. The user formulates the optimization problem in a drag, drop and connect environment. A palette of simulation models and optimizers, populated from the Model Suite and Optimization Technique Suite respectively, is used to choose the components that make up the optimization case. A tools palette is also provided for simple functional blocks such as adders, subtractors, constants, and so on to enable the user to formulate any complex problem. Connection wires and buses are then used to link the optimizer to the models and modules to complete the optimization schematic. The objectives, optimization variables and constraints are set up through a windows-based user interface.

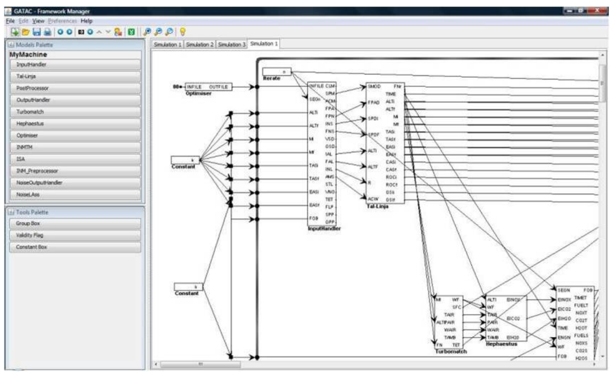


Fig. 5. GATAC Integration Framework GUI

The execution of a complex optimization problem can run into hours if not days. Thus, it is of utmost importance that the user is aware of the status of the optimization process in real-time. A number of graphs can be visualized showing histories and current values of objective functions, score diversity, constraints, and stopping criteria amongst others. From this data, the user can determine the convergence rate of the computation and be able to steer the optimization by pausing and modifying the problem formulation. Therefore, a tool is being developed to support these functions.

4 Results

Whilst GATAC is a software tool that is evolving in the context of the Clean Sky programme, a first version has been rigorously tested and validated at the implementation and functional levels of all the modules making up the framework.

The framework was deployed on the climb phase of a typical aircraft flight profile. For the purpose of this study, the flight profile in question was divided into only four segments. It is, of course, understood, that such a setup does not result in trajectories representative of real flights. Three computational models - an aircraft performance model (APM), an engine performance model (Turbomatch) and an emissions prediction model (Hephaestus) - together with the genetic algorithm optimizer used in reference [3] were installed in the framework. The problem was then formulated such that the four segments of the climb were defined by arbitrarily defining segment lengths, with the overall climb being defined by the cumulative range, start and end altitudes, and Mach numbers (International Standard Atmosphere assumed). During optimization, the intermediate Mach numbers and altitudes were allowed to vary for minimal flight time or fuel burn, with resultant emissions being used in post-optimization analyses. This, of course, resulted in step Mach number changes between segments, which, in effect, represent an effective average value over the relevant segment.

The resulting optimized climb profile yielded numerically identical results (within the margin of error of a GA optimizer) to the work presented by Celis et al. [3], with gains of 16% both in the reduction of flight time and fuel burn. This result together with other similar test cases validated the tool up to this point in development.

5 Conclusions

A first version of GATAC intended for optimizing aircraft trajectories and missions has been developed and tested. The tool has shown to be capable of setting up an optimization problem and performing multi-disciplinary aircraft trajectory optimization. Initial results show that simple trajectory optimization problems give very similar results to customized problem dependent optimization code. This gives confidence that the tool supports numerically correct optimizations opening the way for further work towards optimization of more complex problems that are more representative of actual missions. Moreover, the tool demonstrates to be sufficiently generic in nature to enable the eventual possibility of exploiting it in different domains other than trajectory optimization.

Acknowledgements

This work has been carried out as part of a collaboration between members and associate members involved with GATAC in the SGO ITD of Clean Sky. The authors wish to acknowledge, in particular, the contributions of Dr Jean-Michel Rogero in this work. The project is co-funded by the European Community's Seventh Framework Programme *(FP7/2007-2013)* for the Clean Sky Joint Technology Initiative.

References

[1] Clean Sky JU Grant Agreement No. CSJU-GAM-SGO-2008-001, Systems for Green Operations ITD, 2008.

[2] Advisory Council for Aeronautics Research in Europe. 2008 Addendum to the Strategic Research Agenda, 2008.

[3] Celis C, Long R, Sethi V, and Zammit-Mangion D. On Trajectory Optimization for Reducing the Impact of Commercial Aircraft Operations on the Environment. *19th International Symposium on Air Breathing Engines*, Montreal, Canada, ISABE-2009-1118, 2009.

[4] Salas A and Townsend J. Framework requirements for MDO application development. *7th AIAA/USAF/ NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, Vol. 98, pp. 4740, 1998.

[5] Lee H, Lee J, and Lee J. Development of web services-based multidisciplinary design optimization framework. *Advances in Engineering Software*, Vol. 40, pp 176-183, 2009.

[6] Padula S, Korte J, Dunn H, and Salas A. Multidisciplinary optimization branch experience using iSIGHT software. NASA-99-209714, NASA, 1999.

[7] Kodiyalam S and Sobieszczanski-Sobieski J. Multidisciplinary design optimization - some formal methods, framework requirements, and application to vehicle design. *International Journal of Vehicle Design*, Vol. 25, pp 3-22, 2001.

[8] Gonzalez L, Srinivas K, Periuax J and Whitney E. A generic framework for the design optimization of multidisciplinary UAV intelligent systems using evolutionary computing. *44th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 2006.

Copyright Statement

The authors confirm that they, and/or their company or organization, hold copyright on all of the original material included in this paper. The authors also confirm that they have obtained permission, from the copyright holder of any third party material included in this paper, to publish it as part of their paper. The authors confirm that they give permission, or have obtained permission from the copyright holder of this paper, for the publication and distribution of this paper as part of the ICAS2010 proceedings or as individual off-prints from the proceedings.