Cranfield University

LIBISH KALATHIL BALACHANDRAN

COMPUTATIONAL WORKFLOW MANAGEMENT FOR CONCEPTUAL DESIGN OF COMPLEX SYSTEMS: AN AIR-VEHICLE DESIGN PERSPECTIVE

SCHOOL OF ENGINEERING

PhD

Cranfield University

SCHOOL OF ENGINEERING

PhD THESIS

2007

LIBISH KALATHIL BALACHANDRAN

COMPUTATIONAL WORKFLOW MANAGEMENT FOR CONCEPTUAL DESIGN OF COMPLEX SYSTEMS: AN AIR-VEHICLE DESIGN PERSPECTIVE

Supervisor: PROF. MARIN D GUENOV

Academic Year 2004 to 2007

ABSTRACT

The decisions taken during the aircraft conceptual design stage are of paramount importance since these commit up to eighty percent of the product life cycle costs. Thus in order to obtain a sound baseline which can then be passed on to the subsequent design phases, various studies ought to be carried out during this stage. These include trade-off analysis and multidisciplinary optimisation performed on computational processes assembled from hundreds of relatively simple mathematical models describing the underlying physics and other relevant characteristics of the aircraft. However, the growing complexity of aircraft design in recent years has prompted engineers to substitute the conventional algebraic equations with compiled software programs (referred to as models in this thesis) which still retain the mathematical models, but allow for a controlled expansion and manipulation of the computational system. This tendency has posed the research question of how to dynamically assemble and solve a system of non-linear models. In this context, the objective of the present research has been to develop methods which significantly increase the flexibility and efficiency with which the designer is able to operate on large scale computational multidisciplinary systems at the conceptual design stage.

In order to achieve this objective a novel computational process modelling method has been developed for generating computational plans for a system of non-linear models. The computational process modelling was subdivided into variable flow modelling, decomposition and sequencing. A novel method named Incidence Matrix Method (IMM) was developed for variable flow modelling, which is the process of identifying the data flow between the models based on a given set of input variables. This method has the advantage of rapidly producing feasible variable flow models, for a system of models with multiple outputs. In addition, criteria were derived for choosing the optimal variable flow model which would lead to faster convergence of the system.

Following the variable flow modelling, the decomposition and sequencing of the models were performed. During the decomposition process, the non-hierarchical (strongly coupled) models were identified and grouped into a single set. Subsequently, the strongly coupled sets and the remaining models were sequentially arranged by applying a graph theoretical algorithm. In order to reduce the execution time, the models within

the strongly coupled sets were rearranged using genetic algorithm, where several candidate fitness functions were tried and tested. After extensive tests using different solvers on an aircraft conceptual design test case supplied by industry, it was found that the fitness function for the rearrangement was dependent on the solver used for the strongly coupled sets. In the current context, number of feedback loops was chosen as the fitness function for rearrangement.

A computational framework, the Cranfield Workflow Management Device (CWMD) was designed and developed by the author using the MATLAB programming language. CWMD is an object-oriented framework which provides the computational infrastructure for performing flexible design studies and was used for test and evaluation of the proposed methods.

Results obtained after applying the computational process plan on various test cases, including industrial ones, indicated significant reduction in the iterations required for the convergence of the system. Feedback provided by aircraft conceptual design engineers from industry confirmed the flexibility offered by the CWMD.

Keywords:

Aircraft Conceptual Design, Computational Tools, Constraint Management, Decomposition, Design Structure Matrix, Incidence Matrix, Object Oriented Modelling, Rearrangement, Scheduling, Strongly Connected Components, Variable Flow Modelling

ACKNOWLEDGEMENTS

I would like to thank my supervisor Professor Marin D Guenov for his support and valuable guidance throughout the project. His guidance and direction helped me to well plan my research and successfully achieve the research goals and objectives.

Many thanks to Dr. Helen Lockett for her supervision during the initial days of my research and also for her support in preparing the state-of-the-art report for VIVACE. Many thanks also to Dr. Dunbing Tang who helped me with my research in the first year.

I am grateful to people from Airbus-Toulouse for their support and timely delivery of test cases and the in-house aircraft conceptual design tools. These test cases were very useful for testing the novel methods developed as part of this research. I would like to particularly thank Thierry Druot, Luc Delaire, Claus Brandl, Yannick Caillabet and Bruno Mourguiart from Airbus-Toulouse for their support.

I am grateful to European Union Project-VIVACE and the School of Engineering for funding my studies at Cranfield University.

My special thanks to other PhD students in our research group, Paolo, Yves, Jeremy, Mattia and Marco for their friendship and support. It has been great working with you all.

I would also like to thank the staff of the Cranfield university library, computer centre, accommodation office, administration and also the School of Engineering secretaries, Elizabeth and Diane for helping me at different points during my stay at Cranfield University.

Finally my heartfelt thanks to my beloved parents for supporting and encouraging me through out my academic career. With out your love and support I would not have achieved this goal. Many thanks also to my sisters and their family for their love.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

TABLE OF EQUATIONS

ABBREVIATIONS

CAD- Computer Aided Design

CFD- Computational Fluid Dynamics

CORBA- Common Object Request Broker Architecture

CWMD- Cranfield Workflow Management Device

DM- Dependence Matrix

DO- Data Object

DSM- Design Structure Matrix

FPI- Fixed Point Iteration

GA- Genetic Algorithm

GN- Gauss-Newton

GUI- Graphical User Interface

IM- Incidence Matrix

IMM- Incidence Matrix Method

MDO- Multidisciplinary Design and Optimisation

MO- Model Object

nFdb- Number of Feedback Loops (Feedback Number)

NIvar- Number of Independent Variables

nMm- Number of Modified Models

Noutmod- Sum of the Total Number of Outputs of each Model in System

SCC- Strongly Connected Components

SP- Subprocess Object

ST- Study Object

TNvar- Total number of Variables

TR- Treatment Object

USMAC- Ultra Simplified Model of Aircraft

1 INTRODUCTION

The ever increasing competitiveness and customer demand compel aircraft manufacturers to produce better products in shorter time scales and at reduced cost. In order to achieve these requirements, new and innovative technologies and methods have to be introduced in each stage of the aircraft design.

The aircraft design consists of three consecutive stages; the conceptual, the preliminary and the detailed design stage. Among these three stages, the decisions taken during the conceptual design phase commit to around eighty percent of the product life cycle cost (Howe, 2000). Thus an inaccurate baseline design generated during the conceptual design phase will create significant cost overruns if the selected design has to be altered at the later stages of the design process.

Compared to other design stages where high-fidelity software codes are used, during the conceptual design the physics and characterises of the aircraft are represented usually in a system consisting of hundreds of algebraic equations. This simplicity in the representation tends to reduce the computing time of the system which in turn aids the designer in studying and analysing more feasible configurations of the aircraft within a limited time scale. However, the growing complexity of the aircraft design in recent years has prompted engineers to substitute the conventional algebraic equations, with simple software programs (models). The models retain the algebraic equations, but allow for a controlled expansion of the computational system. However, operating on and solving a system of non-linear models were much complicated compared to those of solving the equations. This, however, has posed the challenge of how to retain the flexibility with which the designer can operate the computational process.

A simplified aircraft system comprising of a number of models and their associated variables is shown in Figure 1-1 in a graphical format. The rectangular boxes are the models and the ovals are the associated variables. Figure 1-1 signifies the complexity involved in managing and solving the models and variables in an aircraft system. It should be emphasised that this example is a simplified version and in real cases the system will be much larger and more complex.

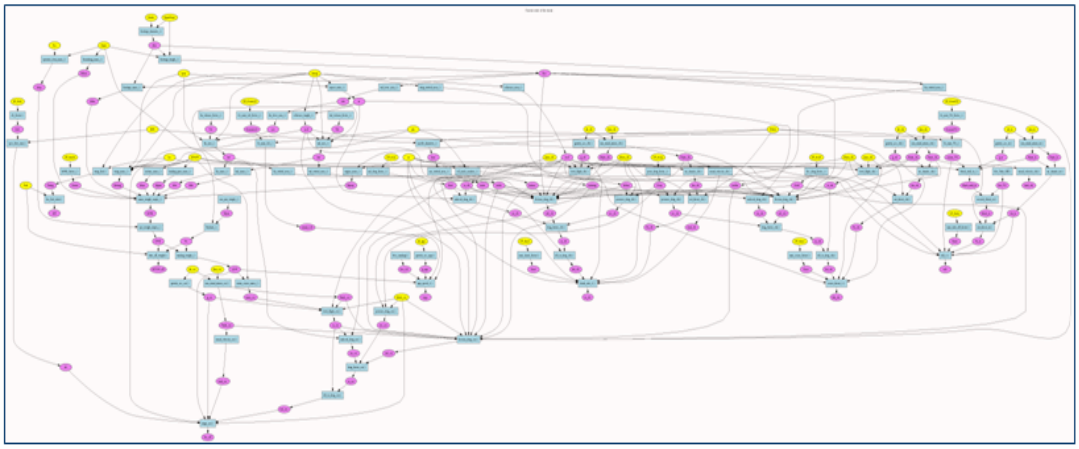


Figure 1-1 A simplified aircraft computational system (adopted from a real test case supplied by industry, (VIVACE, 2005))

A single analysis run of the system of models in the conceptual design phase takes relatively less execution time compared to the other design stages. However, the numerous trade and optimisation design studies conducted during the conceptual design phase, each making hundreds of calls to the system (Figure 1-2), and demanding a converged solution during each call, increases the overall computing time significantly. Thus, reducing the execution time of the system contributes to a considerable reduction in the overall computing time.

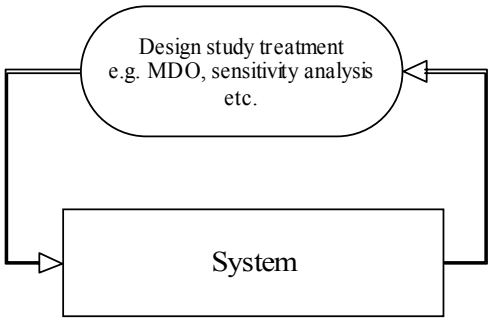


Figure 1-2 Design studies conducted over an aircraft system

In this context, the research is focused on organizing and rearranging the models within a system according to the relevant input variables defined by the designer, which plays a central role in increasing the flexibility and the efficiency of the computational design process.

1.1 Terminology

This subsection provides a brief explanation of the basic terminology which is used in this thesis.

Models: The software programs which are used to represent the physics and characteristics of the aircraft system are referred to as models. Models have either single or multiple inputs and outputs. The models are considered as black boxes since the contained information is generally unavailable (e.g. compiled code). Figure 1-3 shows an example model for calculating the atmospheric pressure and temperature, given the aircraft cruise altitude and air density.

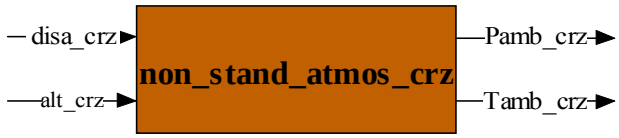


Figure 1-3 Models for calculating atmosphere pressure and temperature

Strongly connected component (SCC): Strongly connected component is a group of models which are coupled through shared variables. SCCs are distinguished because of the computational difficulties associated with their handling and solving, both individually and as part of the overall system. The term SCC is derived from graph theory and formally an SCC is a subset of nodes in a directed graph such that there is a path from every node in the set to every other node.

Subprocess: A subprocess is a collection of models which captures the relevant characteristics of a particular discipline in a mathematical form. An example of a subprocess for an aircraft, which represents the engine discipline, is shown in Figure 1-4. The inputs and outputs of the models are not shown in the figure.

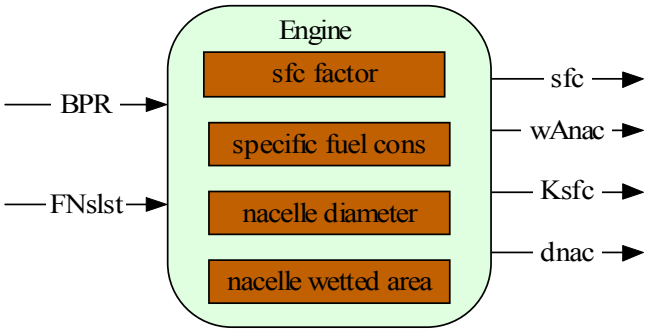
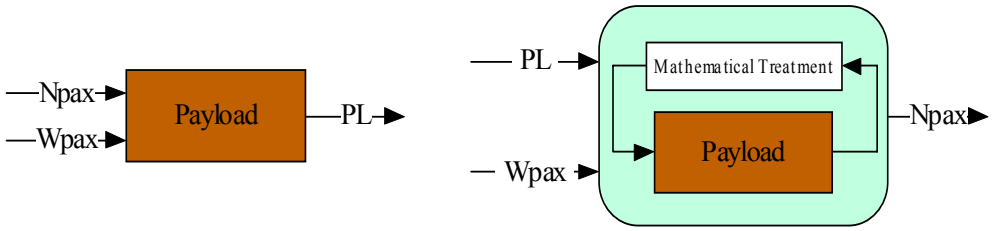


Figure 1-4 An engine subprocess

System: A system is collection of models which captures all the relevant physics and characteristics of the product under development. A mathematical system of an aircraft at the conceptual design stage contains hundred of models in order to capture all the characteristics of the aircraft.

Modified model: A modified model is a model which has some of its input and output variables swapped. The solving of such modified models is accomplished by applying relevant mathematical treatments (solvers). Figure 1-5 shows an example for a modified payload model where the input Npax and output PL are swapped.



(a) Payload model (b) Modified payload model

Figure 1-5 Example of a modified model

Mathematical treatment: A mathematical treatment refers to either the design studies performed over a model, a system or a subprocess, or a solver used to solve an SCC or a modified model. Optimisation, sensitivity analysis, design of experiments etc. are examples of mathematical treatments for design studies. Fixed point iteration, Gauss-Newton methods are the examples of mathematical treatments for solving an SCC or a modified model.

Sub-system: A sub-system is the general term used to signify a subset of model or models in a system, which requires a mathematical treatment to be applied for solving them. An example for a sub-system is an SCC or a modified model.

1.2 Background

1.2.1 Aircraft conceptual design

As mentioned earlier aircraft design consists of three consecutive design phases. The conceptual design phase is the most critical phase. The decisions taken during this stage commit around eighty percentage of the entire life-cycle cost of the aircraft. During the

conceptual phase questions regarding configuration arrangement, size, weight and performance are answered. The conceptual design phase initiates after the requirements for the new aircraft are sufficiently well defined. The requirements definition is based on the (customer) needs that are beyond the capability of the existing aircraft, for example, an extended range. There are cases where the new requirements arise based on the operational experience and limitations identified on a current aircraft. Additionally requirements can be identified within the manufacturing organisation or from the user (if different from the customer). Airline manufactures constantly seek feedback from the airline operators to identify the future needs.

The requirements for a new aircraft are divided into three categories; performance requirements, flight requirements and structural design requirements (Howe, 2004). The performance requirements consist of take-off and landing field lengths, residual climb capacity in case of an engine failure and performance when a landing approach is abandoned. The flight requirements consist of control characteristics and effectiveness, static and dynamic stability and manoeuvre capacity at critical flight phases. The structural design requirements are classified into two categories, the stiffness and strength. These requirements ensure that the airframe will not deform beyond specified limits during various flight manoeuvres which may compromise flight safety.

Once the requirements for the new aircraft are defined the next step is to analyse the way of meeting these. These are identified as;

a) An adaptation or a special version of an existing aircraft

b) A major modification of an existing type

c) A completely new design.

The first case involves alterations in the airframe and change of equipment. The cost involved in this case is relatively small. The second case can be more expensive because of major changes to the airframe including, extended fuselage, new wings, new power plant and also equipment changes. The third phase is the most expensive option and also involves the greatest risk.

To reach a decision to proceed with one of the above three choices, the physical characteristics of the aircraft have to be obtained based on its properties. The

characteristic is a physical parameter and property is an operational parameter (the requirements for the aircraft are generally the properties of the aircraft). In order to obtain the characteristics of the aircraft, hundreds of mathematical models with thousands of associated variables, which represent the physics of the aircraft, are generally operated upon.

The mathematical models which represent the physics of the aircraft usually have the characteristics of the aircraft as input and the aircraft property as the output (Figure 1-6). However, the design process usually has to deal with the reverse problem, that is, some of the properties (requirements) of the aircraft are initially available and the mathematical models are used to calculate the characteristics of the aircraft. Thus the aircraft properties is the available input to the system and the characteristics of the aircraft has to be calculated based on this input. This change of inputs to the system is the root cause of most of the computational complications in an aircraft design or any other complex system. Because of the limited reversibility of the models, alternative aircraft characteristics have to be provided as inputs to the system in order to obtain the necessary aircraft properties. Currently the effects of these changes on the aircraft properties have to be analysed by extensive trade and optimisation studies, in order to satisfy the requirements which acts as constraints over the system.

This reversal of inputs and outputs is one of the problems addressed in this research and is associated with several issues outlined in section1.2.3.

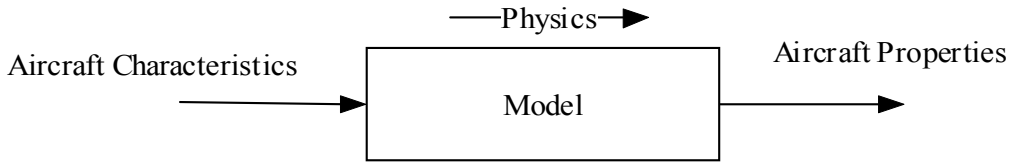


Figure 1-6 A general aircraft mathematical model

The next subsection briefly outlines various trade and optimisation studies conducted during the conceptual design phase of aircraft design.

1.2.2 Design studies during the aircraft conceptual design

Various design studies are performed on the system during the aircraft conceptual design phase in order to obtain a sound baseline, which meets or exceeds the

requirements for the aircraft. This baseline is then passed on to the subsequent design phases for further detailed analyses. The most common design studies conducted at conceptual design stage are trade and optimisation studies.

Trade studies are conducted to explore wide range of alternative configurations in a design process. Here the designer studies the effects of variation of one or more parameters on the values of other parameters. If the range calculated, for example, is 10,000km which is less than the customer requirement, a "range trade" can be calculated if an increase in the take-off gross weight (TOGW) could increase the range to the required level (Raymer, 1999). During trade studies the system is executed each time a new set of inputs have to be evaluated. An example graph plot for a trade study is shown in Figure 1-7

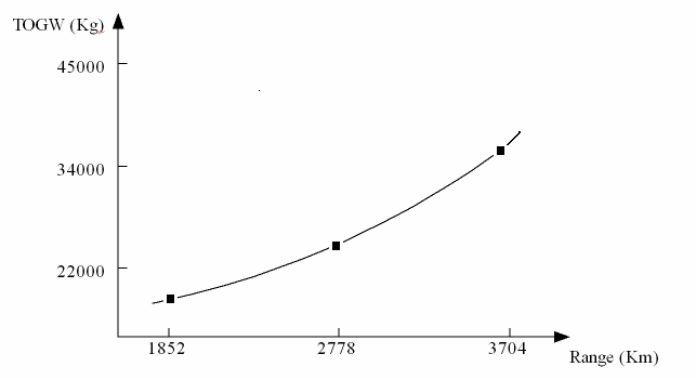


Figure 1-7 Range trade (Raymer, 1999)

Various optimisation studies are performed during the conceptual design phase, which help to meet the requirements for the aircraft and also to exceed the requirements. The optimisation studies assist the designer to explore the global optimum of a design subjected to various constraints. The most common optimisation criteria are the minimisation of the weight and cost (Howe, 2004). During optimisation studies thousands of calls are made to the aircraft system, hence the system execution time plays a significant role in the overall optimisation study execution time.

Optimisation is an evolving field and various methods are currently being researched and developed in this area. Optimisation has developed in the recent years from single objective to multi-criteria, and further to multidisciplinary design optimisation. Some of the commonly used optimisation techniques during conceptual design are NAND

(Nested Analysis and Design) (Balling and Sobieszczanski, 1996), SAND (Simultaneous Analysis and Design) (Balling and Sobieszczanski, 1996), CO (Collaborative Optimisation) (Kroo et.al., 1994), CSSO (Concurrent Subspace Optimization)(Sobieszczanski, 1988) and BLISS (Bi-Level System Synthesis) (Sobieszczanski et.al., 1998), out of which BLISS is latest development in the field of multidisciplinary design optimisation (MDO).

The next section explains the basics of computational process modelling.

1.2.3 Computational process modelling for complex systems

Figure 1-1 illustrated the potential complexity involved in managing and solving the numerous models in a system. We have already explained the importance of reducing the execution time for a system which plays a significant role in reducing the overall design study time.

Computational process modelling is the process of organising a complex system of models in order to calculate quickly the output variables based on the input variables given for the system.

There can be many ways, in terms of computational sequence and information flows, in which a system can be solved to calculate the required output variables. The main task for computational process modelling is to identify the most appropriate organisation and ordering of the models with in the system, so that on execution, the system of models converges quickly and generates the outputs in minimum time.

In this section the basics of computational process modelling are introduced. Computational process modelling is subdivided into three parts; variable flow modelling, system decomposition and scheduling, all subject of the present research. The next three subsections introduce each part in more detail.

1.2.3.1 Variable flow modelling

Variable flow modelling is the process of identifying the information flow among the models based on the input variables selected by the designer for the system. The information flow among the models helps to determine the output variables which could be calculated according to the input variables provided by the designer.

In the system, a model becomes executable when all of its input variables are known. Initially the input variables provided by the designer are the only known variables. The computable variables are determined sequentially with respect to the outputs of the executable models. An example of variable flow modelling operation is shown in Figure 1-8 (Buckley et al., 1992).

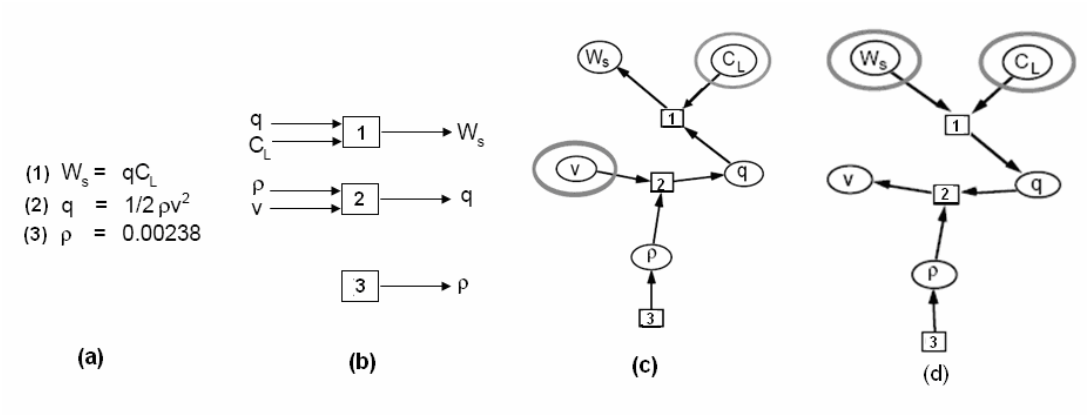


Figure 1-8 Variable flow modelling for a system of three mathematical models (Buckley et al., 1992)

Figure 1-8 (a) and (b) shows a system of algebraic equations and its corresponding models for balancing the weight of an aircraft with its lift. Arrows entering the boxes (Figure 1-8 (b)) represent the inputs and arrows leaving the boxes represent the outputs. Figure 1-8 (c) shows, in graphical format, the variable flow models for the system. In this figure, variable nodes are oval shaped and model nodes are rectangular shaped. An arrow marked from a variable node to the model node denotes that variable is an input of the model. If the arrow is directed from the model to the variable, then that variable is output of the model.

Variables V and C_L (denoted as dual ovals in the figure) are the inputs provided by the designer in this case. Similarly Figure 1-8 (d) shows the variable flow model for the system with Ws and C_L as inputs. For the first case it can be noted that model1 has C_L and q as inputs and Ws as output, model 2 has ρ and V as inputs and q as output and model 3 produces ρ as output.

The sequence for determining the output variables in this case is shown in the Figure 1-9. Here V and C_L are the initial know inputs. In the first step, model3 is identified as executable since model3 does not need any input variables. Hence the output of model

3, ρ is added in the known variables list. Now model2 is executable since all its inputs (ρ and V) are now known. Hence its output q is added in the known variables list. Finally model 1 is identified as executable and its output Ws is added in the list.

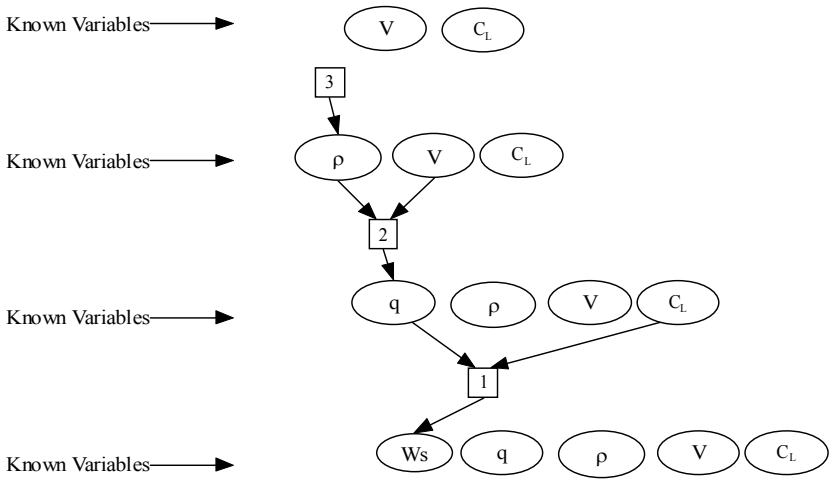


Figure 1-9 Sequence for finding the output variables

For the second case in Figure 1-8 (d), after variable flow modelling, model1 has Ws and C_L as input and q as output. But the actual model1 as shown in Figure 1-8 (b) has q and C_L as input and Ws as output. In this case the inputs and output variables of the model1 are swapped and therefore model1 is considered as a modified model. The term modified model was defined in the section 1.1. Solving a modified algebraic equation can be done by symbolic methods, but for models numerical solving techniques (mathematical treatments) has to be applied, which is another problem tackled in the current research.

The example discussed here is a very simple one. In case of real aircraft conceptual design problems there are hundreds of models and thousands of associated variables. In such cases, obtaining variable flow models will be a complicated process and particular methods are required to obtain the variable flow models. Presence of strongly connected components will further add to the difficulty.

As specified in the terminology section-1.1, ***strongly connected components (SCC)*** are those groups of models which are strongly coupled through shared variables. The term SCC is derived from graph theory and formally an SCC is a subset of nodes in a directed graph such that there is a path from every node in the set to every other node

(Buckley et al., 1992). In the context of a system of models the SCCs are those clusters of models in which each model requires input from one or more models from the same cluster. Presence of SCCs in a system adds to the complexity of the computational process modelling, including variable flow modelling, decomposition and scheduling. In addition, SCCs requires iterative solving of the constituent models. Therefore numerical solving schemes has to be applied which adds to the computational burden on the system.

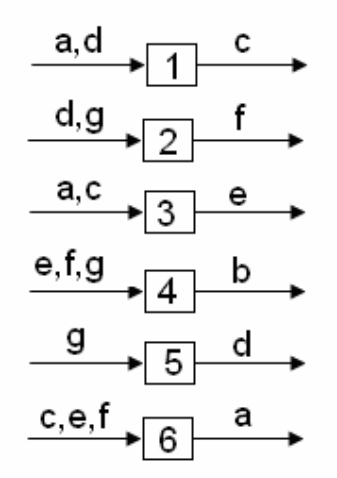


Figure 1-10 An example for a system of models

The process of sequentially determining the known variables, as described earlier, cannot be carried out in the presence of SCC. For example for the system of models in Figure 1-10, if variable 'g' is declared as the independent (known), the procedure for determining the outputs variables is shown in Figure 1-11.

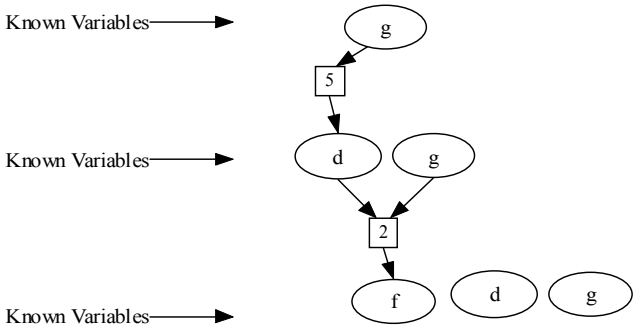


Figure 1-11 Sequence for finding the output variables

Figure 1-11 shows that with variable g given as the input, only variables f and d can be calculated from the system of models. However, there is a variable flow model for this system which helps to calculate all the variables of the system. This variable flow model is displayed in Figure 1-12. It should be noted that it could not be identified using procedure adopted in the Figure 1-11 because of the presence of a SCC. In this example models 1, 3 and 6 are strongly connected. Variable flow modelling in the presence of SCC is another issue which is addressed in this research.

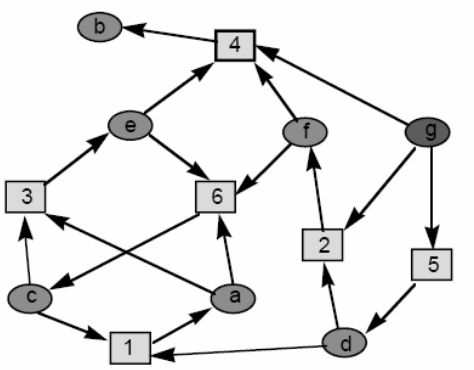


Figure 1-12 Directed Bipartite Graph for models in Figure 1-10 with variable 'g' as input (Buckley et al., 1992)

A number of variable flow methods have been developed earlier for finding the information flow for system algebraic equations. The variable flow models generated for the equations consider that a single variable is calculated by an equation i.e., each equation generates one output. However models generated multiple outputs and hence the available variable flow modelling method has to be reformed to apply in this context.

1.2.3.2 System decomposition

System decomposition is the process of decomposing a complex system into a number of sub-problems. Decomposing reduces the complexity of solving a large-scale system into a number of sub-problems. Decomposition has a wide range of applications in concurrent engineering for identifying the manufacturing processes that can be run in parallel.

In the context of solving an aircraft system, the system decomposition corresponds to identifying the models which are strongly connected and grouping them as sub-systems.

The system of models in Figure 1-10 after decomposition is shown in Figure 1-13. In this example models 1, 3 and 6 are strongly connected and represent a subsystem. Decomposition is another issue which is addressed in this research.

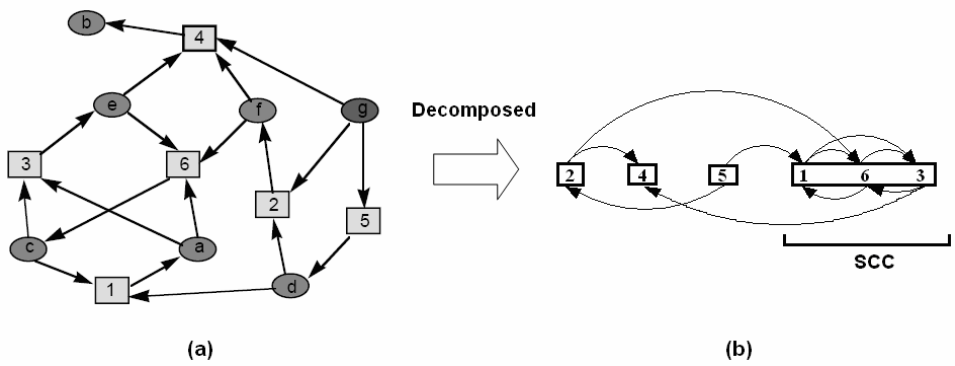


Figure 1-13 Decomposition of a system of models

1.2.3.3 Scheduling

Scheduling is the process of sequencing the models in a system for the purpose of executing them after eliminating or reducing the feedback loops among the models. In the decomposed system shown in Figure 1-13(b), if the execution sequence is considered to be from left to right, the arrows shown pointing from right to left below the models denote the feedback loops. A feedback loop requires certain models to be executed before all their inputs are available. For example, in Figure 1-13(b), model 2 requires an input from model 5 which follows later on in the execution sequence.

In complex design systems a complete elimination of iterative feedback loops by scheduling the models is usually not always possible. Those group of models whose feedback loops cannot be eliminated are the strongly connect components. In such cases the models which were identified as SCCs during the decomposition process are grouped into sub-systems. These sub-systems along with the remaining models are further sequenced during the scheduling process.

In the example in Figure 1-13(b) the models and subsystem after scheduling are shown in Figure 1-14. Compared to the arrangement in the Figure 1-13(b), the number of feedback loops (also called as feedback number) has reduced from four to two. The remaining two feedback loops are part of the SCC and these loops cannot be eliminated.

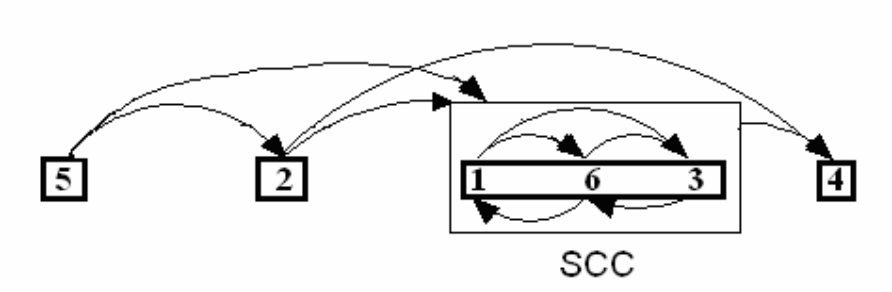


Figure 1-14 The mathematical models of Figure 1.6 after scheduling

Even though the feedback loops belonging to the SCC cannot be eliminated, rearranging these models can lead to a reduced number of feedback loops. This issue of reducing or eliminating the feedback loop in an SCC, which plays a significant role in reducing the convergence cost of the SCC, was a fundamental issue addressed in this research.

1.3 Summary of Aims and Objectives

The growing complexity of aircraft design in recent years has prompted engineers to substitute the conventional algebraic equations with software programs (black boxes or models) which still retain the mathematical models, but allow for a controlled expansion and manipulation of the computational system. This tendency has posed the research question of how to dynamically assemble and solve a system of non-linear models. In this context, the overall aim is to develop a method which significantly increases the flexibility and efficiency with which the designer is able to operate on large scale computational multidisciplinary systems at the conceptual design stage.

In support of the aim the following objectives have been identified:

- Develop a method for variable flow modelling for a system of models with multiple outputs, which will assist the designer in choosing alternative combination of inputs for the system. In addition, criteria need to be derived for choosing from the alternative variable flow models for a system, the one which would lead to faster convergence.
- Develop methods for decomposing and scheduling the models in a system
- Combine variable flow modelling, decomposition and sequencing into a novel method for computational process modelling.
- Develop methods for reducing the execution time for SCCs.

- Investigate and develop mathematical treatments for the modified models in order to achieve the swapping of input and output variables of the models.
- Develop a prototype computational framework for performing flexible design studies which is to be used for test and evaluation of the proposed methods.

1.4 Thesis Structure

This thesis is subdivided into seven chapters. The first chapter introduces the aircraft conceptual design stage and the design studies conducted during this stage. The basic terminology which is widely used in the thesis is also covered in this chapter. Thereafter an introduction regarding the computational process modelling and its subdivisions, the variable flow modelling, scheduling and decomposition are presented. This is followed by the aim and objectives of this research.

The second chapter is the 'literature survey' which gives a comprehensive review of the various currently available computational methods and tools, which have the potential to be applied in the current research context. A critical analysis of the various computational process modelling methods, including variable flow modelling, decomposition and sequencing is given in this chapter. A critical review of the various computational tools commonly used in aircraft design is also presented.

The third chapter 'computational process modelling for complex systems' covers the novel methods developed as part of this research for computational process modelling. The individual features of the computational process modeller, which is developed for computational process modelling, including the novel variable flow modelling method along with the decomposition and scheduling methods, are described in the corresponding sections. A comprehensive example is given which demonstrates the working structure of the entire computational process modeller.

The fourth chapter 'solvers for sub-systems' is focussed on the mathematical treatments, used for solving the sub-systems. Application of the Gauss-Newton method for solving the modified model is presented. Further, application of fixed point iteration and the Gauss-Newton method for solving the strongly connected components are also presented. A method for solving the modified models and strongly connected components together at the system level is also described.

The fifth chapter presents the various tests performed for evaluating methods and approaches developed as part of this research. USMAC, an aircraft conceptual design test case which was widely used for testing is explained initially. Following this, an evaluation of the two objective functions which were chosen for scheduling the coupled models by genetic algorithm is given. Thereafter a detailed description regarding the testing conducted in order to evaluate the proposed computational process modeller is given. Finally, the conclusions are presented.

The sixth chapter describes the object oriented software framework which has been developed in Matlab for implementing and testing the proposed methods and approaches. The various modules of CWMD along with its GUIs, which assist in the easy implementation of the computational plans developed by the computational process modeller, are also explained.

The seventh chapter describes the summary and conclusions of this research. This chapter also explains the current limitations of the proposed methods and applications developed in this research. In addition, the areas which will be of interest for future work are also outlined in this chapter.

2 LITERATURE SURVEY

2.1 Introduction

The aim of the literature survey was to investigate and evaluate various computational methods and tools, which had the potential to be applied in the current research context. Section-2.2 investigates and performs a critical analysis of the various currently available computational process modelling methods, including variable flow modelling, decomposition and scheduling. Section-2.3 provides a critical review of the various computational tools commonly used in aircraft design. Conclusions are drawn in Section-2.4.

2.2 Computational Process Modelling Methods

Computational process modelling is the process of organising a complex system of models in order to calculate the output variables based on the selected input variables. During the literature review it was identified that the various methods available for computational process modelling can be classified into variable flow modelling, decomposition and scheduling methods.

2.2.1 Variable flow modelling methods

Variable flow modelling is the process of identifying the information flow among the models based on the system input variables selected by the designer. Constraint propagation approaches have been utilised by several researchers for variable flow modelling in conceptual design systems. In this approach the equations are represented as constraints between the variables, and the changes in the variable's values are propagated across the constraint network.

Serrano (1987), developed a graph theoretical approach to constraint management. The constraint networks were represented as directed graphs, where nodes represent parameters and arcs represent constraint relationships. The direction of the arcs represented the dependencies among the parameters. Parameter dependencies (variable flow modelling) were generated automatically using bipartite matching representation. Serrano reports two algorithms for bipartite matching. The first one is modelling the

bipartite matching as a network flow problem and the second is bipartite matching based on linear programming.

Modelling the bipartite matching as a network flow problem involves maximizing the flow from a source node to the sink node in a graph representation of the constraint management problem, without exceeding the capacity (maximum allowable flow on the arc) of any one arc on the path. This problem is commonly known as maximal flow problems (Cormen et.al, 1991, p.643). The method used by Serrrano for maximal flow problem is the Max-flow min-cut theorem (Cormen et.al, 1991, p.657). There are many other algorithms for maximal flow problems, some of these are, Ford-Fulkerson algorithm, Brute-force search, Edmonds-Karp algorithm, Relabel-to-front algorithm, etc., (Cormen et.al, 1991). The Max-flow min-cut theorem is described below.

The max-flow min-cut theorem is a statement in optimization theory about maximal flows in flow networks. It states that "The maximal amount of flow is equal to the capacity of a minimal cut".

Suppose G(V,E) is a finite directed graph and every edge (u,v) has a capacity c(u,v) (a non-negative real number). Further assume two vertices, the source s and the sink t. A cut is a split of the nodes into two sets S and T, such that s is in S and t is in T. The value of the cut is the sum of all the flows of the arcs that are separated by the cut. The cut that produces the smallest flow will ensure the maximum flow for the networks. The arcs along the cut carry their maximum flow. For bipartite matching problems the capacity of each arc is assigned a value of 1.

The second method for bipartite matching, linear programming, is performed by remodelling the matching problem into a linear optimisation problem. Each arc in the bipartite graph has an upper limit which is the capacitance value u_{ij}. The unknown in the optimisation problem are the flows x_{ij} from node i to node j. The optimisation problem is represented as

Maximise: $x_{so\text{-}si;}$ here so-source, si-sink

Subject to: $\sum x_{ij} - \sum x_{jk} = 0\,;\ 0 \le x_{ij} \le u_{ij}$

Serrano's graph theoretical approach was developed primarily for systems consisting of algebraic equations. The approach did not address the computational complications that would have been encountered if the system contained models. The variable flow models generated for the equations consider that a single variable is calculated by an equation i.e., each equation generates one output. Models generate multiple outputs and hence require modification in the variable flow modelling scheme mentioned above.

However, Serrano's approach had an implicit reference to variable flow modelling for models with multiple inputs and outputs, but lacked a detailed demonstration or testing. This approach also did not explore the different feasible variable flow models that can be generated for a system.

Bouchard et al. (1988) used directed constraints between design variables and numerical solution approaches to allow rapid production of trade-off studies. The limitation of this approach was that the designer had to decide in advance the input and output variables.

Buckley et.al (1992) has developed a bipartite graph method for variable flow modelling. The system was represented in a bipartite graph with square nodes representing equations and the oval shaped nodes representing variables. The edges in the graph connect equation nodes to the variable nodes, which indicate that the variable is present in the equation. An example bipartite graph is shown in the Figure 2-1.

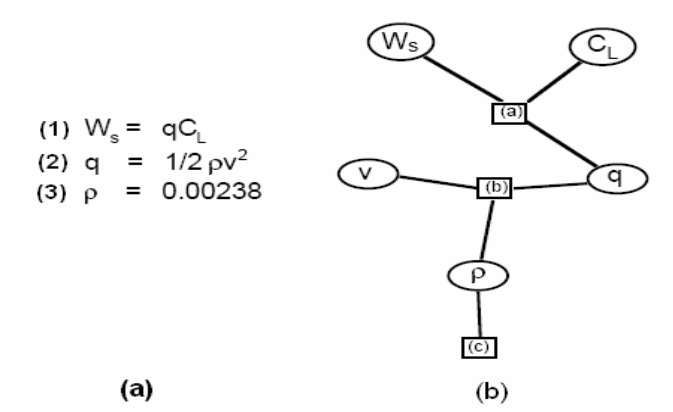


Figure 2-1 A system of non-linear equations with its corresponding undirected bipartite graph (adapted from Buckley et.al, 1992)

The directing of the graph, based on the known variables, is accomplished using a variant of the Ford-Fulkerson algorithm (Cormen et.al, 1991) for finding maximal matching on bipartite graphs. Ford-Fulkerson algorithm as mentioned earlier is one of the methods for maximum flow problem. The algorithm finds an initial pairing of equations and the variable nodes. Then it finds the paths in the graph where directions can be reversed to improve the matching. Figure 2-2 shows the directed graph for the example given in Figure 2-1 with variables Ws and C_L specified as the known variables by the designer.

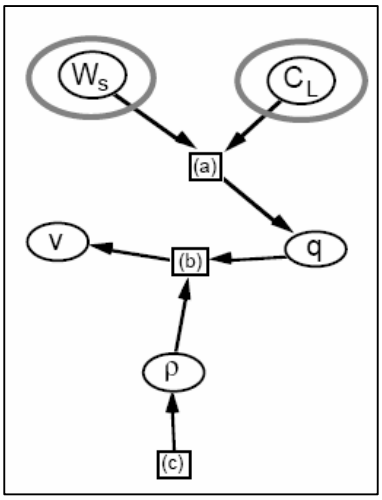


Figure 2-2 Directed bipartite graph (adapted from Buckley et.al, 1992)

The Ford-Fulkerson algorithm computes the maximum flow in a flow network (Cormen et.al, 1991, p.651). The idea behind the algorithm is "As long as there is a path from the source (start node) to the sink (end node), with available capacity on all edges in the path, we send flow along one of these paths. Then we find another path, and so on". For a graph G(V,E), with capacity c(u,v) and flow f(u,v) for the edge from u to v, in order find the maximum flow from the source s to the sink t, after every step in the algorithm(which searches for an augmented path) the following is maintained.

$f(u,v) \le c(u,v)$. The flow from u to v is limited to its maximum capacity

$f(u,v) = -f(v,u)$. Conserves the net flow between u and v. (value of f(u,v) is initially set to zero)

$\sum_{v} f(u,v) = 0 \Leftrightarrow f_{in}(u) = f_{out}(v)$ for all nodes u except s and t. Flow into a node is equal to a flow leaving a node.

The path search is performed by a breadth-first search or a depth-first search algorithm (Cormen et.al, 1991).

As in the previous cases, Buckley et al. (1992) focused on obtaining variable flow models for algebraic equations and not models. In addition, the alternative solutions (variable flow models) for a system were not explored.

Ramaswamy and Ulrich (1993) have developed an adjacency matrix based heuristic algorithm for variable flow modelling. Serrano's (1987) work considered the case where the set of known and unknown variables were specified completely. Ramaswamy's work dealt with situations where only some members of the known variables set have been specified and the remaining variables could be chosen by the designer. The basic idea behind the Ramaswamy's algorithm was, "when a variable in an equation is changed we can use another variable in the same equation to compensate for the change and thus keep the equation satisfied. Since variables may appear in several equations, the effect of changing any variable must be propagated". Ramaswamy's algorithm had the restriction that the functional forms must be either algebraic or transcendental functions. Recursive functions and iterative computer programs (models) were explicitly excluded.

2.2.2 System decomposition and scheduling methods

System decomposition is the process of decomposing a complex system into a number of sub-problems and scheduling is the process of sequencing the models for the purpose of executing them after eliminating or reducing the feedback loops among the models.

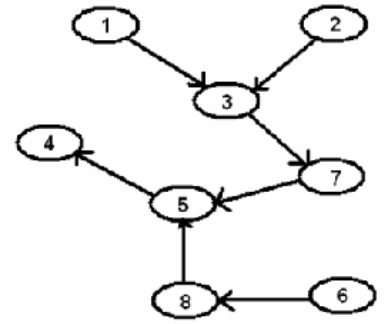
Decomposition and scheduling methods have application in various fields of engineering. Here the author reviews the decomposition and scheduling methods in two areas; the first one is for process management in industries and the second one for the mathematical models in a computational system.

The decomposition and scheduling methods are reviewed in a single section because most of the research conducted in these two areas had reference to each other.

2.2.2.1 Decomposition and Scheduling Methods for Process Management

Process management methods are used for arranging the processes involved in the manufacturing of a product in an industrial environment. The rearrangement identifies the execution sequence of the processes and also the processes which could be executed concurrently. By this arrangement the manufacturing time of the final product could be reduced significantly. Since there is a requirement for arranging the models in a system which is similar to process management, various methods in this area have been reviewed during this research.

Most of the process management methods have evolved from graph theory (Rogers, 1999). Graph representation consists of nodes and edges. An edge connects two nodes. Directed, non-directed, cyclic, acyclic, bipartite, etc., are various types of graph representations. Graphs can be represented in numerous forms, and the most commonly used representation is the adjacency matrix. Adjacency matrix is a square matrix with values of either 1's or 0's. A value of 1 denotes a link from the process in the row to the process in the corresponding column. If there is no link then the value will be zero. An example graph and its corresponding adjacency matrix are shown in Figure 2-3.



Graph

	1	2	3	4	5	6	7	8
1	0	0	1	0	0	0	0	0
2	0	0	1	0	0	0	0	0
3	0	0	0	0	0	0	1	0
4	0	0	0	0	0	0	0	0
5	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	0	1
7	0	0	0	0	1	0	0	0
8	0	0	0	0	1	0	0	0

Adjacency Matrix

Figure 2-3 Graph representation of a process and the corresponding adjacency matrix

One of the earliest available tools for process management was PERT (Rogers, 1999). PERT network is a directed, weighted, acyclic graph. The weights of the edges in a PERT network represent the time needed to complete that process. An example PERT network is shown in Figure 2-4.

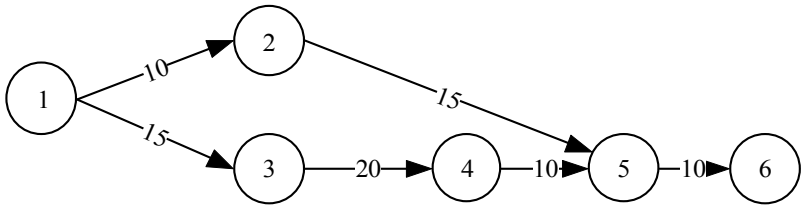


Figure 2-4 Sample PERT Chart

In the figure, for example, the time needed to complete the process represented by the arc (3, 4) is 20 minutes. It can be noted that process (5, 6) cannot be started until process (4, 5) is finished. Even though the path (1, 2, 5) takes only 25 minutes, process (5, 6) cannot be started until the process (1, 3, 4, 5), which takes 45 minutes is completed. This implies that any delay in the process in the path (1, 3, 4, 5) will delay the entire project. However, some time delays are acceptable for the processes in the path (1, 2, 5). A path which will cause project to be delayed if any of its constituent processes (tasks) is delayed is called a critical path. The main objective of a PERT chart is to reduce the critical path which will aid in reducing the execution time for the entire project. The PERT tool is applicable only to sequential activities and cannot handle non-sequential activities. This makes PERT chart unsuitable for aircraft conceptual design, where non-sequential activities (SCC) are commonly encountered.

Steward (1981) had developed another tool for displaying a process sequence called the design structure matrix (DSM). DSM serves as a highly efficient tool for process management. DSM is similar to the adjacency matrix and is derived from graph theory. An example DSM is displayed in Figure 2-5.

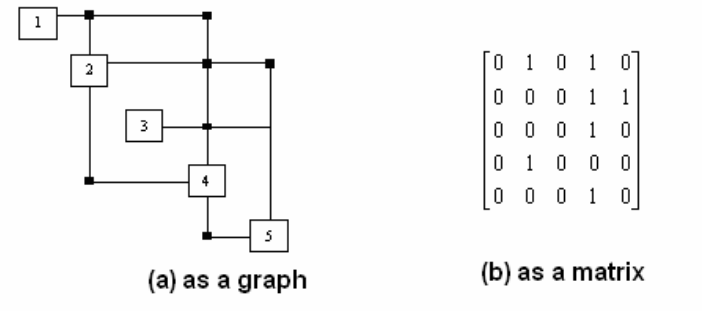


Figure 2-5 Design Structure Matrix

In the DSM shown in Figure 2-5(b), the processes are shown in the numbered boxes along the diagonal. The off-diagonal small black squares that join the horizontal and

vertical lines represent the coupling between the processes, which means the output from the process connected to the horizontal line is given as input to the process attached to the corresponding vertical line. Squares above the diagonal denotes feed forward coupling and those below the diagonal indicates feedback coupling. The data is fed forward in a feed forward coupling, which means that the data required to run a process is available from the process which appears earlier in the sequence. However, in a feedback coupling the data required for executing a process is not available beforehand and it has to be obtained from a process which appears later in the sequence. The advantage of DSM compared to PERT is the capability to group and display the iterative sub-cycles (SCC) found in a design project.

Tang et al. (2000) have introduced a DSM based method for decomposition, which is the identification of the processes which form iterative sub-cycles (SCCs) in a system. The paper also introduces a method for sequentially arranging the decoupled activities and identifying the processes which could be executed in parallel. Tang's algorithm was developed to arrange the process in a manufacturing environment. For identifying the iterative sub-cycles first a binary DSM, 'A' of the system was created. Binary DSM consists of a matrix with row and columns representing the processes. An element '1' in the matrix denotes the process of the column has an input from the process in the row. Further the following sequence of operations was performed on the matrix A:

$$P = A^T; K = \sum_{n=1}^{j} P^n; J = K \circ K^T$$

In the final matrix J if the values of the rows are equal, then the processes which represented the corresponding rows have been identified as strongly connected.

In Tang's paper, the sequential arrangement of the processes is performed by applying a theorem from Xiao and Fei (1997). Before performing this operation, the rows and columns which represent the strongly connected processes and their corresponding variables are collapsed into a single row and column in the matrix P.

Tang's algorithm does not take into account the arrangement of the processes which forms iterative sub-cycles. Nevertheless the sequential arrangement algorithm for the non coupled process has been utilised in the present research.

Kusiak and Wang (1993) have developed a method for decomposing design processes in order to enhance concurrency. Incidence matrix was used to represent the dependency between tasks and parameters in a design process. The rows represented the tasks and columns the parameters. In the matrix, an entry '*' denotes that the parameter in the column is dependent on the task in the corresponding row. The objective is to group the rows and columns in such a way that the matrix separates into mutually exclusive sub-matrices. This grouping was accomplished by using the Cluster identification (CI) algorithm (Kusiak and Chow, 1987). In case of non-decomposable systems, this grouping was not possible with the CI algorithm. The paper proposes an improved CI algorithm (Kusiak and Cheng, 1990), which identifies the overlapping parameters (or tasks) which are removed from the matrix for further grouping. In addition to the task-parameter matrix, this method had application to complex design problems involving large number of constraints and variables. When applied to such cases the matrix had constraints in the rows and variables in the columns. For a non-decomposable constraint-variable incidence matrix, the coupling variables were identified and removed from the matrix using the earlier mentioned improved CI algorithm. Examples of decomposed incidence matrices are shown in Figure 2-6.

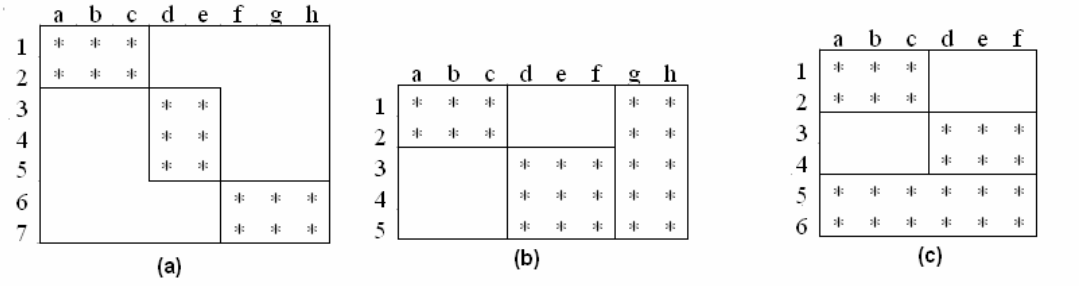


Figure 2-6 (a) Decomposed incidence matrix, (b) Decomposed incidence matrix which has overlapping constraints (c)Decomposed incidence matrix with overlapping variables (Kusiak and Wang, 1993)

Kusiak's algorithm is suitable for decomposing large non-hierarchical systems into smaller mutually exclusive sub-systems for parallel execution. The arrangement of the processes within the sub-systems is not addressed in this work.

Chen et al. (2005) have proposed a formal two-phase method for decomposing complex design problems to more tractable sub-problems. Unlike Kusiak's approach, this method

decouples the decomposition process into mutually exclusive function components: dependency analysis and matrix partitioning. Dependency analysis is achieved via an extended Hierarchical Cluster Analysis (HCA) and matrix partitioning by a Partition Point Analysis (PPA). Further to the application of these two algorithms, a non-organised incidence matrix is decomposed into a block-angular structured matrix. In the first phase the extended HCA is applied to reorder the unorganized incidence matrix into a banded diagonal matrix. The extended HCA comprises of the Binary Tree Construction (BTC), Binary Tree Branch Association (BTBA) and Binary Tree Association (BTA) algorithms. In the second phase, the PPA is applied to further transform the banded diagonal matrix into a block-angular matrix according to decomposition criteria set by the user. The function of each algorithm and the overall view of the formal two-phase algorithm are shown in the Figure 2-7.

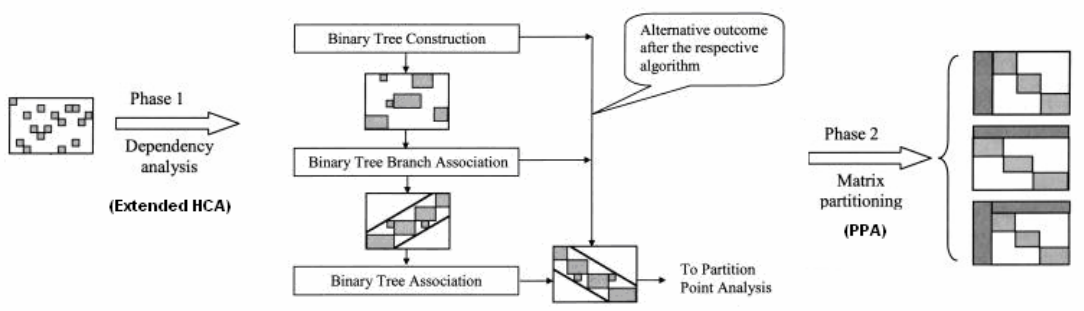


Figure 2-7 Workflow of the two-phase decomposition method (Chen et.al., 2005)

Chen et al.'s algorithm, like Kusiak's, did not address the arrangement of the processes within the subsystems and focuses only on the decomposition of the system into sub-systems. However, this algorithm provides the user with the flexibility in the choice of the different settings of the decomposition criteria, by which the incidence matrix can be decomposed into either a column based, row-based or a hybrid block-angular structure matrix. Chen et al. claims this algorithm will be more efficient than Kusiak's since recursive solving of the matrix is not involved for the decomposition.

2.2.2.2 Decomposition and Scheduling Methods for computational systems

This sub-section provides the decomposition and scheduling methods for design problems represented in computational systems.

Rogers (1997) has developed a software tool named Design Manager's aid for Intelligent Decomposition (DeMAID), for decomposing and sequencing of the models. DSM has been widely used in this tool as a method for sequencing and decomposition. Initially a knowledge based tool was developed for sequencing the models (Rogers, 1989). The knowledge based approach could only examine a limited number of orderings of the models which are part of iterative sub-cycles (SCC). In order to overcome this limitation Rogers in his later work (Rogers, 1997) has introduced a genetic algorithm (GA) based rearrangement method for rearranging the models in the iterative sub-cycles. GA scans a large number of orderings of the models and obtains an optimised ordering based on computational cost. Initially, GA created populations of strings with each string representing an ordering of the design process. The subsequent generation of population was created based on the selection, crossover and mutation operation on the current population. Selection of the string from the pool of population was based on the value of the objective function of each string. The strings with best objective values were selected for crossover. Crossover is the operation of mating of two strings in the hope of producing a child string with better objective values. Cross over operation was accomplished by position based cross over (Syswerda, 1990). Finally, the mutation operation was performed through the order-based mutation operation. The objective function used by GA is shown in Equation 2-1, where f is the number of feedback, c is the number of crossovers, time is the total time required to converge the circuit, cost is the total cost to converge the circuit; and wf, wc, wtime and wcost are user-definable weights.

Equation 2-1 Fitness function for rearrangement of the DSM

$$fitness = 1.0/((wf * f + wc * c + wtime * time + w\cos t * \cos t) **4)$$

Figure 2-8 shows an example DSM before and after rearrangement. It can be noted that the total cost has reduced from 19,640 to 3,950 units and time from 21,340 to 4,570 units.

Rogers's approach required an estimate of the strength of the feedback loops for each arrangement of the design process, to calculate the objective values. The strength of the feedback loops was estimated based on the number of iterations required for the convergence of each loop in the system. This operation can turn out to be

computationally expensive, since each arrangement has to be executed at least once to obtain the strengths. Furthermore, excluding the strength of the feedback loops from the objective function could reduce the quality of the objective function.

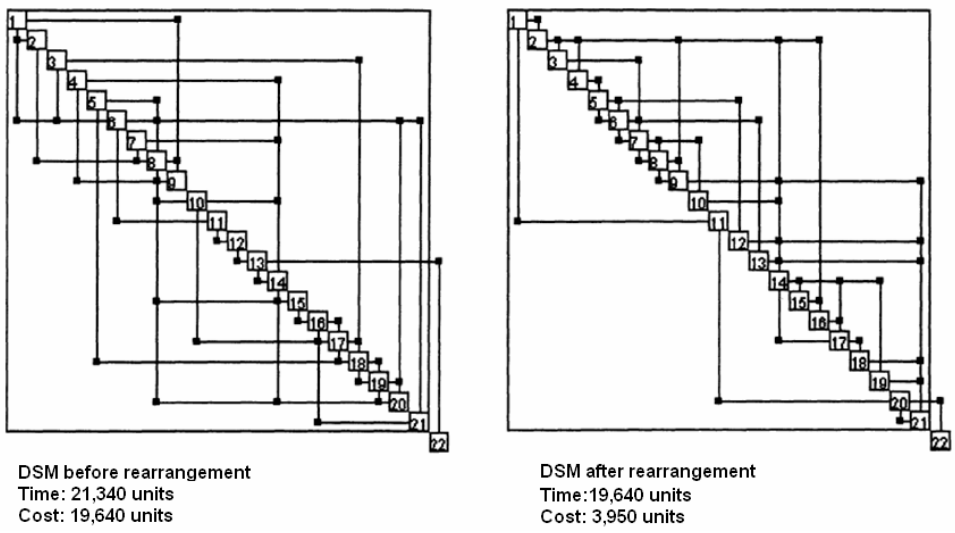


Figure 2-8 An example DSM before and after rearrangement (Rogers, 1997)

Altus et al. (1996) have developed a computer program called 'A GENetic algorithm for Decomposition of Analyses' (AGENDA). The methods used by Rogers for decomposition and sequencing of the design problem were further developed in this tool. Unlike Rogers's work this research was focused on the decomposition of the design problems for conducting multidisciplinary design optimization (MDO) studies. Furthermore, the approach focused also on decomposing large design problems into sub-problems besides arranging the analysis subroutines for efficient execution. The benefit of this type of decomposition was that the sub-problems could be executed in parallel, thus reducing the total computational time. In addition, the solving efforts in each sub-problem can focus on local information, and temporarily the details of other sub-problems can be ignored. A GA based decomposition approach was developed in this work. Similar to the Roger's approach, the GA population consists of string representing the ordering of the analysis subroutines. Crossover operation was accomplished by position based crossover and the mutation operation was performed through the order-based mutation operation. The decomposition to sub-problems was achieved by introducing "breaks", or boundaries between sub-problems. Including m breaks creates m+1 sub-problems. Then the string generated will be of length m+n

where n represents the number of analysis subroutines. For example, a task with 7 subroutines to be split into 3 sub-problems, n=7 and m=2. Figure 2-9 shows a sample genetic string and its corresponding computational system.

The objective function was formulated based on the Dependence matrix (DM). Dependency matrix is an extension of the DSM with integers in the off-diagonal elements. A value *DM(i,j)* represents the number of outputs from routine *i* which are inputs to routine *j*. The feedback length, which was the objective function that was formulated in this approach, is shown in Equation 2-2.

The feedback length which is used as the objective function in Altus's approach provides only a rough estimate of the iteration required for solving the system. The convergence of a coupled system depends on the execution time of each subroutine, the solver used, the strengths of the coupling, starting points and so forth. These aspects were not addressed in the formulated objective function in Equation 2-2. However, AGENDA offers the provision for user-defined objective functions.

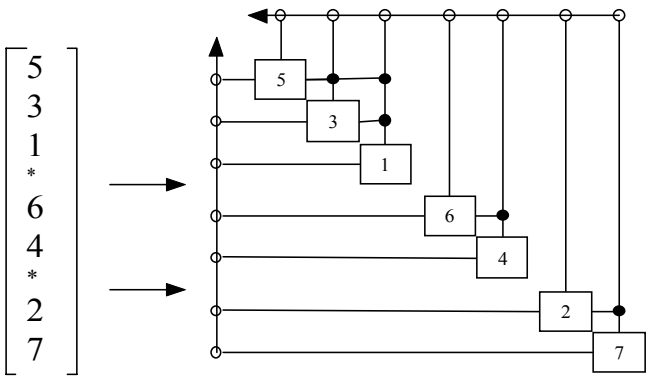


Figure 2-9 Decoding of genetic string into subroutine order and sub-problems

Equation 2-2 Objective function (feedback length) (Altus et al., 1996)

$$j = \sum_{i=2}^{n} \sum_{j=1}^{i-1} DM\ (i, j)(i - j)$$

Papalambros (1995) in his research has formulated the decomposition of a large scale system into subsystems, as an optimization problem. In his optimal model-based partitioning (OMBP) scheme the focus was to obtain a balance between the size or number of sub-problems and the interdependence among them. Fewer number of sub-systems interaction allowed concurrency, but more subsystems required more co-ordination effort. OMBP was formulated as an optimization problem with multiple objectives. The first objective was to minimize the size of the master problem by minimizing the number of linking variables and the second objective was to minimize the size of sub-problems by maximizing number of partitions of similar size which accounted for load balancing among sub-problems. The paper discusses Network Reliability Optimization formulation which is an OMBP formulation. The objectives here were to maximize the number of functioning links and to minimise a measure of network reliability. The formulation corresponds to identifying the critical linking variables and assigning their control to the master problem. The control of the remaining variables was assigned to the sub-problems. Figure 2-10 shows an example of pareto points obtained for each optimal partition.

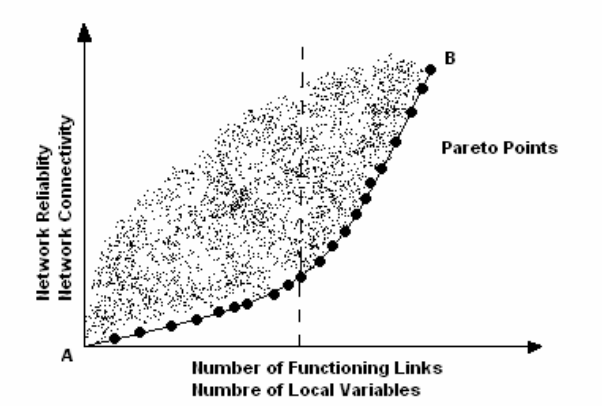


Figure 2-10 Pareto solution of an optimal decomposition problem (Papalambros, 1995)

At point A each variable is a linking one and the problem is fully disconnected. At point B each variable is a local and the problem is fully connected. Intermediate Pareto points are compromises and the selection can be performed based on the trade-offs between the size of the master problem and the sub-problems. The formal Pareto model for the optimal partitioning is shown in Equation 2-3.

Equation 2-3 Formal Pareto model for optimal partitioning

$$\min\left\{-\sum_{i=1}^{m} e_i ; \Phi_{PC}(e)\right\}$$
$$e \in \{0,1\}^m$$
$$\Phi_A^j(e) = 0 or 1$$

Φ_A and Φ_{PC} represent the measures of all terminal and pair-connected reliability and are functions of the binary edge indicator vector e, e_i=1 (0) if edge i is functioning (failed). *All-terminal* recognizes the edges that partition the network and *pair-connected* measures the equality in the size of the partitions.

Papalambros has solved the decomposition of large scale systems as a multi-objective optimization problem. He has studied the trade-offs of solving the problem as a whole or solving it as sub-systems. However, during conceptual design studies where a number MDO studies are conducted, introducing additional objective and constraints (for decomposition) can significantly increase the computational burden during this design stage.

2.3 Computational tools for Complex Systems

There are various computational design tools available for each stage of aircraft design. The literature review conducted by the author in this area has identified that the number of commercial tools available for preliminary and detailed stage were much higher compared to the number of tools available for conceptual design stage. Research in innovative concepts is conducted generally for those products which have a highly competitive market and requires continuous changes and improvement so as to meet or exceed the customer requirements. Examples of such products are automotive, aircraft etc. However, majority of the other products has their concepts being fixed and further research is conducted on these products for better performance mostly through the detailed design process. This could be one of the reasons for the limited number of industries which use computational tools for conceptual design compared to Computer Aided Design (CAD) and Computer Aided Engineering (CAE) tools which has application during the conceptual as well as the detailed design process. In addition, it should be also noted that some industries, especially aerospace have their own in-house

tools for conceptual design, but the details of the majority of these are unpublished because of confidentiality issues.

This section investigates some of the computational tools which have already been used or have the potential to be applied to computational studies in conceptual design. The section is divided into two sub-sections; the first part investigates the tools which can be applied to solve simple mathematical models of complex systems. The second part investigates the tools which are used for integrating high-fidelity mathematical models (CFD, CAD, etc.). The tools reviewed in the second part were some of the well known tools which have been used primarily during preliminary and detailed design stages. In contrast, the present research investigates the applicability of such high-fidelity integration tools to the conceptual design stage.

2.3.1 Computational tools for low-fidelity mathematical models

This section investigates computational tools for solving low-fidelity mathematical models at conceptual design stage. The variable flow modelling and system decomposition and scheduling methods used in some of the tools were discussed in the previous chapter and will not be discussed further.

2.3.1.1 Concept modeller

Concept modeller (Serrano, 1987) was one of the earliest tools developed for constraint management. The design problems were represented and solved as a constraint satisfaction problem. Graph theoretical methods were applied for constraint management. The variable flow modelling methods used in this tool were discussed earlier in section 2.2.1. The tool had the ability to detect the under and over-constrained systems and also was able to identify the redundant and conflicting constraints. In addition, the tool also had the capability to handle equality and inequality constraints.

Concept modeller was developed for constraint management, but was not robust enough to manage the dynamic solving capability required for conducting conceptual design studies. The system was developed mainly for solving system of non-linear equations and had very little reference to solving models.

2.3.1.2 Design Sheet

Design-Sheet (Buckley et.al, 1992) is a computational tool for conducting flexible trade-off studies during conceptual design stage. The tool was developed mainly on the ideas from Concept Modeller, which were advanced further to make it suitable for conceptual design studies. Concept Modeller was a tool resulting from academic research, while Design Sheet is a commercial tool.

Design sheet permitted the user to enter a set of algebraic equations and create computable systems based on the input variables defined by the user. The designer was also given the flexibility to change the input variables and the tool would create the corresponding computable system. The computational process modelling diagram for design sheet is shown in Figure 2-11.

The tool also had the provision for conducting optimisation, trade studies and sensitivity analysis on the system models.

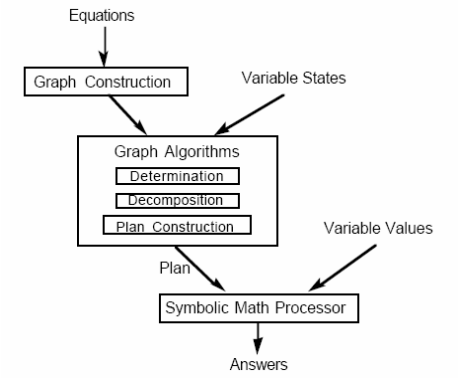


Figure 2-11 Workflow diagram for Design Sheet (Buckley et.al, 1992)

As in the case of Concept Modeller, Design Sheet was developed for solving algebraic equations and not models. The variable flow modelling method used by this tool, which was discussed in section 2.2.1, considers only single output produced by the algebraic equation which would not be the case for models, where multiple outputs are generated. Also Design sheet does not investigate the alternative variable flow models that could be produced for a given set of input variables. Arrangement of the execution sequence of the equations which are strongly connected, and which could have made significant reduction on the convergence time of the system, was not investigated either.

2.3.1.3 DeMAID

Design Manager's Aid for Intelligent Decomposition (DeMAID) is a tool developed by NASA for decomposing large, complex multidisciplinary processes (Rogers, 1997). DeMAID identifies the iterative sub-cycles in a design process, arranges these to reduce the feedback couplings and sequences the iterative sub-cycles and in a hierarchical order. The sequencing of the iterative sub-cycles is performed using GA. The sequencing scheme used in DeMAID is explained briefly in section 2.2.2.2. In DeMAID, the user is able to enter the details of the models, decompose and finally execute these to obtain the results.

DeMAID software was more a decomposition and sequencing tool rather than a conceptual design tool where design studies could be conducted. The flexibility provided by Design Sheet in terms of choosing the input variables for the system was limited in DeMAID. Here, the input variables of the system are fixed according to those variables which are only inputs to the models in the system and were not output of any of the models in the system under consideration. The variable flow modelling which could improve the dynamicity in selecting the input variables for the system was not considered in this tool. This limitation can significantly affect the flexibility required in choosing the inputs while performing conceptual design studies.

2.3.1.4 AGENDA

A GENetic algorithm for Decomposition of Analyses (AGENDA) was a computer program developed at Stanford University for decomposition of system into subsystems (Altus et.al., 1996). The decomposition scheme applied in this tool was described earlier in the section 2.2.2.2. The decomposition scheme could reduce a system to subsystems based on the decomposition criteria set by the designer. AGENDA has the provision for the designer to input his/her own decomposition criteria for splitting the system into subsystems. Compared to DeMAID, AGENDA exhibited similar drawbacks in terms of flexibility in choosing the input variables to the system.

2.3.2 Computational tools for high-fidelity mathematical models

This section investigates several well known tools for integrating and solving high-fidelity mathematical models (e.g. CFD, FEA tools) which are generally used during the

preliminary and detailed design stages of the design. The review of these tools was performed to investigate their applicability to the conceptual design stage.

2.3.2.1 Fiper

Fiper is a software framework for integrating various design tools in a coordinated fashion into a single environment (Wujek et al., 2002). This aids the designer to conduct an overall design analysis in a single space, and focuses more on the product analysis and development rather than on the tool integration aspects. Fiper was developed by Engineous Inc. in a collaborative effort with General Electric, Goodrich, Parker Hannifin, Ohio University, the Ohio Aerospace Institute and Stanford University. Figure 2-12 shows a pictorial view of the FIPER architecture.

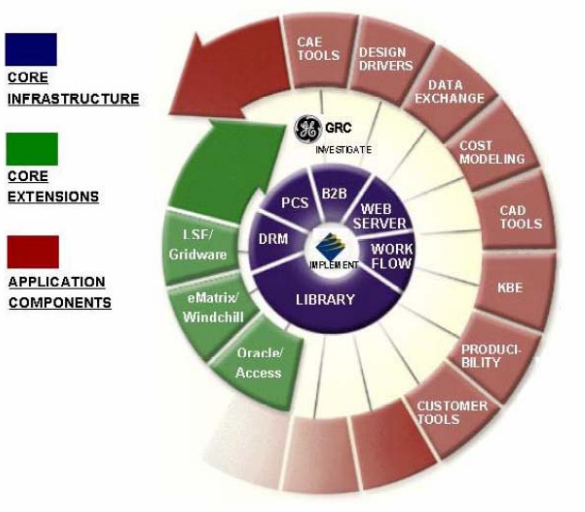


Figure 2-12 Overview of the FIPER architecture (Wujek et al., 2002)

Fiper incorporates various tools as components, by using a Java-based wrapping mechanism. This component-based architecture provides seamless integration of various heterogeneous tools located locally and also in a geographically distributed environment. Fiper also provides parallel execution of tools where necessary thus reducing the computing time.

Fiper is meant for integrating high-fidelity analysis tools and was developed with focus on the preliminary design stages of the design rather than conceptual design. The facility for automated integration of the tools was not addressed in Fiper. However the

wrapping mechanism incorporated in this tool could be a potential application which can be incorporated in a tool for conceptual design stage.

2.3.2.2 Spineware

Spineware (Vankan and Laban, 2002) is a software tool developed at NLR for integrating and executing high and low fidelity design and analysis tools in a distributed and heterogeneous computing environment. Spineware CDE (Computational Design Engine) was developed as part of the MOB (Morris, 2002) project and has played a significant role in the design and analysis of the BWB (Blended Wing Body) aircraft. Spineware is defined as "An object-oriented system that supports the construction and usage of user-defined working environments in heterogeneous computer networks".

Spineware consists of two layers. The first layer is the system level implementation of the Computational Design Engine (CDE), which includes a number of design and analysis tools. The second layer, the SPINEware User Shell, facilitated a more user-friendly environment for the designer to operate on the CDE. The second layer was developed based on object-oriented concepts and the object interaction was accomplished using CORBA standards. The java applet implementation of the SPINEware User Shell provided accessibility of the objects via web browsers. Figure 2-13 (b) shows an example for Spineware object browser.

Spineware was developed for integrating high fidelity analysis tools, primarily for the preliminary design stage of the aircraft design. In the current research context hundred of simple mathematical models were needed to be integrated and solved in a dynamic environment.

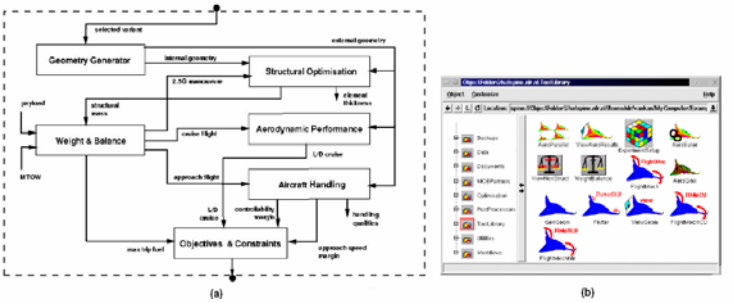


Figure 2-13 (a) Functional structure of the design process of the BWB in the CDE (b) An example for Spineware object browser (Vankan and Laban, 2002)

2.3.2.3 Phoenix Integration

Phoenix integration is a software tool for integrating heterogeneous computing applications to generate a more efficient engineering process (Scott, 2001). The software is developed based on a client-server configuration. The software has three components for integrating the tool:

- The Enterprise Analysis Server (EAS) is used to create wrappers for the various tools which have to be integrated into a single environment. The wrapping allows for converting the analysis software into reusable tools that can be published for others to use. EAS is java-based software server. The published components can be accessed from any computer on the network regardless of the platform.
- The ModelCenter has the capability to access the wrapped applications and build the engineering process. ModelCenter provides a graphical user interface for linking the wrapped applications. It also provides an open Application Protocol Interface (API), which can be called from C++, Java or COM, for the designers to access the integration engine behind the user-interface. ModelCenter has various in-built design exploration tools such as optimisation, design of experiments and response surface modelling.
- ModelRunner is built in mind for the designers who do not want to wrap their code with EAS, but want to perform trade-studies on already built models. ModelRunner is provided with a graphical user interface. Most of the design exploration tools mentioned earlier is also present in the ModelRunner. ModelRunner is provided with an API for integrating third party trade study algorithms.

Limitations of FIPER which makes it unsuitable to be applied in the conceptual design studies are encountered in Phoenix integrations also.

2.3.2.4 AML

The Adaptive Modelling Language (AML) is an object-oriented, knowledge-based engineering modelling framework. AML is the main product of Technosoft Inc. AML

consists of the following components, each tailored for various purposes (Zweber, 2002):

- TIE provides a completely graphical environment for product and process modelling, integration, and optimization. TIE contains a wide selection of modules for interface with common engineering tools.
- AMOpt is a suite of tools for performing optimization and probabilistic design studies within AML applications and TIE models. AMOpt enables multidisciplinary system-level trade studies such as cost versus performance. The AMOpt suite includes the following methods and algorithms: Multi-Objective Genetic Algorithm (GA), Design of Experiments (DOE), Sequential Quadratic Programming (SQP), Powell Method, Nelder-Mead Simplex Method, Monte Carlo Simulation and Response Surface Methodology (RSM)

Similar to FIPER and Phoenix Integration, AML is also a tool for integrating various design and analysis software. The limitations in FIPER and Phoenix Integration are also found in AML, which makes it currently unsuitable for the conceptual design phase.

2.4 Summary and Conclusions

The majority of the methods for variable flow modelling reviewed in this chapter are applicable to algebraic equations. These methods need modification for application in the current research context, where models are used instead of algebraic equations. Furthermore, most of the methods reviewed here focussed on obtaining single feasible variable flow model for solving a system, while there can be multiple feasible variable flow models for a system. This limits the chances of obtaining the one which could lead to relatively shorter execution time.

The above mentioned drawbacks pose the research need for developing a novel variable flow modelling method which could overcome the limitations of solving the models and also for exploring multiple feasible variable flows.

The decomposition and scheduling section was sub-divided into methods for process management and computational systems. The methods reviewed in process management

were relevant for identifying strongly coupled processes. These methods have the potential for identifying the SCCs in a system, in the current research context.

The methods reviewed in the computational systems section are used for decomposing and scheduling the models for faster convergence. Except the Rogers method, all other methods were for decomposing the strongly coupled models into sub-problems in order to execute them in parallel for faster convergence. Rogers's method was for scheduling the models belonging to iterative sub-cycles (SCC), based on the values of the objective function calculated for various arrangements of the constituent models. These objective functions are design problem and solver dependent, and therefore needs further investigation for applying to the current research context.

The current computational process modelling methods which were available for models have focussed only on decomposition and scheduling and not on variable flow modelling (e.g. DeMAID). This has limited the flexibility for choosing the user defined combination of inputs to the system of models, and thus limits the effectiveness of conducting design studies. Therefore there is a research need to combine the variable flow modelling methods and the decomposition and scheduling methods to generate a novel computational process modeller which ensures the flexibility in choosing the inputs to the system and to improve the efficiency (through decomposition and scheduling) of solving the system.

Various computational tools for low and high fidelity mathematical models were reviewed to identify the potential of those tools in the conceptual design stage. Among the low-fidelity tools, Concept Modeller and Design Sheet were developed for solving algebraic equations and AGENDA and DeMAID for solving the models. Even though the latter tools were able to solve the models, they lacked the flexibility in selecting the inputs for the system.

High-fidelity tools included various novel mechanisms for integrating different CAE tools into a single environment. However, these tools lacked the automated integration capability and the flexibility required for integrating numerous simple models in a dynamic environment. This poses a need for the development of a conceptual design tool which ensures the above mentioned limitations are tackled.

3 COMPUTATIONAL PROCESS MODELLING FOR COMPLEX SYSTEMS

3.1 Introduction

Presented in this chapter is a novel method for computational process modelling for complex systems. The limitations of the most recent methods were described in detail in section 2.4. The novel process modeller tackles these limitations and generates a computational arrangement for the system, which leads to shorter execution time. Section 3.2 describes the computational process modeller in detail with the help of a flow chart. The individual features of the computational process modeller including the novel variable flow modelling method along with the decomposition and scheduling methods are described in the corresponding sub-sections. Section 3.3 describes a comprehensive example which demonstrates the working structure of the entire computational process modeller. Finally summary and conclusions are presented in Section 3.4.

3.2 The Computational Process Modeller

Computational process modelling is the process of organising a complex system, in order to efficiently compute the output variables, according to the specific input variables selected by the designer. As described in Chapter 1, while performing trade studies the designer selects various combinations of input variables and further computational process modelling has to be performed to obtain the computational plan for the system.

In this section a computational process modeller is introduced for computational process modelling and is depicted in the Figure 3-1.

A detailed explanation of the flow chart is given in the later sections. Each box in the flowchart is briefly explained here as follows.

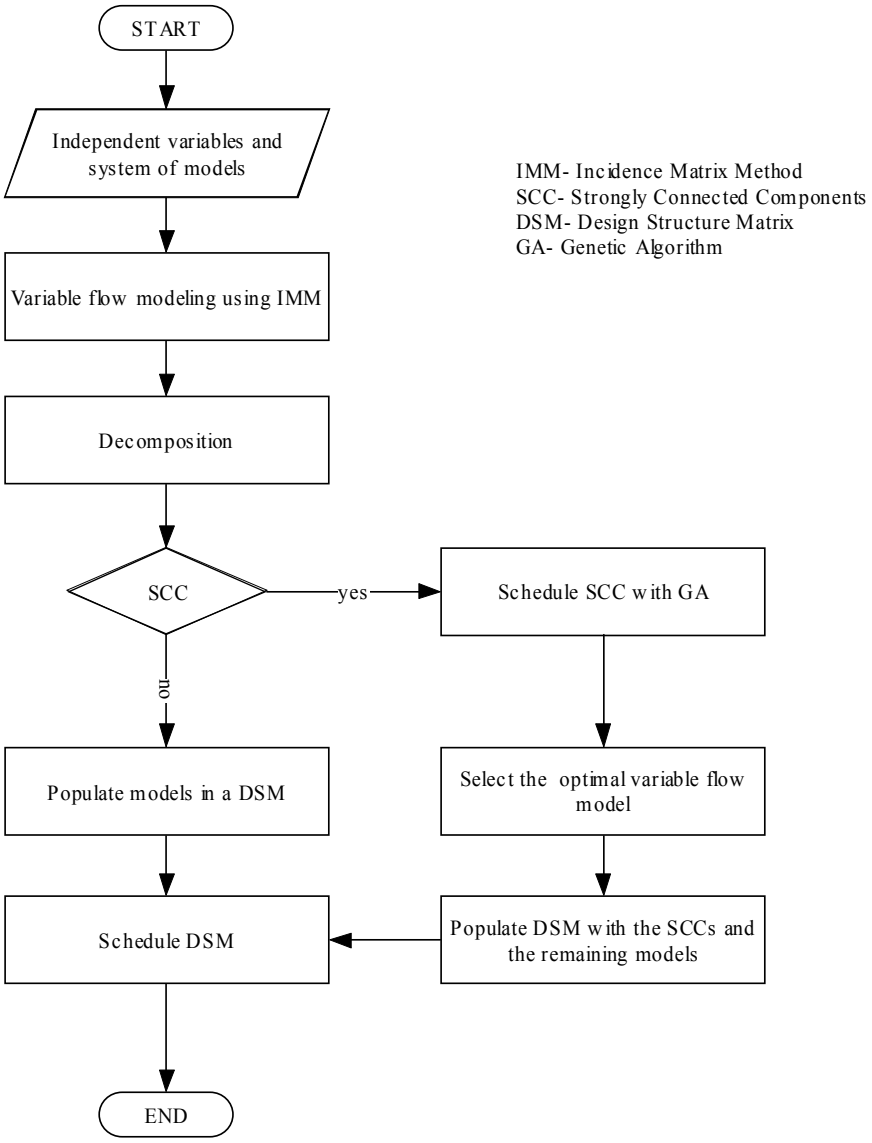


Figure 3-1 The computational process modeller

START

- ***Independent variables and system of models:*** Initially, the designer provides the system of models with a choice of independent (input) variables for computational process modelling. Independent variables are those variables from among all the variables associated with the models in the system, for which the values are known (e.g. the requirement for the aircraft (e.g. Range, MTOW etc), which are known in advance, can be defined as independent during the design study process) and which the designer decides

to provide as inputs to the system. Hence these variables should be always inputs to the models in the system so that recalculating of these variables by the models within the system can be avoided.

- ***Variable flow modelling using IMM:*** Variable flow modelling is performed using the incidence matrix method (IMM) to determine the information (data) flow among the models. All feasible variable flow models of the system are explored in this step.
- ***Decomposition:*** Each variable flow model generated is decomposed into hierarchically decomposable and non-hierarchically decomposable system of models. Non-hierarchically decomposable systems are also called as strongly connected components (SCC).
- ***Schedule SCCs with GA:*** Given a SCC, its constituent models are rearranged by means of Genetic Algorithm (GA).
- ***Select the optimal variable flow model:*** The optimal variable flow model is selected in this step based on the value of the objective function (obtained after rearranging the constituent SCCs) and the number of modified models in each variable flow model.
- ***Populate DSM with the SCCs and the remaining models****:* Each of the rearranged SCCs is regarded as a single model and along with the remaining models and is populated in a DSM. The population is based on the data flow depicted in the selected optimal variable flow model.
- ***Populate models in a DSM:*** If SCCs do not exist, then the models are populated directly in a DSM based on data flow obtained from the variable flow model.
- ***Schedule DSM:*** The DSM is rearranged into a lower triangular matrix based on a graph theoretical algorithm. This rearrangement eliminates the feedback loops and thus the final computational plan is obtained for the system.

.***END***

The following sub-sections explain in detail the significant individual tasks mentioned above.

3.2.1 Variable flow modelling

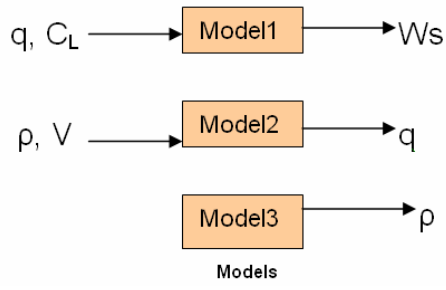
Variable flow modelling is the process of identifying the data flow among the models within a system according to the input (independent) variables selected by the designer.

A novel incidence matrix method (IMM) is proposed here which dynamically obtains the information flow within the system. The first sub-section of this section explains the incidence matrix method, which populates the matrix on a rules based approach. The following sub-section describes the formalisation of this method.

3.2.1.1 Variable flow modelling using incidence matrix method

In mathematics, an incidence matrix is a matrix that shows the relationship between two classes of objects. If the first class is X and the second is Y, the matrix has one row for each element of X and one column for each element of Y. The entry in row x and column y is 1 if x and y are related (called incident in this context) and 0 if they are not (Weisstein, 2007).

In the current research context, the two classes of objects are the models and the variables. In the incidence matrix, each row denotes a model and each column a variable. An entry '*' in the matrix symbolises that the variable representing the corresponding column is either an input or output of the model in the corresponding row. An example of initial incidence matrix for a simple system is shown in Figure 3-2.



	Ws	C_L	q	ρ	V
Model1	*	*	*		
Model2			*	*	*
Model3				*	

Incidence Matrix

Figure 3-2 Incidence matrix for a system of three models

The aim of the incidence matrix method is to substitute the '*' in the matrix with either 'o' (signifying output) or 'i' (signifying input) depending on whether the variable in the

corresponding column will be an input or an output of the model in the corresponding row. This replacement procedure will be called 'population of incidence matrix'. The incidence matrix is populated from its initial state according to the rules stated below.

Rule1: An independent variable should be always input to the models.

Rule2: If a variable is associated with only one model and if it is not an independent variable, then it should be the output of that model.

Rule3: If a model is associated with only one variable, that variable should be output of that model

Rule4: Each variable should be output of at most one model in the system.

Rule5: The number of outputs identified through variable flow modelling, for a particular model, should correspond to the number of outputs of the original model.

Given below is further explanation of the meaning of each rule.

Rule1: This rule implies that all the '*'s in the columns of the independent variables should be replaced with 'i's.

This replacement ensures that the independent variables are always input to the system.

Rule2: This rule implies that if the elements of a column are all empty except a single '*' entry, and if the corresponding variable, which represents the column, is not an independent variable then that '*' should be replaced with an 'o'.

In this case the variable is associated with only a single model, and if this variable is not marked as independent (which means the variable is not input to the model), then the only relation for the variable to model is to be an output.

Rule3: This rule implies that if the elements of a row are all empty except a single '*' entry, and if the corresponding data variable in the column is not an independent variable, then the '*' should be replaced with an 'o'.

The case signifies a model with a single variable. If this variable is not output of the particular model then there was no requirement for such a model in the system and hence the variable is as marked as output

Rule4: This rule implies that except for the columns of the independent variables, all other columns should have 'o' marked in exactly one element.

Each column represents a variable, and since each variable has to be computed, they should be output of a model from the system.

Rule5: This rule implies that every row should have the same number of ‘o’s as the number of outputs of the associated model.

As mentioned earlier certain models have their input and output variables interchanged as a result of variable flow modelling. These modified models are solved using non-linear least-square problem solvers (e.g. Gauss-Newton method) (refer section 4.2). Rule5 ensures that these modified models are determined (not under or over determined) and therefore solvable using the non-linear least square problem solvers.

The above five rules are applied on the incidence matrix in a particular sequence thereby populating the matrix in order to obtain the final information flow. It has to be ensured that while populating the incidence matrix by applying a particular rule (applied for a ‘*’ considering its row/column), other rules (when verified for the same ‘*’ considering its column/row) are not violated. The flow chart which explains the sequence of application of the rules is shown in Figure 3-3.

The flow chart is briefly explained as follows:

1. Incidence matrix is initially created for the models and the variables.
2. The ‘*’s in the columns of the independent variables are replaced with ‘i’s based on Rule1.
3. Further Rule2 is applied which replaces the ‘*’ with ‘o’ in the single-element-columns of the matrix.
4. Rule3 is then applied on the matrix which replaces the ‘*’ with ‘o’ in the single-element-rows of the matrix.
5. Further, each remaining ‘*’ in the matrix is scanned and examined once, whether they could be replaced with either ‘i’ or ‘o’, based on the logic from rules 4 and 5.
6. Once all the ‘*’s are scanned and examined, and if there are still some ‘*’s remaining with out being replaced, then the preceding step (step5) is repeated iteratively until all the ‘*’s are substituted.

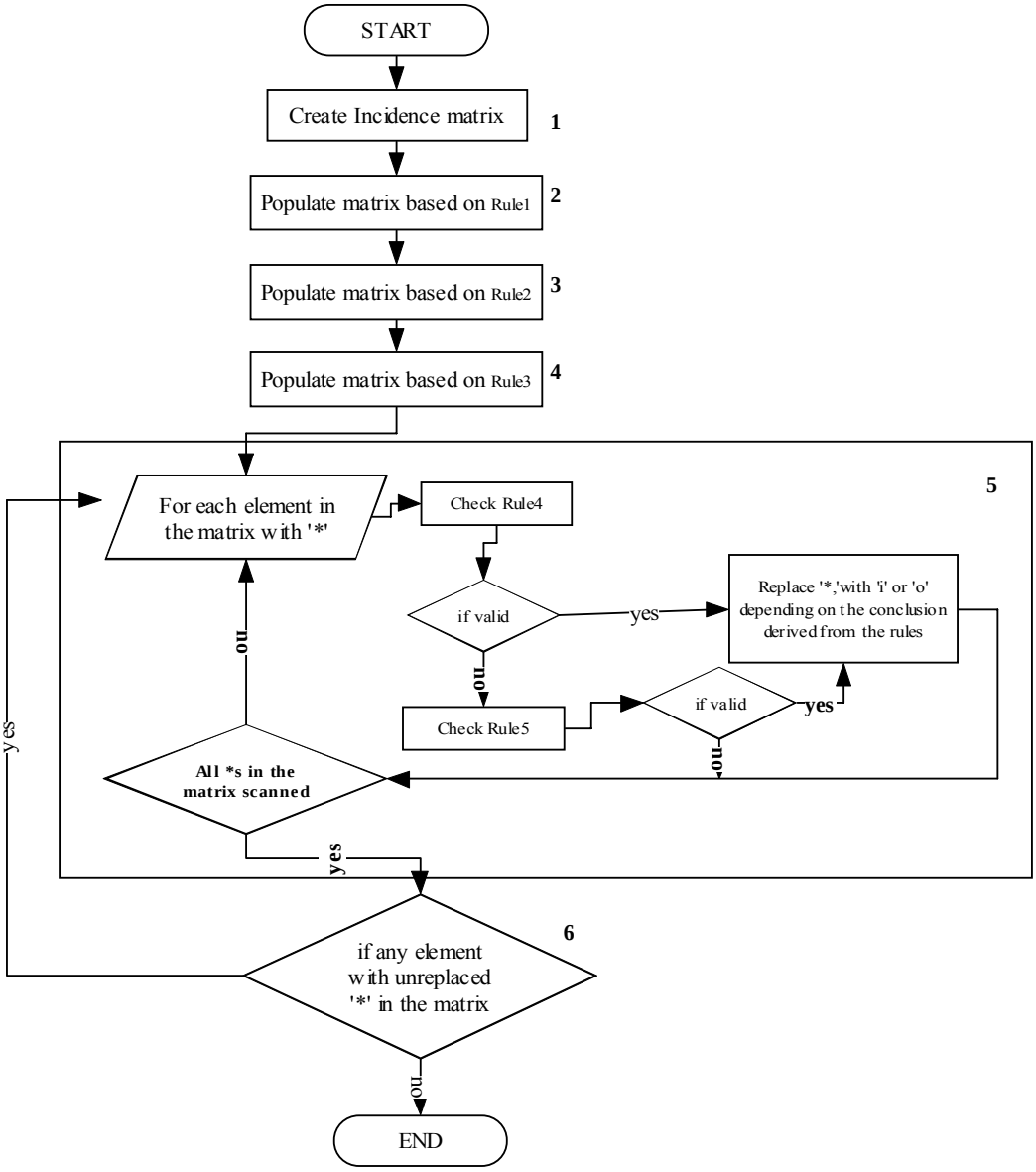


Figure 3-3 Flow chart for incidence matrix method (IMM)

The procedure is explained with two simple examples. The first example has a set of three models in the system. The second example has four models with one of the model generating multiple outputs.

Example1

Figure 3-4 shows a simple set of models for balancing the weight of an aircraft with its lift (Buckley et al., 1992). The example models are for demonstration purpose only. Data variables entering the models are the inputs and data variables leaving the models

are the outputs. In this example, Ws and V are chosen as independent variables for the system (The aim of IMM is to generate variable flow models for a system, given any set of independent variables. In order to demonstrate this capability of IMM, in the current example Ws and V were randomly chosen as independent from the set of variables in the system).

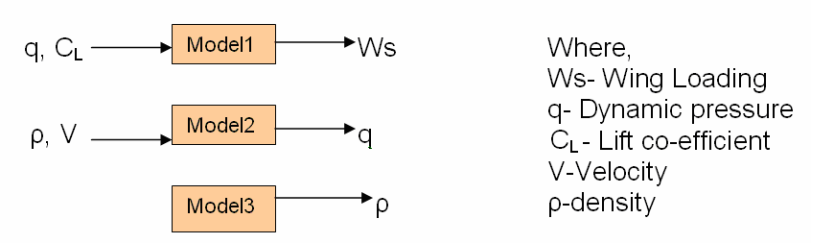


Figure 3-4 Models balancing the weight of aircraft with its lift

Population of the corresponding incidence matrix for the models in Figure 3-4, based on the six steps depicted in Figure 3-3, is described below.

Step1:

The initial incidence matrix is created as shown in Figure 3-5.

	Ws	C_L	q	ρ	V
Model1	*	*	*		
Model2			*	*	*
Model3				*	

Figure 3-5: Initial Incidence matrix for models in Figure 3-4

Step2:

Since Ws and V are the independent variables, according to rule1, the '*'s in the corresponding columns of the variables are substituted with 'i'. The updated matrix is shown in Figure 3-6.

	Ws	C_L	q	ρ	V
Model1	i	*	*		
Model2			*	*	i
Model3				*	

Figure 3-6 Incidence matrix for system in Figure 3-4 with independent variables defined

Step3

According to rule2, the '*' in the single-element-column (column2 in the current example) is replaced with 'o'. The updated matrix is shown in Figure 3-7.

	Ws	C_L	q	ρ	V
Model1	i	o	*		
Model2			*	*	i
Model3				*	

Figure 3-7 Incidence matrix for system in Figure 3-4 after applying rule2 on element (1,2)

Step4

According to rule3 the '*' in the single-element-row is to be replaced with 'o'. The updated matrix is shown in Figure 3-8.

	Ws	C_L	q	ρ	V
Model1	i	o	*		
Model2			*	*	i
Model3				o	

Figure 3-8 Incidence matrix for system in Figure 3-4 after applying rule3 on element (3,4)

Step5

In this step each '*' in the incidence matrix is scanned to examine whether it could be replaced with either an 'i' or 'o'. The elements (1, 3), (2, 3) and (2, 4) are scanned in this step. The first element (1,3), based on rule 5, is replaced with 'i'. Since here the number of outputs for the original model1 is one, and since variable C_L is already defined as the output of model1, the variable q is defined as input for the model. The updated incidence matrix is given in the Figure 3-9.

	Ws	C_L	q	ρ	V
Model1	i	o	i		
Model2			*	*	i
Model3				o	

Figure 3-9 Incidence matrix for system in Figure 3-4 after applying rule5 on element (1,3)

The second element (2,3) in the list is now replaced with 'o' based on rule4. Each variable in the system has to be output of at least one model and since the variable q is already defined as input to model1 it has to be output of model2. The updated incidence matrix is given in the Figure 3-10.

	Ws	C_L	q	ρ	V
Model1	i	o	i		
Model2			o	*	i
Model3				o	

Figure 3-10 Incidence matrix for system in Figure 3-4 after applying rule4 on element (2,3)

The third element (2,4) in the list is now replaced with 'i' based on rule4. The updated incidence matrix is given in the Figure 3-11.

	Ws	C_L	q	ρ	V
Model1	i	o	i		
Model2			o	i	i
Model3				o	

Figure 3-11 Incidence matrix for system in Figure 3-4 after applying rule4 on element (2,4)

Step6

Since all the '*'s in the incidence matrix have been replaced, there is no requirement for further iteration through step 5.

Figure 3-11 shows the final populated matrix obtained for the system. The final matrix indicates that model1 has Ws and q as input and C_L as output, model2 has ρ and V as input and q as output and model3 has ρ as output.

Example 2

This example has four models with model 'std_atmos' producing multiple outputs. The models are shown in Figure 3-12. The models are part of an aircraft sizing problem. Variables Tamb, alt, MTOW, Aref, Kvs, rho and FNslst are selected as independent variables for this case.

Step by step population of the corresponding incidence matrix for this system of models is shown in Figure 3-13 . It can be noted that the matrix was fully populated after the first pass of step5 of the algorithm shown in Figure 3-3. Further iterations were not necessary in this case also.

The two examples demonstrated above were trivial. Solving the incidence matrix was straightforward and always led to a solution. This is not the case for systems which are either under or over-determined (explained in the next paragraph) or which have SCCs.

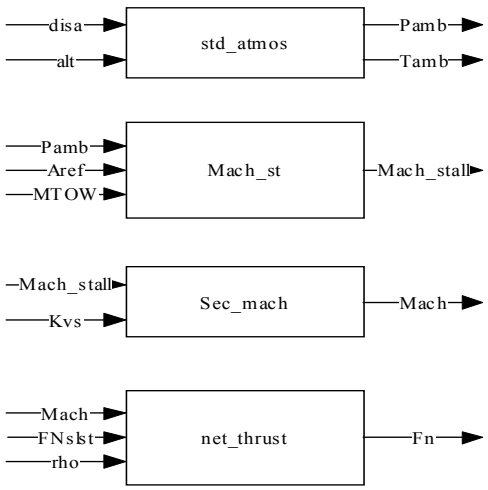


Figure 3-12 Models for example2

(a) Step1

	Tamb	Alt	Pamb	disa	MTOW	Aref	Mach_stall	Kvs	Mach	FNslst	Fn	rho
std_atmos	*	*	*	*								
Mach_st			*		*	*	*					
Sec_mach							*	*	*			
net_thrust									*	*	*	*

(b) Step2

	Tamb	Alt	Pamb	disa	MTOW	Aref	Mach_stall	Kvs	Mach	FNslst	Fn	rho
std_atmos	i	i	*	*								
Mach_st			*		i	i	*					
Sec_mach							*	i	*			
net_thrust									*	i	*	i

(c) Step3

	Tamb	Alt	Pamb	disa	MTOW	Aref	Mach_stall	Kvs	Mach	FNslst	Fn	rho
std_atmos	i	i	*	o								
Mach_st			*		i	i	*					
Sec_mach							*	i	*			
net_thrust									*	i	o	i

(d) Step5

	Tamb	Alt	Pamb	disa	MTOW	Aref	Mach_stall	Kvs	Mach	FNslst	Fn	rho
std_atmos	i	i	o	o								
Mach_st			i		i	i	o					
Sec_mach							i	i	o			
net_thrust									i	i	o	i

Figure 3-13: Population of the incidence matrix for the models in the system in Figure 3-12.

For these system certain '*'s could never be replaced by iterating through step 5 and 6 of the incidence matrix method. Hence the IMM flow chart in Figure 3-3 generates feasible variable flow models only for determined systems which do not have the presence of SCCs. These three cases along with their resolution schemes are explained in the following paragraphs.

Over and under-determined systems

Under-determined systems are those systems where the number of independent variables set by the designer is less than the number required for solving the models in order to compute all unknown variables in the system.

Over-determined systems are those systems where the number of independent variables set by the designer is greater than the number required for solving the models in order to compute all unknown variables.

For a system of algebraic equations, to derive a unique solution, the number of unknowns should be equal to the number of equations. The same theory is applicable for a system of non-linear models. However, since the models can have multiple outputs, the criteria for determining the solvability of a system are shown in Equation 3-1.

Equation 3-1 Criteria for determining the solvability of a system

$TNvar - NIvar\text{-}Noutmod = 0 \rightarrow$ *determined system*

$TNvar - NIvar\text{-}Noutmod > 0 \rightarrow$ *under-determined system*

$TNvar - NIvar\text{-}Noutmod < 0 \rightarrow$ *over-determined system*

Where,

TNvar-Total number of variables

NIvar- Number of independent variables

Noutmod- Sum of the total number of outputs of each model in a system

Noutmod in Equation 3-1 accounts for the multiple outputs generated by the models.

The Equation 3-1 is derived from the following equation

Equation 3-2 Derivation for Equation 3-1

Nmod= TNvar-NIvar-(Noutmod-Nmod)

The RHS of the Equation 3-2 calculates the number of unknown variables in the system, taking into account the multiple outputs generated by certain models. Since the multiple outputs for a model can be calculated simultaneously once that model's inputs are known, these outputs are considered as a single unknown variable in the above equation. This is taken in to account by the term Noutmod-Nmod. The value calculated in the RHS (Number of models) should equal to the LHS (Number of unknowns) signifies thus a determined system.

Thus for example 1 (Figure 3-4)

TNvar=5

NIvar=2

Noutmod=3

TNvar –NIvar-Noutmod = 5-2-3 = 0

Thus the system is determined.

And for example 2 (Figure 3-12)

TNvar=12

NIvar=7

Noutmod=5

TNvar –NIvar-Noutmod = 12-7-5 = 0

Thus the system in the example 2 is also determined.

It can be noted that in both examples if the number of independent variables (NIvar) provided by the designer was less than the one specified, then the system would have become under-determined, and if NIvar was greater than the one specified, the system would have been over-determined. In these cases a variable flow model could not have been achieved by following the flow chart shown in Figure 3-3.

For resolving the under-determined system the designer will have to choose additional variables as independent so that the value obtained from 'TNvar –NIvar-Noutmod' will equal zero.

After population, the incidence matrix for the system in example1, with only Ws chosen as independent, is displayed in Figure 3-14 . Iterations through step5 and step 6 will not resolve the '*' in the matrix. This is because replacing the '*' (in element (2,2)) based on rule4, which states that each variable should be output of one model, by 'o', will violate rule5 for the model in the column. Alternatively, replacing the '*' with 'i' based on rule5 for the row will violate rule 4 for the column.

	Ws	q	C_L	V	ρ
model1	*i*	*i*	*o*		
model2		*		*o*	*i*
model3					*o*

Figure 3-14 Populated incidence matrix for an under-determined system.

Here, *TNvar –NIvar-Noutmod* (=5-1-3=1), is greater than zero and hence according to Equation 3-1 the system is under determined. The option to resolve this system to make it determined is to declare an additional variable, either q, C_L, V or ρ, as independent along with Ws.

Over-determined systems can be resolved by deselecting relevant independent variables so that the value obtained from 'TNvar –NIvar-Noutmod' will equal zero. For example, in Figure 3-15 variables Ws, C_L and V are declared independent, and the figure shows the corresponding populated incidence matrix. Equation 3-1 proves this system as over-determined (*TNvar –NIvar-Noutmod* =5-3-3= -1 < 0). If the '*' in the matrix is replaced with 'o', based on rule5, will violate rule 4. Alternatively, if the '*'is replaced with 'i', based on rule 4, will violate rule 5.

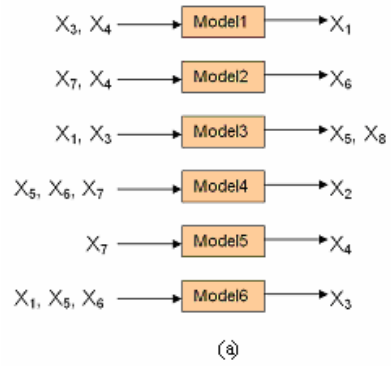
	Ws	q	C_L	V	ρ
model1	*i*	*o*	*i*		
model2		*i*		*i*	*
model3					*o*

Figure 3-15 Populated incidence matrix for an over-determined system.

The option to resolve such a system is to deselect the surplus independent variables so that the system remains determined. In the above example removing any one variable (Ws, C_L or V) from the set of independent variables will make the system determined.

Strongly connected components (SCC)

Presence of SCCs in a system also leads to unresolved '*'s in the populated incidence matrix. Unlike under and over determined systems, in the presence of SCCs the system is proven determined on applying Equation 3-1. However one or more of the constituent models of a SCC requires inputs from at least one other model from the same group. This characteristic of the SCC is the reason for the unresolved '*'s. An example of a system with a SCC along with its corresponding populated incidence matrix is shown in Figure 3-16.



	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Model1	*		i	*				
Model2				*		*	*	
Model3	*		i		*			o
Model4		o			i	i		
Model5				*			*	
Model6	*		i		*	*		

(b)

Figure 3-16 (a) System of models (b) Corresponding populated incidence matrix with X_3 as the independent variable

In this example variable X3 is given as the independent variable. Applying Equation 3-1;

TNvar –NIvar-Noutmod⇒ 8-1-7 = 0

The value obtained implies that the system is determined. However, Figure 3-16 (b) shows some '*'s which are unresolved, after applying IMM. This situation has arisen as a result of models 1,2,3,5 and 6 being strongly connected through shared variables. Hence those models which have still *'s remaining in the matrix after applying IMM is considered as part of SCC. Thus one additional advantage of the incidence matrix method is that the presence of at least one SCC can be identified while performing variable flow modelling. However, mutually exclusive SCCs that can occur in a system

could not be identified during incidence matrix method, and hence a decomposition scheme has to be applied for this purpose. The decomposition scheme is explained further in section 3.2.2.

As mentioned earlier if the system is proven determined, there is a solution existing for the system (see section 1.2.3.1). However, since the presence of SCC leads to unresolved incidence matrix (presence of unreplaced *s, after attempting to fully populate it), the next step is to resolve this problem. This is done by guessing the inputs and output variables of any one of the models belonging to the SCC and thereafter populating the matrix by applying IMM. Guessing the input and output variables can be made in a number of alternative ways and therefore each different guess will lead to a different variable flow model. Since our aim is to reduce the number of modified models (modified models leads to high execution time for the system, because of the numerical solving involved in the case of modified models (refer section 4.2)) in the system as much as possible, the criteria for selecting the models (for guessing the inputs and outputs) should be based on this objective. A new rule is introduced to account for this objective.

Rule6: Among the models which are part of a SCC for which not all '*' have been replaced after applying the first five rules of the IMM, the models for which the new inputs differ from the original ones are selected for guessing. If no such model exists, the incidence matrix is populated with the original inputs and outputs of the models.

Rule6 limits the unnecessary generation of modified models in a system and also the creation of alternative variable flow models to the ones which have minimum modified models. If there is more than one model which have its variables already modified then any of those models can be selected for guessing, one at a time.

In the presence of SCC the additional steps necessary for populating the matrix are depicted in Figure 3-17.

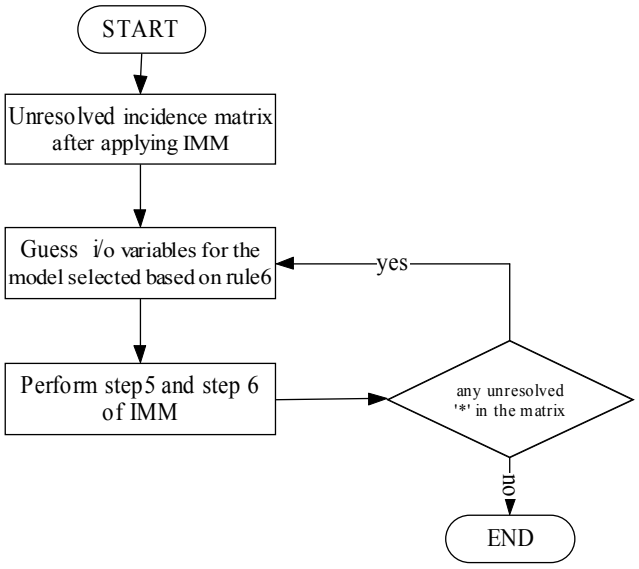


Figure 3-17 Additional steps for IMM in the presence of SCC

Example3.

The system shown in Figure 3-16(a) is considered in this example. Figure 3-16(b) shows the corresponding populated incidence matrix obtained, after applying the IMM. In the matrix, models 1,2,3,5 and 6 are strongly connected.

According to Figure 3-16(b) variable X_3 is input to model6, but the real model6 has X_3 as output (see Figure 3-16(a)). Such a modification is not present in any other constituent model of the SCC. Hence, based on rule 6, model6 is chosen for guessing the input and output variables. There are three guesses possible as shown in Figure 3-18.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Model6	o		i		i	i		

(a)

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Model6	i		i		o	i		

(b)

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Model6	i		i		i	o		

(c)

Figure 3-18 Alternative guesses for i/o variables of model6

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Model1	i		i	o				
Model2				i		o	i	
Model3	i		i		o			o
Model4		o			i	i		
Model5				i			o	
Model6	o		i		i	i		

(a)

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Model1	i		i	o				
Model2				i		o	i	
Model3	o		i		i			o
Model4		o			i	i		
Model5				i			o	
Model6	i		i		o	i		

(b)

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Model1	o		i	i				
Model2				*		i	*	
Model3	i		i		o			o
Model4		o			i	i		
Model5				*			*	
Model6	i		i		i	o		

(c)

Figure 3-19 (a) Populated incidence matrix after the arrangement of model6 as shown in Figure 3-18(a), (b) Populated incidence matrix after the arrangement of model6 as shown in Figure 3-18(b), (c) Incidence matrix after the arrangement of model6 as shown in Figure 3-18(c)

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Model1	o		i	i				
Model2				o		i	i	
Model3	i		i		o			o
Model4		o			i	i		
Model5				i			o	
Model6	i		i		i	o		

(a)

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Model1	o		i	i				
Model2				i		i	o	
Model3	i		i		o			o
Model4		o			i	i		
Model5				o			i	
Model6	i		i		i	o		

(b)

Figure 3-20 Populated incidence matrix for the arrangement of model6 as shown in Figure 3-18(c), in the second iteration

Each arrangement for model6 replaces in turn the sixth row of model6 of the populated matrix shown in Figure 3-16(b). Following this, step 5 and 6 of IMM is applied to populate the new matrix. The result obtained for each case is shown in Figure 3-19.

In Figure 3-19(c) it can be noted that no further population is possible and the matrix has reached the similar situation as before, with unresolved '*'s. This matrix now has to go through another iteration as specified in Figure 3-17. In the second iteration from the unresolved models 2 and 5, the former is chosen for guessing based on rule6. There are two possible alternative guesses of input and outputs variables for model2. After replacing the guessed variables in the matrix and further applying step6 of IMM the two flow models obtained are shown in Figure 3-20.

Thus in total, four alternative variable flow models are obtained for solving the system. These are shown in Figure 3-19(a), Figure 3-19(b), Figure 3-20(a) and Figure 3-20(b), respectively.

Example 4

This example is for the system shown in Figure 3-16(a) with variable X_7 as the selected independent variable. The corresponding populated incidence matrix after applying IMM is shown in Figure 3-21.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Model1	*		*	i				
Model2				i		o	i	
Model3	*		*		*			o
Model4		o			i	i		
Model5				o			i	
Model6	*		*		*	i		

Figure 3-21 Populated incidence matrix, for the system of models in Figure 3 17(a), with X7 as the independent variable.

The matrix shows that model1, model3 and model6 are strongly connected. However, here inputs or outputs currently defined for any of the models in the SCC are not different from those of the original model (Figure 3-16(a)).According to rule 6(the second sentence), the incidence matrix is populated with inputs and outputs of the original models. Thus the final populated matrix is shown in Figure 3-22 .

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Model1	o		i	i				
Model2				i		o	i	
Model3	i		i		o			o
Model4		o			i	i		
Model5				o			i	
Model6	i		o		i	i		

Figure 3-22 Populated incidence matrix, for the system of models in Figure 3-16(a), with X_7 as the independent variable, after resolving the SCC.

3.2.1.2 Variable flow modelling formalisation

In the previous section the incidence matrix method was explained in terms of six rules. This section describes an improved formal incidence matrix method which populates a numerical matrix instead of a character matrix. This method operates in a similar fashion to the earlier one, except that the rules are reformed into a mathematical procedure. This method reduces the memory (RAM) required since numerical arrays are used instead of character arrays. In addition, computer programming of the rules is straightforward since these are in a mathematical form.

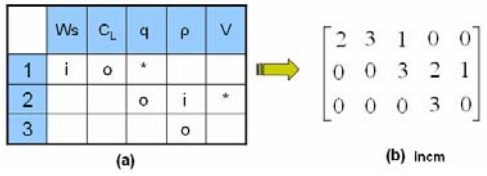


Figure 3-23 Incidence matrix earlier and latest representations.

Figure 3-23 shows the incidence matrix with its corresponding new representation, named as 'incm'. Presence of a variable in the model is represented with '1' ('*' used previously), an input by '2' ('i' used previously) and output with '3'('o' used previously) and '0' denotes no relation between the variable and the model. The population of the incidence matrix in this case signifies replacing 1 with either 2 or 3. The algorithm for populating the numerical incidence matrix is given in the flow chart shown in Figure 3-24.

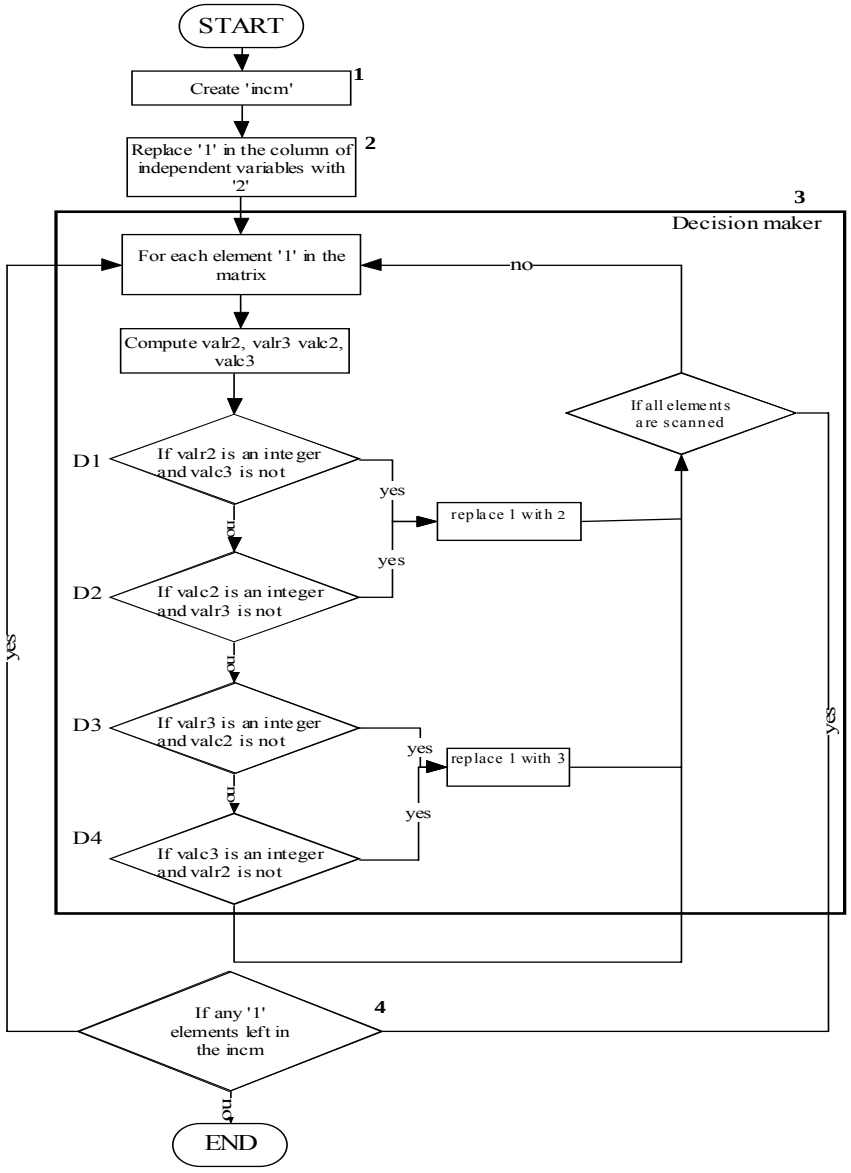


Figure 3-24 Flow chart for improved formal IMM

In the Figure 3-24;

incm- incidence matrix which will be populated for obtaining the variable flow model . An example for incm is given in Figure 3-23(b). Intially incm will have only ‘1’s and ‘0’s, on population using formal IMM the ‘1’s are replace with either ‘2’s or ‘3’s.

incmf- foundation incidence matrix, which corresponds to the real inputs and outputs of the model. The foundation matrix has the elements filled with ‘2’ and ‘3’ depending on inputs and outputs of the original model. The incmf for the system in Figure 3-4 will be,

$$\text{incmf}=\begin{bmatrix} 3 & 2 & 2 & 0 & 0 \\ 0 & 3 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Equation 3-3 Equation for calculating valrf(r)

$$valrf(r)=\prod_{c=1}^{m} incmf(r,c) \qquad ;incmf(r,c)\neq 0$$

Equation 3-4 Equation for calculating valcf(c)

$$valcf(c)=\begin{cases} 3 & \text{if } incmprod=2 \\ incmprod & \text{if } incmprod\neq 2 \end{cases}$$

$$Here,\ incmprod=\prod_{r=1}^{n} incmf(r,c)\ \ ;\ incmf(r,c)\neq 0$$

Equation 3-5 Equation for calculating valr(r)

$$valr(r)=\prod_{c=1}^{m} incm(r,c) \qquad ;incm(r,c)\neq 0$$

Equation 3-6 Equation for calculating valc(c)

$$valc(c)=\prod_{r=1}^{n} incm(r,c) \qquad ;incm(r,c)\neq 0$$

Equation 3-7 Equation for calculating valr2(r,c)

$$valr2(r,c)=\frac{\log\left(\frac{valrf(r)}{valr(r)}\right)}{\log(2)}$$

Equation 3-8 Equation for calculating valr3(r,c)

$$valr3(r,c)=\frac{\log\left(\frac{valrf(r)}{valr(r)}\right)}{\log(3)}$$

Equation 3-9 Equation for calculating valc2(r,c)

$$valc2(r,c) = \frac{\log\left(\frac{valcf(c)}{valc(c)}\right)}{\log(2)}$$

Equation 3-10 Equation for calculating valc3(r,c)

$$valc3(r,c) = \frac{\log\left(\frac{valcf(c)}{valc(c)}\right)}{\log(3)}$$

Here;

r represents the row number

c represents the column number

n represents the number of rows of the incm matrix

m represents the number of columns of the incm matrix

valrf(r) is the product of non-zero elements of row r of incmf

valcf(c) is the product of non-zero elements of column c of incmf

valr(r) is the product of non-zero elements of row r of incm

valc(c) is the product of non-zero elements of column c of incm

Values of valr2, valr3, valc2 and valc3 determine whether the '1' in the incm matrix should be replaced with either 2 or 3.

Explanation

Equation 3-7 could be rewritten as

$$2^{valr2(r,c)} = \frac{valrf(r)}{valr(r)}$$

The right hand side (RHS) of the above equation calculates the product of the values of the elements of row 'r' of the incm matrix with current value 1, as if these were replaced with combinations of '2's and '3's. If valr2(r,c) is an integer this means that RHS can

be represented as multiples of 2. It signifies that the values which can replace the '1's in the row r should all be '2's.

Equation 3-8 could be rewritten as

$$3^{valr3(r,c)} = \frac{valrf(r)}{valr(r)}$$

The RHS of the above equation calculates the product of the values, that the elements of the row 'r' with value 1 could be replaced, in the incm matrix. If valr3(r,c) is an integer means RHS is all multiples of 3. It signifies that the value that could be replaced for '1' in the row r should be all '3's.

Equation 3-9 could be rewritten as

$$2^{valc2(r,c)} = \frac{valcf(c)}{valc(c)}$$

The RHS of the above equation calculates the product of the values, that the elements of the column 'c' with value 1 could be replaced, in the incm matrix. If valc2(r,c) is an integer means RHS is all multiples of 2. It signifies that the value that could be replaced for '1' in the column c should be all '2's.

Equation 3-10 could be rewritten as

$$3^{valc3(r,c)} = \frac{valcf(c)}{valc(c)}$$

The RHS of the above equation calculates the product of the values, that the elements of the column 'c' with value 1 could be replaced, in the incm matrix. If valc2(r,c) is an integer means RHS is all multiples of 3. It signifies that the value that could be replaced for '1' in the column c should be all '3's.

The queries in the decision box (D1 to D4) of Figure 3-24 checks whether the replacement of 1s in the incm, with either 2 or 3 is achievable. The first sentence in the queries ensures that the element 1, which is in consideration for substitution, can be replaced with either 2 or 3 based on remaining elements in the column (row). This is

indicated by the value calculated by the corresponding equation. The second sentence of the queries makes sure that, during the replacement the orthogonal row (column) of the element can accept the changes.

For example, the first decision box (D1) in the Figure 3-24 states: 'if valr2 is an integer and valc3 is not'. If true, the first sentence in the query, 'if valr2 is an integer', ensures that the '1' can be substituted only with '2'. Based on this decision while replacing the '1' with '2' the second part of the query, 'and valc3 is not', ensures that the replacement does not intervene with values of the column orthogonal to the element which is being replaced. valc3 not being an integer ensures that the replacement is not strictly restricted to '3' with regard to the other elements in the column. Similar explanations can be derived for the queries in the other decision boxes.

The equations and the queries together, implicitly satisfy the five rules stated in the earlier IMM.

An example demonstrating the improved incidence matrix method is given below.

Example 5

The example used for demonstrating the earlier IMM is reused here. The system considered is given in Figure 3-4. In this example, Ws and V are chosen as independent variables.

Step 1

The initial incidence matrix, incm, and foundation incidence matrix incmf are given below

$$\text{incm}=\begin{bmatrix}1 & 1 & 1 & 0 & 0\\ 0 & 0 & 1 & 1 & 1\\ 0 & 0 & 0 & 1 & 0\end{bmatrix}$$

$$\text{incmf}=\begin{bmatrix}3 & 2 & 2 & 0 & 0\\ 0 & 0 & 3 & 2 & 2\\ 0 & 0 & 0 & 3 & 0\end{bmatrix}$$

The above matrix representation is based on the layout of the models and variables in Figure 3-5.

Step 2

In the next step, the non-zero elements of the corresponding columns of the independent variables (Ws and V) are replaced with 2.

$$\text{Incm}=\begin{bmatrix} 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Step 3

Each element 1 in the matrix is now scanned and analysed to check whether it could be replaced. The details are given below

For incm(1,2)

$$valrf(1)=\prod_{c=1}^{5} incmf(1,c) \;\; for \;\; incmf \neq 0$$
$$=3\times 2\times 2$$
$$=12$$

$$valcf(2)=3 \quad ; \text{since } incmprod=\prod_{r=1}^{m} incmf(r,2) \;\; ; incmf(r,2) \quad =2$$

$$valr(1)=\prod_{c=1}^{5} incm(1,c) \;\; for \;\; incm \neq 0$$
$$=2\times 1\times 1$$
$$=2$$

$$valc(2)=\prod_{r=1}^{3} incm(r,2) \;\; for \;\; incm \neq 0$$
$$=1$$

$$valr2(1,2)=\frac{\log\left(\frac{valrf(r)}{valr(r)}\right)}{\log(2)}=\frac{\log\left(\frac{12}{2}\right)}{\log(2)}=2.5850$$

$$valr3(1,2)=\frac{\log\left(\frac{valrf(1)}{valr(1)}\right)}{\log(3)}=\frac{\log\left(\frac{12}{2}\right)}{\log(3)}=1.6309$$

$$valc2(1,2) = \frac{\log\left(\frac{valcf(2)}{valc(2)}\right)}{\log(2)} = \frac{\log\left(\frac{3}{1}\right)}{\log(2)} = 1.5850$$

$$valc3(1,2) = \frac{\log\left(\frac{valcf(2)}{valc(2)}\right)}{\log(3)} = \frac{\log\left(\frac{3}{1}\right)}{\log(3)} = 1$$

Now the queries in the decision boxes D1 to D4 are checked.

Decision box D1

valr2=2.508 → non-integer

valc3=0.6309→ non-integer

Therefore D1is unsatisfied

Decision box D2

valc2=1.5850 → non-integer

valr3=1.6309→ non-integer

Therefore D2 is unsatisfied

Decision box D3

valr3=1.6309→ non-integer

valc2=1.5850 → non-integer

Therefore D3 is unsatisfied

Decision box D4

valc3=1 → integer

valr2=2.5850→ non-integer

D4 is satisfied.

Decision box D4 is satisfied therefore the element(1,2) is replaced with 3. The updated incidence matrix is shown below.

$$\text{incm}=\begin{bmatrix} 2 & 3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

For incm(1,3)

valrf(1)=12

valcf(3) =6

valr(1)=6

valc(3)=1

valr2(1, 3)=1

valr3(1, 3)=0.6309

valc2(1, 3)=2.5850

valc3(1, 3)=1.6309

Here decision box D1 is satisfied therefore incm(1,3) is replaced with 2.

$$\text{Incm}=\begin{bmatrix} 2 & 3 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

For incm(2,3)

valrf(2)=12

valcf(3) =6

valr(2)= 2

valc(3)=2

valr2(2, 3)=2.5850

valr3(2, 3)=1.6309

valc2(2, 3)=1.5850

valc3(2, 3)=1

Here decision box D4 is satisfied therefore incm(2,3) is replaced with 3.

$$\text{incm}=\begin{bmatrix}2 & 3 & 2 & 0 & 0\\ 0 & 0 & 3 & 1 & 2\\ 0 & 0 & 0 & 1 & 0\end{bmatrix}$$

For incm(2,4)

valrf(2)=12

valcf(4) =6

valr(2)= 6

valc(4)=1

valr2(2,4)=1

valr3(2,4)=0.6309

valc2(2,4)=2.5850

valc3(2,4)=1.6309

Here decision box D1 is satisfied therefore incm(2,4) is replaced with 2.

$$\text{incm}=\begin{bmatrix}2 & 3 & 2 & 0 & 0\\ 0 & 0 & 3 & 2 & 2\\ 0 & 0 & 0 & 1 & 0\end{bmatrix}$$

For incm(3,4)

valrf(3)=3

valcf(4) =6

valr(3)= 1

valc(4)=2

valr2(3,4)=1.5850

valr3(3,4)=1

valc2(3,4)=1.5850

valc3(3,4)=1

Here decision box D3 (and D4) is satisfied therefore incm(3,4) is replaced with 3.

$$\text{incm}=\begin{bmatrix} 2 & 3 & 2 & 0 & 0 \\ 0 & 0 & 3 & 2 & 2 \\ 0 & 0 & 0 & 3 & 0 \end{bmatrix}$$

Step 4

Since all the '1' in the incm matrix are now replaced with either 2's or 3's, there is no requirement to iterate through step 3 again.

The final matrix indicates that model1 has Ws and q as input and C_L as output, model2 has ρ and V as input and q as output and Model3 has ρ as output. The result obtained here is the same as the one obtained by applying the earlier IMM (see Figure 3-11). The populated matrix is obtained here in a single iteration, for larger cases more iteration may be required to arrive at a solution.

In the current formal IMM, the over-determined, under-determined and strongly connected components are resolved in the same manner as explained earlier. The additional steps required for resolving SCC for the formal IMM are shown in Figure 3-25. The chart is similar to the one in Figure 3-17, but is modified according to the notations used in the formal IMM.

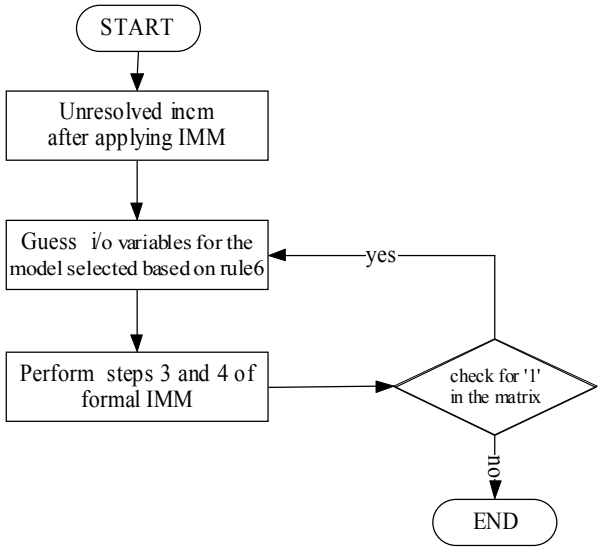


Figure 3-25 Additional steps for formal IMM in the presence of SCC

3.2.2 System Decomposition

The next step, after the variable flow models have been generated, is to perform system decomposition. This is the process of decomposing a complex system into a number of sub-problems. In the context of solving an aircraft system, the system decomposition corresponds to identifying the models which are strongly connected and grouping these as sub-systems.

In the incidence matrix section it was shown that for determined systems during variable flow modelling, the models which had unresolved '*' (or 1's in case of formal IMM) after applying the IMM, were considered as strongly connected. This dynamic identification of SCC is a significant advantage. However, since IMM cannot identify mutually exclusive SCCs we have to employ an additional method for dealing with this problem.

Here we adopt an algorithm (Tang et al., 2000) used for identifying the coupled activities in a manufacturing environment, in order to identify the SCCs. The algorithm was briefly explained in the literature review chapter. A more detailed explanation follows.

A design process can be represented in a directed graph; it consists of a set of nodes, representing the design activities and a set of directed lines connecting these nodes. The directed lines represent the linkage between the design processes.

In the proposed algorithm the design activities are represented in a binary design structure matrix (DSM). In the DSM both rows and column represent a design activity (models in our case). In the matrix an element '1' denotes, the model representing the column of the element has an input from the model representing the corresponding row. '1' marked above the diagonal denotes the feed forward loop and '1' below the diagonal denotes the feedback loop.

The problem of recognising coupled activities set is translated into the problem of seeking SCC in a directed graph.

Let D denotes the DSM,

Accessibility Matrix P= $\sum_{n=1}^{j} D^n$ where j= number of design activities.

In the accessibility matrix the values of the elements which are greater than 1 are replaced with 1. Then the Hadamard (entry wise) product of P and P^T is performed.

$$P \circ P^T = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix} \circ \begin{bmatrix} p_{11} & p_{21} & \cdots & p_{n1} \\ p_{12} & p_{22} & \cdots & p_{n2} \\ \cdots & \cdots & \cdots & \cdots \\ p_{1n} & p_{2n} & \cdots & p_{nn} \end{bmatrix} = \begin{bmatrix} p^2_{11} & p_{12}.p_{21} & \cdots & p_{1n}.p_{n1} \\ p_{21}.p_{12} & p^2_{22} & \cdots & p_{2n}.p_{n2} \\ \cdots & \cdots & \cdots & \cdots \\ p_{n1}.p_{1n} & p_{n2}.p_{2n} & \cdots & p^2_{nn} \end{bmatrix}$$

Here P^T is the transpose of P. P_{ij}=1 denotes that design activity representing the row (i) is accessible from the design activity in the column (j). Similarly if the design activity representing the column (j) is accessible from the design activity in the row (i) then P_{ji}=1. Thus both these design activities are accessible to each other if and only if Pij.Pji=1. Therefore in the matrix $P \circ P^T$, if the non-zero elements in the i^{th} row are in the j_1^{th}, j_2^{th},… j_k^{th} column then the design activities representing the rows 1,2..k are strongly coupled.

The above mentioned method is applied in the current context to identify SCC in a system. An example is given below which explains the procedure.

Example 6

The system shown in Figure 3-16 (a) is considered in this example. The variable flow model obtained for this system, given X6 as the independent variable, is shown in Figure 3-26.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Model1	o		i	i				
Model2				o		i	i	
Model3	i		i		o			o
Model4		o			i	i		
Model5				i			o	
Model6	i		o		i	i		

Figure 3-26 Populated incidence matrix for the system shown in Figure 3-16 with X6 given as the independent variable

The DSM for the system, populated based on the data flow from the variable flow model (Figure 3-26) is given below.

$$D = \begin{array}{c} \\ \text{mod}\,el1 \\ \text{mod}\,el2 \\ \text{mod}\,el3 \\ \text{mod}\,el4 \\ \text{mod}\,el5 \\ \text{mod}\,el6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

Accessibility matrix P, is calculated as follows.

$$P = D^1 + D^2 + D^3 + D^4 + D^5 + D^6 = \begin{bmatrix} 144 & 0 & 232 & 138 & 0 & 232 \\ 144 & 63 & 169 & 138 & 63 & 169 \\ 144 & 0 & 232 & 138 & 0 & 232 \\ 0 & 0 & 0 & 6 & 0 & 0 \\ 88 & 63 & 81 & 94 & 63 & 81 \\ 88 & 0 & 144 & 94 & 0 & 144 \end{bmatrix}$$

The non-zero elements of the matrix are now replaced with 1.

$$P = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Now the Hadamard product of P and P^T is,

$$P \circ P^T = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \circ \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

In the above matrix rows 1, 3 and 6 are equal and so are rows 2 and 5. Hence according to the method described before, the models which represent these rows are strongly coupled. More specifically in the current example there are two mutually exclusive

SCCs. model1, model3 and model6 belong to the first SCC and model2 and model5 to the second. Thus system is now decomposed into the following

- SCC_1 (model1, mode3, model6)
- SCC_2 (model2, model5)
- model4.

3.2.3 System scheduling

Scheduling is the process of sequencing the models in a system for the purpose of executing them after eliminating or reducing the feedback loops among the models. As mentioned earlier, in complex systems an entire elimination of feedback loops may not be possible. System decomposition identified those set of models in which feedback loops cannot be eliminated, as SCCs.

The first subsection explains how these coupled sets of models belonging to SCCs are rearranged so that feedback loops are reduced.

In the section 3.2.1.1 it was proven that a system can have multiple feasible variable flow models, in the presence of SCCs. In this section we also explain how the optimal variable flow model could be selected from the feasible ones. Selection of the optimal variable flow model is related to the rearrangement of models of the SCC, and hence it is described in this section.

The second subsection describes the arrangement of non-coupled models in a sequence.

3.2.3.1 Scheduling of coupled models

Presence of feedback loops in the SCCs makes it necessary to employ iterative methods to solve them. The more feedback loops, the more time and computational cost is required for solving. Feedback loops are formed when a model, requires input from another model which comes later in the execution sequence. Reducing the feedback loops can thus reduce the time and computational cost for solving a SCC. Rearranging the models in the SCC can reduce the feedback loops.

A genetic algorithm based approach for ordering complex design processes (Rogers, 1997) is used here for the rearrangement. Here we consider the number of feedback loops as the objective function to be minimised (This objective function has been

chosen after conducting extensive testing with different candidate objective functions on an aircraft conceptual design test case. The tests conducted are explained in section 5.3.). Genetic algorithm has been chosen since compared to other scheduling methods, it is independent of problem formulation, and therefore different objective functions can be formulated for different scheduling architectures.

Genetic algorithms work on the principle of evolution. They search a population of design points, with each design point having a vector of design parameters. Successive populations are generated by selection, crossover and mutation operations. Selection determines those members of the population that are fit to participate in the production of members of the next generation. Selection is based on the value of the objective function of each member, and the members with better objective survive. Crossover is the process of mating of members selected through selection operation, hoping to produce children which have better objective level than the parents. The final mutation operation prevents the genetic algorithm from reaching local minimum by preventing the search space becoming too narrow.

Traditionally binary coding is used in GA; the values of design variables are coded as binary numbers and then concatenated. This approach works well with numerical problems, but is not efficient for sequencing problems. In case of rearranging a SCC, the order in which models are to be executed form the genetic string. For example, if there are three models model1, model2 and model3.The initial string 1, 2, 3 indicate the execution sequence as model1, model2 and then model3. Then a string 2, 3, 1 will denote model2, model3 and then model1 as the execution order. In this case special operators are required for mutation and cross over operations.

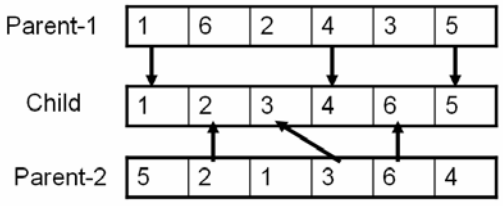


Figure 3-27 Position based crossover

Position-based cross over (Syswerda, 1990) is used here and is shown in Figure 3-27. From the first parent several models are selected to be passed on to the same location in the child string. From the second string those models which were not selected from the first parent are passed on to the spaces in the child string. The final child string will now have one copy of each model.

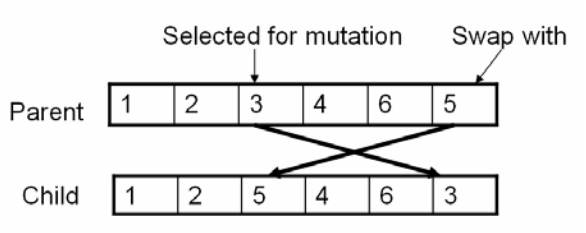


Figure 3-28 Order based mutation

Order based mutation operation is shown in Figure 3-28. Each string position is polled and if it is chosen for mutation then it is swapped with a randomly selected position in the same string.

The equation formulated for calculating the objective function (feedback number) is given in Equation 3-11(a). In the equation, D is a *reformed* DSM created from the incidence matrix of a SCC. In the reformed DSM, '1' marked above the diagonal denotes the feed forward loop and '1' below the diagonal denotes the feedback loop. Additionally, the number '1' marked on the diagonal elements denotes modified models. This number '1' represents the additional computational expenditure required for modified models. The rest of the diagonal elements are marked '0'.

Equation 3-11 Equations for calculating number of feedback loops and number of modified models

a) Number of feedback loops, $\text{nFdb} = \sum_{i=2}^{n} \sum_{j=1}^{i-1} D(i,j)$

b) Number of modified models, $\text{nMm} = \sum_{k=1}^{n} D(k,k)$

The models are rearranged based on the objective function represented in Equation 3-11(a).

Selecting the optimal variable flow model

The Equation 3-11(b) suffices the purpose of selecting the optimal variable flow model from the group of variable flow models created by IMM, which is explained in the following paragraph.

In the section 3.2.1 it was proved that there can be multiple feasible variable flow models for a system, in the presence of SCC. To choose the optimal variable flow model which can lead to the shortest execution time, the number of feedback loops of and the number of modified models in the constituent SCCs of each variable flow model is taken into consideration. The flow model which has the lowest number of modified models in the constituent SCC is chosen as the optimal and is selected for further solving. If there are multiple variable flow models with same number of modified models then the one which have lesser number of feedback loops is chosen as the optimum. The first criteria of number of modified models for selecting the optimal variable flow model compared to the number of feedback loops has been decided after conducting various tests (These tests are explained later in section 5.4). These tests identified that the modified models add more to the computing cost compared to the feedback number. Higher the number of modified models higher was the computational cost. However, the number of feedback loops plays a significant role in rearranging the models of the SCC in each variable flow model, which significantly reduces the computational cost for the SCC.

The scheduling procedure thus satisfies two aims: scheduling the models in the SCC and choosing the optimum variable flow model.

Example 7

This example is a continuation of example3. In example 3 there were four variable flow models obtained for solving the SCC as shown in Figure 3-19(a), Figure 3-19(b), Figure 3-20(a) and Figure 3-20(b). After system decomposition it was identified that for all four variable flow models, models 1,2,3,5 and 6 are strongly coupled.

The current example shows how the constituent SCCs of each variable flow model are rearranged to reduce the number of feedback loops. In addition, this example also shows how the optimal variable flow model can be chosen based on the criteria defined earlier.

Figure 3-29(a) to (d) shows the corresponding DSM representation of the four variable flow models represented in Figure 3-19(a), Figure 3-19(b), Figure 3-20(a) and Figure 3-20(b). For the flow model in Figure 3-19(a), model1, model5 and model6 are modified models and hence the diagonal elements of the corresponding DSM in the Figure 3-29(a) are given a value of 1. Similarly for the other DSMs the respective diagonal elements are given values of 1, according to the modified models in the corresponding variable flow models.

	Model1	Model2	Model3	Model5	Model6
Model1	1	1	0	1	0
Model2	0	0	0	0	1
Model3	0	0	0	0	1
Model5	0	1	0	1	0
Model6	1	0	1	0	1

(a) Number of feedback loops=1+1+1=3
Number of modified models=1+1+1=3

	Model1	Model2	Model3	Model5	Model6
Model1	1	1	0	1	0
Model2	0	0	0	0	1
Model3	1	0	1	0	1
Model5	0	1	0	1	0
Model6	0	0	1	0	1

(b) Number of feedback loops=1+1+1=3
Number of modified models=1+1+1+1=4

	Model1	Model2	Model3	Model5	Model6
Model1	0	0	1	0	1
Model2	1	1	0	1	0
Model3	0	0	0	0	1
Model5	0	1	0	1	0
Model6	0	1	0	0	1

(c) Number of feedback loops=1+1+1=3
Number of modified models=1+1+1=3

	Model1	Model2	Model3	Model5	Model6
Model1	0	0	1	0	1
Model2	0	1	0	1	0
Model3	0	0	0	0	1
Model5	1	1	0	0	0
Model6	0	1	0	0	1

(d) Number of feedback loops=1+1+1=3
Number of modified models=1+1=2

Figure 3-29 DSM representation of the variable flow models obtained in example 3

Each DSM in Figure 3-29 is rearranged using genetic algorithm with number of feedback loops as the objective function to be minimised. The final rearranged DSMs are shown in the Figure 3-30(a) to (d).It can be noted that the number of feedback loops of all the DSMs is reduced as a result of the rearrangement.

	Model1	Model5	Model2	Model6	Model3
Model1	1	1	1	0	0
Model5	0	1	1	0	0
Model2	0	0	0	1	0
Model6	1	0	0	1	1
Model3	**0**	0	0	1	0

(a) Number of feedback loops=1+1=2
Number of modified models=3

	Model3	Model1	Model5	Model2	Model6
Model3	1	1	0	0	1
Model1	0	1	1	1	0
Model5	0	0	1	1	0
Model2	0	0	0	0	1
Model6	1	0	0	0	1

(b) Number of feedback loops=1
Number of modified models=4

	Model1	Model3	Model6	Model2	Model5
Model1	0	1	1	0	0
Model3	0	0	1	0	0
Model6	0	0	1	1	0
Model2	1	0	0	1	1
Model5	**0**	0	0	1	1

(c) Number of feedback loops=1+1=2
Number of modified models=3

	Model5	Model1	Model3	Model6	Model2
Model5	0	1	0	0	0
Model1	0	0	1	1	0
Model3	0	0	0	1	0
Model6	0	0	0	1	1
Model2	1	0	0	0	1

(d) Number of feedback loops=1
Number of modified models=2

Figure 3-30 Rearranged DSMs of Figure 3-29

The next step is to identify the optimal variable flow model based on the number of modified model and the number of feedback loops in the constituent SCC of each variable flow model. Since the first criteria for choosing the optimal variable flow model is the number of modified models, the variable flow model corresponding to the fourth DSM (Figure 3-30 (d)) which has the least number of modified models of 2 is chosen as the optimal. Hence the corresponding variable flow model, which is chosen as the optimum, is given in Figure 3-20(b). The final execution sequence for models of the SCC for this variable flow model is model5→model1→model3→model6→model2, which is based on the arrangement shown in the fourth DSM (Figure 3-30(d)).

In this example since there are no variable flow models which have equal number of modified models, the second criteria, number of feedback loops, was not required to be considered for choosing the optimal variable flow model.

3.2.3.2 Scheduling of non-coupled models

In the previous section scheduling of the coupled models were explained. This section explains the scheduling of non-coupled models in a sequential order. The models which belong to SCC, which are already arranged using the method described in the previous sub-section, are confined into a single subsystem and are sequentially arranged along with the remaining non-coupled models.

Here we also adopt an algorithm (Tang et al., 2000) used for sequentially arranging the design process in concurrent engineering. The algorithm works according to the following theorem;

Theorem (Xiao and Fei, 1997): *If P is the accessibility matrix (explained in the previous section) of a directed graph G,* $P.E_{r-1}=(p_1,p_2....p_m)^T$, *where* $r\geq 1$, $1\leq m\leq n$ *(m is the number of nodes in the graph); the m-dimension vector* $E_0=(1,1,....,1)^T$; $E_r=(e_1,e_2,...,e_m)^T$, *where*

$$e_i=\begin{cases}0 & p_i\in\{0,1\};\\ 1 & p_i\notin\{0,1\};\end{cases}\qquad (i=1,2,....,m)$$

The necessary and sufficient condition of Lr= {v_i} is p_i=1, where Lr indicates that the level of node v_i is r in the graph G. Here level refers to the position in the rearranged sequence of the nodes.

The above theorem sequentially arranges the models considering the following

- If all the elements in a column of the DSM are zero then the models corresponding to that column should be executed as early as possible since they do not need any input from other activities.
- If all the elements of a row are zero, the model representing that row should be executed after all other non all zero element models because it provides no input to any other models
- The ultimate objective of the rearrangement is to schedule the rows and columns of the DSM into a lower triangular form.

The following example will explain the procedure in detail

Example8

The system shown in Figure 3-16(a) with X_7 as the chosen independent variable (corresponding incidence matrix, shown in Figure 3-22) is considered in this example for sequential arrangement. On decomposition of the incidence matrix models 1, 3 and 6 were identified as strongly coupled. The SCC which contains these models and the remaining non-coupled models are represented in Figure 3-31. The scheduling of models in the SCCs was explained in the previous section.

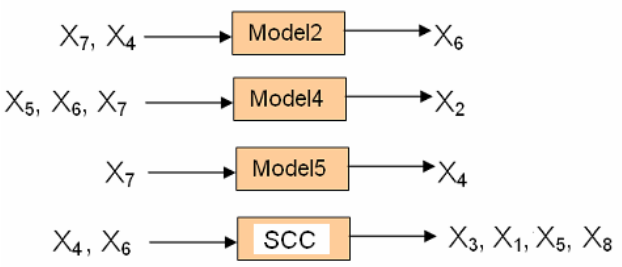


Figure 3-31: Confined SCC and the remaining models

The theorem mentioned above is applied to this case for rearranging it sequentially. The DSM for the system populated based on the variable flow model shown in the incidence

matrix in Figure 3-22 is given below. It has to be noted that the models belonging to SCC are regarded as a single model, with inputs and outputs as shown in Figure 3-31.

$$D = \begin{matrix} \text{model2} \\ \text{model4} \\ \text{model5} \\ \text{SCC} \end{matrix} \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$D^1 + D^2 + D^3 + D^4 = \begin{bmatrix} 4 & 20 & 0 & 10 \\ 0 & 4 & 0 & 0 \\ 10 & 25 & 4 & 20 \\ 0 & 10 & 0 & 4 \end{bmatrix}$$

Replacing non-zero values in the above matrix with 1

$$\text{Accessibility matrix P} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

The order levels of all the models are figured out as follows:

$E_0 = (1,1,1,1)^T$
$P.E_0 = (3,1,4,2)^T$
$L_1 = \text{model4}$

Here the second element of $P.E_0$ had 1. Hence according to the theorm model4 was added in the level L_1. Similarly the models for the subsequent levels are identified as follows;

$E_1 = (1,0,1,1)^T$
$P.E_1 = (2,0,3,1)^T$
$L_2 = \text{SCC}$

$E_2 = (1,0,1,0)^T$
$P.E_2 = (1,0,2,0)^T$
$L_3 = \text{model2}$

$E_3 = (0,0,1,0)^T$
$P.E_3 = (0,0,1,0)^T$
$L_4 = \text{model5}$

L_1, L_2, L_3 and L_4 represent the order levels obtained. The DSM for the order obtained from the above method is

$$D = \begin{array}{c} \text{model4} \\ \text{SCC} \\ \text{model2} \\ \text{model5} \end{array} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

The rearranged DSM is in a lower triangular form. It has to be noted that since the above theorem rearranges the DSM into a lower triangular form, the execution sequence for the activities representing the rows will be from bottom to top. The sequential execution order for the system will therefore be mode5→model2→SCC→model4.

3.3 Computational Process Modelling Example

The individual methods incorporated in the computational process modeller were explained in the previous sections with specific examples. This section demonstrates the application and implementation of the entire computational process modeller, for generating computational plans for systems, with the help of an aircraft sizing example.

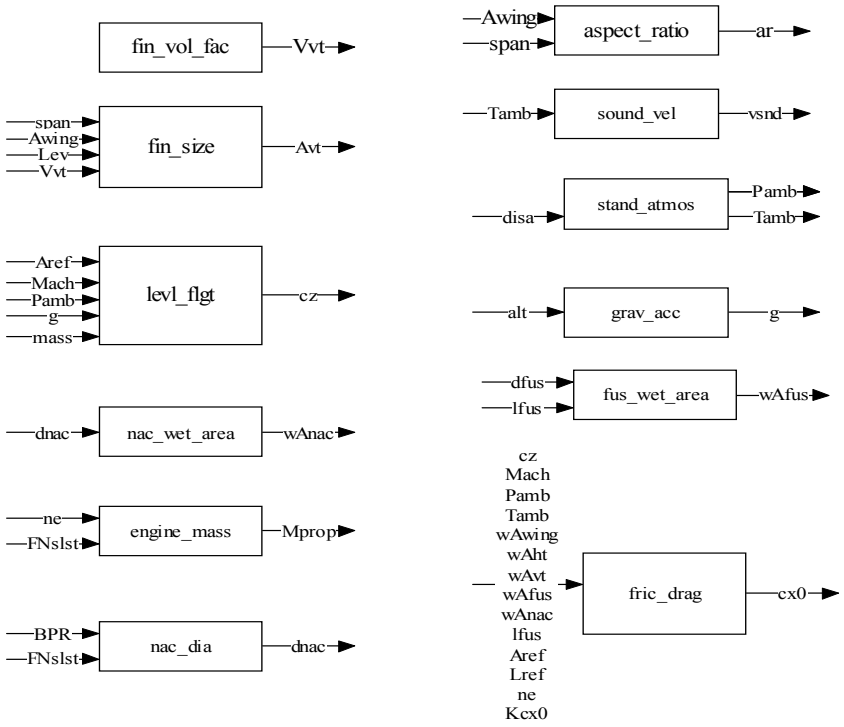


Figure 3-32 System of models

The case considered here is a subset of an aircraft system. The explanation is in correspondence with the flowchart for computational process modeller shown in Figure 3-1.

Figure 3-32 shows the system of models considered for this example. In the system there are 12 models and 31 variables. Variables Aref, Avt, BPR, Kcx0, Lev, Lref, Mach, Mprop, ar, cx0, cz, lfus, mass, vsnd, wAfus, wAht, wAvt and wAwing, which were chosen randomnly, are considered as independent variables.

The first step in the computational process modeller is to perform the variable flow modelling. Formal incidence matrix method is used in this example for variable flow modelling. Figure 3-33 shows the initial incidence matrix representation of the system with the relevant element of the columns of the independent variables replaced with 2 (after applying step2 of the formal IMM).

	'Aref'	'Avt'	'Awing'	'BPR'	'FNslst'	'Kcx0'	'Lev'	'Lref'	'Mach'	'Mprop'	'Pamb'	'Tamb'	'Vvt'	'alt'	'ar'	'cx0'	'cz'	'dfus'	'disa'	'dnac'	'g'	'lfus'	'mass'	'ne'	'span'	'vsnd'	'wAfus'	'wAht'	'wAnac'	'wAvt'	'wAwing'
fin_vol_fac													1																		
aspect_ratio			1												2										1						
fin_size_		2	1				2						1												1						
sound_vel												1														2					
levl_flgt	2								2		1						2				1		2								
stand_atmos											1	1		1					1												
grav_acc														1							1										
fus_wet_area																		1				2					2				
nac_wet_area																				1									1		
fric_drag	2					2		2	2		1	1				2	2					2		1			2	2	1	2	2
engine_mass					1					2														1							
nac_dia				2	1															1											

Figure 3-33 Initial incidence matrix for the system in Figure 3-32 (Blank cells denote '0')

Figure 3-34 shows the incidence matrix populated after applying steps 3 and 4 of the formal IMM (Figure 3-24).

	'Aref'	'Avt'	'Awing'	'BPR'	'FNslst'	'Kcx0'	'Lev'	'Lref'	'Mach'	'Mprop'	'Pamb'	'Tamb'	'Vvt'	'alt'	'ar'	'cx0'	'cz'	'dfus'	'disa'	'dnac'	'g'	'lfus'	'mass'	'ne'	'span'	'vsnd'	'wAfus'	'wAht'	'wAnac'	'wAvt'	'wAwing'
fin_vol_fac													3																		
aspect_ratio			1												2										1						
fin_size_		2	1				2						2												1						
sound_vel												3														2					
levl_flgt	2								2		1						2				1		2								
stand_atmos											1	2		1					3												
grav_acc														1							1										
fus_wet_area																		3				2					2				
nac_wet_area																				1									1		
fric_drag	2				-	2		2	2		1	2				2	2					2		1			2	2	1	2	2
engine_mass					1					2														1							
nac_dia				2	1															1											

Figure 3-34 Populated incidence matrix according to formal IMM (Blank cells denote '0')

	'Aref'	'Avt'	'Awing'	'BPR'	'FNslst'	'Kcx0'	'Lev'	'Lref'	'Mach'	'Mprop'	'Pamb'	'Tamb'	'Vvt'	'alt'	'ar'	'cx0'	'cz'	'dfus'	'disa'	'dnac'	'g'	'lfus'	'mass'	'ne'	'span'	'vsnd'	'wAfus'	'wAht'	'wAnac'	'wAvt'	'wAwing'
fin_vol_fac													3																		
aspect_ratio			3												2										2						
fin_size_		2	2				2						2												3						
sound_vel												3														2					
levl_flgt	2								2		3						2				2		2								
stand_atmos											2	2		3					3												
grav_acc														2							3										
fus_wet_area																		3				2					2				
nac_wet_area																				2									3		
fric_drag	2					2		2	2		2	2				2	2					2		3			2	2	2	2	2
engine_mass					3					2														2							
nac_dia				2	2															3											

(a)

	'Aref'	'Avt'	'Awing'	'BPR'	'FNslst'	'Kcx0'	'Lev'	'Lref'	'Mach'	'Mprop'	'Pamb'	'Tamb'	'Vvt'	'alt'	'ar'	'cx0'	'cz'	'dfus'	'disa'	'dnac'	'g'	'lfus'	'mass'	'ne'	'span'	'vsnd'	'wAfus'	'wAht'	'wAnac'	'wAvt'	'wAwing'
fin_vol_fac													3																		
aspect_ratio			3												2										2						
fin_size_		2	2				2						2												3						
sound_vel												3														2					
levl_flgt	2								2		3						2				2		2								
stand_atmos											2	2		3					3												
grav_acc														2							3										
fus_wet_area																		3				2					2				
nac_wet_area																				3									2		
fric_drag	2					2		2	2		2	2				2	2					2		2			2	2	3	2	2
engine_mass					2					2														3							
nac_dia				2	3															2											

(b)

	'Aref'	'Avt'	'Awing'	'BPR'	'FNslst'	'Kcx0'	'Lev'	'Lref'	'Mach'	'Mprop'	'Pamb'	'Tamb'	'Vvt'	'alt'	'ar'	'cx0'	'cz'	'dfus'	'disa'	'dnac'	'g'	'lfus'	'mass'	'ne'	'span'	'vsnd'	'wAfus'	'wAht'	'wAnac'	'wAvt'	'wAwing'
fin_vol_fac													3																		
aspect_ratio			3												2										2						
fin_size_		2	2				2						2												3						
sound_vel												3														2					
levl_flgt	2								2		2						2				3		2								
stand_atmos											2	2		2					3												
grav_acc														2							2										
fus_wet_area																		3				2					2				
nac_wet_area																				3									2		
fric_drag	2					2		2	2		2	2				2	2					2		2			2	2	2	2	2
engine_mass					3					2														2							
nac_dia				2	2															3											

(c)

	'Aref'	'Avt'	'Awing'	'BPR'	'FNslst'	'Kcx0'	'Lev'	'Lref'	'Mach'	'Mprop'	'Pamb'	'Tamb'	'Vvt'	'alt'	'ar'	'cx0'	'cz'	'dfus'	'disa'	'dnac'	'g'	'lfus'	'mass'	'ne'	'span'	'vsnd'	'wAfus'	'wAht'	'wAnac'	'wAvt'	'wAwing'
fin_vol_fac													3																		
aspect_ratio			3												2										2						
fin_size_		2	2				2						2												3						
sound_vel												3														2					
levl_flgt	2								2		2						2				3		2								
stand_atmos											3	2		2					3												
grav_acc														3							2										
fus_wet_area																		3				2					2				
nac_wet_area																				2									3		
fric_drag	2					2		2	2		2	2				2	2					2		3			2	2	2	2	2
engine_mass					3					2														2							
nac_dia				2	2															3											

(d)

Figure 3-35 Four different variable flow models obtained for the system in Figure 3-32(Blank cells denote ‘0’)

In the Figure 3-34 there are some '1's still existing in some rows of the matrix. Since the system is determined (*TNvar –NIvar-Noutmod* $\Rightarrow$ *31-18-13 = 0*), those rows indicate the presence of at least one SCC. aspect_ratio, fin_size, levl_flgt, stand_atmos, grav_acc, nac_wet_area, fric_drag, engine_mass and nac_dia are the unresolved models in the matrix and hence are part of the SCC.

The additional steps required for the formal IMM in the presence of SCC are further performed according to the steps given in the flowchart of Figure 3-25. Subsequently four different variable flow models are obtained for the system. They are represented in Figure 3-35 (a) to (d). The matices not only show the models of the SCC but also other models.

The next step is to perform the decomposition of the system for each variable flow model. For the first variable flow model (Figure 3-35 (a)), the various matrix manipulations performed for decomposition are given below.

DSM, D=

fin_vol_fac	1	0	1	0	0	0	0	0	0	0	0	0
aspect_ratio	0	1	1	0	0	0	0	0	0	0	0	0
fin_size_	0	1	1	0	0	0	0	0	0	0	0	0
sound_vel	0	0	0	1	0	1	0	0	0	1	0	0
levl_flgt	0	0	0	0	1	1	0	0	0	1	0	0
stand_atmos	0	0	0	0	0	1	1	0	0	0	0	0
grav_acc	0	0	0	0	1	0	1	0	0	0	0	0
fus_wet_area	0	0	0	0	0	0	0	1	0	0	0	0
nac_wet_area	0	0	0	0	0	0	0	0	1	1	0	0
fric_drag	0	0	0	0	0	0	0	0	0	1	1	0
engine_mass	0	0	0	0	0	0	0	0	0	0	1	1
nac_dia	0	0	0	0	0	0	0	0	1	0	0	1

Accessibility Matrix, P=

1	1	1	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	1	1	1	1	1
0	0	0	0	1	1	1	0	1	1	1	1	1
0	0	0	0	1	1	1	0	1	1	1	1	1
0	0	0	0	1	1	1	0	1	1	1	1	1
0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1

$$P \circ P^T =$$

1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1

From the PoP^T matrix, there are three sets of similar rows (2,3), (5,6,7) and (9,10,11,12). Hence there are three SCCs in this system; 1)aspect_ratio and fin_size 2)levl_flgt, stand_atmos and grav_acc 3)nac_wet_area, fric_drag, engine_mass and nac_dia.

Similar decomposition is performed on the rest of the variable flow models. In all the three cases the same SCCs were identified. Therefore, the decomposition procedure for the other three variable flow models is not described here.

The next step is to perform the scheduling of coupled models (SCCs). The scheduling of the SCCs (by genetic algorithm) of each of the four variable flow models is given below.

For the first variable flow model (Blank cells denotes '0'):

	BEFORE SCHEDULING	AFTER SCHEDULING
SCC 1	aspect_ratio 1 1 fin_size_ 1 1 Number of feedback loops=1 Number of modified models= 2	*No Change*
SCC 2	levl_flgt 1 1 stand_atmos 1 1 grav_acc 1 Number of feedback loops=1 Number of modified models= 2	*No Change*
SCC 3	nac_wet_area 1 fric_drag 1 1 engine_mass 1 1 nac_dia 1 Number of feedback loops=1 Number of modified models= 2	*No Change*

The total number of feedback number is 3(1+1+1) and total number of modified models is 6(2+2+2)

For the second variable flow model (Blank cells denotes '0'):

	BEFORE SCHEDULING	AFTER SCHEDULING
SCC 1	aspect_ratio 1 1 fin_size_ 1 1 Number of feedback loops=1 Number of modified models=2	*No Change*
SCC 2	levl_flgt 1 1 stand_atmos 1 1 grav_acc 1 Number of feedback loops=1 Number of modified models=2	*No Change*
SCC 3	nac_wet_area 1 1 fric_drag 1 1 engine_mass 1 1 nac_dia 1 1 Number of feedback loops=3 Number of modified models=4	engine_mass 1 1 fric_drag 1 1 nac_wet_area 1 1 nac_dia 1 1 Number of feedback loops=1 Number of modified models=4

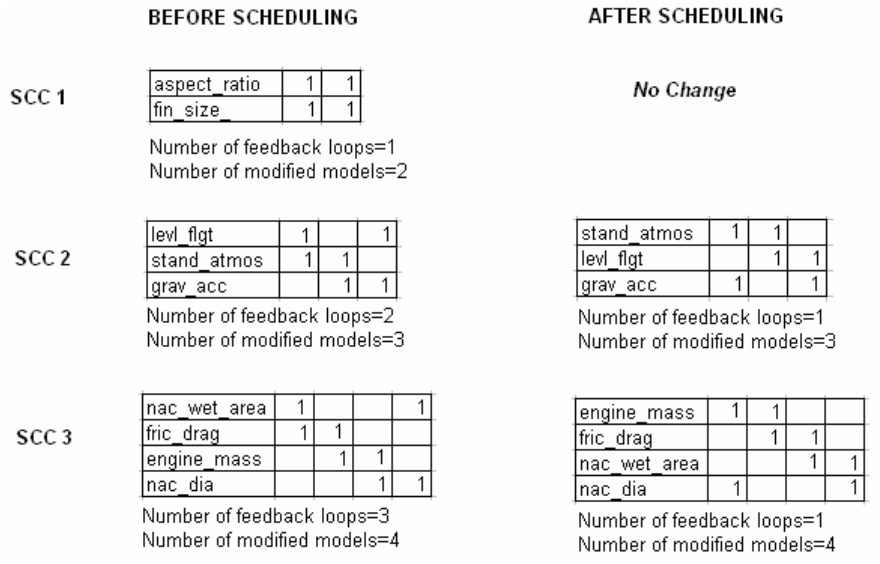
The total number of feedback loops is 3(1+1+1) and total number of modified models is 8(2+2+4).

For the third variable flow model (Blank cells denotes '0'):

	BEFORE SCHEDULING	AFTER SCHEDULING
SCC 1	aspect_ratio 1 1 fin_size_ 1 1 Number of feedback loops=1 Number of modified models=2	*No Change*
SCC 2	levl_flgt 1 1 stand_atmos 1 1 grav_acc 1 1 Number of feedback loops=2 Number of modified models=3	stand_atmos 1 1 levl_flgt 1 1 grav_acc 1 1 Number of feedback loops=1 Number of modified models=3
SCC 3	nac_wet_area 1 fric_drag 1 1 engine_mass 1 1 nac_dia 1 Number of feedback loops=1 Number of modified models=2	*No Change*

The total number of feedback loops is 3(1+1+1) and the total number of modified models is 7(2+3+2).

Finally for the fourth variable flow model (Blank cells denotes '0'):



The total number of feedback loops is 3(1+1+1) and the total number of modified models is 9(2+3+4).

The first criterion for choosing the optimal variable flow model is the number of modified models. Compared to the others the first variable flow model which has the least number of modified models of 6 is hence chosen as the optimal. The corresponding variable flow model, which is therefore chosen as the optimum, is given in Figure 3-35 (a).

The execution sequences for the models in the SCCs are given below.

SCC 1 : aspect_ratio→fin_size

SCC 2 : levl_flgt→stand_atmos→grav_acc

SCC 3: nac_wet_area→fric_drag→engine_mass→nac_dia

In this example since there are no variable flow models which have equal number of modified models, the second criteria, number of feedback loops, was not required to be considered for choosing the optimal variable flow model.

The next step is to sequentially arrange the SCCs with the remaining non-coupled modes. The models belonging to SCC are regarded as a single model with inputs and output. The SCCs and the remaining models are represented in Figure 3-36.

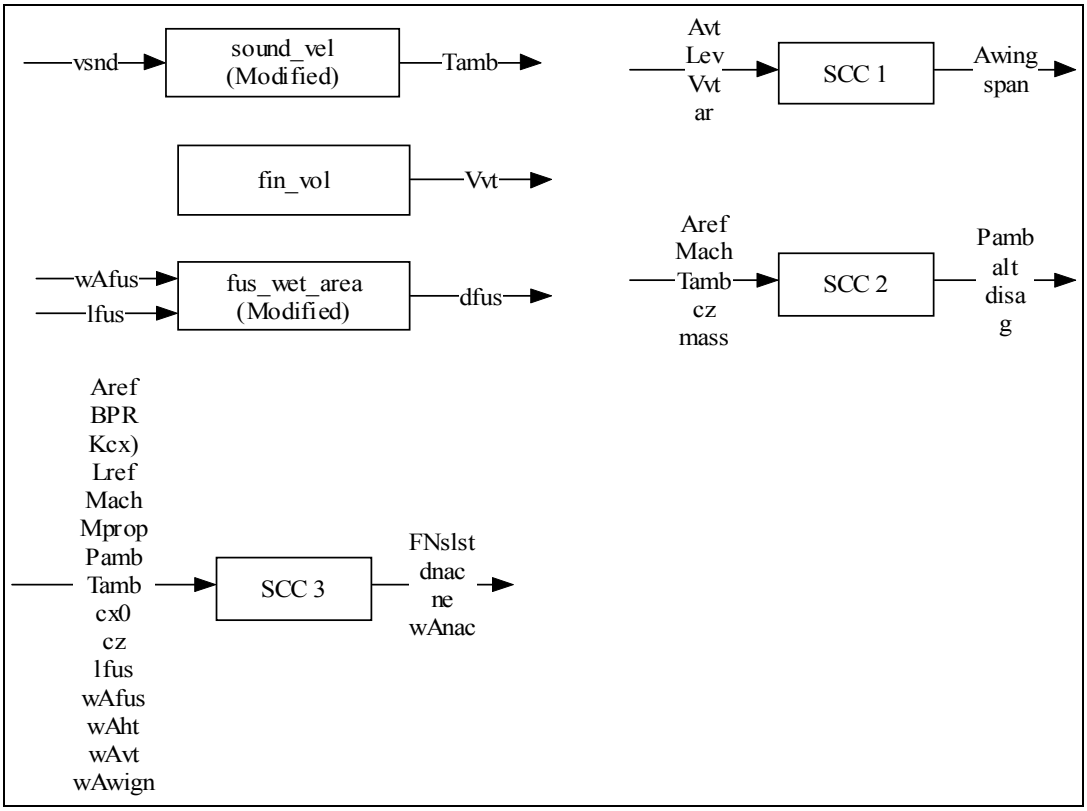


Figure 3-36 SCCs and the remaining models

The corresponding DSM populated based on the optimal variable flow model shown in Figure 3-35 (a) is given below.

$$D = \begin{array}{c} \\ SCC1 \\ SCC2 \\ SCC3 \\ sound\_vel \\ fus\_wet\_area \\ fin\_vol \end{array} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Accessibility Matrix, $P = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$

The order levels of all the models are determined as follows:

$$E_0 = (1,1,1,1,1)^T$$
$$P.E_0 = (2,1,1,3,1,2)^T$$
$$L_1 = SCC2, SCC3, fus\_wet\_area$$

$$E_1 = (1,0,0,1,0,1)^T$$
$$P.E_1 = (1,0,0,2,0,1)^T$$
$$L_2 = SCC1, fin\_vol$$

$$E_2 = (0,0,0,1,0,0)^T$$
$$P.E_2 = (0,0,0,1,0,0)^T$$
$$L_3 = sound\_vel$$

The DSM for the order obtained from the above method is

$$D = \begin{array}{c} \\ SCC2 \\ SCC3 \\ fus\_wet\_area \\ SCC1 \\ fin\_vol \\ sound\_vel \end{array} \begin{bmatrix} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

It can be noted that the rearranged DSM is in a lower triangular form. Hence the execution sequence for the system is sound_vel→ fin_vol→ SCC1 (aspect_ratio→ fin_size)→ fus_wet_area→ SCC3 (nac_wet_area→ fric_drag→ engine_mass→ nac_dia)→ SCC2 (levl_flgt→ stand_atmos→ grav_acc).

3.4 Summary and Conclusions

Presented in this chapter is a novel method for computational process modelling. The computational process modeller was subdivided into variable flow modelling, decomposition and scheduling.

The IMM and the modified formal IMM for variable flow modelling are introduced. Compared to IMM, the formal IMM is easier to implement and takes less memory space while computing. These novel methods have the advantage of generating multiple variable flow models for a system which consists of models with multiple outputs. . Until now this feature has not been available even in computational process modelling

methods for algebraic equations, were the focus has always been on generating a single feasible variable flow model.

Decomposition was performed based on an algorithm from concurrent engineering for identifying coupled design processes.

Scheduling the coupled models (SCC) was performed by genetic algorithm with number of feedback loops as the objective function to be minimised. The number of modified models and the value of the objective function of the constituent SCCs of the multiple flow models, generated during variable flow modelling, were considered as the criteria for choosing the optimum flow model. As a result this scheduling procedure satisfied two aims: scheduling the models in the SCC and choosing the optimum variable flow model.

The non-coupled models were sequentially arranged based on an algorithm from concurrent engineering for arranging design tasks.

An example system consisting of 13 models and 31 variables demonstrated the step by step procedure for computational process modelling.

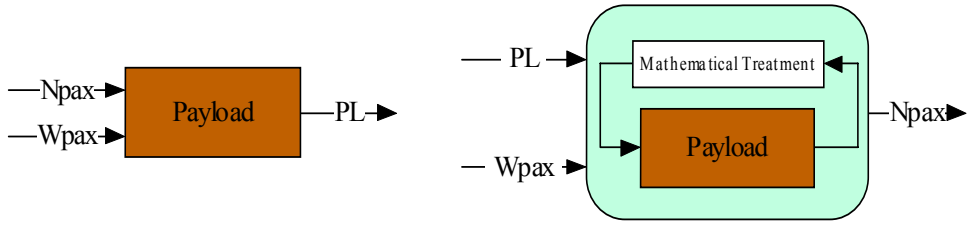
4 SOLVERS FOR SUB-SYSTEMS

4.1 Introduction

The computational process modeller generates the optimum execution sequence of the models in a system. The system is then solved based on the obtained execution sequence to compute the unknown variables. The solving is carried out by the sequential execution of the models, SCCs and modified models in the system. Unlike the models, where the execution is straightforward, SCCs and modified models require mathematical treatments for solving them. The treatments and approaches used in this research for solving these sub-systems are discussed below.

4.2 Scheme for Solving Modified Models

A modified model is a model which has some of its input and output variables swapped as a result of variable flow modelling. Since the original model accepts only its real inputs, numerical solving has to be relied on in order to accomplish the required modification. An example of a modified model is given in Figure 4-1.



(a) Payload model (b) Modified payload model

Figure 4-1 An example for modified model

A model *m* can be represented as

$$y = m(x),\ where\ x = x_1,.....,x_n\ \ and\ \ y = y_1,.....y_m$$

Here x is the input variables and y the output variables. In the case of a modified model a subset of variables in x and y is interchanged. Let *xm* and *ym* denote that subset (number of elements *xm* is equal to the number of element in *ym*). Hence a modified model can be represented as

$(yr,\ xm) = mm(xr,\ ym)$

where

$xr = x \cap xm$

$yr = y \cap ym$

Here *xr* and *ym* are the known values and the aim is to compute *yr* and *xm*.

Various schemes which are available for solving a system of non-linear equations can be reformed to be applied for solving the modified models. In this research solvers available for non-linear least-squares problem are considered for solving the modified models. The Gauss-Newton method, which is one of the available methods for solving the non-linear least squares problem, is chosen for solving the modified models. There are various solving schemes such as the Newton method, Gauss-Newton Algorithm, Levenberg-Marquardt algorithm, etc., available for solving non-linear least-square problems (Dennis and Robert, 1983). The Newton method is generally used for unconstrained minimisation and the other two for solving system of non-linear equations. Since this research did not have the aim of choosing the most appropriate solver for modified model no much investigation was performed in choosing the solver. Gauss-Newton method was chosen because of its proven efficiency as a solver for non-linear least-squares problems. The plan described below for solving the modified models will be similar if any other solvers are used. Hence Gauss-Newton method can be replaced with any solver with out any modification in the plan, for solving the modified models.

For solving the modified model, initially guessed values for *xm* along with the known *xr* variables are given as inputs to the model *m*. On execution, the values of the output variables, generated by *m* are compared with the corresponding known inputs *ym* of the modified model. If the difference is more than the required tolerance then a new set of guessed values for *xm* which are calculated based on the Gauss-Newton method, are provided again as input to model *m*.

This iterative procedure is repeated until convergence is attained. The convergence here means the difference between the output variables generated by *m* and the known input variables *ym* become less than the minimum required tolerance level set by the designer. Figure 4-2 shows a flow chart depicting the solving of the modified models.

In the flow diagram xm^c represents the initial guessed values for *xm* and *tol* is the tolerance level. ym^d is the output generated by model *m*, which has to be compared with *ym*. If the difference is above the required tolerance then the iteration is repeated until convergence is reached.

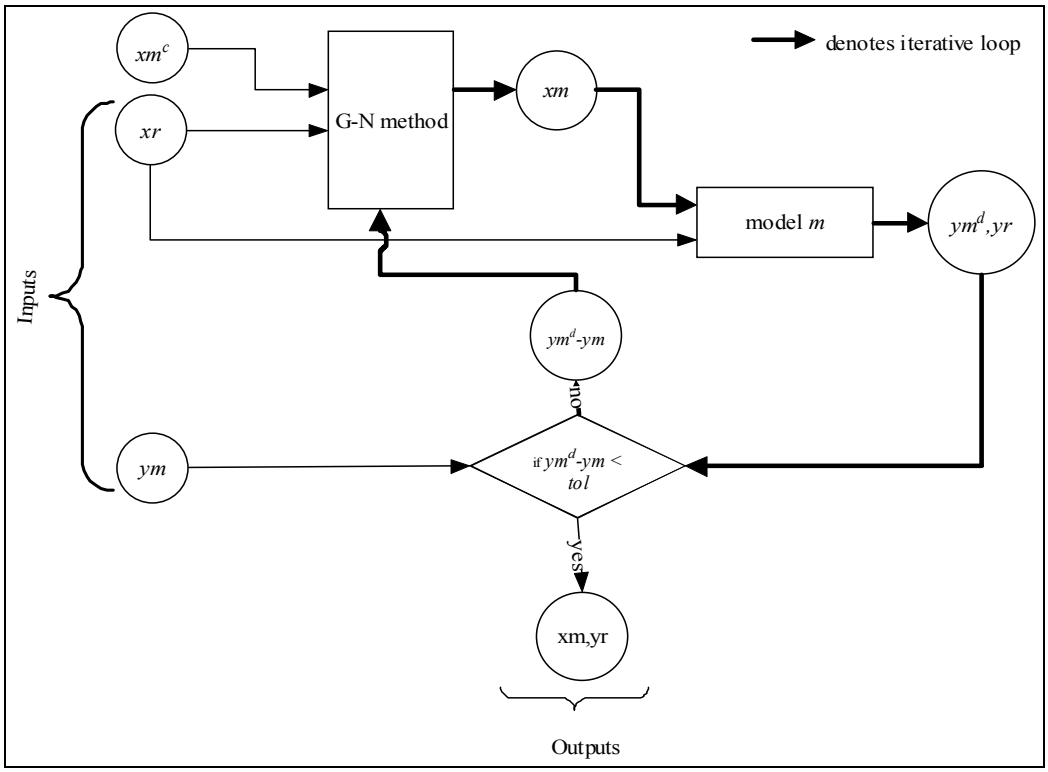


Figure 4-2 Flowchart for solving modified models

In the Gauss-Newton method the vector *xm* during each iteration is calculated by the Equation 4-1.

Equation 4-1 Equation for calculating values of the iterative variables of modified models, using Gauss-Newton method

$$xm^{k+1} = xm^k - \left(J_f(xm^k)^T . J_f(xm^k)\right)^{-1} . J_f(xm^k)^T . f(xm^k)$$

Here *k* represents the iteration number and $J_f(xm)$ is the jacobian of function *f* at *xm*. Function *f(x)* involves one single execution of the original model m (with *xr* and *xm* as input) and the output of *f(x)* is the vector containing ym^d-*ym*.

Since models are black-boxes and are not directly differentiable, finite difference method is employed for obtaining the Jacobians in the Gauss-Newton method. For more details on the Gauss-Newton method see Appendix-II.

Example

Modified model given in Figure 4-1 is solved as described below.

For this modified model,

$$x = \{Npax, Wpax\}$$
$$y = \{PL\}$$
$$xm = \{Npax\}$$
$$ym = \{PL\}$$

Hence

$$xr = \{Wpax\}$$
$$yr = \{\ \}$$

Given the inputs PL=17250 and Wpax=115, the output Npax is calculated as follows.

The initial guessed value for *xm, xm^c* is given as 1. On solving, based on the flowchart in Figure 4-2, 150 is the solution obtained for PL. Table 4-1 shows the progress of different values (depicted in the flow chart), during each iteration.

Table 4-1 Variation in the swapped variables during the solving of 'Payload' modified model

Iteration	*xm(Npax)*	*ym^d(PL)*	*ym^d-ym*
1	1	115	-17135
2	2	230	-17020
3	4.5	517.5	-16732
4	10.75	1236.3	-16014
5	26.375	3033.1	-14217
6	65.4375	7525.3	-9724.7
7	150	17250	2.5705e-004

Here the convergence is attained in seven iterations. The tolerance level is kept 1e-3, hence the iteration terminated when the values of ym^d-ym reached 2.5705e-4.

The limitation of solving the modified models using Gauss-Newton method is that the solution obtained is greatly dependent upon the starting points (xm^c). For example, Figure 4-3 shows a curve for the function y=sin(x). Let us consider that this function is modified in order to generate output x, given input y. Considering a case where input y is given a value of 0.6, a starting guess of 200 (for x) will generate 143.1301 as output (for x). However, if a value of 400 is given as the initial guess, then 396.8695 will be the computed output for x.

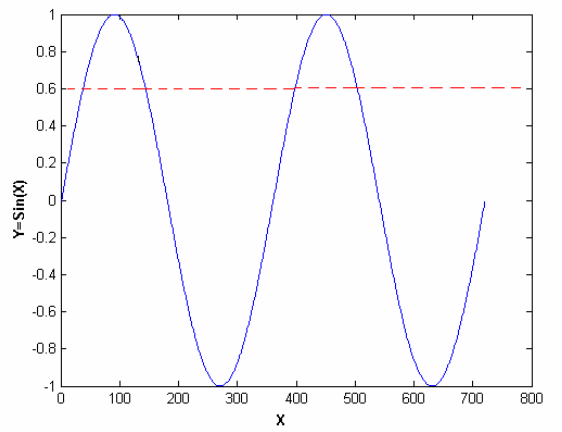


Figure 4-3 Sine curve

Additionally, by using Gauss-Newton method the iterations may not always converge to a root, especially when the starting points are too far away from the solution.

A problem associated with solving the modified model is that, if the switched variables in a modified model are insensitive to each other, then a solution will not be achievable. The problem is discussed in the next paragraph.

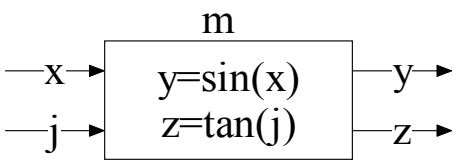


Figure 4-4 A model for demonstrating sensitivity of switched variables.

In Figure 4-4, for the modelm, x and j are the inputs and y and z are the outputs. Models are black boxes and the constituent equations are generally hidden, but in the Figure 4-4

these are shown only for demonstration purpose.It is clear from the equations that the input x and output z of the model m will be unrelated. This means that variable z is insensitive to variable x. If after variable flow modelling, these variables are switched, then obtaining a solution for x, given z as input is impossible. Such situation will lead to an unsolvable modified model. The same applies to input j and output y.

The above example is a case where the switched variables are totally insensitive to each other. There will be cases where the switched variables will be weakly sensitive to each other. Switching these variables will lead to numerous iterations required for convergence for a modified model which adds to the computing cost for the system.

In complicated aircraft design problems similar situations as the cases described above might arise. Unlike the above example, the information within the models is generally unavailable. Therefore to obtain the sensitivity of the input variables to the output variables of the models, sensitivity analysis (Cacuci, 2003) has to be carried out. The results obtained should be used to decide which variable can be switched in a model. The sensitivity of the switched variables of the modified models in practice should be taken into consideration while generating the variable flow models in order to avoid unsolvable modified models.

The systems used for testing in this research did not have any models which totally insensitive inputs and outputs variables. Hence the sensitivity of the switched variables was not considered while generating the variable flow models.

The solvability of a modified model is dependent on the sensitivity of the switched variables, the starting point chosen for the unknown variables and various other issues which are yet to be identified. Normalising the models based on this information will assist in grading these models in the system according to its solvability and thus providing a criterion for intelligently choosing the models which are to be modified (i.e. choosing that variable flow model as the optimum, in which the contained modified models are quickly solvable) during the computational design process modelling. Conducting sensitivity analysis for each model and normalising them was out of scope of the current research context and therefore is left as a future work.

4.3 Schemes for Solving Strongly Connected Components

This section describes the application of two different methods for solving the SCCs, the Fixed-point-iteration (FPI) method and the Gauss-Newton (GN) method. The Fixed-point-iteration method is applied in two different manners for solving the SCCs.

4.3.1 Solving SCC using Fixed point iteration method

SCCs contain iterative loops, the presence of which necessitate certain variables to be known before they actually become available as output from models which appear later in the execution sequence. A simple SCC is shown in Figure 4-5.

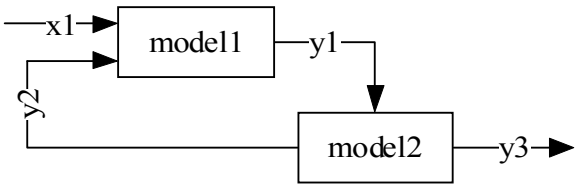


Figure 4-5 A simple SCC

For this SCC, *model1* has *x1* and *y2* as input and *y1* as output, and *model2* has *y1* as input and *y2* and *y3* as outputs. The execution sequence for this SCC is *model1*→*model2*. It can be noted from the execution sequence that *model1* requires *y2* as input in order to be executed. However, *y2* is computed by *model2* which appears later in the execution sequence. This situation is always encountered in a SCC and is the main difficulty when solving it.

The fixed point iteration method (FPI) tackles this situation by starting with an initial guess of the unknown inputs. Thereafter once the real values of these variables are available from the models which appear later in the execution sequence, these values are fed back into the earlier model and the execution is repeated. The iterative execution sequence is continued until the difference between the guessed values and the real values obtained are below the required tolerance level.

For the above example, initial guessed value of $y2^g$ is given as input to *model1*. Thereafter *model1* and then *model2* is executed. If the difference between *y2*, obtained as output from *model2*, and $y2^g$ is above the required tolerance then the new *y2* is given as input to *model1* and the execution is repeated. The iteration is continued until *y2* equals $y2^g$ or their difference is below the minimum tolerance level.

A detailed explanation of the fixed point iterative method is given in the Appendix-III.

Now we explain two different ways of executing the models in a SCC while solving it using Fixed-point-iterative method. These will be called as Fixed-point-iterative first method and Fixed-point-iterative second method

In the first method, the sequence of executing the models is determined according to the presence of the feedback variables and their convergence. The models covered by a particular feedback loop are executed iteratively until the feedback variable attains convergence. Once the convergence is established the execution is continued with the subsequent models.

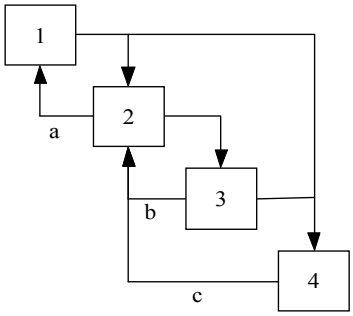


Figure 4-6 An example SCC with three feedback loops

In the SCC shown in the Figure 4-6, the iterative loops are represented as *a*, *b* and *c*. The execution sequence for this sub-system is *1→2→3→4*. However, according to the method described above, the execution of the models is re-established based on the feedback loops. In this example, initially models *1* and *2*, covered by the feedback loop *a* are iteratively executed until the feedback variable attains convergence. Thereafter the subsequent model *3* is executed, which is again part of another loop *b* and hence execution is diverted back to model *2*. The execution of model *2* with the new inputs obtained from model *3* might create a new value for the feedback variable of loop *a* and thus violating the convergence attained earlier. Therefore the iteration of loop *a* is continued further. This iterative solving procedure is continued until all the feedback variables attain convergence.

In the second method for solving the SCC using Fixed-point-iteration method, the execution sequence is not re-established as in the previous method. Here the feedback variables are updated at the end of each iteration. Here a single iteration involves

executing once all the models in the SCC. For example, for the SCC shown in Figure 4-6, one iteration involves executing the models *1→2→3→4*, thereafter feedback variables of the loops *a, b, c* are updated with the new values obtained after the execution. Subsequently this iterative procedure is repeated until the convergence is attained i.e., the values of the feedback variable obtained as output after the execution equals to values which were given as inputs before execution.

The convergence of Fixed-point-iteration is highly dependent on the starting guess provided for the feedback variables by the designer. The nearer the guessed value to the solution the faster is the convergence.

4.3.2 Solving SCC using Gauss-Newton method

Solving the SCC using Gauss-Newton method involves solving it as a non-linear least-square problem.

In the above mentioned FPI –second method, the latest computed values were used for the feedback variables for the subsequent iteration. In contrast, while applying the Gauss-Newton method the values of the feedback variables during each iteration are calculated by the Equation 4-2. However, the execution sequence for the models is similar to FPI-second method.

Equation 4-2 Equation for calculating values of the iterative variables of SCC, using Gauss-Newton method

$$x^{k+1} = x^k - \left(J_f(x^k)^T . J_f(x^k) \right)^{-1} . J_f(x^k)^T . f(x^k)$$

Here x is the vector of feedback variables, k represents the iteration number and $J_f(x)$ is the jacobian of function f at x. Function *f(x)* involves one single sequential execution of all models in the SCC (with x as inputs) and the output of *f(x)* is the vector containing the difference of the feedback variables in the previous and the current iteration.

Solving the SCC involves executing all the models in a sequence and thereafter iterating with the new feedback variables generated by the Equation 4-2 until convergence is attained. The aim is to make the output of function *f(x)* to zero.

Here also solution obtained is greatly dependent upon the starting points of the feedback variables. Additionally, by using Gauss-Newton method the iterations may not always

converge to a root, especially when the starting points are too far away from the solution.

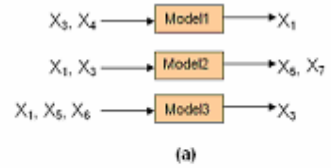
4.4 Solving SCC and Modified Models at System Level

The presence of modified models adds complexity when solving the SCC in a system. This is because the modified models have to be solved during each iteration of solving the SCC.

There are two approaches to solve such cases. The first approach is rather straight-forward: here the SCC and the modified models are solved independently. The difficulty is that the modified models have to be solved during each iteration of the SCC and this will add significantly to the computing time in order to attain total convergence. If there are multiple modified models then the problem amplifies.

The second method is to solve SCC and the modified models jointly at the system level. Since the Gauss-Newton method is applicable for solving both the SCC and modified models, the problem has been reformed to solve these together.

The feedback variables of the SCC and the *xm* variables of the modified models are solved together. During each iteration of the SCC all these variables are computed by the Gauss-Newton method and passed on to the required models. The iterative solving is continued until convergence is attained. The example given below explains the procedure in detail.



(a)

	X_1	X_3	X_4	X_5	X_6	X_7
Model1	i	i	o			
Model2	i	i		o		o
Model3	o	i		i	i	

(b)

$$D = \begin{array}{c} \\ \mathrm{mod}\,el1 \\ \mathrm{mod}\,el2 \\ \mathrm{mod}\,el3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \end{array}$$

(c)

Figure 4-7 An example SCC with modified models

The Figure 4-7 shows a SCC with three models. From the incidence matrix it is clear that *model3* and *model1* are modified. The graphical view of the system is shown in Figure 4-8.

Here,

Feedback variable for the SCC= {*X1*}

xm for *model1*= {*X4*}

xm for *model3*={*X1*}

The equation for computing the iterative variables is the same as Equation 4-2. How ever here, the vector *x* will contain the feedback variables of SCC and also the *xm* variables of the modified models. The function *f* involves executing once the models in the SCC. The output of the function will be the vector containing the difference in the feedback variables in the previous and current iteration (for the SCC) and also $ym\text{-}ym^d$ (which was described earlier) (for the modified models).

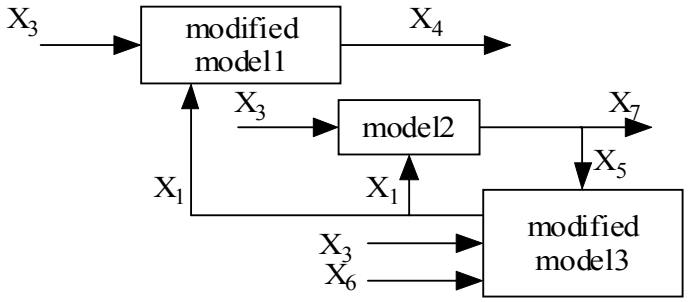


Figure 4-8 Graphical view of the SCC with modified models

In the above example, the vector *x* which has to be computed during each iteration of the SCC using the Gauss-Newton method will be {X1, X4}. In this particular case the Jacobian matrix for the Gauss-Newton method will be non-square because the length of the vector of output (3 in this case) of function *f* is different from the input vector (2 in this case).

Solving the SCC sub-system in this manner has shown faster convergence rate compared to the first method.

4.5 Summary and Conclusions

This chapter has explained the various methods that were used in this research for solving the modified models and the SCCs.

Gauss-Newton method, used for solving non-linear least-squares problems, is introduced here for solving the modified models.

Two methods, the fixed-point-iterative method and the Gauss-Newton method, used in this research for solving the SCC are described in this chapter. In addition, two different ways for applying the fixed-point-iterative method to solve SCC, are also explained.

An approach was also introduced to solve the SCC and inherent modified models jointly by the Gauss-Newton method.

It is also shown here that while conducting variable flow modelling, the sensitivity of the switched variables of the modified models has to be taken into consideration in order to avoid unsolvable modified models. The implementation of this aspect is left as a future work.

5 RESULTS AND DISCUSSION

5.1 Introduction

This chapter explains the various tests conducted in order to evaluate the methods and approaches developed as part of this research. The chapter has four sections. The first section explains the USMAC, an aircraft conceptual design test case which was widely used for testing. The second section evaluates the two objective functions which were chosen for scheduling the coupled models by genetic algorithm. The third section describes the testing of the proposed computational process modeller on two test cases. Finally, the conclusions are presented.

5.2 The USMAC Test Case

The Ultra Simplified Model of Aircraft (USMAC) is an aircraft conceptual design test case supplied by the industrial partner (VIVACE, 2005) and has been widely used for testing the various methods developed in this research. Even though it is a simplified version, the case incorporates the most relevant aspects of aircraft conceptual design, such as the multidisciplinary nature of computation, heterogeneous data and non-linear models. The simplification is done so that the computation can be as simple and as fast as possible while still being representative of real design practice.

The USMAC is a set of 97 models and 124 variables; most of the models are not longer than single line expressions.

Example: Range estimation based on Breguet-Leduc formula

Model: RA = range_(Kra,TOW,LDW,Mach,vsnd,g,lod,sfc)

RA = Kra*(vsnd*Mach*lod)/(sfc*g)*log(TOW/LDW)

End

The details of the models and the variables in USMAC are given in a tabular format in the Appendix-IV.

5.3 Evaluation of the Objective Function

A genetic algorithm based approach was presented in section 3.2.3.1 for scheduling the coupled models (SCC). Number of feedback loops number (feedback number) was considered as the objective function to be minimised.

In this section we analyse the most commonly used objective functions for rearrangement, which are the number of feedback loops (Rogers, 1997) and the feedback length (Altus et al., 1996), in the context of aircraft conceptual design. The equation to calculate the feedback length is given in Equation 2-2. In the equation the term DM is the same as the term D in Equation 3-11.

The testing is performed by studying the effect of feedback length and number of feedback loops, on the computing time of the SCC in a system.

The USMAC test case is setup for testing with the following independent variables.

Independent Variables: Awing, BPR, FNslst, Fuel, Mach_clb, Mach_crz, Mach_cth, Naisle, Npax, NpaxFront, alt_app, alt_clb, alt_crz, alt_cth, alt_to, disa_clb, disa_crz, disa_cth, disa_to, ne, phi, span, tuc.

After computational process modelling, out of the 97 models, 15 models are identified as strongly connected. They are:

take_off_weight_,system_mass_,landing_gear_mass_,wing_mass_,manu_weight_empty_,ope_weight_empty_,landing_weight_,mean_cruise_mass_crz,level_flight_crz,friction_drag_crz,induced_drag_crz,drag_factor_crz,lift_to_drag_crz,range_crz,operator_item_mass_.

In this case there was only a single variable flow model identified for the system and there was no modified model present in the system.

In order to evaluate the objective functions, models belonging to the SCC are arranged in various sequences. Each sequence has its corresponding feedback length and feedback number (i.e. the number of feedback loops). Each arrangement of the SCC and the remaining non-coupled models are assembled into fully executable systems which can compute all the unknown variables. Since the convergence of the SCC can be solver dependent, each SCC sub-system has been solved individually with Fixed-point iteration first method, Fixed-point iteration second method and Gauss-Newton method,

as solvers. (The features of those solvers were explained in section 4.3). The number of calls made to each model in the SCC while solving them are considered as the criteria for deciding the computational cost involved in solving the SCC. Different tests conducted have proven that these 'number of calls' correspond to the computing time and cost for the system.

For testing the efficiency of the feedback number as the objective function, each system is arranged according to the increasing feedback number for its constituent SCC and further executed. The sum of the calls made to each model of the SCC, while solving the SCCs using each of the three solvers, which corresponds to its computational cost are given in Table 5-1. In the table, the systems are arranged in the increasing order of the feedback number. The corresponding graph (curve fit) is given in Figure 5-1. It can be noted that there are many arrangements possible for the models in the SCC for the same feedback number. For example there were 12 ways of arranging the models in the SCC with feedback number of 4. All these arrangement were taken into consideration so that the net effect can be observed rather considering just a single arrangement for a particular feedback number.

In Table 5-1, the fifth column for Gauss-Newton method represents the sum of calls made to each model in the SCC for solving, excluding the ones for calculating Jacobians. The subsequent sixth column sums up all the calls. The starting points for the feedback variables, while solving the SCC, for all the three solvers are given the value of unity.

Table 5-1 Number of calls made by each solver to solve the SCC

		Sum of the calls made to each model in the SCC			
USMAC System No.	**Feedback number**	**FPI- first method**	**FPI-second method**	**Gauss-Newton method**	**Gauss-Newton method (including calls for calculating jacobians)**
1	2	387	360	165	540
2	3	428	630	285	1200
3	3	441	435	195	840
4	4	486	570	285	1440
5	4	486	555	285	1440
6	4	486	615	285	1500
7	4	1279	660	375	1890

8	4	1304	660	375	1890
9	4	1309	660	375	1890
10	4	1324	660	375	1890
11	4	1334	660	375	1890
12	4	1349	660	375	1890
13	4	1349	660	375	1890
14	4	1354	660	375	1890
15	4	1359	660	375	1890
16	5	1443	645	285	1800
17	5	1468	630	285	1725
18	5	1493	630	285	1725
19	5	1498	645	285	1800
20	5	1513	660	285	1725
21	5	1523	630	285	1725
22	5	1528	630	285	1725
23	5	1533	630	285	1725
24	5	1538	630	285	1725
25	5	3197	735	390	2355
26	5	3314	735	390	2355
27	5	3382	735	390	2355
28	5	3398	735	390	2355
29	5	3398	735	390	2355
30	5	3420	735	390	2355
31	5	3455	735	390	2355
32	5	3466	735	390	2355
33	5	3474	735	390	2355
34	5	3485	735	390	2355
35	5	3485	735	390	2355
36	5	3542	735	390	2355
37	6	3741	735	285	2100
38	6	3798	735	285	2100
39	6	3809	720	285	2100
40	6	3825	720	285	2100
41	6	3828	735	285	2100
42	6	3828	720	285	2100
43	6	3844	720	285	2100
44	6	3882	720	285	2100
45	6	3901	720	285	2100
46	6	3912	735	285	2100
47	6	3931	735	285	2100
48	6	4010	810	390	2745
49	6	4113	810	390	2745
50	6	4137	810	390	2745
Average calls		**2475.74**	**681.6**	**328.5**	**1984.8**

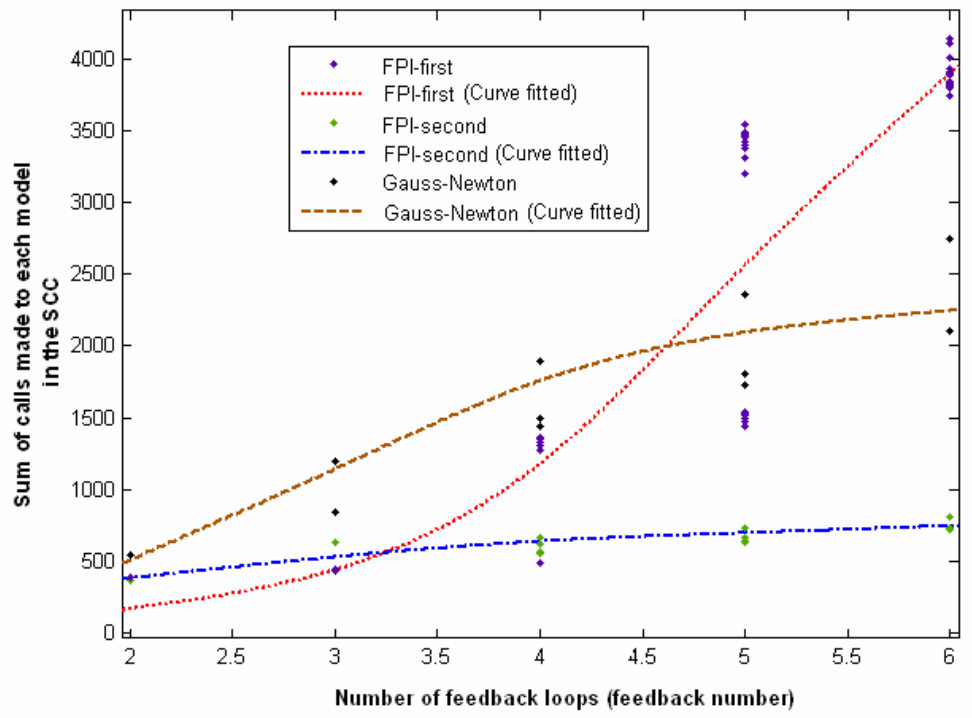


Figure 5-1 Feedback number versus sum of the calls made to each model in the SCC by each solver (The graph is curve fitted)

From the graph in the Figure 5-1 it is clear that, while solving the SCC in the system with FPI-first solver and Gauss-Newton, there is an increase in the sum of the calls to the models with increase in the feedback number. For FPI-second solver the effect of the feedback number on the sum of the calls to the models is negligible.

Compared to the other two solvers FPI-second solver has taken the least average number of calls to solve the SCC (Average number of calls: FPI-first: 2475, FPI-second: 681 and GN: 1984 calls). This indicates that FPI-second is the most efficient solver for SCC in this case.

As the next step, in order to test the efficiency of the feedback length as the objective function, each system is arranged according to the increasing feedback length for its constituent SCC and further executed. The sum of the calls made to each model of the SCC for solving it, for each system arrangement, by each of the three solvers, are given in the Table 5-2. In the table, the systems are arranged in the increasing order of the feedback length. The corresponding graph (spline fit) representation is in Figure 5-2.

Table 5-2 Number of calls made by each solver to solve the SCC

		Sum of the calls made to each model in the SCC			
USMAC system No.	**Feedback length**	**FPI- first method**	**FPI-second method**	**Gauss-Newton method**	**Gauss-Newton method (including calls for calculating jacobians)**
1	23	610	255	285	870
2	23	610	255	285	870
3	24	611	210	150	495
4	25	657	210	150	495
5	25	675	435	285	1200
6	25	702	255	285	870
7	25	976	255	285	870
8	25	976	255	285	870
9	26	663	210	150	495
10	26	703	210	150	495
11	26	748	255	285	870
12	26	1037	255	285	870
13	27	195	210	165	360
14	27	699	210	150	495
15	27	749	210	150	495
16	29	996	300	285	1155
17	29	1083	300	285	1155
18	30	747	240	285	870
19	30	1078	300	285	1155
20	31	315	255	180	585
21	31	590	255	180	780
22	31	761	240	285	870
23	31	1117	300	285	1155
24	31	1638	300	285	1155
25	32	1182	255	180	780
26	33	328	255	180	585
27	33	643	255	180	780
28	33	1155	255	180	780
29	33	2185	255	180	780
30	33	2903	300	285	1155
31	34	1173	255	180	780
32	34	1283	300	285	1155
33	35	671	300	285	1155
34	35	696	255	180	780
35	35	1329	300	285	1155
36	35	1989	300	285	1155
37	36	717	300	285	1155
38	36	732	300	285	1155

39	36	780	255	180	780
40	37	1332	255	180	780
Average calls		**975.85**	**264.375**	**232.875**	**860.25**

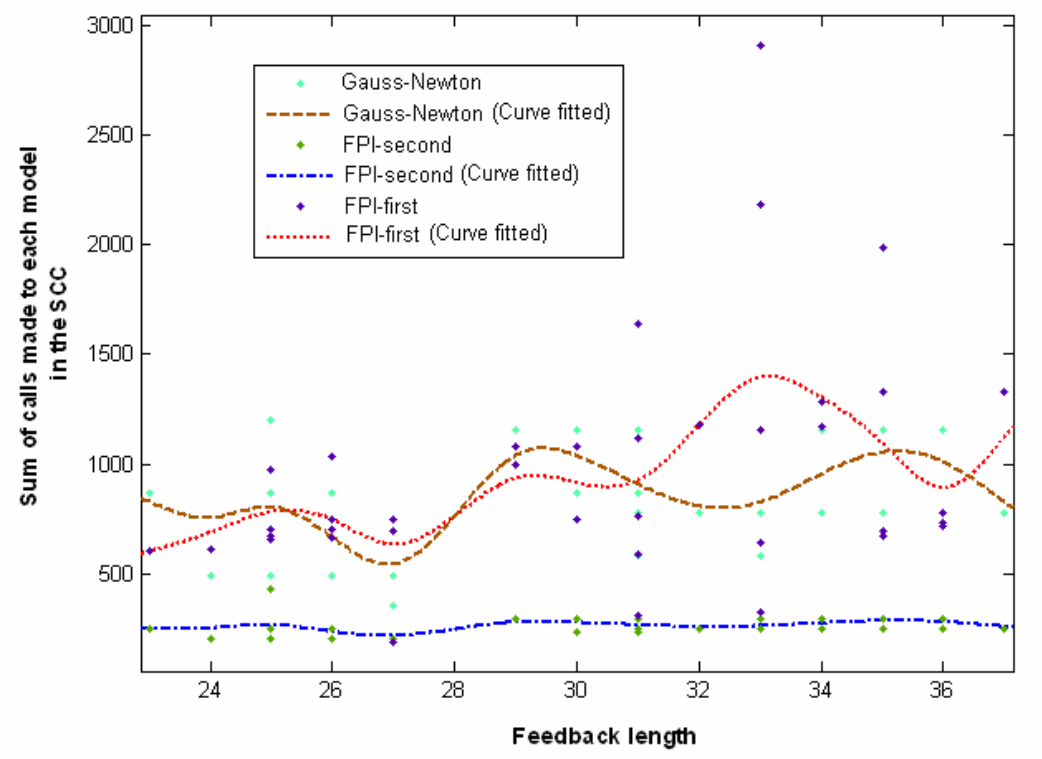


Figure 5-2 Feedback length versus sum of the calls made to each model in the SCC by each solver (The graph is curve fitted)

In this case, the sum of the calls made to the models (Figure 5-2), by each of the three solvers, for solving the SCC, did not show any particular pattern of influence from the feedback length. The effect was much more evident in the previous case of feedback number.

As in the previous case, compared to the other two solvers, FPI-second solver has taken the least average calls to solve the SCC (Average number of calls: FPI-first: 975, FPI-second: 264 and GN: 860 calls).

From the above two case setup it can be concluded that, compared to the feedback length, increase in the feedback number lead to increased computational cost for the SCC especially when FPI-first method is used for solving it.

The conclusions derived from the testing performed above prove that during the rearrangement of the models in the SCC by GA, reducing the feedback number actually reduces the computing cost for the SCC. This confirms the appropriateness of feedback number being chosen as the objective function while rearranging the SCC.

Even though FPI-second solver had the least average calls in solving the SCC, the chosen objective functions did not correspond with the computational cost associated with the FPI-second solver. From this observation it can be concluded that the objective function is solver depended and cannot be generalised for any solver. Since the currently tested objective function of number of feedback loops has shown good correspondence with the FPI-first solver, they were used as the objective function and solver during the various tests conducted during this research.

5.4 Computational Process Modelling Evaluation

The computational process modeller for computational process modelling of complex systems was presented in Chapter 3. This section analyses the effectiveness of this process plan in generating optimal computational plan for the systems by applying and testing it on a simple sizing test case and the USMAC case.

After applying the computational process modeller to each case the computational cost for the optimal computational plan obtained is compared with the non-optimal ones generated while performing the computational process modeller. Here the computational plan refers to the data flow obtained after variable flow modelling and also the execution sequence obtained for the models after the decomposition and scheduling. The non-optimal computational plan refers to thecomputational plan based on those variable flow models for the SCCs which were not selected as the optimal. The comparison is performed on the basis of the number of calls made to the models of the SCC during the solving process. Here, the number of calls corresponds to the computational cost for the system and hence chosen as the standard for comparison.

The strongly connected components are solved using FPI- first method, since the objective function (feedback number) used for rearranging the coupled models in the SCC fits well with this solver. This was proven in the previous section. In all the cases the unknown variables are given a starting point of unity while solving.

The next two sections describe the testing of the computational process modeller on a simplified sizing test case and the USMAC test case.

5.4.1 Simplified sizing test case

A simplified set of aircraft sizing equations from Buckley (1992) is considered for testing the computational process modeller. The equations are shown in Figure 5-3.

1. $We=Wo*2.61*Wo(-0.1)*(Wo/Sref)(-0.05)$
2. $Wo=Wf+We$
3. $Walt=0.985*WLO$
4. $W_x=0.995*W_{ec}$
5. $W_f=1.06*(1-W_x/W_o)*W_e$
6. $W_{LO}=0.97*W_o$
7. $Wec=Exp[0.00043R]*Walt$

Figure 5-3 Simplified aircraft sizing problem (test case)

Here W_e is the empty weight, W_o is the gross take off weight, S_{ref} is the wing area and R represents the range. This set of equations is for the mission profile shown in Figure 5-4. The symbols $W_{\#}$ in Figure 5-4 represent the weight of aircraft at each position of the mission. These equations have been considered for testing purpose only and may not represent a real aircraft case.

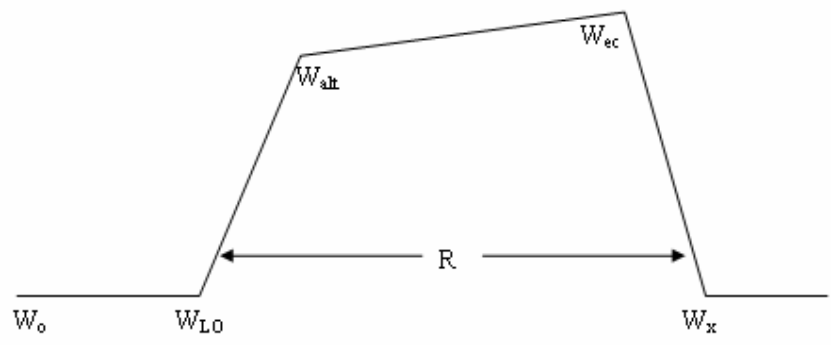


Figure 5-4 Mission profile for the sizing problem

Since in this research the focus is on generating computational plans for models, these equations are compiled and formed as models. The inputs and outputs for each model are shown in Figure 5-5.

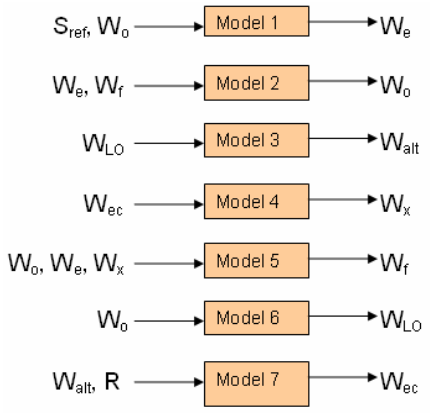


Figure 5-5 Models for simplified aircraft sizing problem

This test case is set up with two different sets of independent variables. Both cases have two variables as inputs, which leads to a determined system:

TNvar –NIvar-Noutmod = 9-2-7=0 → determined system

Table 5-3 Independent variables selected for each case

	Independent variables
Case1	R, W_e
Case 2	R, Sref

The set of input variables in each case are given in Table 5-3. The two cases are diverse in terms of the generated computational plan. The first case has six models and the second case has all the seven models constituting the SCC.

5.4.1.1 Case1

The final matrix obtained after applying IMM with R and W_e as independent variables is shown in the Figure 5-6.

	R	Sref	Walt	We	Wec	Wf	Wlo	Wo	Wx
model1		3		2				2	
model2				2		1		1	
model3			1				1		
model4					1				1
model5				2		1		1	1
model6							1	1	
model7	2		1		1				

Figure 5-6 Incidence matrix for Case1 of the sizing test case (Blank cells denote '0').

From the figure it is clear that models 2, 3, 4, 5, 6 and 7 are part of the SCC, since there are 1s stills remaining in the corresponding rows of the matrix (The decomposition performed later has shown that these models belong to a single SCC). On further solving for the SCC, three variable flow models are generated. The variable flow models and the corresponding rearranged DSMs obtained for the SCC are shown in Figure 5-7. In the DSMs shown in the figure, value of 1 on the diagonal elements represents the models which are modified as a result of variable flow modelling.

Varaible flow model 1
NFdb=2, nMm=2

	R	Sref	Walt	We	Wec	Wf	Wlo	Wo	Wx
model2				2		3		2	
model3			3				2		
model4					2				3
model5				2		2		3	2
model6							3	2	
model7	2		2		3				

DSM

model5	1	1			1	
model6			1			
model3				1		
model7					1	1
model2	1				1	
model4	1					

Variable flow model 2
nFdb=1, nMm=6

	R	Sref	Walt	We	Wec	Wf	Wlo	Wo	Wx
model2				2		3		2	
model3			2				3		
model4					3				2
model5				2		2		2	3
model6							2	3	
model7	2		3		2				

DSM

model4	1	1				
model7		1	1			
model3			1	1		
model6				1	1	1
model2					1	1
model5	1					1

Variable flow model 3
nFdb=1, nMm=0

	R	Sref	Walt	We	Wec	Wf	Wlo	Wo	Wx
model2				2		2		3	
model3			3				2		
model4					2				3
model5				2		3		2	2
model6							3	2	
model7	2		2		3				

DSM

model2		1				1
model6			1			
model3				1		
model7					1	
model4						1
model5	1					

Figure 5-7 Variable flow models and corresponding rearranged DSMs of the SCCs for case 1 of the sizing test case (Blank cells denote '0').

Solving

For a comparative testing each variable flow model obtained for the SCC (shown in Figure 5-7) are setup and executed. Table 5-4 provides the details of the variable flow models for the SCC, along with their execution and other relevant details.

Table 5-4 Details of computational process modeling and solving of SCC for case 1 of the sizing test case.

Computational process modelling				Solving	
Variable flow model	Number of feedback loops (nFdb)	Number of modified models (nMm)	Optimal flow model (chosen by process plan)	Number of calls to the models in SCC	% additional computational cost
1	2	2		340	529% more
2	1	6		434	703% more
3	1	0	■	54	base

Discussion

The optimal variable flow model selected by the computational process modeller is the third one which has the least number of modified models (zero in this case). On executing, from the number of calls made to the models belonging to the SCC for each of the three different variable flow models it is clear that the third variable flow model has made the least number of calls. This agrees with the computational process modeller's choice of optimal flow model.

Since there were no variable flow models which have equal number of modified models, there was no requirement for comparing and selecting from them the optimal one based on the feedback number. However it has to be noted that the SCC for each variable flow model was rearranged using genetic algorithm in order to obtain the arrangement with least feedback number.

From the table it is also understandable that as the number of modified models increases the computational cost increases.

The final computational plan generated by the computational process modeller, after applying the final scheduling, is given below.

Computational Plan

Inputs for the system

R, W_e

Outputs of the system

Sref,Walt,Wec,Wf,Wlo,Wo,Wx

Execution sequence

1) **scc_1**, 2) **modified_model1**

In the list above, the models with prefix 'modified' means, modified models.The SCC in the system are represented with prefix 'scc'. The process number 1 in the process execution sequence given above is a strongly connected component. The details of this process are given below.

Strongly Connected Components

1. scc_1

Inputs to SCC

R, W_e

Outputs of SCC

Walt, Wec, Wf, Wlo, Wo, Wx

Execution sequence

model2, model6, model3, model7, model4, model5

Treatment (Solver)

FPI-first method

The variable flow model for this SCC is the variable flow model number 3 shown in the Figure 5-7.

5.4.1.2 Case2

The final matrix obtained after applying IMM with R and Sref as independent variables is shown in the Figure 5-8. From the Figure 5-8 it is clear that all the models in the system are part of the SCC, since there are 1s stills remaining in each row of the matrix (The decomposition performed later has shown that these models belong to a single SCC). On further solving for the SCC, four variable flow models are generated.

	R	Sref	Walt	We	Wec	Wf	Wlo	Wo	Wx
model1		2		1				1	
model2				1		1		1	
model3			1				1		
model4					1				1
model5				1		1		1	1
model6							1	1	
model7	2		1		1				

Figure 5-8 Incidence matrix for Case2 of the sizing test case (Blank cells denote '0').

The variable flow models and the corresponding rearranged DSMs obtained are shown in the Figure 5-9.

Variable flow model 1
nFdb=1, nMm=6,

	R	Sref	Walt	We	Wec	Wf	Wlo	Wo	Wx
model1		2		3				2	
model2				2		3		2	
model3			2				3		
model4					3				2
model5				2		2		2	3
model6							2	3	
model7	2		3		2				

DSM

model3	1	1				
model6		1	1	1	1	
model1				1	1	
model2				1	1	
model5					1	1
model4						1
model7	1					1

Variable flow model 2
nFdb=2, nMm=0

	R	Sref	Walt	We	Wec	Wf	Wlo	Wo	Wx
model1		2		3				2	
model2				2		2		3	
model3			3				2		
model4					2				3
model5				2		3		2	2
model6							3	2	
model7	2		2		3				

DSM

model2		1			1	1
model6			1			
model3				1		
model7					1	
model4						1
model1	1					1
model5	1					

Variable flow model 3
nFdb=2, nMm=2

	R	Sref	Walt	We	Wec	Wf	Wlo	Wo	Wx
model1		2		2				3	
model2				3		2		2	
model3			3				2		
model4					2				3
model5				2		3		2	2
model6							3	2	
model7	2		2		3				

DSM

model2	1	1					1
model1	1	1	1				1
model6				1			
model3					1		
model7						1	
model4							1
model5	1						

Variable flow model 4
nFdb=2 nMm=3

	R	Sref	Walt	We	Wec	Wf	Wlo	Wo	Wx
model1		2		2				3	
model2				2		3		2	
model3			3				2		
model4					2				3
model5				3		2		2	2
model6							3	2	
model7	2		2		3				

DSM

model1	1	1				1	1
model6			1				
model3				1			
model7					1		
model4							1
model2						1	1
model5	1					1	1

Figure 5-9 Variable flow models of the SCCs for case 2 of the sizing test case (Blank cells denote '0').

Solving

Table 5-5 provides the details of the variable flow models for the SCC, along with their execution and other relevant details. The variable flow models are given in the Figure 5-9.

Table 5-5 Details of computational process modeling and solving of SCC for case 2 of the sizing test case.

Computational process modelling				Solving	
Variable flow model	nFdb	nMm	Optimal flow model (chosen by process plan)	Number of calls to the models in SCC	% additional computational cost
1	1	6		1167	495% more
2	2	0	■	196	base
3	2	2		579	195% more
4	2	3		853	335% more

Discussion

The optimal variable flow model selected by the computational process modeller is the second one which has the least number of modified models (zero in this case). While solving, the second variable flow model for the SCC has the least number of calls to the models. This supports the computational process modeller's choice of the second variable flow model as the optimum one.

As in the previous case, from the table it is also understandable that as the number of modified models increases the computational cost increases.

The final computational plan generated by the computational process modeller, after applying the final scheduling, is given below.

Computational Plan

Inputs for the system

R, Sref

Outputs of the system

W_e,Walt,Wec,Wf,Wlo,Wo,Wx

Execution sequence

1) scc_1

The process number 1 in the process execution sequence given above is a strongly connected component. The details of this process are given below.

Strongly Connected Components

1. scc_1

Inputs to SCC

R, Sref

Outputs of SCC

We,Walt, Wec, Wf, Wlo, Wo, Wx

Execution Sequence

model2, model6, model3, model7, model4, model1, model5

Treatment (Solver)

FPI-first method

The variable flow model for this SCC is the variable flow model number 2 (which was chosen as the optimum) shown in the Figure 5-9.

5.4.2 USMAC case

The USMAC test case is set up in three different ways with each one having a different set of variables as inputs. All the three cases had 23 variables as inputs, which leads to a determined system.

TNvar –NIvar-Noutmod = 124-23-101=0→ determined system

The three cases were setup in order to demonstrate the capability of the computational process modeller in generating computational plans for the system with randomly chosen inputs. The three cases were diverse in terms of the generated computational plan. The first two cases had one SCC and the last case had two SCCs in the final computational plan. The contained models of the SCC were also different in the three cases.

As in the previous cases the unknown variables are given a starting point of unity while solving. The detailed explanations regarding how the final computational plan for each

case was generated are not given here. However, the details of the final computational plan obtained are given for each case.

5.4.2.1 Case1

The inputs variables given for system are Awing, Leh, MTOW, Mach_clb, Mach_crz, Mach_cth, PL, RA, alt_clb, alt_crz, alt_cth, alt_to, dfus, disa_clb, disa_crz, disa_cth, disa_to, dnac, g_app, ne, phi, span and tuc.

After decomposition thirteen models are identified as strongly connected in the system. They are;

level_flight_crz,induced_drag_crz,mean_cruise_mass_crz,friction_drag_crz,drag_factor_crz,landing_weight,ope_weight_empty,manu_weight_empty,lift_to_drag_crz,engine_mass, range_crz, spec_fuel_cons and nacelle_diameter.

There were twelve variable flow models generated for the SCC. However, only four variable flow models produced a converged solution and hence only these will be discussed here. The non-converged once were those variable flow models which have higher number of modified models and feedback numbers. The details of one of the twelve variable flow models which gave a non converged solution are also given for completeness. The incidence matrices and the corresponding rearranged DSMs of these five flow models are given in the Appendix-V.i (Figure A- 3 to Figure A- 12).

Table 5-6 Details of computational process modeling and solving of SCC for case 1

Computational process modelling				**Solving**	
Variable flow model	nFdb	nMm	Optimal flow model (chosen by process plan)	Number of calls to the models in SCC	% additional computational cost
1	3	6		117	95% more
2	5	11		158	163 % more
3	6	3	■	60	base
4	5	9		198	230% more
5	8	11		Non converged	-

Solving

Table 5-6 provides the details of the variable flow models for the SCC, along with their execution and other relevant details.

Discussion

The variable flow model 3 was chosen by the computational process modeller as the optimal one, since this flow model has the least number of modified models. Since there were no variable flow models which have equal number of modified models, there was no requirement for comparing and selecting from them the optimal one based on the feedback number.

It is shown in the Table 5-6, that the selected optimal flow model has the lowest computational cost for the SCC in comparison with the other variable flow models.

From the Table 5-6 it is also clear that when the number of modified models increases the computational cost for the SCC also increases. However, for the variable flow model number 4, even though the number of modified models is less than the one for the flow model 2, it has taken more calls to obtain a converged solution.

This discrepancy was studied in detail. It was identified that they occurred because the convergence of the SCC was not only depended on the number of modified models and the feedback number, but also on various other factors such as, the starting point for the unknown variables, mutual sensitivity of the switched variables of the modified models (see section 4.2 for more details) and other factors which are yet to be discovered. In the current research, the focus has been on the modified models and the feedback number.

The variable flow model 5 which had the highest number of modified models and feedback number lead to a non-converged solution. This approves the fact that higher the number of modified models and feedback number, higher will be the computational cost for the system.

The final computational plan obtained for USMAC after performing the computational process modelling using the computational process modeller, is given in the next

paragraph. The incidence matrix which displays the variable flow model generated by IMM, for this system is shown in Figure A- 1 in the Appendix-V.i. In the figure models belonging to SCC are confined into a single process.

Computational Plan

Inputs for the system

Awing, Leh, MTOW, Mach_clb, Mach_crz, Mach_cth, PL, RA, alt_clb, alt_crz, alt_cth, alt_to, dfus, disa_clb, disa_crz, disa_cth, disa_to, dnac, g_app, ne, phi, span, tuc

Outputs of the system

lod_crz, lod_cth, mass_clb, mass_crz, mass_cth, rho_clb, rho_cth, rho_to, sfc, tofl, vapp, vsnd_clb, vsnd_crz, vsnd_cth, vz_clb, wAfus, wAht, wAnac, wAvt, wAwing

Execution sequence

1)aspect_ratio_,2)fin_lever_arm_,3)fin_volume_factor_,4)tail_volume_factor_,5)reference_length_,6)gravity_acc_cth,7)non_stand_atmos_cth,8)top_of_climb_mass_cth,9)gravity_acc_clb,10)non_stand_atmos_clb,11)top_of_climb_mass_clb,12)one_pax_weight_,13)Cz_max_TO_factor_,14)fin_size_,15)tail_size_,16)**modified_tail_lever_arm__**tr,17)reference_area_,18)gravity_acc_to,19)non_stand_atmos_to,20)level_flight_cth,21)level_flight_clb,22)non_stand_atmos_crz,23)**modified_Payload__**tr,24)Cz_max_TO_,25)ref_mach_number_,26)fric_drag_factor_,27)ind_drag_factor_,28)press_drag_factor_,29)fin_wetted_area_,30)tail_wetted_area_,31)wing_wetted_area_,32)fus_wetted_area_,33)nac_wetted_area_,34)Mach_stall_to_,35)Kvs_Take_Off,36)friction_drag_cth,37)induced_drag_cth,38)pressure_drag_cth,39)friction_drag_clb,40)induced_drag_clb,41)pressure_drag_clb,42)pressure_drag_crz,43)gravity_acc_crz,44)sound_velocity_crz,45)MWE_factor_,46)operator_item_mass_,47)furnishing_mass_,48)fin_mass_,49)tail_mass_,50)wing_mass_,51)system_mass_,52)landing_gear_mass_,53)fuselage_mass_,54)sfc_factor,55)secured_Mach_to,56)air_density_to,57)drag_factor_cth,58)air_density_cth,59)drag_factor_clb,60)air_density_clb,61)Cz_max_LD_facto

r_,62)**scc_test_smac_tr**,63)Kvs_Landing,64)max_take_off_factor,65)net_thrust_to,66)net_thrust_cth,67)lift_to_drag_cth,68)max_climb_factor,69)net_thrust_clb,70)lift_to_drag_clb,71)sound_velocity_clb,72)max_cruise_factor,73)**modified_take_off_weight**__tr,74)wing_fuel,75)Cz_max_LD_,76)**modified_fuselage_length**__tr,77)app_speed_1,78)**modified_gravity_acc_app**_tr,79)tofl_1,80)cruise_thrust_1,81)sound_velocity_cth,82)climb_rate_1,83)time_crz,84)fus_fuel_ratio_,85)**modified_fuselage_diameter**__tr

The process number 62 in the process execution sequence given above is a strongly connected component. The details of this process are given below.

Strongly Connected Components

62. scc_test_smac_tr

Inputs to SCC

Aref, Kcx0, Kind, Kmwe, Ksfc, Lref, MTOW, Mach_crz, Mfurn, Mfus, Mgear, Mht, Mop, Msys, Mvt, Mwing, PL, Pamb_crz, RA, Tamb_crz, ar, cxc_crz, dnac, g_crz, lfus, ne, vsnd_crz, wAfus, wAht, wAnac, wAvt, wAwing

Outputs of SCC

BPR, FNslst, LDW, MWE, Mprop, OWE, cx0_crz, cx_crz, cxi_crz, cz_crz, lod_crz, mass_crz, sfc

Execution sequence

level_flight_crz,induced_drag_crz,mean_cruise_mass_crz,friction_drag_crz,drag_factor_crz,landing_weight,ope_weight_empty,manu_weight_empty,lift_to_drag_crz,engine_mass,**modified_range_crz_tr**, **modified_spec_fuel_cons**, **modified_nacelle_diameter**

Treatment (Solver)

FPI-first method

The variable flow model for this SCC is the variable flow model number 3 (which was chosen as the optimum) shown in the Figure A- 8.

The values of the input and output variables obtained after executing the system are given in the Table 5-7.

Table 5-7 Values of the input and output variables obtained after executing the system for case1

Input Variables	Output Variables			
Awing=158.9697	Aht=27.8359	Mach_stall_to=0.18096	Vht=0.7	g_to=9.8337
Leh=19.801	Aref=158.9697	Mach_to=0.20448	Vvt=0.05	kfn_cth=0.87825
MTOW=91596.0416	Avt=26.8834	Mchar=0.79638	Wpax=115	kvs_LD=1.23
Mach_clb=0.78	BPR=6	Mfurn=3787.5	alt_app=0	kvs_TO=1.13
Mach_crz=0.78	FNslst=147061.7694	Mfus=10288.5733	ar=6.4797	lfus=40.25
Mach_cth=0.78	Fn_clb=42913.3285	Mgear=3680.6213	cx0_clb=0.015652	lod_clb=17.4065
PL=17250	Fn_cth=42913.3285	Mht=711.3937	cx0_crz=0.015652	lod_crz=17.1901
RA=7203619.4473	Fn_to=120065.3339	Mop=7550.3395	cx0_cth=0.015652	lod_cth=17.4065
alt_clb=10668	Fuel=23999.9997	Mprop=7958.6491	cx_clb=0.030353	mass_clb=87016.2395
alt_crz=10668	Fwing=19414.3782	Msys=5431.0733	cx_crz=0.028114	mass_crz=79596.0418
alt_cth=10668	Kcx0=1.4	Mvt=693.7676	cx_cth=0.030353	mass_cth=87016.2395
alt_to=0	Kcxp=1	Mwing=10244.124	cxc_clb=0.00098888	rho_clb=0.3796
dfus=4.1	KczmaxLD=1	Naisle=0.99998	cxc_crz=0.00098888	rho_cth=0.3796
disa_clb=0	KczmaxTO=1	Npax=150	cxc_cth=0.00098888	rho_to=1.225
disa_crz=0	Kff=0.80893	NpaxFront=6	cxi_clb=0.013713	sfc=1.7111e-005
disa_cth=0	Kind=1	OWE=50346.0419	cxi_crz=0.011474	tofl=1708.8739
disa_to=0	Kmcl=0.7	Pamb_clb=23842.2717	cxi_cth=0.013713	vapp=61.2978
dnac=2.2353	Kmcr=0.65	Pamb_crz=23842.2717	cz_clb=0.52834	vsnd_clb=296.5339
g_app=9.8337	Kmto=1	Pamb_cth=23842.2717	cz_crz=0.48329	vsnd_crz=296.5339
ne=2	Kmwe=1	Pamb_to=101325	cz_cth=0.52834	vsnd_cth=296.5339
phi=0.38173	Ksfc=1	RA_time=31144.529	czmax_LD=2.7488	vz_clb=3.006
span=32.0948	LDW=67596.0419	Tamb_clb=218.808	czmax_TO=2.4395	wAfus=445.5678
tuc=0.10813	Lev=19.801	Tamb_crz=218.808	g_clb=9.8008	wAht=43.6909
	Lref=4.9531	Tamb_cth=218.808	g_crz=9.8008	wAnac=32.1095
	MWE=42795.7024	Tamb_to=288.15	g_cth=9.8008	wAvt=54.1282
				wAwing=248.5631

5.4.2.2 Case2

In this case the inputs variables given for system are Awing, Leh, MTOW, MWE, Mach_clb, Mach_cth, Mprop, Npax, RA, alt_app, alt_clb, alt_crz, alt_cth, alt_to, dfus, disa_clb, disa_crz, disa_cth, disa_to, ne, sfc, span and tuc.

After decomposition seven models are identified as strongly connected in the system. They are;

level_flight_crz,range_crz,lift_to_drag_crz,induced_drag_crz,friction_drag_crz,pressure_drag_crz,drag_factor_crz

There were seven variable flow models generated for the SCC. However, only three variable flow models gave a converged solution and hence only these will be discussed here. The non-converged once were those variable flow models which had higher number of modified models and feedback numbers. The details of one of the seven variable flow models which gave a non converged solution are also given for completeness. The incidence matrices and the corresponding rearranged DSMs of these four flow models are given in the Appendix-V.ii (Figure A- 15 to Figure A- 22).

Solving

Table 5-8 provides the details of the SCC and its variables flow models, along with their solving details.

Table 5-8 Details of computational process modeling and solving of SCC for case 2

Computational process modelling				Solving	
Variable flow model	nFdb	nMm	Optimal flow model selected by computational process modeller	Number of calls to the models in SCC	% additional computational cost
1	1	4		220	base
2	1	5		900	309% more
3	2	3	■	253	15% more
4	2	5		Non-converged	-

Discussion

Since the third flow model has the lowest number of modified models (three), this was chosen during computational process modeller as the optimal one. However after execution it was found that, variable flow model 1 had lower computational cost compared to flow model 3 even though the number of modified models was higher for this flow model. The reasons for this have been explained in the previous case-1. Nevertheless, the computational process modeller has chosen a variable flow model 3 which still has much less computational cost compared to the flow model number 2. It can also be noted that the variable flow model number 4 which had the highest number of modified model and feedback number lead to a non converged solution.

The final computational plan obtained for USMAC is given in the next paragraph. The incidence matrix, which displays the variable flow model generated by IMM for this system is given in Figure A- 13 in the Appendix-V.ii. In the figure models belonging to SCC are confined into a single process.

Computational Plan

Inputs to the system

Awing, Leh, MTOW, MWE, Mach_clb, Mach_cth, Mprop, Npax, RA, alt_app, alt_clb, alt_crz, alt_cth, alt_to, dfus, disa_clb, disa_crz, disa_cth, disa_to, ne, sfc, span, tuc

Outputs of the system

Aht, Aref, Avt, BPR, FNslst, Fn_clb, Fn_cth, Fn_to, Fuel, Fwing, Kcx0, Kcxp, KczmaxLD, KczmaxTO, Kff, Kind, Kmcl, Kmcr, Kmto, Kmwe, Ksfc, LDW, Lev, Lref, Mach_crz, Mach_stall_to, Mach_to, Mchar, Mfurn, Mfus, Mgear, Mht, Mop, Msys, Mvt, Mwing, Naisle, NpaxFront, OWE, PL, Pamb_clb, Pamb_crz, Pamb_cth, Pamb_to, RA_time, Tamb_clb, Tamb_crz, Tamb_cth, Tamb_to, Vht, Vvt, Wpax, ar, cx0_clb, cx0_crz, cx0_cth, cx_clb, cx_crz, cx_cth, cxc_clb, cxc_crz, cxc_cth, cxi_clb, cxi_crz, cxi_cth, cz_clb, cz_crz, cz_cth, czmax_LD, czmax_TO, dnac, g_app, g_clb, g_crz, g_cth, g_to, kfn_cth, kvs_LD, kvs_TO, lfus, lod_clb, lod_crz, lod_cth, mass_clb, mass_crz, mass_cth, phi, rho_clb, rho_cth, rho_to, tofl, vapp, vsnd_clb, vsnd_crz, vsnd_cth, vz_clb, wAfus, wAht, wAnac, wAvt, wAwing

Execution sequence

1)aspect_ratio_,2)tail_volume_factor_,3)reference_length_,4)fin_volume_factor_,5)fin_lever_arm_,6)tail_size_,7)**modified_tail_lever_arm**__tr,8)fin_size_,9)tail_mass_,10)system_mass_,11)sfc_factor,12)landing_gear_mass_,13)fuselage_mass_,14)furnishing_mass_,15)fin_mass_,16)MWE_factor_,17)**modified_spec_fuel_cons**_tr,18)**modified_manu_weight_empty**__tr,19)**modified_engine_mass**__tr,20)**modified_wing_mass**__tr,21)top_of_climb_mass_cth,22)top_of_climb_mass_clb,23)reference_area_,24)operator_item_mass_,25)one_pax_weight_,26)non_stand_atmos_cth,27)non_stand_atmos_clb,28)nacelle_diameter_,29)gravity_acc_cth,30)gravity_acc_clb,31)Cz_max_TO_factor_,32)wing_wetted_area_,33)tail_wetted_area_,34)ref_mach_number_,35)press_drag_factor_,36)ope_weight_empty_,37)non_stand_atmos_to,38)nac_wetted_area_,39)level_flight_cth,40)level_flight_clb,41)ind_drag_factor_,42)gravity_acc_to,43)fus_wetted_area_,44)fric_drag_factor_,45)fin_wetted_area_,46)Payload_,47)Cz_max_TO_,48)pressure_drag_cth,49)pressure_drag_clb,50)non_stand_atmos_crz,51)landing_weight_,52)induced_drag_cth,53)induced_drag_clb,54)friction_drag_cth,55)friction_drag_clb,56)Mach_stall_to_,57)Kvs_Take_Off,58)sound_velocity_crz,59)secu

red_Mach_to,60)mean_cruise_mass_crz,61)gravity_acc_crz,62)drag_factor_cth,63)drag_factor_clb,64)air_density_to,65)air_density_cth,66)air_density_clb,67)Cz_max_LD_factor_,68)wing_fuel,69)**modified_take_off_weight__**tr,70)sound_velocity_clb,71)net_thrust_to,72)net_thrust_cth,73)net_thrust_clb,74)max_take_off_factor,75)max_cruise_factor,76)max_climb_factor,77)lift_to_drag_cth,78)lift_to_drag_clb,79)gravity_acc_app,80)**modified_fuselage_length__**tr,81)**scc_1_tr**,82)Kvs_Landing,83)Cz_max_LD_,84)tofl_1,85)time_crz,86)sound_velocity_cth,87)**modified_fuselage_diameter__**tr,88)fus_fuel_ratio_,89)cruise_thrust_1,90)climb_rate_1,91)app_speed_1

The process number 81 in the execution sequence given above is a strongly connected component. The details of this SCC are given below.

Strongly Connected Components

81. scc_1_tr

Inputs to SCC

Aref, Kcx0, Kcxp, Kind, LDW, Lref, MTOW, Mchar, Pamb_crz, RA, Tamb_crz, ar, g_crz, lfus, mass_crz, ne, sfc, vsnd_crz, wAfus, wAht, wAnac, wAvt, wAwing

Outputs of SCC

Mach_crz, cx0_crz, cx_crz, cxc_crz, cxi_crz, cz_crz, lod_crz

Execution Sequence

modified_level_flight_crz,**modified_range_crz**,**modified_lift_to_drag_crz**,induced_drag_crz,friction_drag_crz,pressure_drag_crz,drag_factor_crz

Treatment (Solver)

FPI-first method

The variable flow model for this SCC is the variable flow model number 3 (which was chosen as the optimum) shown in the Figure A- 20.

The values of the input variables, and output variables obtained after executing the system, are given in the Table 5-9.

Table 5-9 Values of the input variables, and output variables obtained after executing the system for case2

Input variables	Output variables			
Awing=158.9697	Aht=27.8359	Mgear=3680.6213	cxc_crz=0.0009894	tofl=1708.8725
Leh=19.801	Aref=158.9697	Mht=711.3937	cxc_cth=0.00098892	vapp=61.2977
MTOW=91596.0416	Avt=26.8834	Mop=7550.3395	cxi_clb=0.013713	vsnd_clb=296.5339
MWE=42795.6932	BPR=6	Msys=5431.0733	cxi_crz=0.011473	vsnd_crz=296.5339
Mach_clb=0.78	FNslst=147061.77	Mvt=693.7676	cxi_cth=0.013713	vsnd_cth=296.5339
Mach_cth=0.78	Fn_clb=42913.3287	Mwing=10244.1147	cz_clb=0.52834	vz_clb=3.006
Mprop=7958.6491	Fn_cth=42913.3287	Naisle=0.99998	cz_crz=0.48327	wAfus=445.5678
Npax=150	Fn_to=120065.3438	NpaxFront=6	cz_cth=0.52834	wAht=43.6909
RA=7203619.4473	Fuel=24000.0089	OWE=50346.0327	czmax_LD=2.7488	wAnac=32.1096
alt_app=0	Fwing=19414.3782	PL=17250	czmax_TO=2.4395	wAvt=54.1282
alt_clb=10668	Kcx0=1.4	Pamb_clb=23842.2717	dnac=2.2353	wAwing=248.5631
alt_crz=10668	Kcxp=1	Pamb_crz=23842.2717	g_app=9.8337	
alt_cth=10668	KczmaxLD=1	Pamb_cth=23842.2717	g_clb=9.8008	
alt_to=0	KczmaxTO=1	Pamb_to=101325	g_crz=9.8008	
dfus=4.1	Kff=0.80893	RA_time=31144.0861	g_cth=9.8008	
disa_clb=0	Kind=1	Tamb_clb=218.808	g_to=9.8337	
disa_crz=0	Kmcl=0.7	Tamb_crz=218.808	kfn_cth=0.87825	
disa_cth=0	Kmcr=0.65	Tamb_cth=218.808	kvs_LD=1.23	
disa_to=0	Kmto=1	Tamb_to=288.15	kvs_TO=1.13	
ne=2	Kmwe=1	Vht=0.7	lfus=40.25	
sfc=1.7111e-005	Ksfc=1	Vvt=0.05	lod_clb=17.4064	
span=32.0948	LDW=67596.0327	Wpax=115	lod_crz=17.1898	
tuc=0.10813	Lev=19.801	ar=6.4797	lod_cth=17.4064	
	Lref=4.9531	cx0_clb=0.015652	mass_clb=87016.2395	
	Mach_crz=0.78001	cx0_crz=0.015652	mass_crz=79596.0371	
	Mach_stall_to=0.18096	cx0_cth=0.015652	mass_cth=87016.2395	
	Mach_to=0.20448	cx_clb=0.030353	phi=0.38172	
	Mchar=0.79638	cx_crz=0.028114	rho_clb=0.3796	
	Mfurn=3787.5	cx_cth=0.030353	rho_cth=0.3796	
	Mfus=10288.5733	cxc_clb=0.00098892	rho_to=1.225	

5.4.2.3 Case3

In this case the inputs variables given for system are Fwing, LDW, Lev, Mach_crz, Mach_cth, Mfurn, alt_app, alt_clb, alt_crz, alt_cth, alt_to, cx_crz, czmax_TO, disa_clb, disa_crz, disa_cth, disa_to, dnac, lod_clb, mass_crz, ne, span and tuc.

After decomposition here two SCCs were identified in this case. The first SCC had seven models and the second SCC had six models. The models belonging to each SCC are given below.

SCC 1

nacelle_diameter,spec_fuel_cons,range_crz,operator_item_mass,ope_weight_empty, manu_weight_empty, engine_mass

SCC 2

lift_to_drag_clb,induced_drag_clb,level_flight_clb,friction_drag_clb,pressure_drag_clb ,drag_factor_clb

For the first SCC, two variable flow models were generated and for the second one there were six variable flow models. For the second case three variable flow models produced a converged solution and hence only these will be discussed here. The incidence matrix

and the corresponding rearranged DSM of these flow models are given in the Appendix-V.iii (Figure A- 25 to Figure A- 34).

Solving

Table 5-10 provides the details of all the SCCs and their variables flow models, along with the execution details.

Table 5-10 Details of computational process modeling and solving of SCCs for case 2

Computational process modelling					Solving	
SCC	Variable flow model	nFdb	nMm	Optimal flow model	Number of calls to the models in SCC	% additional computational cost
SCC 1	1	1	4	■	110	base
	2	1	5		110	equal
SCC2	1	1	3		74	base
	2	1	4		320	332% more
	3	1	2	■	86	16.2% more

Discussion

In this case, for the first SCC, the computational process modeller has selected the first variable flow model as the solution. For the second SCC the third variable flow model was the selected solution. From the sixth column in Table 5-10 it is clear the flow model selected by the computational process modeller was not the optimum in terms of computational cost for the second SCC, because variable flow model 1 made less number of calls to system compared to flow model 3 which was chosen as the optimum. But it is clear that these selections were reasonable choice since the chosen flow models lead to faster convergence of the SCCs compared to variable flow model 2 which produces a converged solution after 320 calls to the models. The reasons for these sub-optimal choices were discussed earlier.

The final computational plan obtained for USMAC is given in the next paragraph.. The incidence matrix for this system, which displays the variable flow model generated by

IMM, is given in the Figure A- 23 in the Appendix-V.iii. In the figure models belonging to SCC are confined into a single process.

Computational Plan

Inputs for the system

Fwing, LDW, Lev, Mach_crz, Mach_cth, Mfurn, alt_app, alt_clb, alt_crz, alt_cth, alt_to, cx_crz, czmax_TO, disa_clb, disa_crz, disa_cth, disa_to, dnac, lod_clb, mass_crz, ne, span, tuc

Outputs of the system

Aht, Aref, Avt, Awing, BPR, FNslst, Fn_clb, Fn_cth, Fn_to, Fuel, Kcx0, Kcxp, KczmaxLD, KczmaxTO, Kff, Kind, Kmcl, Kmcr, Kmto, Kmwe, Ksfc, Leh, Lref, MTOW, MWE, Mach_clb, Mach_stall_to, Mach_to, Mchar, Mfus, Mgear, Mht, Mop, Mprop, Msys, Mvt, Mwing, Naisle, Npax, NpaxFront, OWE, PL, Pamb_clb, Pamb_crz, Pamb_cth, Pamb_to, RA, RA_time, Tamb_clb, Tamb_crz, Tamb_cth, Tamb_to, Vht, Vvt, Wpax, ar, cx0_clb, cx0_crz, cx0_cth, cx_clb, cx_cth, cxc_clb, cxc_crz, cxc_cth, cxi_clb, cxi_crz, cxi_cth, cz_clb, cz_crz, cz_cth, czmax_LD, dfus, g_app, g_clb, g_crz, g_cth, g_to, kfn_cth, kvs_LD, kvs_TO, lfus, lod_crz, lod_cth, mass_clb, mass_cth, phi, rho_clb, rho_cth, rho_to, sfc, tofl, vapp, vsnd_clb, vsnd_crz, vsnd_cth, vz_clb, wAfus, wAht, wAnac, wAvt, wAwing

Execution sequence for the models

1)**modified_wing_fuel**_tr,2)Cz_max_TO_factor_,3)reference_area_,4)non_stand_atmos_crz,5)gravity_acc_crz,6)aspect_ratio_,7)**modified_Cz_max_TO**__tr,8)tail_volume_factor_,9)reference_length_,10)ref_mach_number_,11)press_drag_factor_,12)level_flight_crz,13)ind_drag_factor_,14)fin_volume_factor_,15)**modified_fin_lever_arm**__tr,16)tail_size_,17)pressure_drag_crz,18)induced_drag_crz,19)fin_size_,20)wing_wetted_area_,21)tail_wetted_area_,22)**modified_tail_lever_arm**__tr,23)nac_wetted_area_,24)**modified_mean_cruise_mass_crz**_tr,25)fric_drag_factor_,26)fin_wetted_area_,27)**modified_drag_factor_crz**_tr,28)top_of_climb_mass_cth,29)one_pax_weight_,30)non_stand_atmos_cth,31)gravity_acc_cth,32)**modified_furnishing_mass**__tr,33)**modified_friction_drag_crz**_tr,34)non_stand_atmos_to,35)level_flight_cth,36)gravity_acc_to,37)**modified_**

fus_wetted_area__tr,38)Payload_,39)wing_mass_,40)top_of_climb_mass_clb,41)tail_mass_,42)system_mass_,43)sound_velocity_crz,44)sfc_factor,45)pressure_drag_cth,46)non_stand_atmos_clb,47)lift_to_drag_crz,48)**modified_landing_weight**__tr,49)landing_gear_mass_,50)induced_drag_cth,51)gravity_acc_clb,52)fuselage_mass_,53)friction_drag_cth,54)fin_mass_,55)Mach_stall_to_,56)MWE_factor_,57)Kvs_Take_Off,58)secured_Mach_to,59)**scc_1_tr**,60)drag_factor_cth,61)**scc_2_tr**,62)air_density_to,63)air_density_cth,64)air_density_clb,65)Cz_max_LD_factor_,66)**modified_take_off_weight**__tr,67)sound_velocity_clb,68)net_thrust_to,69)net_thrust_cth,70)net_thrust_clb,71)max_take_off_factor,72)max_cruise_factor,73)max_climb_factor,74)lift_to_drag_cth,75)gravity_acc_app,76)**modified_fuselage_length**__tr,77)Kvs_Landing,78)Cz_max_LD_,79)tofl_1,80)time_crz,81)sound_velocity_cth,82)**modified_fuselage_diameter**__tr,83)fus_fuel_ratio_,84)cruise_thrust_1,85)climb_rate_1,86)app_speed_1

In this case there are two SCCs. These are the process number 59 and 61 in the list of the execution sequence of the models. The details of these SCCs are given below.

Strongly Connected Components

59. scc_1_tr

Inputs to SCC

Kmwe, Ksfc, LDW, MTOW, Mach_crz, Mfurn, Mfus, Mgear, Mht, Msys, Mvt, Mwing, Npax, OWE, dnac, g_crz, lod_crz, ne, vsnd_crz

Outputs of SCC

BPR, FNslst, MWE, Mop, Mprop, RA, sfc

Execution Sequence

modified_nacelle_diameter,spec_fuel_cons,range_crz**,**operator_item_mass,**modified_ope_weight_empty,modified_manu_weight_empty**,**modified_engine_mass_**

Treatment (Solver)

FPI-first method

61. scc_2_tr

Inputs to SCC

Aref, Kcx0, Kcxp, Kind, Lref, Mchar, Pamb_clb, Tamb_clb, ar, g_clb, lfus, lod_clb, mass_clb, ne, wAfus, wAht, wAnac, wAvt, wAwing

Outputs of SCC

Mach_clb, cx0_clb, cx_clb, cxc_clb, cxi_clb, cz_clb,

Execution Sequence

modified_lift_to_drag_clb,induced_drag_clb,**modified_level_flight_cl**

b,friction_drag_clb,pressure_drag_clb,drag_factor_clb

Treatment (Solver)

FPI-first method

The variable flow model for the first SCC is the variable flow model number 1 (which was chosen as the optimum) shown in the Figure A- 26. The variable flow model for the second SCC is the variable flow model number 3 (which was chosen as the optimum) shown in the Figure A- 34.

Table 5-11 Values of the input and output variables obtained after executing the system for case3

Input Variables	Output Variables			
Fwing=19414.3782	Aht=27.8359	Mgear=3680.621	cx_cth=0.030352	tofl=1708.9373
LDW=67596.0327	Aref=158.9697	Mht=711.3937	cxc_clb=0.00098892	vapp=61.2981
Lev=19.801	Avt=26.8834	Mop=7550.3515	cxc_crz=0.00098822	vsnd_clb=296.5339
Mach_crz=0.78	Awing=158.9697	Mprop=7958.462	cxc_cth=0.00098822	vsnd_crz=296.5339
Mach_cth=0.78	BPR=6	Msys=5431.073	cxi_clb=0.013711	vsnd_cth=296.5339
Mfurn=3787.5	FNslst=147058.5577	Mvt=693.7676	cxi_crz=0.011474	vz_clb=3.0056
alt_app=0	Fn_clb=42912.095	Mwing=10244.2965	cxi_cth=0.013713	wAfus=445.5675
alt_clb=10668	Fn_cth=42912.3914	Naisle=0.99998	cz_clb=0.52832	wAht=43.6909
alt_crz=10668	Fn_to=120062.5369	Npax=150	cz_crz=0.48329	wAnac=32.1095
alt_cth=10668	Fuel=24000	NpaxFront=6	cz_cth=0.52834	wAvt=54.1282
alt_to=0	Kcx0=1.4	OWE=50346.0327	czmax_LD=2.7487	wAwing=248.5631
cx_crz=0.028114	Kcxp=1	PL=17250	dfus=4.1	
czmax_TO=2.4395	KczmaxLD=1	Pamb_clb=23842.2717	g_app=9.8337	
disa_clb=0	KczmaxTO=1	Pamb_crz=23842.2717	g_clb=9.8008	
disa_crz=0	Kff=0.80893	Pamb_cth=23842.2717	g_crz=9.8008	
disa_cth=0	Kind=1	Pamb_to=101325	g_cth=9.8008	
disa_to=0	Kmcl=0.7	RA=7203668.6537	g_to=9.8337	
dnac=2.2353	Kmcr=0.65	RA_time=31144.7418	kfn_cth=0.87825	
lod_clb=17.4064	Kmto=1	Tamb_clb=218.808	kvs_LD=1.23	
mass_crz=79596.0327	Kmwe=1	Tamb_crz=218.808	kvs_TO=1.13	
ne=2	Ksfc=1	Tamb_cth=218.808	lfus=40.25	
span=32.0948	Leh=19.801	Tamb_to=288.15	lod_crz=17.1902	
tuc=0.10813	Lref=4.9531	Vht=0.7	lod_cth=17.4068	
	MTOW=91596.0327	Vvt=0.05	mass_clb=87016.2311	
	MWE=42795.6812	Wpax=115	mass_cth=87016.2311	
	Mach_clb=0.78002	ar=6.4797	phi=0.38184	
	Mach_stall_to=0.18096	cx0_clb=0.015651	rho_clb=0.3796	
	Mach_to=0.20448	cx0_crz=0.015652	rho_cth=0.3796	
	Mchar=0.7964	cx0_cth=0.015652	rho_to=1.225	
	Mfus=10288.5673	cx_clb=0.030352	sfc=1.7111e-005	

The values of the input and output variables obtained after executing the system are given in the Table 5-11.

5.5 Conclusion

Presented in this chapter are the tests conducted in order to evaluate the methods and approaches developed as part of this research.

Initially the tests were performed in order to finalise the objective function for scheduling the coupled models. Feedback length was proven inefficient to provide an

estimate of the computational cost for solving the SCC in a system. In contrast, feedback number provided a good quality estimate, if the solver for SCC is either FPI-first or GN. This, however, proves that the objective function should be formulated based on the solver and cannot be generalised for solving SCCs using any solver. In the current context of testing the computational process modeller, FPI-first was used as the solver for SCC and hence feedback number, which was proven to fit well with this solver, was chosen as the objective function for scheduling the coupled models.

The different tests performed on the simple sizing and the USMAC test case for evaluating the computational process modeller, have confirmed the effectiveness of this modeller for generating optimal execution plans for complex aircraft conceptual design systems. Even though there were a few sub-optimal choices, when tested on USMAC, with regard to computational cost, all choices for computational flow made by the system were among the best. The sub-optimal were inevitable since the computational cost for a system was found to be not just depended on the feedback number and the number of modified models (which were the focus of this research), but also on the starting guess for the unknown variables, mutual sensitivity of the switched variables of the modified models (see section 4.2) and possibly other factors which are yet to be discovered. The various tests also proved that in the majority of the cases an increase in the number of modified models in the system increased the computational cost. This finding confirms the choice of incorporating the number of modified models also as a criterion for choosing the optimal variable flow model.

The three tests with USMAC were performed on the same system of models, but with different set of input variables. The computational process modeller was able to generate sound execution plans for all of the cases, in order to compute the unknown variables. This demonstrates the ability of the computational process modeller to provide flexibility for the designer in choosing the independent (input) variables during the design process.

6 FRAMEWORK DESIGN AND DEVELOPMENT: CRANFIELD WORKFLOW MANAGEMENT DEVICE (CWMD)

6.1 Introduction

This chapter describes the implementation of an object-oriented framework for dynamically setting up the computational plans that are generated by the computational process modeller. The framework has been developed for test and evaluation of the proposed methods and approaches.

6.2 Overview of CWMD

The CWMD is an object-oriented framework for conducting design studies on mathematical models, which represent the physics and other characteristics of an aircraft. The software tool is implemented in Matlab programming language. Matlab has a variety of inbuilt functions which make programming easier compared to other languages. Even though Matlab is not a fully object-oriented programming language compared to C++ or C#, it has the capabilities to incorporate object-oriented programming.

CWMD accepts the models and their associated variables and wraps these in the form of specific objects. These objects can be further modified and grouped, according to the plan generated by computational process modeller, in order to form executable systems. These executable systems can be saved as objects in CWMD. Further, various mathematical treatments, can be applied to these objects for conducting design studies on the system. The object model of the framework is presented in the next section.

6.3 Object Oriented Modelling of CWMD

6.3.1 Basic Concepts of Object Oriented Modelling

Fundamental terms used in object oriented programming, which are used in this thesis, are given below:

Class

A class defines an entity, including the entity's characteristics (e.g., attributes, fields or properties) and the things it can do (e.g., behaviours, or methods or features) (Booch, 1993). Classes provide modularity and structure in an object-oriented computer program. Also, the code for a class should be relatively self-contained. Collectively, the properties and methods defined by a class are called members.

Object

An object is a particular instance of a class. The set of values of the attributes of a particular object is called its state. The object consists of state and behaviour.

Method

A method represents an object's abilities. The method is implemented as a function associated with the object.

6.3.2 CWMD Class Diagram

The class diagram developed for CWMD is shown in Figure 6-1. Here five classes are identified; data, model, subprocess, treatment and study. A brief description of each class is given below. A more detailed description of these is followed in the next section.

1. *Data (DO)*
   Data object contains all information required to describe a data (variable) element.
2. *Model (MO)*
   A model object is an elementary black box (a simple software program or model) with inputs, outputs and a program. Inputs and outputs of the model objects are data objects.
3. *Subprocess (SP)*
   A subprocess is an object that defines mathematical treatment on one or more model objects (and/or subprocess objects). A subprocess can also be a container of a group of discipline specific models (and/or subprocess objects), without a mathematical treatment.

4. *Treatment (TR)*

   A treatment object contains all the information of a particular mathematical treatment, which can applied on a model or a subprocess object. For example, an optimisation treatment applied on a computational process or a solver applied to solve a SCC or a modified model.

5. *Study (ST)*

A study object is an assembly of subprocesses (or models) with one or more treatments in order to conduct a design study over the entire system.

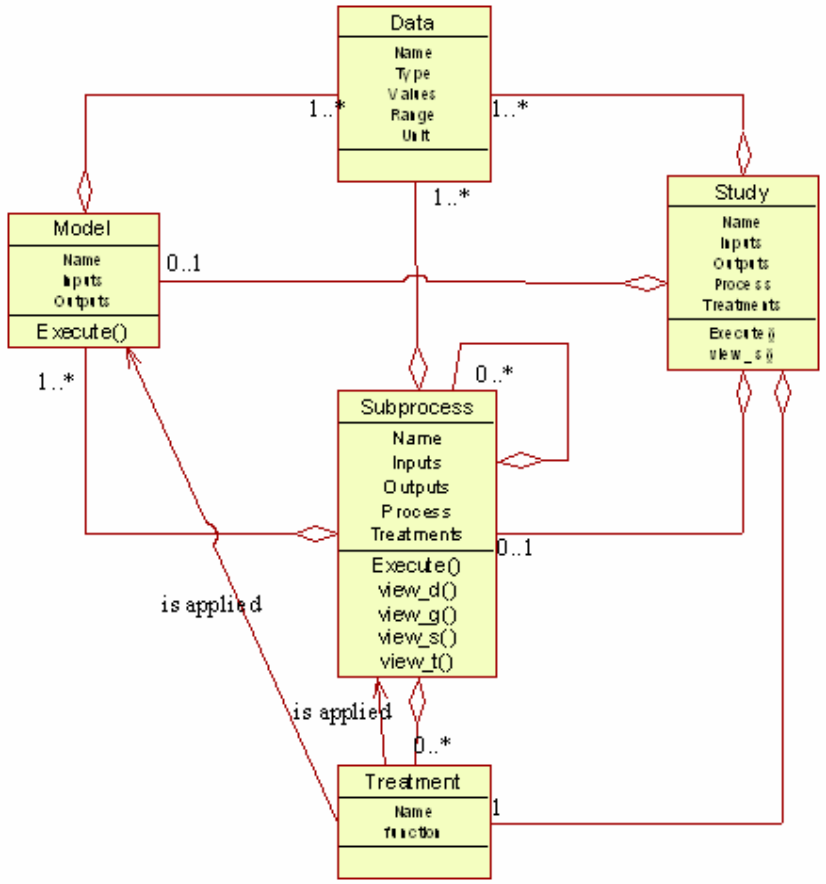


Figure 6-1 Class diagram for CWMD

The links in the class diagram shown in the Figure 6-1 are described in the Table 6-1.

Table 6-1 Links and description of the class diagram for CWMD

Links	Description
MO-DO	MO contains one or more DOs
SP-DO	SP contains one or more DOs
SP- MO	SP can contain one or more MOs
SP- TR	SP can contain zero or more TRs
SP-SP	SP can contain zero or more SPs
ST -DO	ST can contain one or more DOs
ST- MO	ST can contain zero or one MOs
ST-SP	ST can contain zero or one SPs
ST- TR	ST can contain one TR

6.3.3 CWMD Object Models

This section describes in more detail the various objects in CWMD, their associated methods and attributes.

6.3.3.1 Data object

Input and output variables associated with models are modelled as data objects in CWMD. A data object contains all the relevant information regarding the corresponding variable. The following are the attributes currently available for a data object.

Attribute	Explanation
Name	Name of the variable
Type	Type of the variable (scalar, array or structure)
Values	Value associated with variable
Range	Range of the variable (minimum and maximum value)
Unit	Unit of the variable (e.g., Kg, Kg/m^2)

An example for the attributes of a data object, MTOW (Maximum takeoff weight) is given below.

Attribute	Example
Name	MTOW
Type	scalar
Values	91596
Range	Min:91500, Max:99000
Unit	Kg

There are no methods associated with a data object.

6.3.3.2 Model object

A model object is an elementary black box (a simple software program or model) with inputs, outputs and a program. Inputs and outputs of the model object are data objects. The following are the attributes for a model object.

Attribute	Explanation
Name	Name of the model object (this also refers to the name of the software program which will be executed on executing the model)
Inputs	Input data objects for the model
Outputs	Output data objects for the model

An example for a model object is given below.

Name: Engine_mass

Inputs: ne, FNslst

Outputs: Mprop

Here ne, FNslst and Mprop are data objects.

The methods associated with the model objects are given below.

Methods	Explanation
Execute	Executes the software program associated with the model object with inputs from the values of the input data objects, and thereafter updates the output data objects with the outputs of obtained after executing the program.

6.3.3.3 Subprocess object

A subprocess object can be of two types; the first type is for modelling a modified model or an SCC. This is done by integrating the constituent model (or models in the

case of SCC) with the corresponding mathematical treatment object used to solve the modified model (or SCC). The second type is for assembling a set of models, or a set of models and subprocesses, into a higher level subprocess. In this case subprocess contains only models with out any treatment. This wrapping allows the multilevel and hierarchical arrangement of a system which minimises the difficulty involved in managing thousands of models in a complex system.

The computational plan generated after applying the computational process modeller for systems is saved as subprocess in CWMD.

The attributes of a subprocess are given below

Attribute	**Explanation**
Name	Name of the subprocess
Inputs	Input data objects of the subprocess
Outputs	Output data objects of the subprocess
Process	Model and subprocess objects included in this subprocess
Treatments	Treatment objects included in this subprocess

The methods associated with the subprocess are given below.

Methods	**Explanation**
Execute	If treatment attribute is empty Executes the objects in the process atttribute (in the sequence in which it is added in the process attribute), with inputs from the values of the input data objects. On execution, the outputs obtained are updated in output data objects of the subprocess. If treatment attribute is not empty The treatment function is executed with the models (and/or subprocess) in the process attribute of the subprocess object given as inputs.
view_d	Plots the subprocess as a design structure matrix
view_g	Plots the subprocess in a graphical format
view_i	Plots the subprocess as an incidence matrix
view_t	Plots the subprocess in a tabular format

More details on the methods associated with the plotting of the subprocesses (view_d, view_g, view_i, view_t) are given in section 6.5.

Examples for the two types of subprocess objects are given below.

Subprocess for assembling models and/or subprocess

This example demonstrates the assembling of four models into a subprocess. The four model objects are given in the table below

Name of the models	**Inputs**	**Outputs**
sfc_factor	-	Ksfc
spec_fuel_con	Ksfc,BPR	sfc
nacelle_dia	BPR,FNslst	dnac
nac_wet_area	dnac	wAnac

The attributes for the subprocess in which these four models are grouped, are given below.

Name: Engines

Inputs: BPR, FNslst

Outputs: sfc, wAnac, dnac, Ksfc

Process: sfc_factor, spec_fuel_con, nacelle_dia, nac_wet_area

Treatments: Nil

This subprocess 'Engines' now performs like a model with inputs BPR and FNslst and generating sfc, wAnac, dnac and Ksfc as outputs, on execution.

This 'engines' subprocess, can now be assembled with other models or subprocesses in order to create a higher level subprocess.

Subprocess for modified models or SCCs

An example for a modified model is given in Figure 4-1

The attributes of the subprocess for modelling this modified model are:

Name: modified_Payload

Inputs: PL, Wpax

Output: Npax

Process: Payload

Treatments: modifier (Gauss-Newton method)

Thus the 'modified_Payload' subprocess performs like a model with inputs PL and Wpax and generating Npax as output, on execution.

In the case of SCCs, the treatment applied will be different (e.g., sccsolver) and the process attribute contains the models and/or subprocesses which are strongly connected.

The advantage of this subprocess representation is that the modified models and SCC can be treated as simple models with inputs and outputs, concealing the complications involved in solving and simplifying the difficulty involved in modelling these, each time a new computational plan is generated.

6.3.3.4 Study object

A study object is similar to a subprocess object, however, the distinction is made since the study object represents a top level design study process, while the treatment applied to the subprocess (or model) is a design study function (optimisation or design of experiment etc).

The attributes of a study object are given below:

Attribute	**Explanation**
Name	Name of the study
Inputs	Input data objects for the study
Outputs	Output data objects of the study
Process	Model and subprocess objects included in this study
Treatments	Treatment objects included in this study

The methods associated with the study are given below.

Methods	**Explanation**
Execute	The treatment function(in the treatment attribute) is executed with the models (and or subprocess) in the process attribute of the study object provided as inputs.
view_s	Plots the study object as a block diagram

The example given below represents a study object. This object is for conducting optimisation study on the 'engines' subprocess.

Name: engines_optimise

Inputs: BPR, FNslst

Output: sfc, wAnac, dnac, Ksfc

Process: Engines

Treatments: Optimiser

6.3.3.5 Treatment object

A treatment is a mathematical operation applied on a model or a subprocess object. The attributes for a treatment object is given below.

Attribute	**Explanation**
Name	Name of the treatment
function	The name of the software program associated with the treatment

There are no methods associated with the treatment.

6.4 Example Case

Figure 6-2 shows a system of models which represents a simplified set of aircraft sizing equations (Buckley et al., 1992). This example demonstrates the object modelling for this system when it is setup in CWMD.

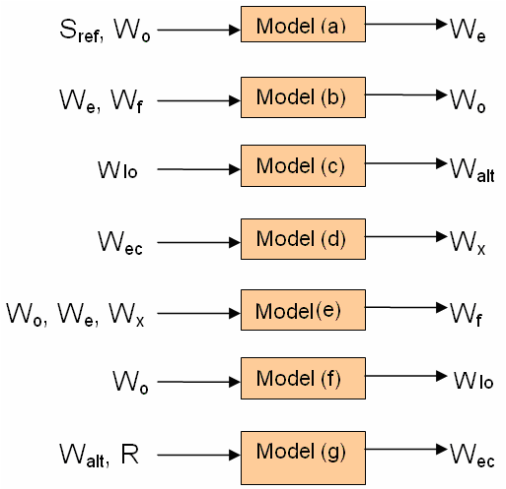


Figure 6-2 System of models

All the models and the variables in the system are modelled in CWMD as model and data objects. The data objects for the variables are given in the table below. Since the case study is only for demonstration purposes, the values and range attributes are not given.

DATA OBJECTS		
Name	Type	Unit
W_e	'Scalar'	Kg
W_{lo}	'Scalar'	Kg
Sref	'Scalar'	m^2
W_f	'Scalar'	Kg
W_o	'Scalar'	Kg
W_{alt}	'Scalar'	Kg
R	'Scalar'	m
W_{ec}	'Scalar'	Kg
W_x	'Scalar'	Kg

The attributes of the models objects for the models are given in the table below

Name of the model	**Inputs**	**Outputs**
a	Sref, W_o	W_e
b	W_e, W_f	W_o
c	W_{lo}	W_{alt}
d	W_{ec}	W_x
e	W_o,W_e,W_x	W_f
f	W_o	W_{lo}

The attributes for the treatment objects are given in the table below

Name	**function**
modifier	gauss-newton
sccsolver	Fixed-point-first
optimiser	gradient-based

For the system shown in Figure 6-2, considering a case where R, W_{alt} are independent variables the corresponding variable flow model generated by the computational process modeller is given below.

	Sref	Wo	We	Wf	Wlo	Walt	Wec	Wx	R
a	o	i	i						
b		i	o	i					
c					o	i			
d							i	o	
e		i	i	o				i	
f		o			i				
g						i	o		i

Figure 6-3 Variable flow models for the system of models shown in Figure 6-2

In this system models b and e are identified as strongly connected and the models c, f, a and b are modified models. The final computational plan obtained for this system after applying the computational process modeller is, g→modified_c→modified_f→d→SCC(e→modified_b)→a. The hierarchical arrangement of the subprocess formed for this system by CWMD is represented symbolically in Figure 6-4. The data objects are not shown in the figure for clarity purpose.

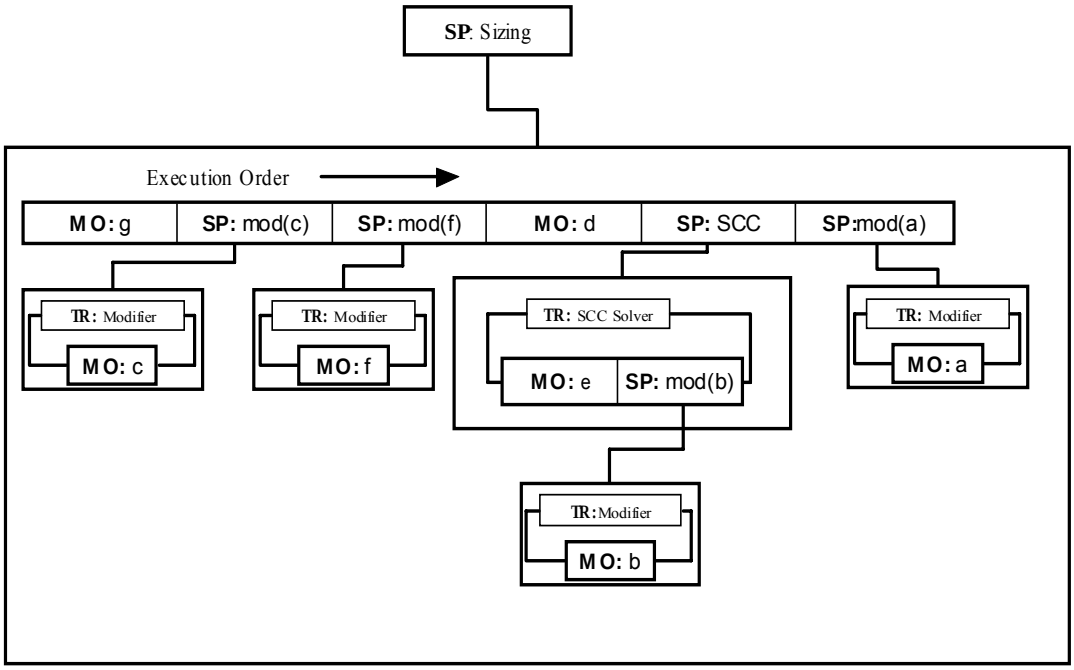


Figure 6-4 Symbolic representation of the subprocess for the system in Figure 6-2 with R and W_{alt} given as input variables.

The attributes of each subprocess in the Figure 6-4 are given in the table below.

Name	Inputs	Outputs	Process	Treatments
Sizing	R W_{alt}	Sref W_e W_{ec} W_f W_{lo} W_o W_x	g mod(c) mod(f) d SCC mod(a)	-
mod(c)	W_{alt}	W_{lo}	c	modifier
mod(f)	W_{lo}	W_o	f	modifier
SCC	W_o W_x	W_e W_f	mod(b) e	sccsolver
mod(b)	W_f	W_e	b	modifier

	W_o			
mod(a)	W_e W_o	Sref	a	modifier

The top-level subprocess 'Sizing' now performs like a model with R and Walt as inputs and the generating the remaining variables as outputs, on execution.

If a design study, for example an optimization study, has to be conducted on this subprocess, then the corresponding attributes for the study object which models this design study, are as given below.

Name: sizing_optimise

Inputs: R, Walt

Output: Sref,W_e,W_{ec},W_f,W_{lo},W_o,W_x

Process: sizing

Treatments: Optimiser

6.5 Modules of CWMD

This section explains the different modules of CWMD which include the creator, executer and the viewer. The creator section explains how the different objects are created. The executor section describes the execution of the model, subprocess and study objects. The viewer section explains the various viewers for subprocess objects available in CWMD. The architecture of the CWMD is given in the Appendix-VI.

Shown in Figure 6-5 is the main interface for CWMD.

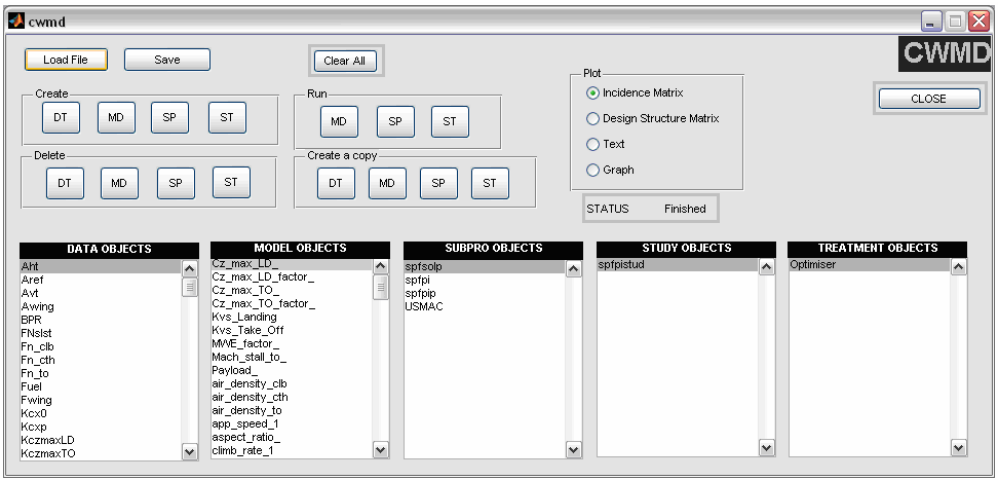


Figure 6-5 CWMD main window

In this main interface the lists boxes (DATA OBJECTS, MODEL OBJECTS, SUBPRO OBJECTS, STUDY OBJECTS, and TREATMENT OBJECTS) display the corresponding objects created. The push buttons in the panel 'Create', 'Run' and 'Delete' are used for creating, executing and deleting the objects. The radio button in the panel 'Plot' plots the selected objects in the chosen format. The 'Save' button saves all the objects in the current window in file name specified by the design, in '.mat' format. The 'Load File' button loads the objects saved in files, into the CWMD window.

Currently there is no provision in CWMD for creating treatment objects and therefore the treatment objects currently available are inbuilt in CWMD. The available treatments are 'Gauss-Newton', 'Fixed point iteration- first', 'Fixed point iteration- second' and 'Optimiser'. The first three treatments are for solving SCCs and modified models, and the last one is for conducting optimisation studies. Gauss-Newton method is also used as treatment for modified models. The 'optimiser' treatment is a gradient based optimiser.

6.5.1 Creator

The buttons in the 'Create' panel of the main CWMD window are used for creating the different objects. The following sections describe how each object is created.

Data object creator

Data object is created using ‘Data creator’ GUI (Graphical User Interface). It has the provision for entering the attributes for a data object. Data Creator GUI can be activated on clicking ‘DT’ button in the ‘Create’ panel of the CWMD window. The GUI is displayed in the Figure 6-6.

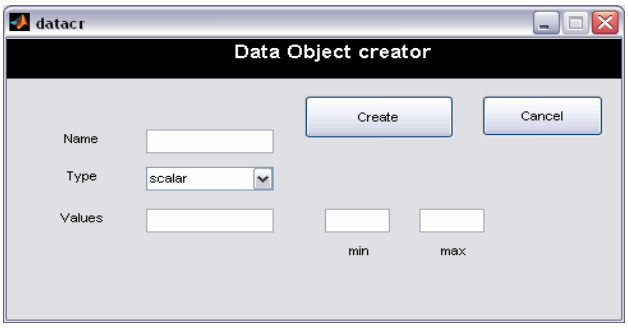


Figure 6-6 Data creator GUI

Clicking on the ‘Create’ button will create the data object, with the entered attributes, and places the object in the data objects list box in the Main (CWMD) window.

Model object creator

Model object is created using ‘Model creator’ GUI. It has the provision for entering the attributes for a Model object. Model Creator GUI can be activated on clicking ‘MD’ button in the ‘Create’ panel of the CWMD window. The GUI is displayed in the Figure 6-7.

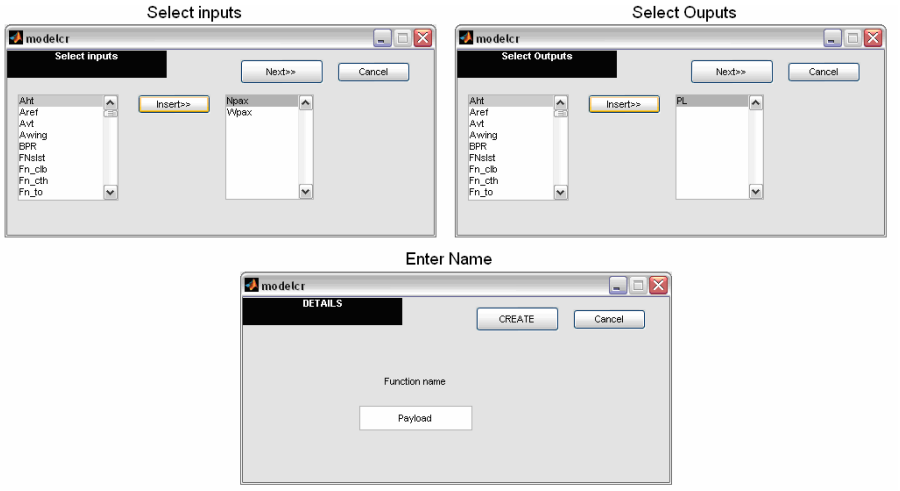


Figure 6-7 Model creator GUI

In the three windows displayed in Figure 6-7, the first window is for selecting the inputs for the models, from the list of available data objects. The second window is for selecting the outputs for the model object and the third window is for entering the name of the model object. Clicking on the 'Create' button will create the model object, with the entered attributes, and places the object in the 'model objects' list box in the Main (CWMD) window.

Subprocess Object Creator

Subprocess object is created using 'Subprocess creator' GUI. This GUI has the provision for entering the attributes for the subprocess object to be created. In addition, the main function of this GUI includes defining the inputs (models and the independent variables) required for generating the computational plan for the selected models by applying the computational process modeller. The GUI is directly interfaced with the code for computational process modeller. Once the computational plan is generated based on the given inputs, the corresponding subprocess object is created which on execution will follow the generated computational plan.

'Subprocess Creator' GUI can be activated on clicking 'SP' button in the 'Create' panel of the CWMD window. The GUI is displayed in Figure 6-8.

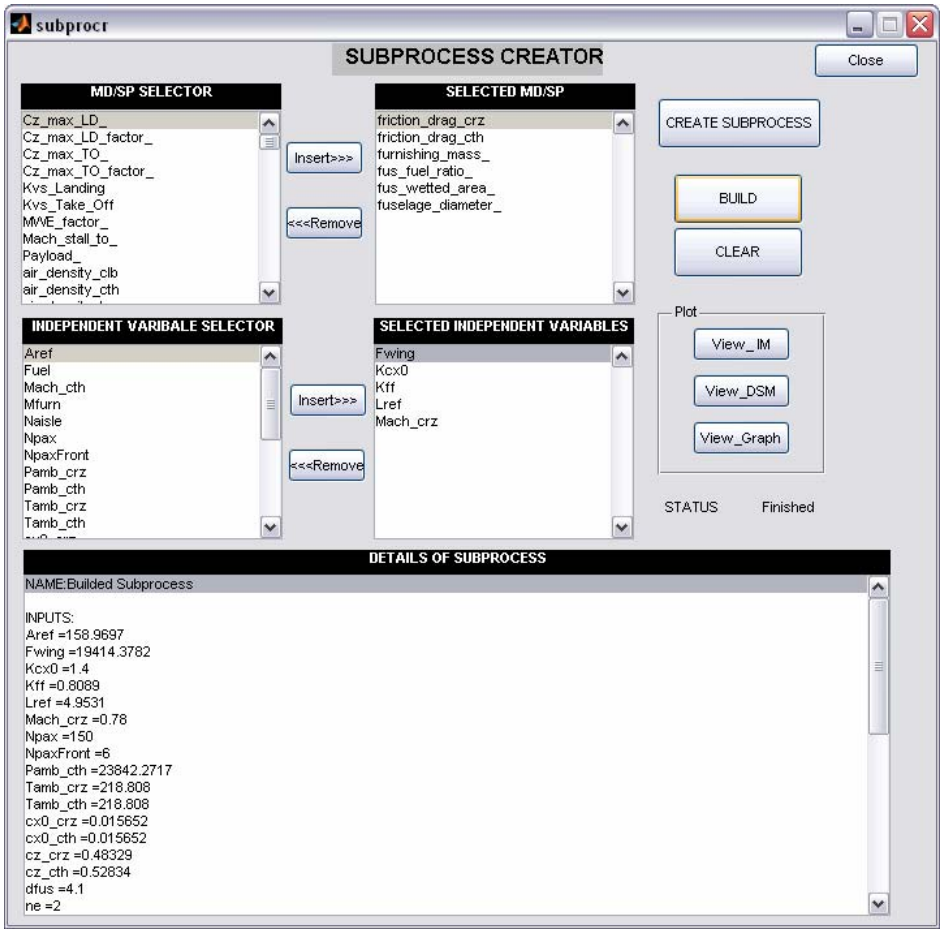


Figure 6-8 Subprocess creator GUI

In this interface the 'Build' push button will activate the computational process modeller with models selected in the 'SELECTED MD/SP' list box and the independent variables in the 'SELECTED INDEPENDENT VARIABLES' list box, as the inputs for the process plan. The computational process modeller generates the computational plan for the selected models and based on this plan a temporary subprocess is created by CWMD. The details of this subprocess will be displayed in the 'DETAILS OF SUBPROCESS' list box. Once the designer is satisfied with the generated subprocess, clicking on the push button 'CREATE SUBPROCESS' will create the subprocess object and places this object in the 'subprocess objects' list box in the Main (CWMD) window. The presence of a SCC in the system will activate the window shown in Figure 6-9 during the creation process. From this window the designer will be able to choose the type of solver for solving the SCC which will be inserted in the treatment attribute

of the SCC subprocess. For modified models, Gauss-Newton method is automatically applied as the treatment for solving these.

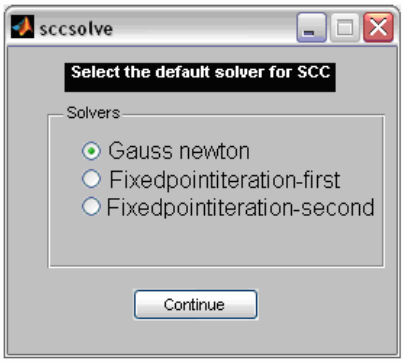


Figure 6-9 GUI for selecting the solver for SCC

The subprocess can be plotted in various formats with the push buttons in the 'Plot' panel.

Study Object Creator

Study object is created using 'Study creator' GUI. It has the provision for entering the attributes for the Study object. 'Study creator' GUI can be activated on clicking 'ST' button in the 'Create' panel of the CWMD window. The GUI is displayed in Figure 6-10.

In this GUI the attributes for the study object which are the process and the treatment, can be selected from the 'MD/SP SELECTOR' and 'TREATMENT SELECTOR' list box. Currently these selections are limited to a single model or subprocess and a single treatment. The inputs and outputs attributes for the study object are the same as that of the selected model or subprocess and therefore entered automatically while the study object is created. The details of the created study object are displayed in the 'DETAILS OF THE STUDY' window.

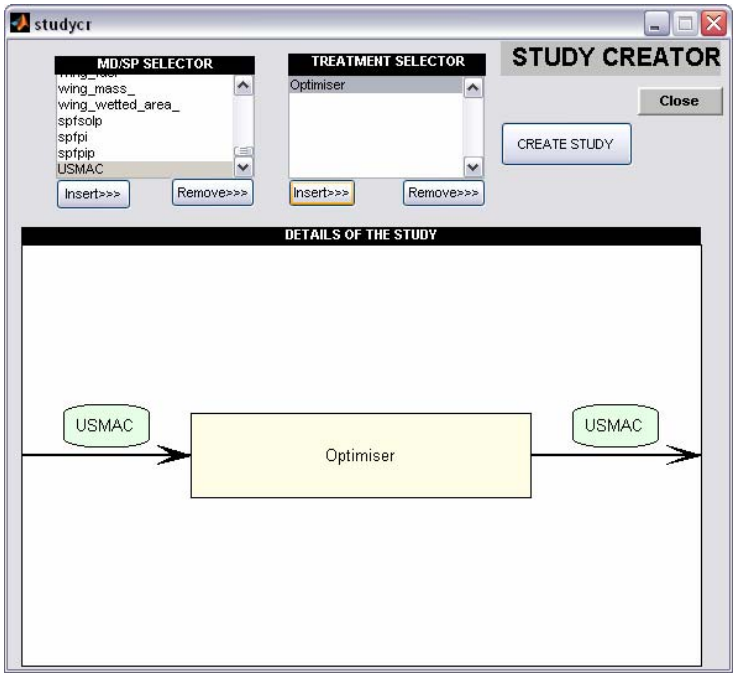


Figure 6-10 Study creator GUI

6.5.2 Executer

The GUI for executing the models or subprocess object can be activated by clicking on the MD (in case of model object) or SP (in case of subprocess object) push button in the 'Run' panel of the CWMD window. The interface for executing the models and the subprocess are the same.

On clicking the MD or SP button in the 'Run' panel in the CWMD window, the GUI shown in Figure 6-11 will appear with the corresponding selected model or subprocess which has to be executed, loaded in it. The values of the input variables for the loaded model or subprocess can be modified by selecting and editing the corresponding variable in the 'INPUTS' list box. The value of the selected variable is displayed in the box on the right hand side of the 'INPUTS' list box which can be edited and further set by clicking the 'SET VALUE' button.

The 'EXECUTE' button will activate the 'execute' method associated with the loaded model or subprocess. Once executed, the outputs obtained are listed in the 'OUTPUTS' list box.

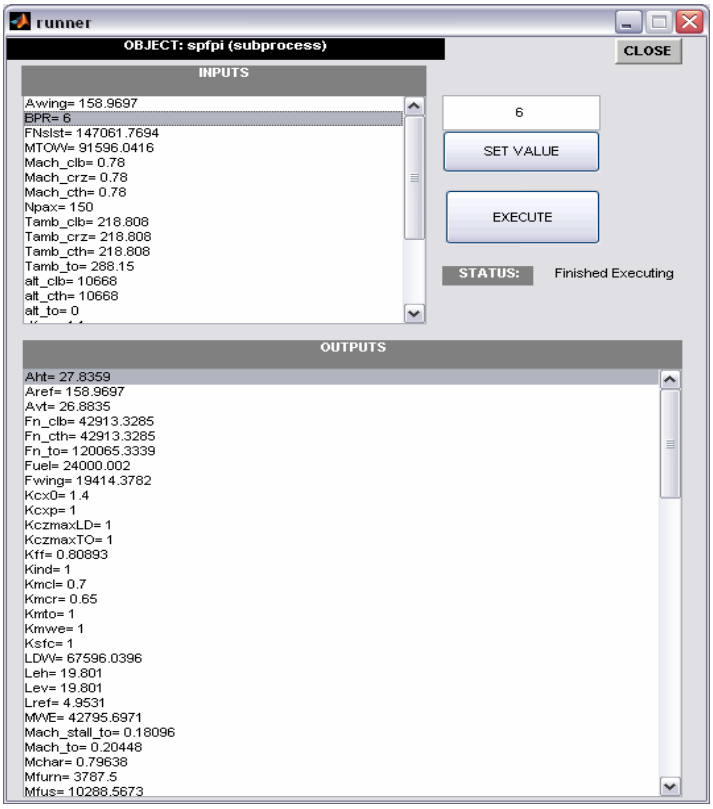


Figure 6-11 GUI for executing the objects

There is currently no interface available for executing the study objects. The selected study object in the 'STUDY OBJECTS' list box in the CWMD window is directly executed (by activating the execute method associated with the study object) when the 'ST' push button in the 'Run' panel is clicked.

6.5.3 Viewer

In CWMD, a subprocess can be viewed in four different formats which are the DSM, Incidence Matrix, Graph and tabular format.

Double clicking the subprocess objects in the objects list box of the CWMD window will display the corresponding objects in the format chosen in the 'Plot' panel.

Plotting as incidence matrix

Incidence matrix plot for a subprocess has the constituent models and/or subprocesses representing the rows, and the constituent variables representing the columns. An 'I'

marked in a cell denotes that variable representing the column of the cell is an input to the model/subprocess representing the corresponding row. Similarly an 'O' marked in a cell shows that the variable in the column is an output of the model/subprocess in the corresponding row. The cells with 'I' and 'O' are coloured in green and red for clarity purpose.

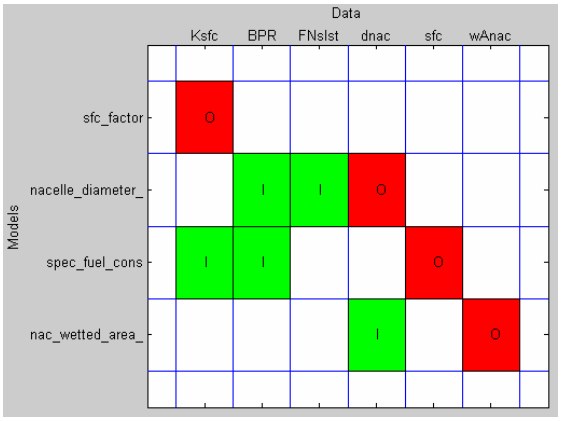


Figure 6-12 Incidence matrix plot by CWMD

This plot is activated by the 'view_i' method of the subprocess object. The incidence matrix display for the 'engines' example subprocess is shown in Figure 6-12.

Plotting as DSM

A DSM plot for a subprocess has models/subprocess in its process attribute, representing both rows and columns of the matrix. A black dot marked in cell denotes the data flow from the model/subprocess representing the row to the model/subprocess representing the column. A dot above the diagonal denotes a feed forward loop and a dot below the diagonal denotes a feedback loop.

This plot is activated by the 'view_d' method of the subprocess object. An example DSM plot is shown in Figure 6-13.

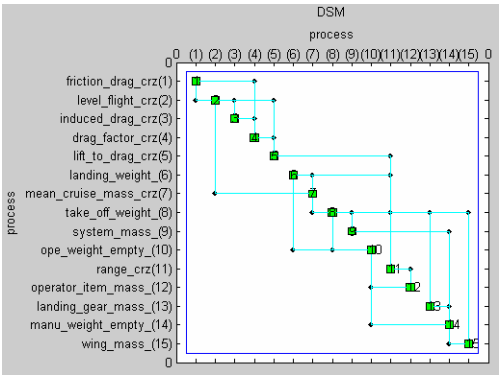


Figure 6-13 Design structure matrix plot by CWMD

Plotting as graph

A Graph plot for a subprocess displays in a graphical format the interaction among different models/subprocesses, through their variables. The models/subprocess and the associated variables in the subprocess are displayed in oval shaped boxes, with directed arrows representing their mutual association. Double clicking on a subprocess in the graph will display the graph plot of that subprocess in another window.

This plot is activated by the 'view_g' method of the subprocess object. An example graph plot is shown in Figure 6-14.

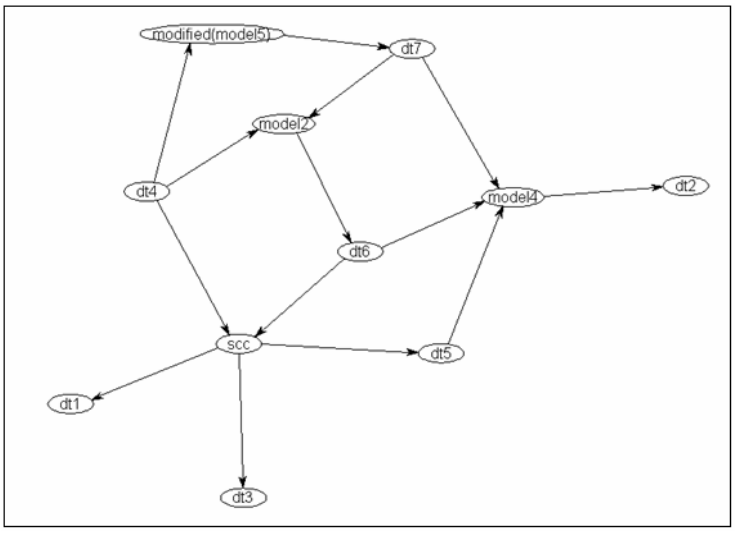


Figure 6-14 Graph plot of a subprocess by CWMD

Plotting in tabular format

A tabular plot for a subprocess displays the attributes of the subprocess in a tabular format. Double clicking on a subprocess listed in the 'PROCESS' list box, will display the tabular plot of that subprocess in the same window.

This plot is activated by the 'view_t' method of the subprocess object. An example tabular plot is shown in Figure 6-15.

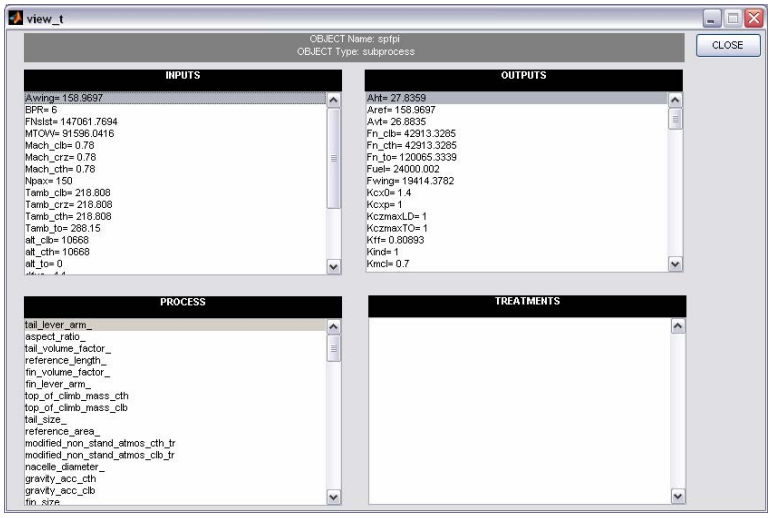


Figure 6-15 Tabular plot of a subprocess by CWMD

6.6 Convergence Monitoring in CWMD

SCC's and modified model's convergence monitoring are implemented in CWMD. For SCCs, while solving using FPI-second method and Gauss-Newton method, the difference in the values of the feedback variables, in the previous and the current iteration, are plotted against the iteration number, for monitoring the convergence. The residual approaching zero indicates convergence. An example plot which shows the convergence of the feedback variables MTOW (Maximum Takeoff Weight) and RA (Range) in an SCC is given in the Figure 6-16.

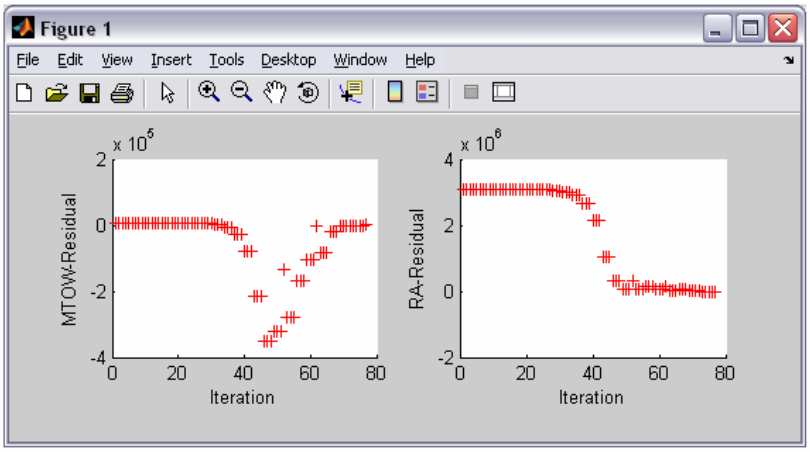


Figure 6-16 Convergence monitoring for SCC

In case if the FPI-first method is used for solving the SCC, CWMD displays two plots. The first one highlights, in a DSM format the loop in which the current iteration is taking place and the second one plots the difference in the value of the feedback variable of that loop, in the previous and current iteration, against the iteration number. An example for this is given in Figure 6-17.

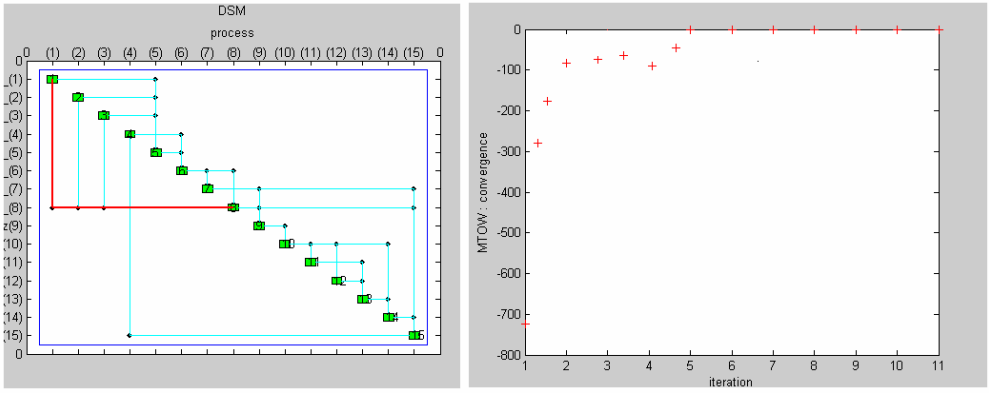


Figure 6-17 Convergence monitoring for SCC when FPI-first solver is used.

For the modified models the convergence monitoring is similar to the one shown in Figure 6-16. Here instead of the feedback variables, $ym\text{-}ym^d$ (which was explained in section 4.2) are plotted against the iteration number.

6.7 Summary and Conclusions

This chapter has described the software framework which has been developed in order to test and evaluate the various methods and techniques developed as part of this research.

The object oriented approach applied to the design and implementation of the CWMD has brought the following advantages:

- The variables were modelled as data objects, which generalised the representation of heterogeneous variables in a system into a common object model. This representation supported similar handling capability for different types of variables.
- The models were represented as model objects, which generalised the heterogeneous models in a system into a common object notation. The general representation assisted easier interaction among the models which was vital during data transfer.
- The subprocess object representation allowed complicated systems to be modelled and saved into executable subprocess objects.
- The commonality in model and subprocess objects allowed easier interaction among these, which also enabled their assembling into higher level subprocesses.
- The subprocess allowed hierarchical organisation of complicated systems, which minimized the chaos of dealing with numerous models.
- The object oriented programming allowed easier implementation of the computational plan developed by the computational process modeller, for systems. This allows the designer to focus more on the design study itself instead on the implementation issues of the computational plans.
- The ability to replace the solvers in the treatment attribute of the subprocess allows SCCs and modified models to be custom solved with different solvers (in case if a particular solver is not leading to convergence) without the need for major modifications.

In addition, CWMD has different types of viewers for subprocess, which helps the designer to visualise and study the process clearly.

The evaluation provided by the industry partner confirmed the flexibility and efficiency offered by the CWMD in terms of choosing the inputs for the system, implementing the computational plans, visualising the process and the solving the systems.

The current limitations for CWMD are listed below.

- Currently only Matlab functions can be implemented as model objects in CWMD. External compiled codes cannot be accessed as model objects.
- Even though a number of treatment objects can be incorporated in a study object, currently it has been tested with only a single treatment.
- 'FMINCON' function of Matlab for optimisation is currently implemented as optimiser treatment in CWMD. Equalities and inequality constraints are not yet implemented.
- External treatment functions cannot be currently modelled in CWMD as treatment objects, through the GUI. However, all the necessary requirements for incorporating this feature are implemented in CWMD.
- The options settings (number of iterations etc.) for SCC solvers, modified model solvers and optimiser are currently hard coded. For all the solvers tolerance is set to 1e-10 and Maximum iterations to 5000.
- There are currently no automated resolution schemes implemented for over and under determined systems. The user is only warned in case of such systems.

7 SUMMARY AND CONCLUSIONS

Presented in this thesis is a computational process modeller for dynamically assembling and solving a system of non-linear models which represent the physics and other characteristics of the aircraft at the conceptual design stage. The primary objective of this research has been to develop methods and techniques which significantly increase the flexibility and efficiency with which the designer is able to operate on large scale computational multidisciplinary systems. This objective has been met through the development of the computational process modeller and the software framework, CWMD.

In this research the computational process modeller was developed with focus on aircraft conceptual design. However, the applicability of the computational process modeller is not just limited to the aircraft domain. It has the potential to be applied to the conceptual design phase of any complex product which can be represented in the form of models (e.g. financial modelling).

7.1 Literature Review

The comprehensive literature review which has been conducted has identified a number of existing methods for computational process modelling. The majority of the methods for variable flow modelling, which is the first step in computational process modelling, were applicable to algebraic equations, but needed modification for application in the current research context, where models are used. In addition, the current computational process modelling methods which were available for models have focussed only on decomposition and scheduling and not on variable flow modelling, which limits the flexibility in choosing the independent variables for the system of models.

Various computational tools for solving low and high fidelity mathematical models were also reviewed to identify the potential of those tools to be applied in the aircraft conceptual design stage. None of the existing low-fidelity tools has the flexibility in selecting the inputs for the system of models. As for the higher-fidelity tools, these currently lack an automated integration capability and the flexibility required for integrating numerous simple models in a dynamic computational environment.

7.2 Computational Process Modelling

A novel computational process modeller which generates optimal computational plans for systems while ensuring the flexibility in choosing the inputs to the system and improving the efficiency in solving the system has been developed. The novel method is an integrated scheme which incorporates variable flow modelling, decomposition and scheduling methods.

For variable flow modelling, a novel incidence matrix method (IMM) has been introduced which has the advantage of rapidly producing feasible variable flow models for systems which contain models that can generate multiple outputs. In addition, the IMM approach is capable of exploring all feasible variable flow models for a system. Until now this feature has not been available even in computational process modelling methods for algebraic equations, were the focus has always been on generating a single feasible variable flow model. Criteria were derived for choosing the optimal variable flow model from the group which would lead to faster convergence of the system. A modified formal IMM was also introduced which, compared to IMM, is easier to implement and takes less memory space while computing. The method also accounted for over and underdetermined systems and derived resolution schemes.

Decomposition was performed based on an algorithm from concurrent engineering for identifying coupled design processes.

Scheduling the coupled models (SCC) was performed by genetic algorithm with the number of feedback loops as the objective function to be minimised. The number of modified models was chosen as the first criteria for selecting the optimal variable flow model. The objective function obtained after rearranging the SCCs was chosen as the second criteria for choosing the optimal variable flow models if there were more than one variable flow model with equal number of modified models. As a result the scheduling procedure satisfied the aims of scheduling the models in the SCC and choosing the optimum variable flow model.

The non-coupled models were sequentially arranged based on an algorithm from concurrent engineering for arranging design tasks.

In this research the computational process modeller was developed with focus on aircraft conceptual design stage. All the testing and evaluation were performed on

aircraft conceptual design test cases. The objective function for rearrangement and the solvers for the sub-systems were chosen after conducting tests on the available models representing the physics and characteristics of the aircraft. However, the applicability of the computational process modeller is not just limited to aircraft conceptual design cases. It has the potential to be applied to any complex product which can be represented in the form of models.

Applying the computational process modeller to any other system will require reformulation of the objective function for rearrangement and also the solvers for the sub-systems. These will depend on the characteristics of models being used. A test, as explained in the results and discussion chapter can be performed in order to formulate the objective function and the appropriate solver in such cases.

7.3 Testing and Evaluation

Different tests were performed in order to finalise the objective function for scheduling the coupled models. After the tests it was identified that the objective function should be formulated based on the specific solver and cannot be generalised for solving SCCs using any solver. Objective functions, feedback length and feedback number were investigated with FPI-first, FPI second and Gauss-Newton as solvers for SCC. Feedback length was proven inefficient to provide an estimate of the computational cost. In contrast, feedback number provided a good quality estimate, if the solver for SCC was either FPI-first or GN.

The various tests conducted for evaluating the computational process modeller also proved that in the majority of the cases an increase in the number of modified models in the system increased the computational cost. Based on this observation the number of modified models in the system was also incorporated in choosing the optimal variable flow model.

The different tests performed on the USMAC test case for evaluating the computational process modeller, have confirmed the effectiveness of this modeller for generating optimal computational plans for complex aircraft conceptual design systems. Even though there were a few sub-optimal choices with regard to computational cost, all choices for computational flow made by the system were among the best.

The different tests conducted have demonstrated the ability of the computational process modeller to provide the flexibility for the designer in choosing the independent variables during the design process. The tests have also proven the improvement in the efficiency of solving the systems by applying the computational plan generated by the computational process modeller.

7.4 Framework: CWMD

The CWMD framework developed for testing and evaluating the various methods and techniques developed in this research has brought in several advantages because of its object-oriented design and implementation. The variables and models which were modelled as objects generalised the representation of heterogeneous variables and models in a system, into a common object model. This representation supported similar handling capability for different types of variables and easier interaction among the models which was vital during data transfer. The subprocess object representation allowed complicated systems to be modelled and saved into executable subprocess objects and also allowed quick implementation of the computational plan developed by the computational process modeller. This capability allows the designer to focus more on the design study itself rather than the implementation issues of the computational plans. The ability to replace the solvers in the treatment attribute of the subprocess allows SCCs and modified models to be custom solved with different solvers (in case if a particular solver is not leading to convergence) without the need for major modification in the implementation. The evaluation provided by the industry partner confirmed the flexibility and efficiency offered by the CWMD in terms of choosing the inputs for the system, implementing the computational plans, visualising the process and the solving systems.

7.5 Current Limitations

The application of the computational process modeller is limited to systems with simple models containing scalar variables.

The FPI-first, FPI-second and Gauss-Newton method were used in this research for solving the SCCs, and the Gauss-Newton method was used for solving the modified models. The limitation identified for solving the SCCs and modified models using these

methods was that the convergence and final results obtained after solving were greatly dependent upon the starting points given during the iterative solving of the unknown variables. The current research focussed only on obtaining a feasible solution and more investigation is necessary to explore the results, where multiple solutions exist. This area was beyond the scope of this research.

Further investigation is necessary in order to formulate an objective function for rearranging the models in SCC when FPI-second is used as the solver. This solver was computationally less expensive compared to other solvers when tested, but has not shown any dependency between the currently formulated objective function, and the computational cost associated with FPI-second solver.

There were a few sub-optimal choices made by computational process modeller when tested on the USMAC test case. These sub-optimal choices of the variable flow models for the SCCs by the computational process modeller were inevitable since the computational cost for a system was found to be not only depended on the feedback number and the number of modified models, but also on the starting guess for the unknown variables, mutual sensitivity of the switched variables of the modified models and possibly other factors which are yet to be discovered. The current criteria for selecting the optimal variable flow model has to be further modified to incorporate the above mentioned factors, which will subsequently further reduce or eliminate altogether the sub-optimal choices.

Since solving the modified models is dependent on the starting points and the mutual sensitivity of the switched variables, certain modified models may require additional computational cost to solve, compared to others. This might challenge the association noticed during our testing between the number of modified models and computational cost, if tested with different starting points for the same problem. This will require modification of criteria for selecting the optimal variable flow model which accounts for the computational expense due to modified models. Currently the first criterion for selecting the optimal variable flow model is set as the minimum number of modified models and the second is the feedback number. The above mentioned challenge suggests further investigation in this area.

There are currently a few limitations of the CWMD which can be improved further. More relevant attributes can be added to objects to capture more details of the element being modelled. An example for this will be an attribute which can capture the propagation of uncertainty while solving the models. Currently only Matlab functions can be implemented as model objects in CWMD. External compiled codes cannot be accessed as model objects.

Even though a number of treatment objects can be incorporated in a study object, currently it has been tested with only a single treatment. Further development of this aspect will enable to model multidisciplinary design optimisation studies in the study object which will allow sequential and cyclic application of various treatments on the systems.

7.6 Future Work

Future work could address the current limitations of the proposed methods for computational process modelling and CWMD, namely;

- Investigation of the influence of the starting guesses for unknown values of the SCCs and modified models on convergence.
- Study of the effect of the mutual sensitivity of switched model variables on solving the modified models, and its implications on the convergence of the derived variable flow models.
- Normalising the models based on its solving complexity. This will help in grading the models in the system and providing a criterion for intelligently choosing the models which are to be modified (i.e. choosing the optimal variable flow model) during the computational design process modelling.
- Identification of other factors which can possibly affect the convergence of the SCCs and modified models.
- Improvement of the criteria for selecting the optimal variable flow model considering the above factors, in order to eliminate the sub-optimal choices by the computational process modeller.

- Investigation of various other solvers for SCCs and modified models, and generating corresponding objective functions for these.
- Exploration of the multiple solutions that can be generated while solving the modified models and SCCs.
- Derivation of improved resolution schemes for under and over determined systems.
- Improvement of the study object in order to incorporate multidisciplinary design optimisation studies, by including multiple treatments.

REFERENCES

Altus.S.S, Kroo I.M, Gage P.J (1996), *A genetic algorithm for scheduling and decomposition of multidisciplinary design problems*, Transactions of ASME, DEC-1996, Vol.118

Balling, R.J., Sobieszczanski-Sobieski, J. (1994), Optimization of Couples Systems: A Critical Overview of Approaches, AIAA-94-4330-CP, *Proceedings of the 5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pp.697-707, Panama City, Florida, September 7-9.

Booch, G (1993). *Object-Oriented Analysis and Design with Applications*, 2nd edition, Addison-Wesley. ISBN 0-8053-5340-2.

Bouchard, E. E., Kidwall, G. H., and Rogan, J. E. (1988), The Application of Artificial Intelligence Technology to Aeronautical System Design, AIAA 88-4426, *AIAA Aircraft Design Systems and Operations Meeting*, Atlanta, Georgia.

Buckley, M.J., Fertig, K.W. and Smith, D.E. (1992), Design Sheet: An environment for facilitating flexible trade studies during conceptual design, AIAA 92-1191, *Aerospace Design Conference*, Irvine, California

Cacuci, D. G. (2003), *Sensitivity and uncertainty analysis. Volume I, Theory*, Chapman & Hall/CRC Press, 1 edition, NW, USA

Chen, L., Ding, Z., and Li, S. (2005), *A formal two-phase method for decomposition of design problems*, Journal of Mechanical Design, 127:184--195, March 2005.

Cormen, T., Leiserson, C., and Rivest, R., 1991, *Introduction to Algorithms.*, McGraw-Hill Book Company, New York.

Denniz, J.E., Jr., Robert, B., 1983, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice –Hall, Inc. Englewood Cliffs, New Jersey

Dyer, C. (2002), Fixed Point Iteration, *Citing internet resources*, http://pathfinder.scar.utoronto.ca/~dyer/csca57/book_P/node34.html ,(accessed 21st August 2007)

Gauss Newton Method (2007), *Citing internet resources*, http://en.wikipedia.org/wiki/Gauss-Newton_algorithm ,(accessed 21st August 2007)

Howe, D. (2000), *Aircraft Conceptual Design Synthesis*, John Wiley & Sons, West Sussex, England

Kroo, I., Altus, S., Braun, R., Gage, P., Sobieski, I. (1994), Multidisciplinary Optimization Methods for Aircraft Preliminary Design, AIAA-94-4325-CP, *Proceedings of the 5th AIAA/ NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pp.697-707, Panama City, Florida, September 7-9

Kusiak, A., and Cheng, C. H. , 1990, *A Branch-and-Bound Algorithm for Solving the Group Technology Problem*, Annals of Operations Research, Vol. 26, pp. S415-431

Kusiak, A., and Chow, W. S. , 1987, *Efficient Solving of the Group Technology Problem*, Journal of Manufacturing Systems, Vol. 6, No.2, pp.117, 124

Kusiak, A., and Wang, J. (1993), *Decomposition of the design process*, Journal of Mechanical Design, Vol. 115, No. 4, 1993

Morris A.M.J. (2002), MOB-A European Distributed Multi-disciplinary Design and Optimisation Project, *9th AIAA/ISSMO Conference*, Atlanta, GA, USA, September 2002. AIAA-2002-5444.

Padula S.L., Korte J.J., Dunn H.J., Salas A.O. (1999), *Multidisciplinary Optimization Branch Experience Using iSIGHT Software*, NASA Technical Report, NASA/TM-1999-209714

Papalambros, P.Y., *Optimal Design of Mechanical Engineering Systems,* Journal of Mechanical Design, 1995. 117(June): p. 55-62.

Ramaswamy, R., and Ulrich, K. (1993), A Designer's Spreadsheet, *Proceedings of the Design Theory and Methodology Conference,* DE-Vol. 53, ASME, New York, pp. 105-113.

Raymer, D.P. (1999), *Aircraft Design: A Conceptual Approach*, AIAA Education Series, Washington, D.C.

Rogers, J.L. (1989), *A Knowledge-Based Tool for multilevel decomposition of a complex design problem,* NASA Technical paper 2903, 1989

Rogers, J.L. (1997), Reducing design cycle time and cost through process resequencing, *Proceedings of the International Conference on Engineering Design*, ICED 97, Tampere, Aug 19-21, 1997

Rogers, J. L. (1999), *Tools and Techniques for Decomposing and Managing Complex Design projects*, Journal of Aircraft, Vol. 36 No. 1, Jan.-Feb. 1999, pp. 266-274.

Scott A.T. (2001), *An evaluation of three commercially available integrated design framework packages for use in the Space Systems Design Lab*, White Paper, Phoenix Integration

Serrano, D., (1987), *Constraint Management in Conceptual Design*, Ph.D. Dissertation, MIT, Department of Mechanical Engineering, Cambridge, Massachusetts

Sobieszczanski-Sobieski, J. (1988), Optimization by Decomposition: A Step from Hierarchic to Non-Hierarchic Systems, *NASA Conference Publication 3031, Part 1, Second NASA/Air Force Symposium on Recent Advances in Multidisciplinary Analysis and Optimization*, Hampton, Virginia, September 28-30

Sobieszczanski-Sobieski, J., Agte, J., and Sandusky, Jr. (1998), Bi-level Integrated System Synthesis (BLISS) , AIAA 98-4916, *7th AIAA/USAF/NASA/ISSMOSymposium on Multidisciplinary Analysis and Optimization*, September 2-4, 1998, St. Louis, MO.

Steward, D. V. (1993), Re-engineering the Design Process, *Proceedings of the Second Workshop on Enabling Technologies*, Infrastructure for Collaborative Enterprises, April, Morgantown, WV,

Steward, D. V. (1981), *Systems Analysis and Management: Structure, Strategy and Design*. Petrocelli Books Inc., New York. 1981.

Syswerda, G. (1990), *Scheduling Optimization Using Genetic Algorithms*, Handbook of Genetic Algorithms, Davis, I., ed. Van Nostrand Reinhold, New York.

Tang, D., Zheng, L. and Zhizhong, L. (2000), *Re-engineering of the design process for concurrent engineering*, Computers and Industrial Engineering 38(2000) 479-491

Vankan W. J. and Laban M. (2002), A Spineware Based Computational Design Engine for Integrated Multi-disciplinary Aircraft Design, AIAA 2002-5445, *Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 4-6 September 2002, Atlanta, Georgia

VIVACE (2005), *Citing internet resources*, http://ww.vivaceproject.com, (accessed 21st August 2007)

Weisstein. E.W. (2007), *Wolfram Mathworld*, http://ww.mathworld.wolfram.com/IncidenceMatrix.html, Retrieved on 21 June 2007

Wujek B.A., Koch P.N., McMillan M., Chiang W.S., (2002), A Distributed, Component-Based Integration Environment for Multidisciplinary Optimal and Quality Design, AIAA-2002-5476, 9th *AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, September 4-6, 2002, Atlanta, Georgia.

Xiao, R., & Fei, Q. (1997), Application of the improved method in structural modelling to comprehensive management of the Mine Bereaus. System Engineering: Theory and Practise, 17(3), 57-62

Zweber J.V., Kabis H., Follett W.W., Ramabadran N. (2002), *Towards An Integrated Environment For Hypersonic Vehicle Design and Synthesis*, AIAA 2002-5172, AIAA/AAAF 11th International Space Planes and Hypersonic Systems and Technologies Conference, 29 Sep - 4 Oct 2002, Orleans.

BIBLIOGRAPHY

AIAA Techical Committee on Multidisciplinary Design Optimization(MDO), (1991), *White Paper on Current State of the ART*, American Institue of Aeronautics and Astronautics, Inc., Januray 15.

Antonie, N., Kroo, I., Willcox, K. and Barter, G. (2004), A Framework for Aircraft Conceptual Design and Environmental Performance Studies, *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 30 August-1 September, Albany, New York.

Arian, E. (1997), Convergence Estimates for Multidisciplinary Analysis and Optimization, *NASA/CR-201752, ICASE Report No.97-57*, NASA Langley Research Center, Hampton, Virginia.

Balachandran, L. K., Fantini, P. F. and Guenov, M. D., (2007), Computational Process Management for Aircraft Conceptual Design, *7th AIAA Aviation Technology, Integration and Operations (ATIO) Conference*, September 18-21, 2007, Belfast, Northern Ireland.

Bela, B. (1979), *Graph Theory: An Introductory Course*., Springer-Verlag, New York.

Blouin, V. Y., Summers, J. D. and Fadel, G. M. (2004), Intrinsic Analysis of Decomposition and Coordination Strategies for Complex Design Problems, *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 30 August-1 September, Albany, New York.

Bouchard, E. E., Kidwell and G. H., Rogan, J. E. (1988), The Application of Artificial Intelligence Technology to Aeronautical System Design, *AIAA/AHS/ASEE Aircraft Design Systems and Operations Meeting*, September 7-9, 1988/Atlanta, Georgia.

Browning, T. R. (2002), Process Integration Using the Design Structure Matrix, Systems Engineering, Volume 5, Issue 3 , Pages 180 – 193, Wiley Periodicals, Inc.

Carty, A. (2002), An Approach to Multidisciplinary Design Analysis and Optimization For Rapid Conceptual Design, AIAA-2002-5438, 9th *AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, September 4-6, 2002, Atlanta, Georgia.

Chanron, V., and Lewis, L. (2005), *A study of Convergence in Decentralized Design Processes*, Research in Engineering Design(2005), 16: 133-145.

Chen, L., Ding, Z., and Li, S. (2005), *Analysis of Decomposability and Complexity for Design Problems in the Context of Decomposition*, Journal of Mechanical Design, Vol. 127, July 2005.

Chen., L., Simon, L. (2005), *Analysis of decomposability and complexity for design problems in the context of decomposition*, Journal of Mechanical design, ASME, 2005, Vol 127/545.

Dent, D., Paprzycki, M. and Kucaba-Pietal, A. (2000), *Recent advances in solvers for nonlinear algebraic equations*, Comput. Assist. Mech. Eng. Sci. (CAMES) 7 (2000) 493-505.

Design Sheet (2004), *Citing internet resources*, http://www.design-sheet.com (accessed 21st August 2007).

English, K. (2002), Combined System Reduction and Sequencing in Complex System Optimization, AIAA-2002-5412, 9th *AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, September 4-6, 2002, Atlanta, Georgia.

Fang, H., Horstemeyer, H. F. (2004), An Integrated Design Optimization Framework Using Object-oriented Programming, AIAA 2004-4499, *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 30 August-1 September, Albany, New York.

Fantini, P. F., Balachandran, L. K., and Guenov, M. D., (2007), Computational Intelligence in Multi Disciplinary Optimization at Feasibility Design Stage, *First International Conference on Multidisciplinary Design Optimization and Applications*, April 2007, Besancon, France.

Fiper (1996), *Citing internet resources*, http://www.engineous.com , (accessed 21st August 2007).

Gould, R. J.(1988), *Graph theory*, Benjamin-Cummings Pub Co, Menlo Park, CA

Guenov, M. D., Balachandran, L. K., Tang, D., Lockett, H.,(2006) Computational Process Modelling, *ICAS conference, 25th Congress of International Council of the Aeronautical Sciences*, September-2006, Hamburg, Germany

Hulme, K. F. (2000), *The Design of a Simulation-based Framework for the Development Approaches in Multidisciplinary Design Optimization*, PhD Dissertation, University at Buffalo, Buffalo, NY

Hulme, K. F. and Bloebaum, C. L (1999), A Comparison of Formal and Heuristic Strategies for Iterative Convergence of a Coupled Multidisciplinary Analysis, *Third World Congress on Structural and Multidisciplinary Optimization*, May, 1999, Amherst, NY.

Hulme, K. F., Bloebaum, C. L. and Nozaki, Y. (2000), A Performance-Based Investigation of Parellel and Serial Approaches to Multidisciplinary Analaysis Convergence, AIAA-2000-4812, *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 8th, , Sept. 6-8, 2000 Long Beach, CA.

Kolb, M. A., (1996), An Object-Oriented Framework for Multidisciplinary Robust Preliminary Design, AIAA-1996-4105, *NASA, and ISSMO, Symposium on Multidisciplinary Analysis and Optimization*, Sept. 4-6, 6th, Bellevue, WA,

Krishnamachari, R. S., and Papalambros, P., (1997), *Optimal Hierarchical Decomposition Synthesis Using Integer Programming*, ASME Journal of Mechanical. Design., 119(4), pp. 440–447.

Krishnan, R. (1998), Evaluation of Frameworks for HSCT Design Optimization, NASA/CR-1998-208731, NASA Langley Research Center, Hampton, Virginia

Kumar, V., Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, http://citeseer.ist.psu.edu/article/kumar92algorithms.html

Laban, M. (2004), Multi-Disciplinary Analysis and Optimisation of Supersonic Transport Aircraft Wing Planforms, AIAA 2004-4542, *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 30 August - 1 September 2004, Albany, New York

Lano, R. J. (1977), *The N^2 Chart*, Systems Engineering and Integration Division One Space Park, Redondo Beach, California.

Malone, B., Woyak, S. (1995), An Object- Oriented Analysis and Optimization Control Environment for the Concpetual Design of Aircraft, AIAA-1995-3862 Aircraft Engineering, Technology, and Operations Congress, 1st, Sept 19-21, 1995 ,Los Angeles, CA,

Matlab 7.0 (2004), *Matlab Graphics*, The Mathworks Inc., Natick MA.

Matlab 7.0 (2004), *Matlab Mathematics*, The Mathworks Inc., Natick MA.

Matlab 7.0 (2004), *Matlab Programming*, The Mathworks Inc., Natick MA.

Matlab 7.0 (2004), *Symbolic Math Tool box*, The Mathworks Inc., Natick MA.

Messac, A., Yahaya, A. I., Mattson, C. A. (2003), *The Normalized Normal Constraint Method for Generating the Pareto Frontier*, Structural and Multidisciplinary Optimization, Vol. 25, No. 2, 2003, pp. 86-98.

Michelena, N. F., and Papalambros, P. Y. (1994), *A network reliability approach to optimal decomposition of design problems*, Advances in Design Automation--1994, B. Gilmore, ed., 1994, ASME, pp. 195—204, vol. 2, New York.

Padula, S. L., Alexandrov, N. and Green, L. L.(1996), MDO Test Suite at NASA Langley Research Center, *Sixth AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Opitmization*, September 4-6, 1996, Bellevue, Washington.

Parashar, S. and Bloebaum, C. L. (2005), Descision Support Tool for Multidisciplnary Design Optimization (MDO) using Multi-Domain Decomposition, *46th*

AIAA/ASME/ASCE/AHS/ASC Structures Strucutral Dynamics and Materials Conference, 18-21 April 2005, Austin, Texas.

Park, H. W., Kim, M. S., Choi, D. H. and Marvis, D. N. (2002), Optimizaing the Parallel Process Flow for the Individual Discipline Feasible Method, AIAA-2002-5411, 9th *AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, September 4-6, 2002, Atlanta, Georgia.

Phoenix Intergration (1995), *Citing internet resources*, http://ww.phoenix-int.com, (accessed 21st August 2007).

Ramaswamy, V. and Lewis, K. E. (1998), Conceptual Design of a Complex Engineering System Through Coupled Selection, AIAA-1998-4882, *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 7th, Sept. 2-4, 1998, St. Louis, MO.

Rao S. S. (1998), Engineering Optimization Theory and Practise, New Age International (P) Limited, New Delhi, India.

Reddy, S. Y. and Fertig K. W. (1996), Design Sheet: A System for Exploring Design Space, Application to Automotive Drive Train Analysis, *Fourth International Conference on Artificial Intelligence in Design (AID'96)*, Standford Univeristy.

Reddy, S. Y. and Fertig, K. W. (1998), Managing Function Constrints in Design Sheet, *Proceedings of DETC'98 1998 Design Engineering Technical Conferences,* September 13-16, 1998, Atlanta, Gerogia.

Reddy, S. Y. and Fertig, K.W. (1995), *Facilitating Infrared Seeker Performance Trade Studies Using Design Sheet*, Rockwell Palo Alto Laboratory technical report, Rockwell Science Center, Palo Alto Laboratory, USA.

Reddy, S. Y., Fertig, K. W. and Smith, D. E. (1996), Constraint Management Methodology for Conceptual Design Trade-off Studies, *Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, August 18-22, 1996, Irvine, California.

Reddy, S. Y., Fertig, K.W. and McCormick D.J (2006), Constrained Exploration of Trade Spaces, *Proceedings of the 2nd IEEE International Conference on Space Mission Challenges for Information Technology*, IEEE Computer Society Washington, DC, USA.

Reddy, S. Y., Fertig, K.W. and Smith D.E. (1996), Constraint Management Methodology for Conceptual Design Tradeoff Studies, *Proceeding of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, August 18-22, 1996, Irvine, California.

Rogers, J. L. (1996), DeMAID/GA - An enhanced design manager's aid for intelligent decomposition, AIAA-1996-4157, NASA, and ISSMO, Symposium on Multidisciplinary Analysis and Optimization, 6th, Sept. 4-6, 1996, Bellevue, WA.

Rogers, J. L., Korte, J. J. and Bilardo, V. J. (2006), Development of a Genetic Algorithm to Automate Clustering of a Dependency Structure Matrix, , *NASA/TM-2006-214279*, NASA Langley Research Center, Hampton, Virginia.

Rogers, J. L., McCulley, C. M. and Bloebaum, C. L. (1996), *Integrating a Genetic Algorithm Into a Knowledge-Based System for Orderign Complex Design Processses*, NASA Techinal Memorandum 110247, NASA Langley Research Center, Hampton, Virginia

Salas, A. O. and Rogers, J. L. (1997), *A Web-Based System for Monitoring and Controlling Multidisciplinary Design Projects*, NASA/TM-97-206287, National Aeronautics and Space Administration, Langley Research Center, Hampton, Virginia.

Sampath, R. and Kolonay, R. M.(2002), 2D/3D CFD Design Optimization Using the Federated Intelligent Product Environment (FIPER) Technology, 9th *AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, September 4-6, 2002, Atlanta, Georgia.

Sobieszczanski-Sobieski, J., Haftka, R. T. (1997), *Multidisciplinary Aerospace Design Optimization: Survey of Recent Development*, Structural and Multidisciplinary Optimization, Springer Berlin / Heidelberg, Volume 14, Number 1 / August, 1997.

Sullivan, B. O. (2002), *Constraint-Aided Conceptual Design,* Engineering Research Series, Professional Engineering Publishing Limited, London, UK.

Townsend, J. C. and Salas, A. O. (2002), Managing MDO Software Development Projects, 9th *AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, September 4-6, 2002, Atlanta, Georgia.

Vankan W. J., Schulthesis, B. C. and Baalbergen, E. H. (2002), *ICT Environment for Integrated Multidisciplinary Aircraft Design Analysis*, *NLR-TP-2001-338*, National Aerospace Laboratory, NLR.

Wujek B.A., Koch P.N. and Chiang W.S., (2002), A Workflow Paradigm for Flexible Design for Flexible Design Process Configuration in Fiper , AIAA-2000-4868, 8th *AIAA/USA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, September 5-8, 2000, Long Beach, CA.

Xie, S., Milthorpe, J. and Smith, W. F.(2004), A Contribution-based Decomposition Method for Multidisciplinary Engineering Optimization, *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 30 August-1 September, Albany, New York.

Yu, L. Y., Yassine, A. A. and Goldberg, D. E.(2003), A Genetic Algorithm for Developing Modular Product Architectures, *Proceedings of DETC'03, ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, September 2-6, 2003, Chicago, Illinois, USA.

APPENDICES

I. Publications by the author related to this thesis

Balachandran, L. K., Fantini, P. F. and Guenov, M. D., (2007), Computational Process Management for Aircraft Conceptual Design, *7th AIAA Aviation Technology, Integration and Operations (ATIO) Conference*, September 18-21, 2007, Belfast, Northern Ireland.

Guenov, M. D., Balachandran, L. K., Tang, D., Lockett, H.,(2006) Computational Process Modelling, *ICAS conference, 25th Congress of International Council of the Aeronautical Sciences*, September-2006, Hamburg, Germany

Fantini, P. F., Balachandran, L. K., and Guenov, M. D., (2007), Computational Intelligence in Multi Disciplinary Optimization at Feasibility Design Stage, *First International Conference on Multidisciplinary Design Optimization and Applications*, April 2007, Besancon, France.

II. Gauss-Newton Method

Given m functions $f_1,\ldots,f_m$ consisting of n parameters $x_1,\ldots,x_n$ with $m \geq n$ and we have to minimise the sum

$$S(x) = \sum_{i=1}^{m} (f_i(x))^2$$

Here x stands for the vector $(x_1,\ldots x_n)$

The aim here is to find x which leads to minimum S. Gauss-Newton algorithm (Dennis and Robert, 1983) (Gauss Newton Method, 2007) is an iterative procedure, the user provides the initial guess for x, denoted as x^0 and the subsequent guesses for x^k are the calculated as follows

$$x^{k+1} = x^k - \left(J_f(x^k)^T . J_f(x^k) \right)^{-1} . J_f(x^k)^T . f(x^k)$$

Here $f = (f_1,\ldots,f_m)$ and $J_f(x)$ is the Jacobian of f at x.

$$J_f(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \cdot & \cdots & \cdot \\ \cdot & & \cdot \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

The matrix inversion in never computed in practise; instead the equation is reformulated to:

$$x^{k+1} = x^k + \delta^k$$

Here δ^k is computed by solving the linear system

$$\left(J_f(x^k)^T . J_f(x^k)\right)\delta^k = -J_f(x^k)^T . f(x^k)$$

The models are black-boxes and cannot be differentiated directly. Therefore finite difference methods are used for computing the Jacobians.

III. Fixed Point Iteration Method

In Fixed point iteration method (Dyer, 2002), for solving an equation *f(x)=0*, the equation is reformed to the format *x=g(x)*. Finding a value for *x* for which *x=g(x)* is thus equivalent to finding a solution of the equation *f(x)=0*. The function g(x) defines a map on the real line over which *x* varies, so that for each value of *x*, *g(x)* maps that point to a new point, *x`* on the real line. Usually *x`* thus obtained and *x* are at some distance apart. If this distance is equal to zero for a particular point $x=x_p$, we call x_p a fixed point of the function *g(x)*. Thus $x_p=g(x_p)$, hence $f(x_p)=0$.

If we are able to choose a point x_0 which lies near to the fixed point x_p, of *g(x)*, where we don't know the value of x_p, the iterative scheme based on fixed point method can be defined as

$$x_{n+1} = g(x_n)$$

Where $n=0,1,\ldots$, the iteration is continued until the difference between successive x_n is as small as the required tolerance. The final value of x_n approximates a fixed point of g(x), and hence approximates a zero of f(x).

IV. Variables and Models of USMAC Test Case

The models and variables of the USMAC testcase are given below. The definitions of the variable and the units of the variables are given in the Table A- 1 for variables. The inputs and outputs and the category in which the model belongs are given in the Table A- 2 for the models.

Table A- 1 Variables of USMAC test case

VARIABLES			
	Variables	**Definition**	**Unit**
1	Aht	Tail size	
2	alt_app	Current altitude	m
3	alt_clb	Current altitude	m
4	alt_crz	cruise altitude	m
5	alt_cth	Current altitude	m
6	alt_to	Current altitude	m
7	ar	Aspect ratio	
8	Aref	Reference wing area	m^2
9	Avt	Fin size	
10	Awing	Wing planform area	m^2
11	BPR	Bypass ratio of the engines	
12	cx_clb	Climb drag factor	
13	cx_crz	Cruise drag factor	
14	cx_cth	Cruise drag factor	
15	cx0_clb	Climb friction drag	
16	cx0_crz	Cruise friction drag	
17	cx0_cth	Cruise friction drag	
18	cxc_clb	Climb pressure drag	
19	cxc_crz	Cruise pressure drag	
20	cxc_cth	Cruise pressure drag	
21	cxi_clb	Climb induced drag	
22	cxi_crz	Cruise induced drag	
23	cxi_cth	Cruise induced drag	
24	cz_clb	level_flight_climb	
25	cz_crz	level_flight_cruise	
26	cz_cth	level_flight_ Cruise	

27	czmax_LD	Maximum lift factor at landing	
28	czmax_TO	Maximum lift factor at take off	
29	dfus	Fuselage diameter	m
30	disa_clb	standard temperature shift	K
31	disa_crz	standard temperature shift	K
32	disa_cth	standard temperature shift	K
33	disa_to	standard temperature shift	K
34	dnac	Nacelle diameter	m
35	Fn_clb	Thrust	N
36	Fn_cth	Cruise Thrust	N
37	Fn_to	Thrust	N
38	FNslst	Sea level static net thrust of one single engine (Newton)	N
39	Fuel	Nominal fuel (kg)	Kg
40	Fwing	Wing fuel tank capacity (kg)	Kg
41	g_app	Current gravity	m/s^2
42	g_clb	Climb gravity	m/s^2
43	g_crz	Cruise gravity	m/s^2
44	g_cth	Cruise gravity	m/s^2
45	g_to	Gravity at take-off	m/s^2
46	Kcx0	Friction drag factor	
47	Kcxp	Pressure drag factor	
48	KczmaxLD	Cz maximum LD factor	
49	KczmaxTO	Cz maximum TO factor	
50	Kff	Proportion of fuel in the fuselage (kg)	Kg
51	kfn_cth	Fuselage length	m
52	Kind	Induced drag factor	
53	Kmcl	Maximum climb factor	
54	Kmcr	Maximum cruise factor	
55	Kmto	Maximum takeoff factor	
56	Kmwe	MWE factor	
57	Ksfc	SFC factor	
58	kvs_LD	Kvs Landing	
59	kvs_TO	Kvs Takeoff	
60	LDW	Nominal landing weight (kg)	Kg
61	Leh	Tail lever arm	m
62	Lev	Fin lever arm	m
63	lfus	Fuselage length	m
64	lod_clb	Lift to drag ratio at climb	
65	lod_crz	Lift to drag ratio at cruise	
66	lod_cth	Mean cruise mass at Cruise	Kg
67	Lref	Reference length	m
68	Mach_clb	current Mach number	
69	Mach_crz	cruise Mach number	

70	Mach_cth	current Mach number	
71	Mach_stall_to	Mach_stall_to	
72	Mach_to	current Mach number	
73	mass_clb	Mean cruise mass at climb	Kg
74	mass_crz	Mean cruise mass at cruise	Kg
75	mass_cth	Number of engines	
76	Mchar	Characteristic Mach number	
77	Mfurn	Furnishing mass	Kg
78	Mfus	Fuselage mass	Kg
79	Mgear	Landing gear mass	Kg
80	Mht	Tail mass	Kg
81	Mop	Operator item mass	Kg
82	Mprop	Engine mass	Kg
83	Msys	System mass	Kg
84	MTOW	Maximum take off weight (kg)	Kg
85	Mvt	Fin mass	Kg
86	MWE	Manufacturer weight empty (kg)	Kg
87	Mwing	Wing mass	Kg
88	Naisle	Number of aisle in the main deck	
89	ne	Number of engines	
90	Npax	Total number of seats	
91	NpaxFront	Maximum number of seats on a row	
92	OWE	Operational weight empty (kg)	Kg
93	Pamb_clb	Climb atmospheric pressure	N/m^2
94	Pamb_crz	Cruise atmospheric pressure	N/m^2
95	Pamb_cth	Cruise atmospheric pressure	N/m^2
96	Pamb_to	takeoff atmospheric pressure	N/m^2
97	phi	Wing characteristic sweep angle (rad)	radians
98	PL	Payload	Kg
99	RA	Nominal range (m)	m
100	RA_time	Cruise time	s
101	rho_clb	Climb air density	Kg/m^3
102	rho_cth	Sound velocity at cruise	m/s
103	rho_to	Takeoff air density	Kg/m^3
104	sfc	Specific fuel consumption (kg/N/s)	kg/N/s
105	span	Wing span (m)	m
106	Tamb_clb	Climb atmospheric temperature	K
107	Tamb_crz	Cruise atmospheric temperature	K
108	Tamb_cth	Cruise atmospheric temperature	K
109	Tamb_to	takeoff atmospheric temperature	K

110	tofl	take off field length	m
111	tuc	Wing characteristic thickness to chord ratio	
112	vapp	approach speed	m/s
113	Vht	Tail volume factor	
114	vsnd_clb	Sound velocity at climb	m/s
115	vsnd_crz	Sound velocity at cruise	m/s
116	vsnd_cth	Cruise speed	m/s
117	Vvt	Fin volume factor	
118	vz_clb	climb speed	m/s
119	wAfus	Fuselage wetted area	m^2
120	wAht	Tail wetted area	m^2
121	wAnac	Nacelle wetted area	m^2
122	wAvt	Fin wetted area	m^2
123	wAwing	Wing wetted area	m^2
124	Wpax	One passenger weight	Kg

Table A- 2 Models of USMAC test case

MODELS				
	Function name	**Outputs**	**Inputs**	**Category**
1	aspect_ratio_	ar	span,Awing	geometry
2	Cz_max_LD_	czmax_LD	KczmaxLD, phi	aerodynamic
3	Cz_max_LD_factor_	KczmaxLD		aerodynamic
4	Cz_max_TO_	czmax_TO	KczmaxTO, phi	aerodynamic
5	Cz_max_TO_factor_	KczmaxTO		aerodynamic
6	engine_mass_	Mprop	ne, FNslst	weight
7	fin_lever_arm_	Lev	Leh	geometry
8	fin_mass_	Mvt	Avt	weight
9	fin_size_	Avt	Awing, span, Lev, Vvt	handling qualities
10	fin_volume_factor_	Vvt		handling qualities
11	fin_wetted_area_	wAvt	Avt	geometry
12	fric_drag_factor_	Kcx0		aerodynamic
13	furnishing_mass_	Mfurn	Npax	weight
14	fus_fuel_ratio_	Kff	Fuel, Fwing	weight
15	fus_wetted_area_	wAfus	dfus, lfus	geometry
16	fuselage_diameter	dfus	NpaxFront, Naisle	geometry

	_			
17	fuselage_length_	lfus	Npax, NpaxFront, dfus	geometry
18	fuselage_mass_	Mfus	dfus, lfus	weight
19	ind_drag_factor_	Kind		aerodynamic
20	Kvs_Landing	kvs_LD		performance
21	Kvs_Take_Off	kvs_TO		performance
22	landing_gear_mass_	Mgear	MTOW	weight
23	landing_weight_	LDW	PL,OWE	weight
24	manu_weight_empty_	MWE	Kmwe, Mfus, Mwing, Mht, Mvt, Mgear, Mprop, Msys, Mfurn	weight
25	MWE_factor_	Kmwe		weight
26	nac_wetted_area_	wAnac	dnac	geometry
27	nacelle_diameter_	dnac	BPR, FNslst	geometry
28	one_pax_weight_	Wpax		weight
29	ope_weight_empty_	OWE	MWE, Mop	weight
30	operator_item_mass_	Mop	RA, Npax	weight
31	Payload_	PL	Npax, Wpax	weight
32	press_drag_factor_	Kcxp		aerodynamic
33	ref_mach_number_	Mchar	phi, tuc	aerodynamic
34	reference_area_	Aref	Awing	aerodynamic
35	reference_length_	Lref	Awing, ar	aerodynamic
36	sfc_factor	Ksfc		engine
37	spec_fuel_cons	sfc	Ksfc, BPR	engine
38	system_mass_	Msys	MTOW	weight
39	tail_lever_arm_	Leh	lfus	handling qualities
40	tail_mass_	Mht	Aht	weight
41	tail_size_	Aht	Awing, Lref, Leh, Vht	handling qualities
42	tail_volume_factor_	Vht		handling qualities
43	tail_wetted_area_	wAht	Aht	geometry
44	take_off_weight_	MTOW_eff	Fuel, PL, OWE	weight
45	wing_fuel	Fwing	Awing, tuc	geometry
46	wing_mass_	Mwing	MTOW, Awing, Lref, span, phi, tuc	weight
47	wing_wetted_area_	wAwing	Awing	Geometry
48	mean_cruise_mass_crz	mass_crz	MTOW,LDW	weight
49	non_stand_atmos	Pamb_crz,Ta	disa_crz,alt_crz	atmosphere

	_crz	mb_crz		
50	sound_velocity_crz	vsnd_crz	Tamb_crz	atmosphere
51	gravity_acc_crz	g_crz	alt_crz	Earth
52	level_flight_crz	cz_crz	mass_crz, g_crz, Mach_crz, Pamb_crz, Aref	performance
53	pressure_drag_crz	cxc_crz	Mach_crz, Mchar, Kcxp	aerodynamic
54	induced_drag_crz	cxi_crz	cz_crz,ar, Kind	aerodynamic
55	friction_drag_crz	cx0_crz	cz_crz,Mach_crz,Pamb_crz,Tamb_crz,wAwing,wAht,wAvt,wAfus,wAnac,lfus,Aref,Lref,ne,Kcx0	aerodynamic
56	drag_factor_crz	cx_crz	cx0_crz,cxi_crz,cxc_crz	aerodynamic
57	lift_to_drag_crz	lod_crz	cz_crz,cx_crz	Lift over Drag definition
58	max_cruise_factor	Kmcr		engine
59	range_crz	RA_eff	MTOW,LDW,Mach_crz,vsnd_crz,g_crz,lod_crz,sfc	mission
60	time_crz	RA_time	RA_eff,Mach_crz,vsnd_crz	Mission
61	top_of_climb_mass_clb	mass_clb	MTOW	weight
62	non_stand_atmos_clb	Pamb_clb,Tamb_clb	disa_clb,alt_clb	atmosphere
63	air_density_clb	rho_clb	Pamb_clb,Tamb_clb	atmosphere
64	sound_velocity_clb	vsnd_clb	Tamb_clb	atmosphere
65	gravity_acc_clb	g_clb	alt_clb	Earth
66	level_flight_clb	cz_clb	mass_clb,g_clb,Mach_clb,Pamb_clb,Aref	performance
67	pressure_drag_clb	cxc_clb	Mach_clb,Mchar,Kcxp	aerodynamic
68	induced_drag_clb	cxi_clb	cz_clb,ar,Kind	aerodynamic
69	friction_drag_clb	cx0_clb	cz_clb,Mach_clb,Pamb_clb,Tamb_clb,wAwing,wAht,wAvt,wAfus,wAnac,lfus,Aref,Lref,ne,Kcx0	aerodynamic
70	drag_factor_clb	cx_clb	cx0_clb,cxi_clb,cxc_c	aerodynamic

			lb	
71	lift_to_drag_clb	lod_clb	cz_clb,cx_clb	Lift over Drag definition
72	net_thrust_clb	Fn_clb	Mach_clb,rho_clb,FNslst	engine
73	max_climb_factor	Kmcl		engine
74	climb_rate_1	vz_clb	mass_clb,Mach_clb,Fn_clb,Kmcl,lod_clb,vsnd_clb,g_clb,ne	Performance
75	top_of_climb_mass_cth	mass_cth	MTOW	weight
76	non_stand_atmos_cth	Pamb_cth,Tamb_cth	disa_cth,alt_cth	atmosphere
77	air_density_cth	rho_cth	Pamb_cth,Tamb_cth	atmosphere
78	sound_velocity_cth	vsnd_cth	Tamb_cth	atmosphere
79	gravity_acc_cth	g_cth	alt_cth	Earth
80	level_flight_cth	cz_cth	mass_cth,g_cth,Mach_cth,Pamb_cth,Aref	performance
81	pressure_drag_cth	cxc_cth	Mach_cth,Mchar,Kcxp	aerodynamic
82	induced_drag_cth	cxi_cth	cz_cth,ar,Kind	aerodynamic
83	friction_drag_cth	cx0_cth	cz_cth,Mach_cth,Pamb_cth,Tamb_cth,wAwing,wAht,wAvt,wAfus,wAnac,lfus,Aref,Lref,ne,Kcx0	aerodynamic
84	drag_factor_cth	cx_cth	cx0_cth,cxi_cth,cxc_cth	aerodynamic
85	lift_to_drag_cth	lod_cth	cz_cth,cx_cth	Lift over Drag definition
86	net_thrust_cth	Fn_cth	Mach_cth,rho_cth,FNslst	engine
87	cruise_thrust_1	kfn_cth	mass_cth,Fn_cth,Kmcr,lod_cth,g_cth,ne	Performance
88	gravity_acc_app	g_app	alt_app	Earth
89	app_speed_1	vapp	LDW,czmax_LD,Aref,g_app,kvs_LD	Performance
90	non_stand_atmos_to	Pamb_to,Tamb_to	disa_to,alt_to	atmosphere
91	air_density_to	rho_to	Pamb_to,Tamb_to	atmosphere
92	gravity_acc_to	g_to	alt_to	Earth
93	Mach_stall_to_	Mach_stall_to	MTOW,Aref,czmax_TO,Pamb_to,g_to	performance
94	secured_Mach_to	Mach_to	kvs_TO,Mach_stall_t	performance

			o	
95	net_thrust_to	Fn_to	Mach_to,rho_to,FNslst	engine
96	max_take_off_factor	Kmto		engine
97	tofl_1	tofl	Fn_to,Kmto,MTOW,czmax_TO,rho_to,ne,Aref,kvs_TO	performance

V. Additional Figures

i. USMAC test case: Case1

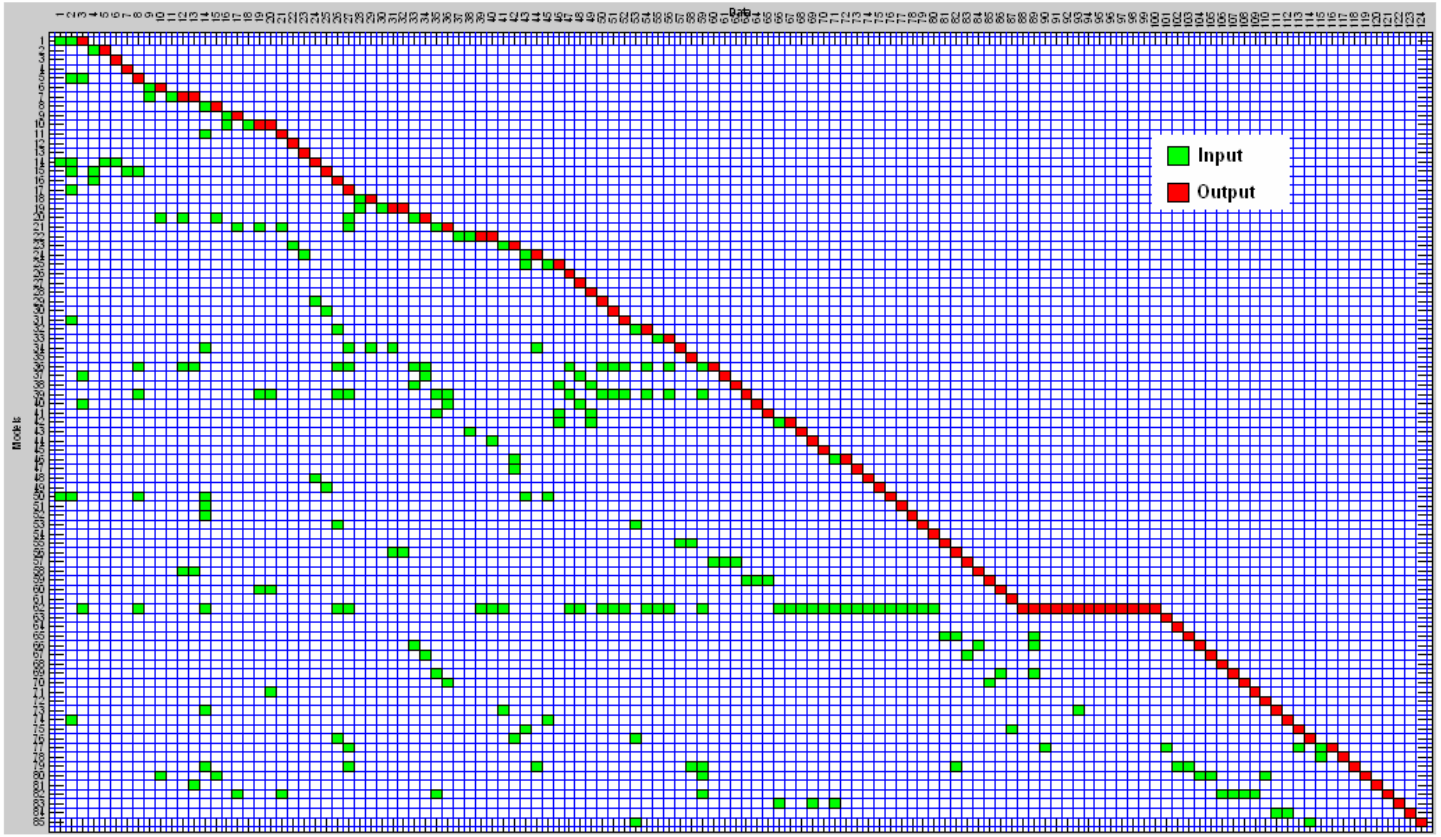


Figure A- 1: Incidence matrix for Case-1 (X and Y labels given in Figure A- 2) of USMAC test case

Y-AXIS LABELS (MODELS)					
1	aspect_ratio_	31	wing_wetted_area_	61	Cz_max_LD_factor_
2	fin_lever_arm_	32	fus_wetted_area_	62	scc_test_smac_tr
3	fin_volume_factor_	33	nac_wetted_area_	63	Kvs_Landing
4	tail_volume_factor_	34	Mach_stall_to_	64	max_take_off_factor
5	reference_length_	35	Kvs_Take_Off	65	net_thrust_to
6	gravity_acc_cth	36	friction_drag_cth	66	net_thrust_cth
7	non_stand_atmos_cth	37	induced_drag_cth	67	lift_to_drag_cth
8	top_of_climb_mass_cth	38	pressure_drag_cth	68	max_climb_factor
9	gravity_acc_clb	39	friction_drag_clb	69	net_thrust_clb
10	non_stand_atmos_clb	40	induced_drag_clb	70	lift_to_drag_clb
11	top_of_climb_mass_clb	41	pressure_drag_clb	71	sound_velocity_clb
12	one_pax_weight_	42	pressure_drag_crz	72	max_cruise_factor
13	Cz_max_TO_factor_	43	gravity_acc_crz	73	modified_take_off_weight__tr
14	fin_size_	44	sound_velocity_crz	74	wing_fuel
15	tail_size_	45	MWE_factor_	75	Cz_max_LD_
16	modified_tail_lever_arm__tr	46	operator_item_mass_	76	modified_fuselage_length__tr
17	reference_area_	47	furnishing_mass_	77	app_speed_1
18	gravity_acc_to	48	fin_mass_	78	modified_gravity_acc_app_tr
19	non_stand_atmos_to	49	tail_mass_	79	tofl_1
20	level_flight_cth	50	wing_mass_	80	cruise_thrust_1
21	level_flight_clb	51	system_mass_	81	sound_velocity_cth
22	non_stand_atmos_crz	52	landing_gear_mass_	82	climb_rate_1
23	modified_Payload__tr	53	fuselage_mass_	83	time_crz
24	Cz_max_TO_	54	sfc_factor	84	fus_fuel_ratio_
25	ref_mach_number_	55	secured_Mach_to	85	modified_fuselage_diameter_
26	fric_drag_factor_	56	air_density_to		
27	ind_drag_factor_	57	drag_factor_cth		
28	press_drag_factor_	58	air_density_cth		
29	fin_wetted_area_	59	drag_factor_clb		
30	tail_wetted_area_	60	air_density_clb		

X-AXIS LABELS (VARIABLES)							
1	span	31	Pamb_to	61	cxi_cth	91	MWE
2	Awing	32	Tamb_to	62	cxc_cth	92	Mprop
3	ar	33	Mach_cth	63	cx0_clb	93	OWE
4	Leh	34	cz_cth	64	cxi_clb	94	cx0_crz
5	Lev	35	Mach_clb	65	cxc_clb	95	cx_crz
6	Vvt	36	cz_clb	66	Mach_crz	96	cxi_crz
7	Vht	37	disa_crz	67	cxc_crz	97	cz_crz
8	Lref	38	alt_crz	68	g_crz	98	lod_crz
9	alt_cth	39	Pamb_crz	69	vsnd_crz	99	mass_crz
10	g_cth	40	Tamb_crz	70	Kmwe	100	sfc
11	disa_cth	41	PL	71	RA	101	kvs_LD
12	Pamb_cth	42	Npax	72	Mop	102	Kmto
13	Tamb_cth	43	phi	73	Mfurn	103	Fn_to
14	MTOW	44	czmax_TO	74	Mvt	104	Fn_cth
15	mass_cth	45	tuc	75	Mht	105	lod_cth
16	alt_clb	46	Mchar	76	Mwing	106	Kmcl
17	g_clb	47	Kcx0	77	Msys	107	Fn_clb
18	disa_clb	48	Kind	78	Mgear	108	lod_clb
19	Pamb_clb	49	Kcxp	79	Mfus	109	vsnd_clb
20	Tamb_clb	50	wAvt	80	Ksfc	110	Kmcr
21	mass_clb	51	wAht	81	Mach_to	111	Fuel
22	Wpax	52	wAwing	82	rho_to	112	Fwing
23	KczmaxTO	53	dfus	83	cx_cth	113	czmax_LD
24	Avt	54	wAfus	84	rho_cth	114	NpaxFront
25	Aht	55	dnac	85	cx_clb	115	g_app
26	lfus	56	wAnac	86	rho_clb	116	vapp
27	Aref	57	Mach_stall_to	87	KczmaxLD	117	alt_app
28	alt_to	58	kvs_TO	88	BPR	118	tofl
29	g_to	59	ne	89	FNslst	119	kfn_cth
30	disa_to	60	cx0_cth	90	LDW	120	vsnd_cth
						121	vz_clb
						122	RA_time
						123	Kff
						124	Naisle

Figure A- 2: X and Y axis labels for the incidence matrix in Figure A- 1

Variable flow model for SCC: 1

Feedback number: 3, Number of modified models: 6

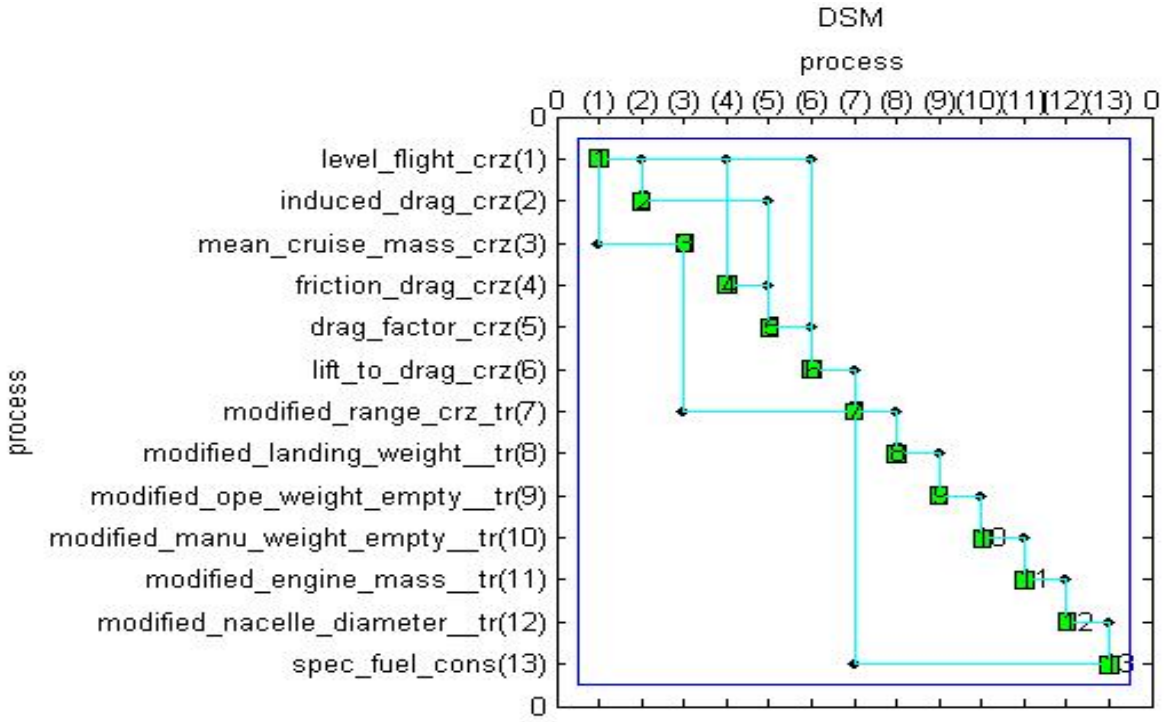


Figure A- 3: Design Structure Matrix of the SCC (Variable flow model-1) for Case-1 of USMAC test case

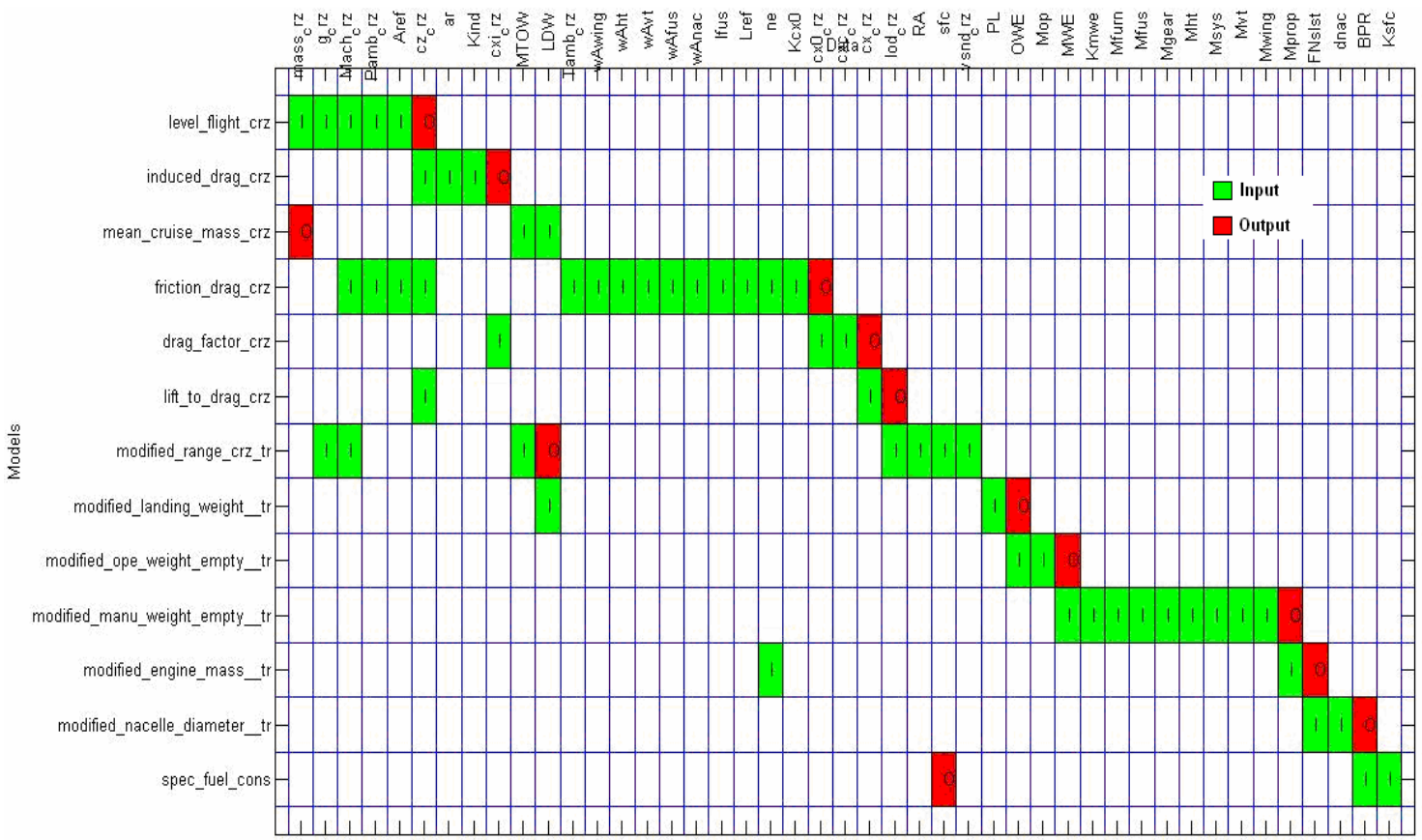


Figure A- 4: Incidence Matrix of the SCC (Variable flow model-1) for Case-1 of USMAC test case

Variable flow model for SCC: 2

Feedback number: 5, Number of modified models: 11

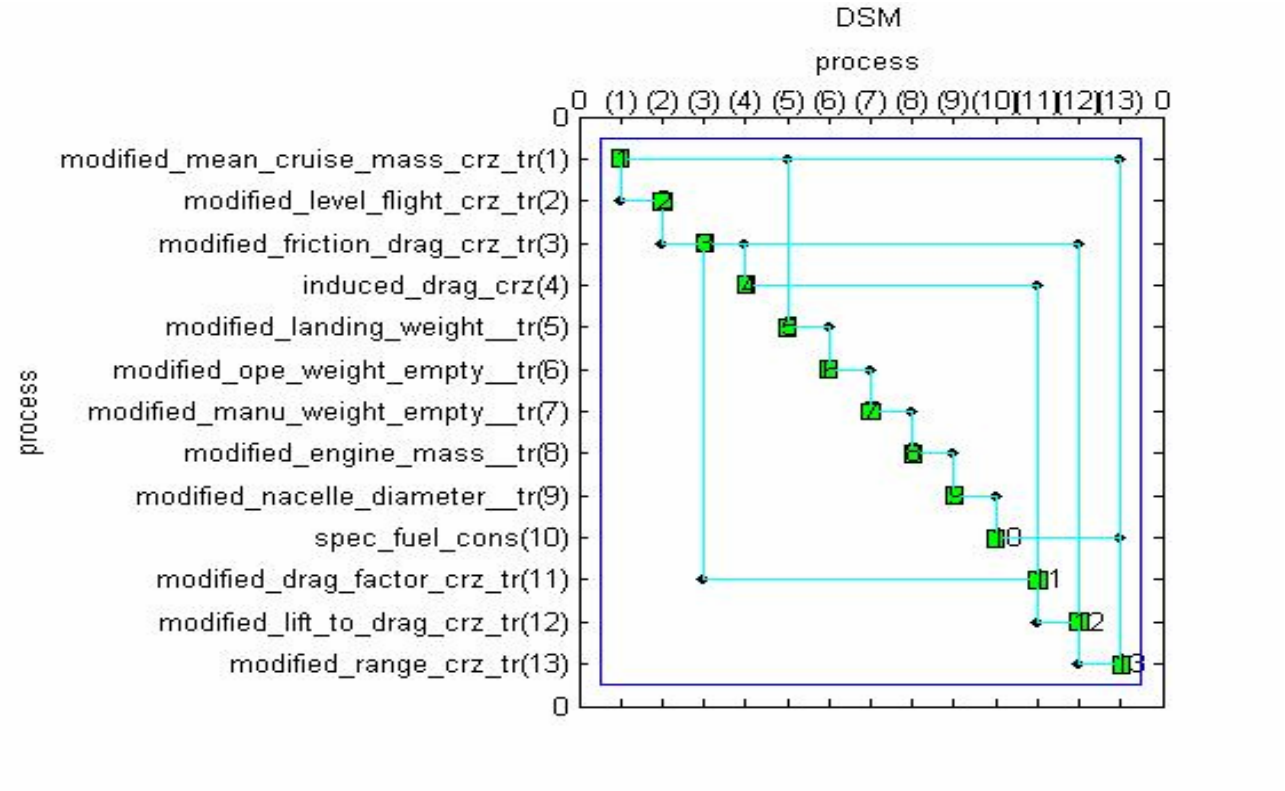


Figure A- 5: Design Structure Matrix of the SCC (Variable flow model-2) for Case-1 of USMAC test case

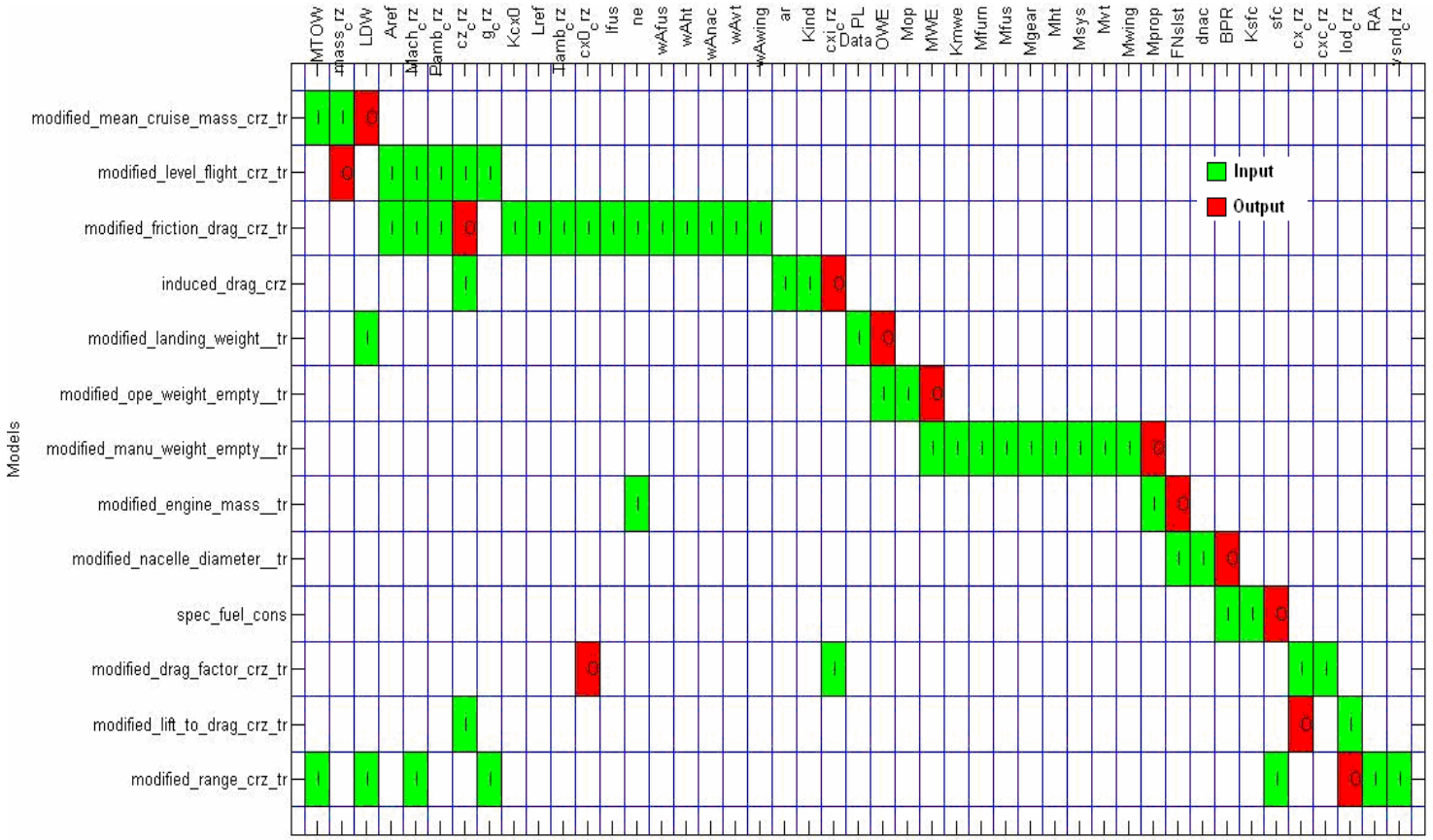


Figure A- 6: Incidence Matrix of the SCC (Variable flow model-2) for Case-1 of USMAC test case

Variable flow model for SCC: 3

Feedback number: 6, Number of modified models: 3

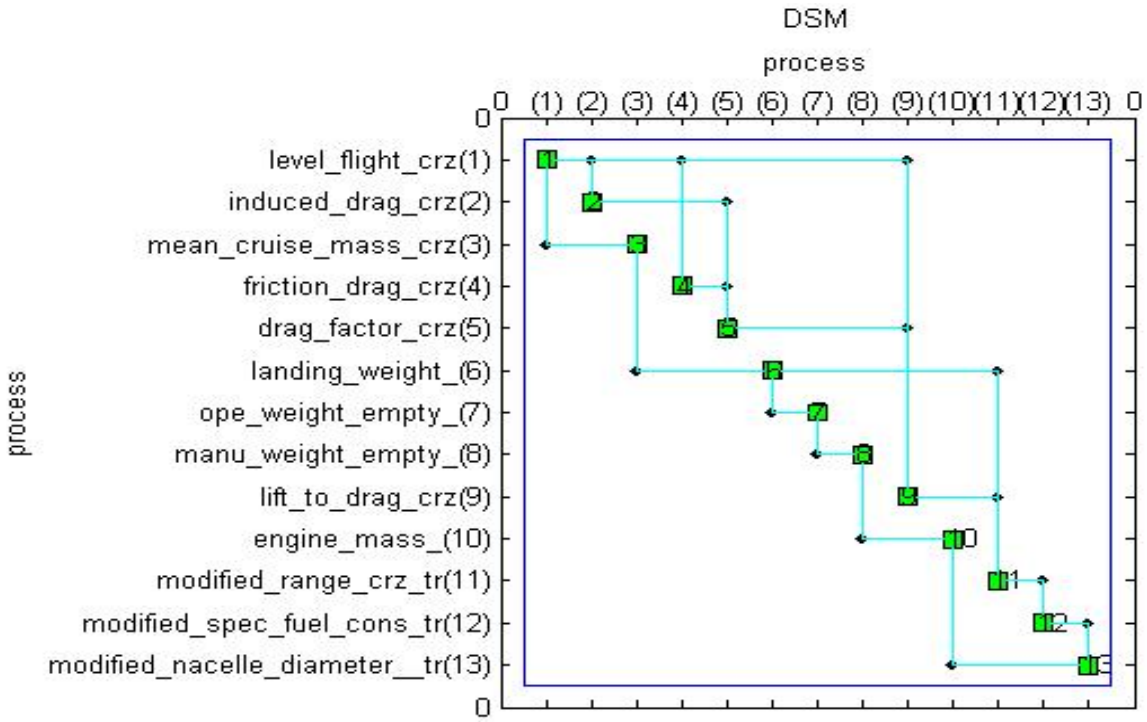


Figure A- 7: Design Structure Matrix of the SCC (Variable flow model-3) for Case-1 of USMAC test case

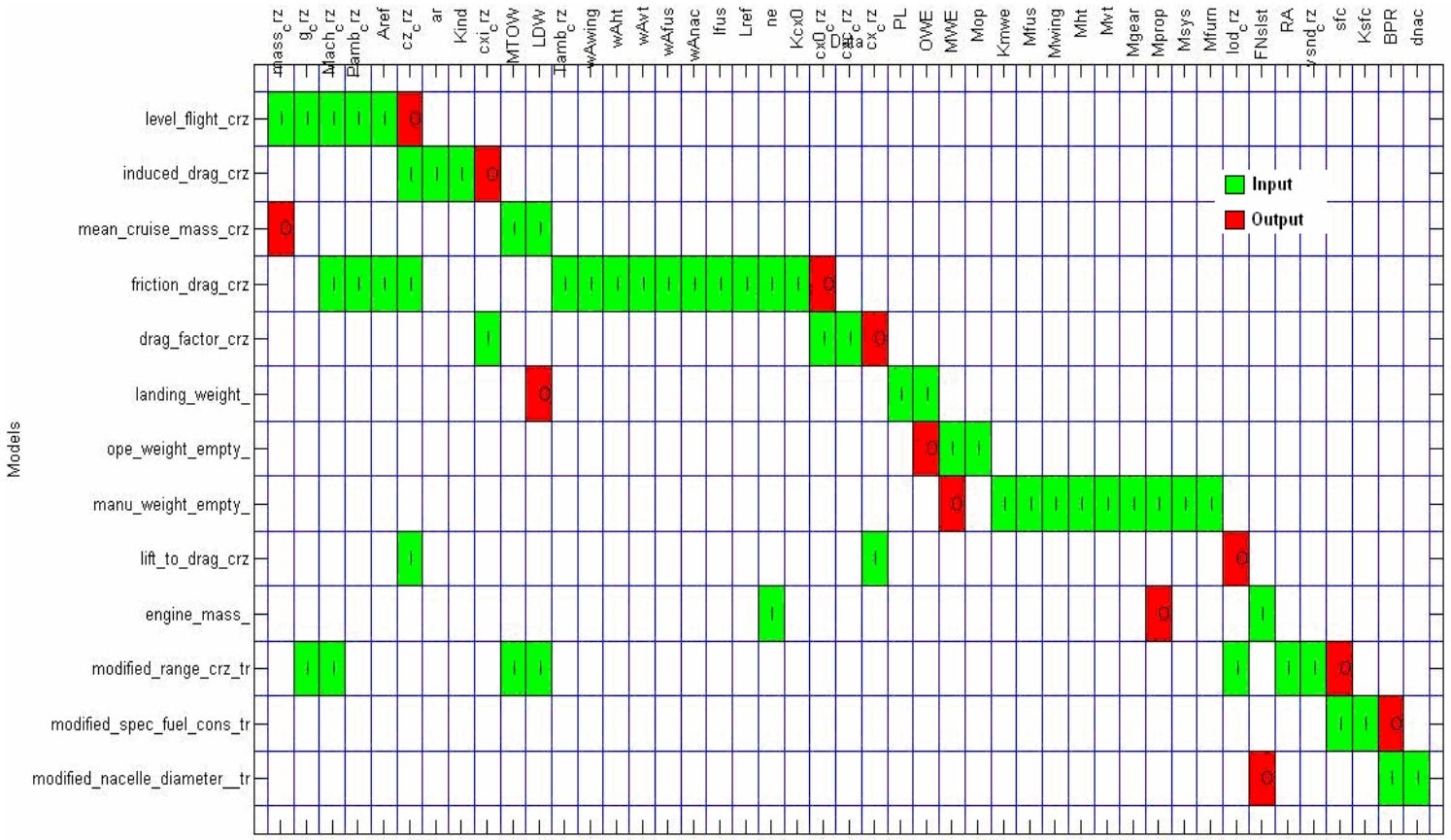


Figure A- 8: Incidence Matrix of the SCC (Variable flow model-3) for Case-1 of USMAC test case

Variable flow model for SCC: 4

Feedback number: 5, Number of modified models: 9

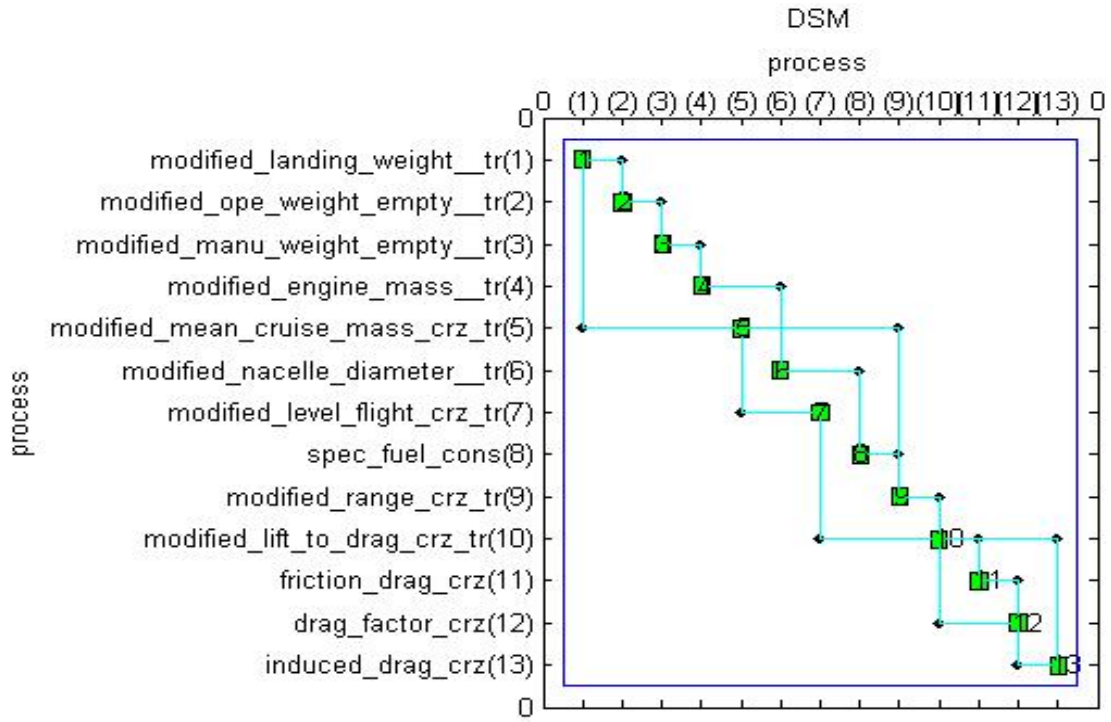


Figure A- 9: Design Structure Matrix of the SCC (Variable flow model-4) for Case-1 of USMAC test case

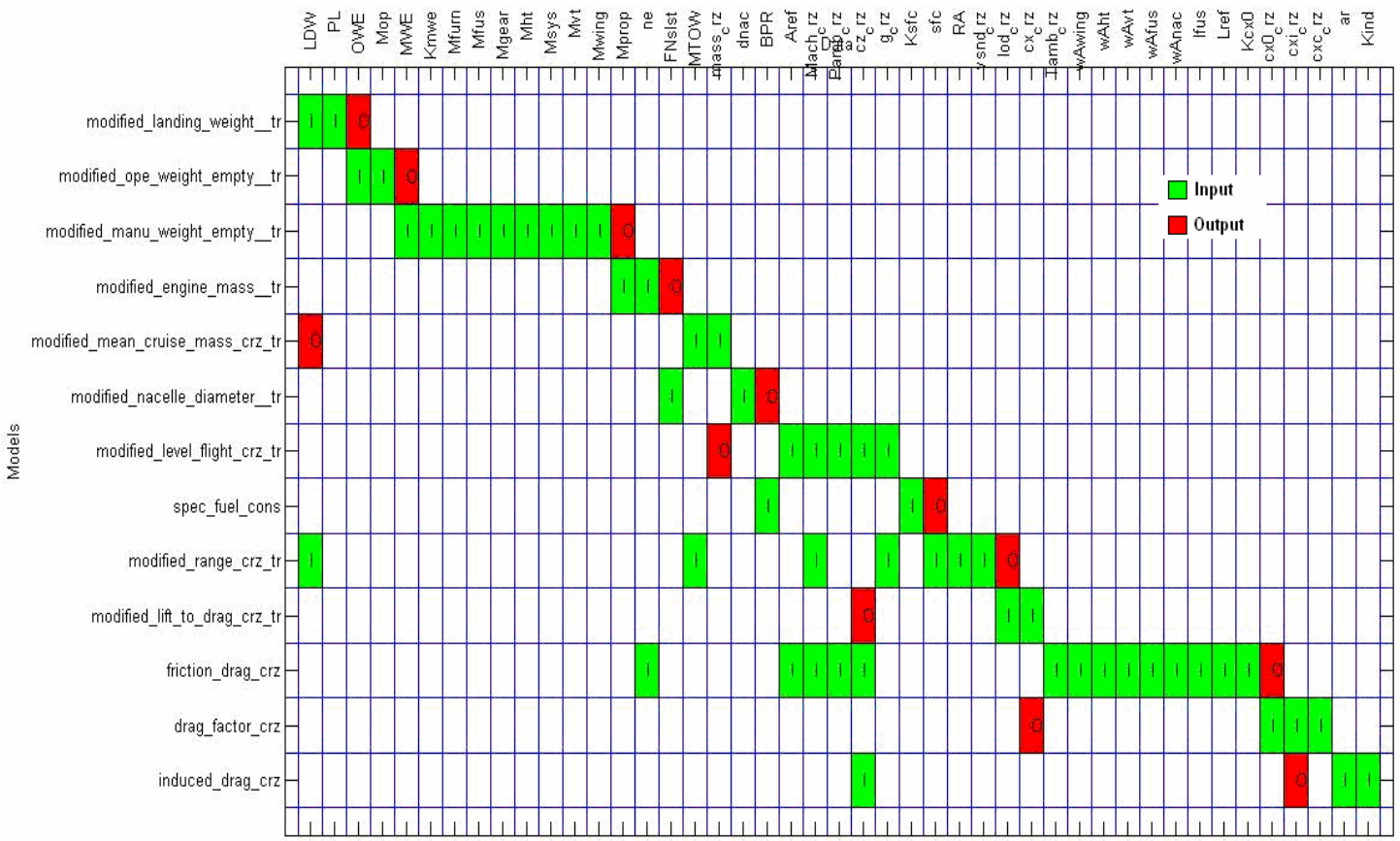


Figure A- 10: Incidence Matrix of the SCC (Variable flow model-4) for Case-1 of USMAC test case

Variable flow model for SCC: 5

Feedback number: 8, Number of modified models: 11

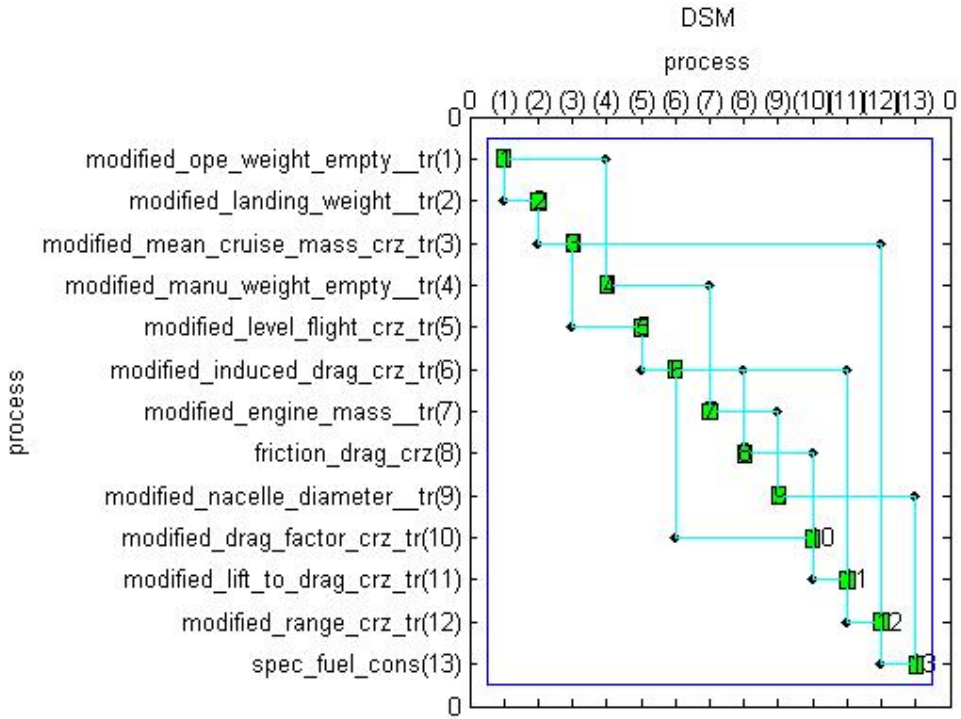


Figure A- 11 Design Structure Matrix of the SCC (Variable flow model-5) for Case-1 of USMAC test case

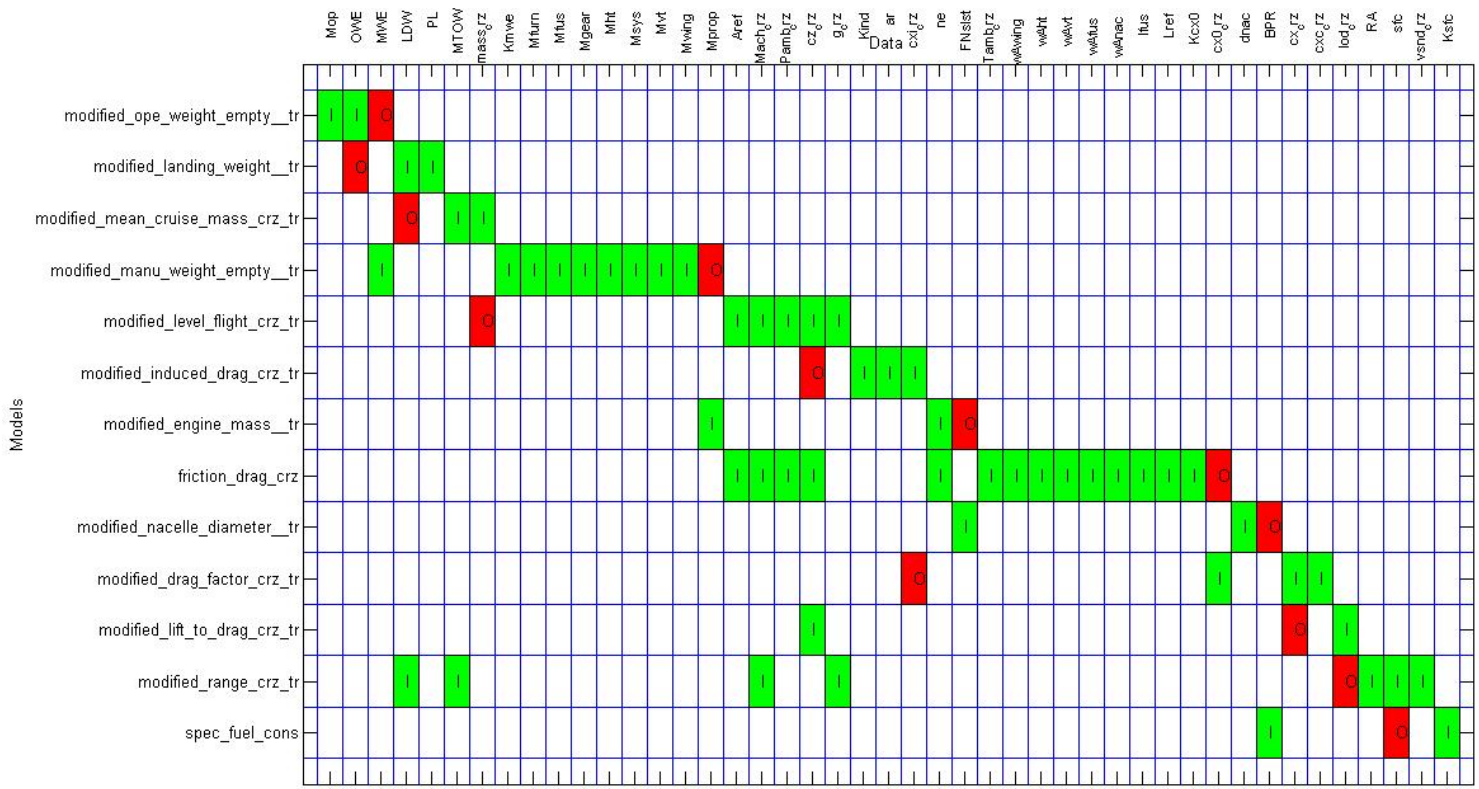


Figure A- 12 Incidence Matrix of the SCC (Variable flow model-5) for Case-1 of USMAC test case

ii. USMAC test case: Case2

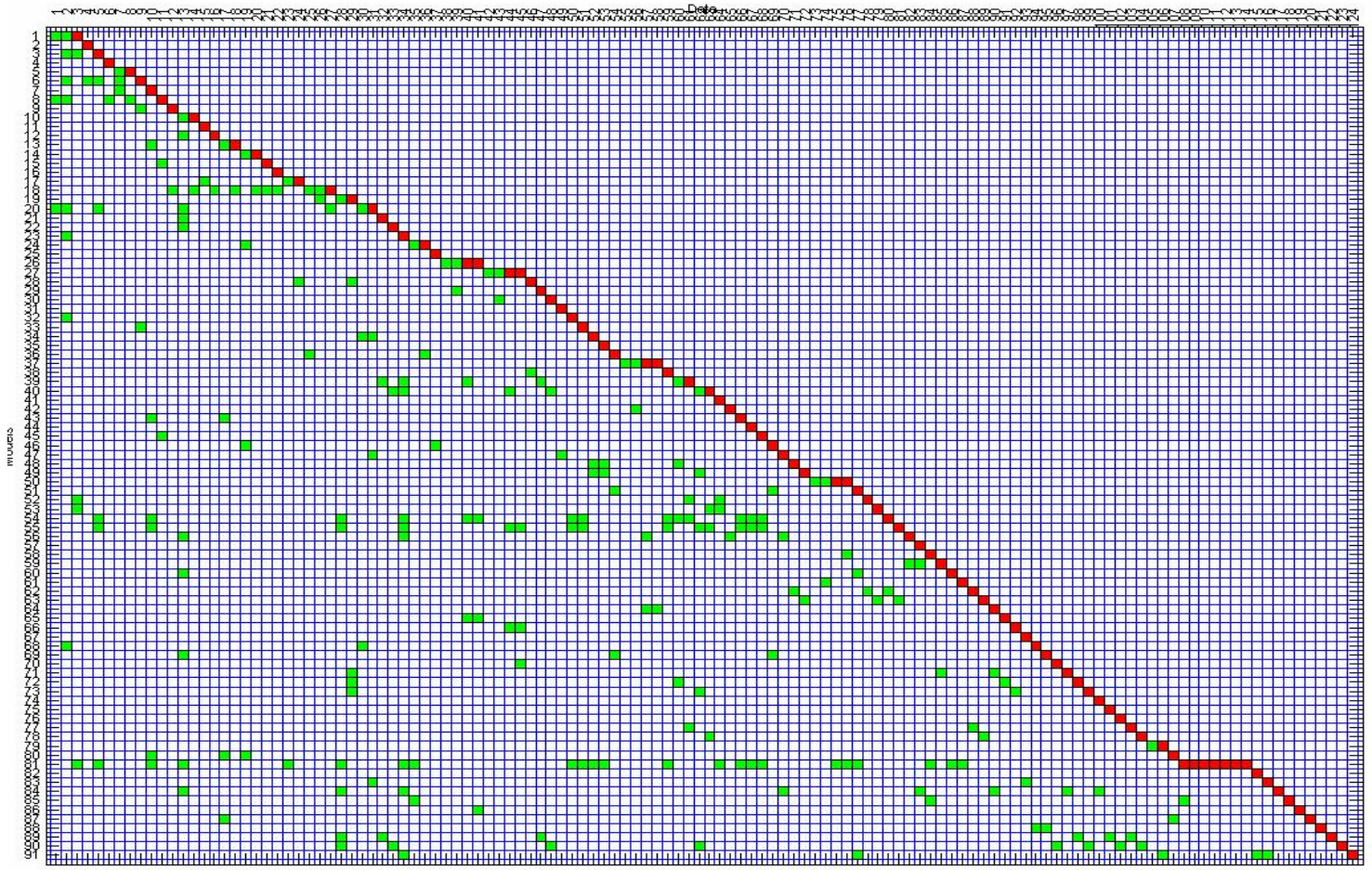


Figure A- 13: Incidence matrix for Case-2 (X and Y labels are given in Figure A- 14) of USMAC test case

Y-LABELS(MODELS)					
1	aspect_ratio_	31	Cz_max_TO_factor_	61	gravity_acc_crz
2	tail_volume_factor_	32	wing_wetted_area_	62	drag_factor_cth
3	reference_length_	33	tail_wetted_area_	63	drag_factor_clb
4	fin_volume_factor_	34	ref_mach_number_	64	air_density_to
5	fin_lever_arm_	35	press_drag_factor_	65	air_density_cth
6	tail_size_	36	ope_weight_empty_	66	air_density_clb
7	modified_tail_lever_arm__tr	37	non_stand_atmos_to	67	Cz_max_LD_factor_
8	fin_size_	38	nac_wetted_area_	68	wing_fuel
9	tail_mass_	39	level_flight_cth	69	modified_take_off_weight__tr
10	system_mass_	40	level_flight_clb	70	sound_velocity_clb
11	sfc_factor	41	ind_drag_factor_	71	net_thrust_to
12	landing_gear_mass_	42	gravity_acc_to	72	net_thrust_cth
13	fuselage_mass_	43	fus_wetted_area_	73	net_thrust_clb
14	furnishing_mass_	44	fric_drag_factor_	74	max_take_off_factor
15	fin_mass_	45	fin_wetted_area_	75	max_cruise_factor
16	MWE_factor_	46	Payload_	76	max_climb_factor
17	modified_spec_fuel_cons_tr	47	Cz_max_TO_	77	lift_to_drag_cth
18	modified_manu_weight_empty__tr	48	pressure_drag_cth	78	lift_to_drag_clb
19	modified_engine_mass__tr	49	pressure_drag_clb	79	gravity_acc_app
20	modified_wing_mass__tr	50	non_stand_atmos_crz	80	modified_fuselage_length__tr
21	top_of_climb_mass_cth	51	landing_weight_	81	scc_1_tr
22	top_of_climb_mass_clb	52	induced_drag_cth	82	Kvs_Landing
23	reference_area_	53	induced_drag_clb	83	Cz_max_LD_
24	operator_item_mass_	54	friction_drag_cth	84	tofl_1
25	one_pax_weight_	55	friction_drag_clb	85	time_crz
26	non_stand_atmos_cth	56	Mach_stall_to_	86	sound_velocity_cth
27	non_stand_atmos_clb	57	Kvs_Take_Off	87	modified_fuselage_diameter__tr
28	nacelle_diameter_	58	sound_velocity_crz	88	fus_fuel_ratio_
29	gravity_acc_cth	59	secured_Mach_to	89	cruise_thrust_1
30	gravity_acc_clb	60	mean_cruise_mass_crz	90	climb_rate_1
				91	app_speed_1

X- LABELS(VARIABLES)							
1	span	31	phi	61	cz_cth	91	rho_cth
2	Awing	32	mass_cth	62	Mach_clb	92	rho_clb
3	ar	33	mass_clb	63	cz_clb	93	KczmaxLD
4	Vht	34	Aref	64	Kind	94	Fwing
5	Lref	35	RA	65	g_to	95	Fuel
6	Vvt	36	Mop	66	wAfus	96	vsnd_clb
7	Leh	37	Wpax	67	Kcx0	97	Fn_to
8	Lev	38	disa_cth	68	wAvt	98	Fn_cth
9	Aht	39	alt_cth	69	PL	99	Fn_clb
10	lfus	40	Pamb_cth	70	czmax_TO	100	Kmto
11	Avt	41	Tamb_cth	71	cxc_cth	101	Kmcr
12	Mht	42	disa_clb	72	cxc_clb	102	Kmcl
13	MTOW	43	alt_clb	73	disa_crz	103	lod_cth
14	Msys	44	Pamb_clb	74	alt_crz	104	lod_clb
15	Ksfc	45	Tamb_clb	75	Pamb_crz	105	alt_app
16	Mgear	46	dnac	76	Tamb_crz	106	g_app
17	dfus	47	g_cth	77	LDW	107	NpaxFront
18	Mfus	48	g_clb	78	cxi_cth	108	Mach_crz
19	Npax	49	KczmaxTO	79	cxi_clb	109	cx0_crz
20	Mfurn	50	wAwing	80	cx0_cth	110	cx_crz
21	Mvt	51	wAht	81	cx0_clb	111	cxc_crz
22	Kmwe	52	Mchar	82	Mach_stall_to	112	cxi_crz
23	sfc	53	Kcxp	83	kvs_TO	113	cz_crz
24	BPR	54	OWE	84	vsnd_crz	114	lod_crz
25	MWE	55	disa_to	85	Mach_to	115	kvs_LD
26	Mprop	56	alt_to	86	mass_crz	116	czmax_LD
27	Mwing	57	Pamb_to	87	g_crz	117	tofl
28	ne	58	Tamb_to	88	cx_cth	118	RA_time
29	FNslst	59	wAnac	89	cx_clb	119	vsnd_cth
30	tuc	60	Mach_cth	90	rho_to	120	Naisle
						121	Kff
						122	kfn_cth
						123	vz_clb
						124	vapp

Figure A- 14: X and Y axis labels for the incidence matrix in Figure A- 13

Variable flow model for SCC: 1

Feedback number: 1, Number of modified models: 4

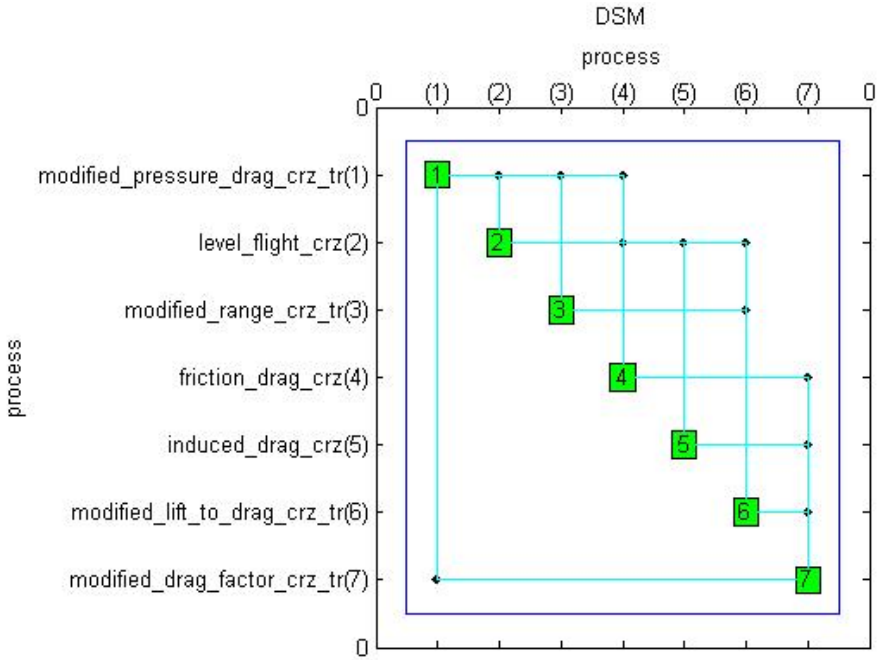


Figure A- 15: Design Structure Matrix of the SCC (Variable flow model-1) for Case-2 of USMAC test case

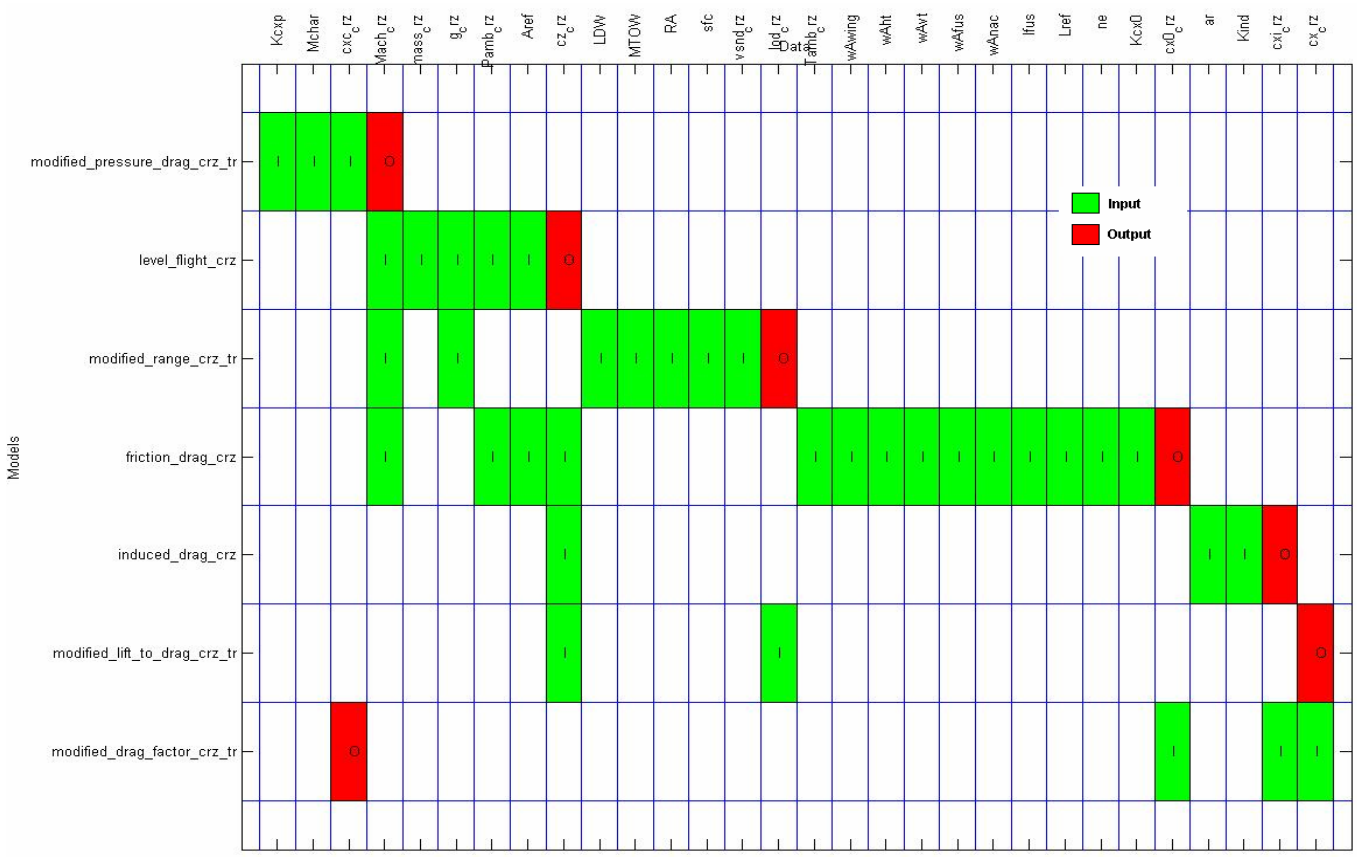


Figure A- 16: Incidence Matrix of the SCC (Variable flow model-1) for Case-2 of USMAC test case

Variable flow model for SCC: 2

Feedback number: 1, Number of modified models: 5

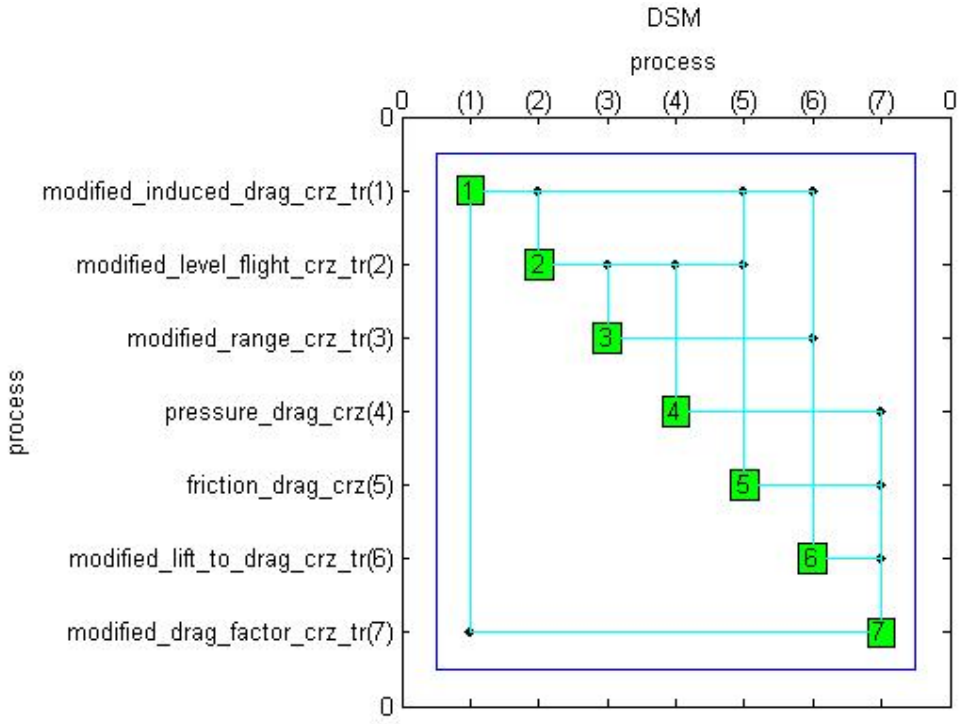


Figure A- 17: Design Structure Matrix of the SCC (Variable flow model-2) for Case-2 of USMAC test case

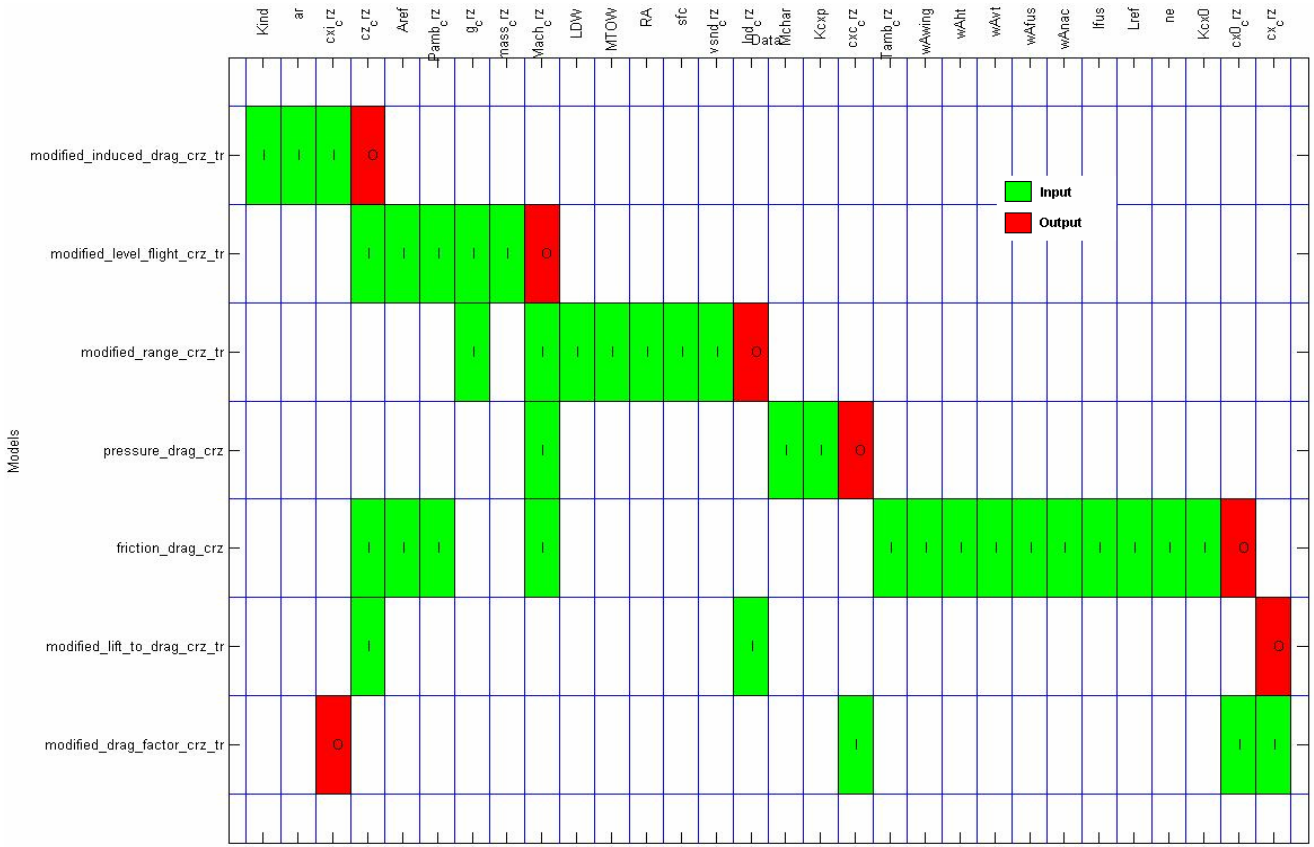


Figure A- 18: Incidence Matrix of the SCC (Variable flow model-2) for Case-2 of USMAC test case

Variable flow model for SCC: 3

Feedback number:2, Number of modified models: 3

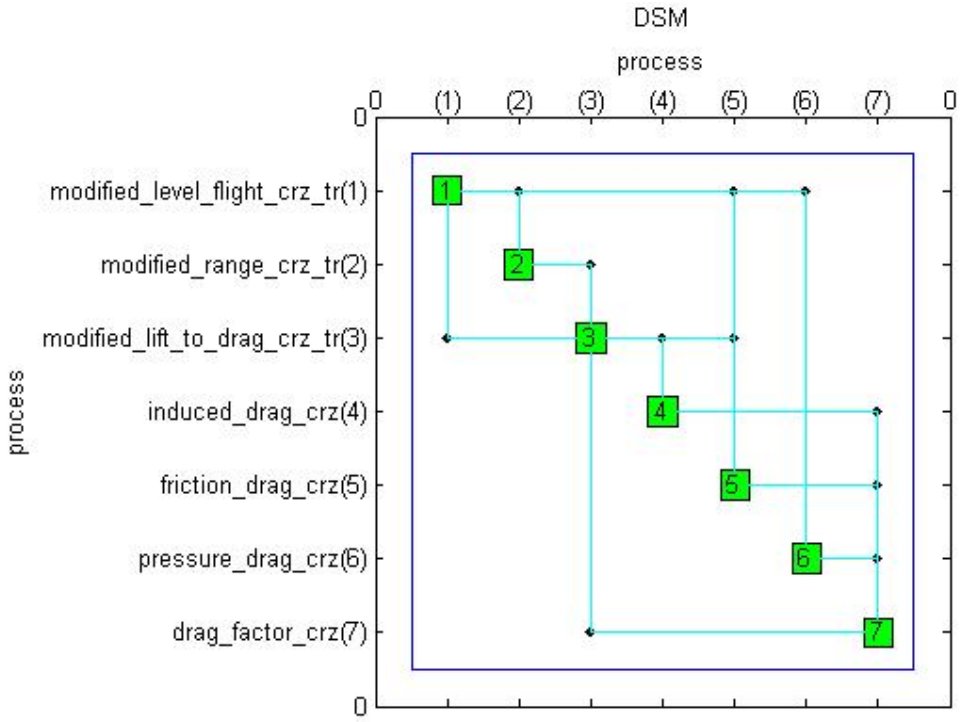


Figure A- 19: Design Structure Matrix of the SCC (Variable flow model-3) for Case-2 of USMAC test case

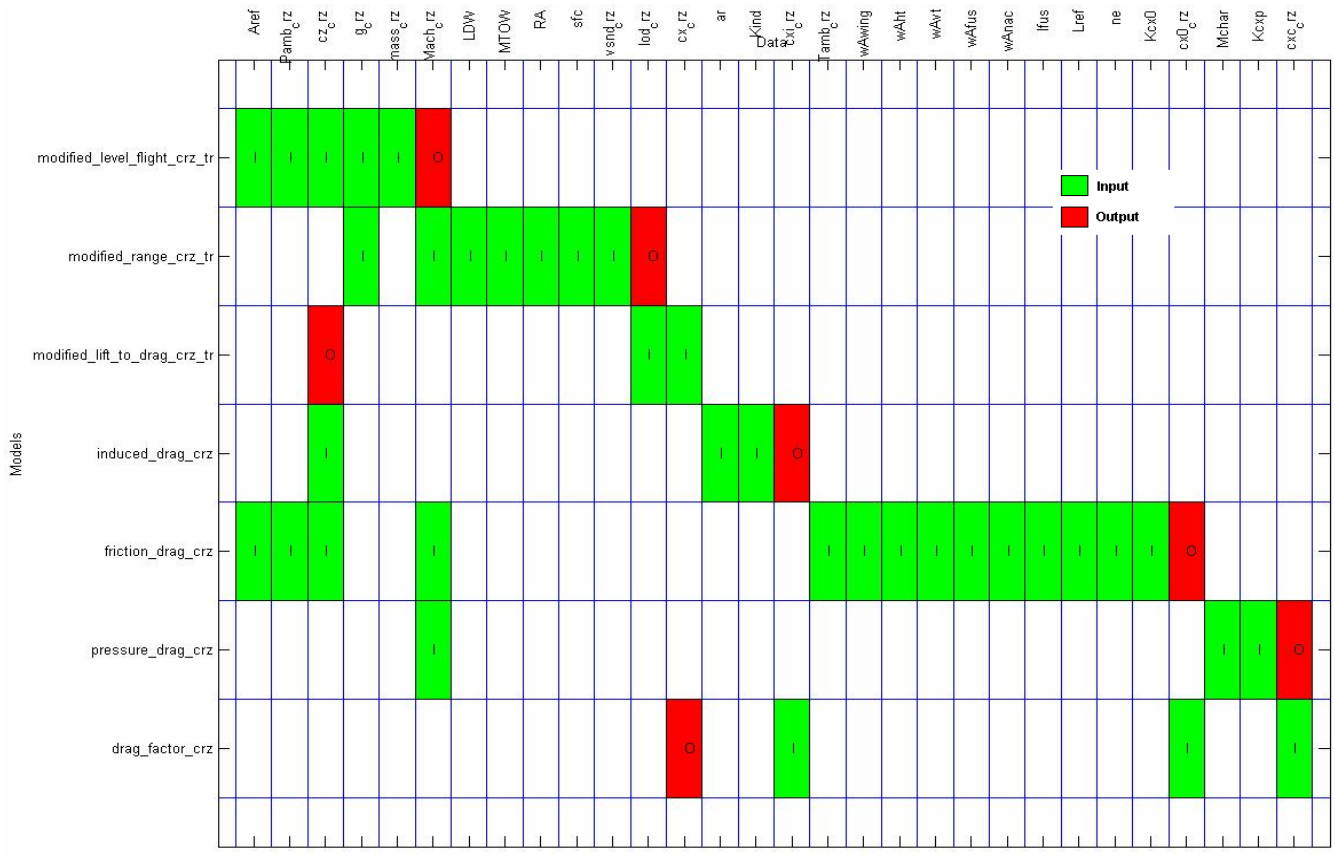


Figure A- 20: Incidence Matrix of the SCC (Variable flow model-3) for Case-2 of USMAC test case

Variable flow model for SCC: 4

Feedback number:2, Number of modified models: 5

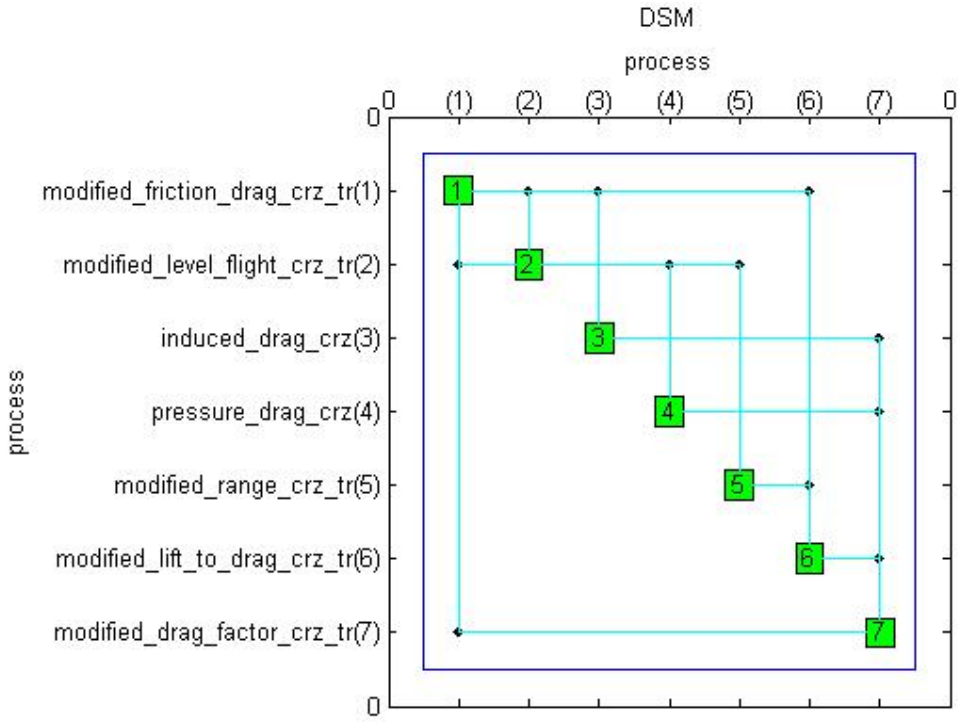


Figure A- 21 Design Structure Matrix of the SCC (Variable flow model-4) for Case-2 of USMAC test case

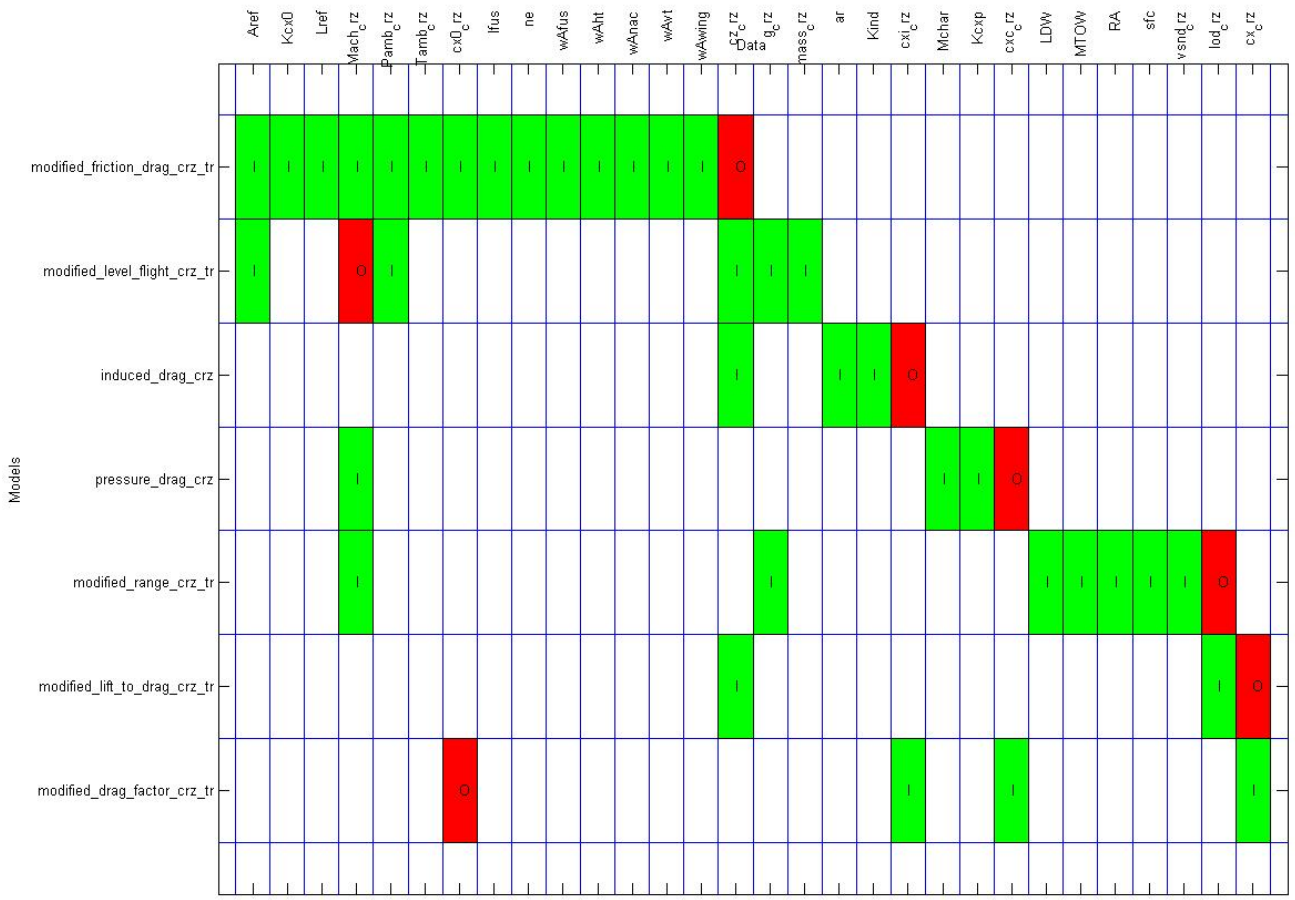
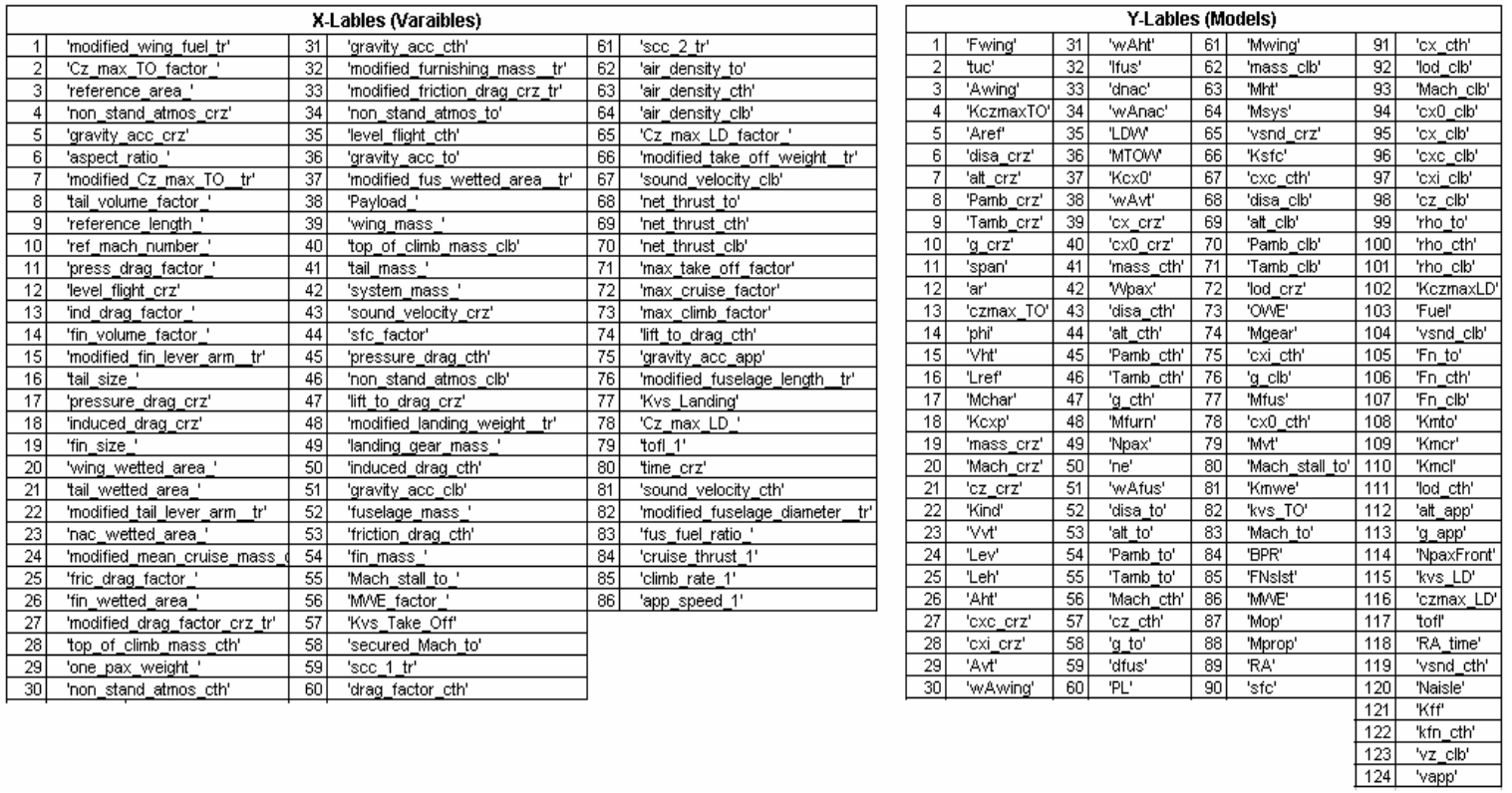


Figure A- 22 Incidence Matrix of the SCC (Variable flow model-4) for Case-2 of USMAC test case

iii. USMAC test case: Case3

Figure A- 23: Incidence matrix for Case-3 (X and Y labels are given in Figure A- 24) of USMAC test case



X-Lables (Varaibles)					
1	'modified_wing_fuel_tr'	31	'gravity_acc_cth'	61	'scc_2_tr'
2	'Cz_max_TO_factor_'	32	'modified_furnishing_mass__tr'	62	'air_density_to'
3	'reference_area_'	33	'modified_friction_drag_crz_tr'	63	'air_density_cth'
4	'non_stand_atmos_crz'	34	'non_stand_atmos_to'	64	'air_density_clb'
5	'gravity_acc_crz'	35	'level_flight_cth'	65	'Cz_max_LD_factor_'
6	'aspect_ratio_'	36	'gravity_acc_to'	66	'modified_take_off_weight__tr'
7	'modified_Cz_max_TO__tr'	37	'modified_fus_wetted_area__tr'	67	'sound_velocity_clb'
8	'tail_volume_factor_'	38	'Payload_'	68	'net_thrust_to'
9	'reference_length_'	39	'wing_mass_'	69	'net_thrust_cth'
10	'ref_mach_number_'	40	'top_of_climb_mass_clb'	70	'net_thrust_clb'
11	'press_drag_factor_'	41	'tail_mass_'	71	'max_take_off_factor'
12	'level_flight_crz'	42	'system_mass_'	72	'max_cruise_factor'
13	'ind_drag_factor_'	43	'sound_velocity_crz'	73	'max_climb_factor'
14	'fin_volume_factor_'	44	'sfc_factor'	74	'lift_to_drag_cth'
15	'modified_fin_lever_arm__tr'	45	'pressure_drag_cth'	75	'gravity_acc_app'
16	'tail_size_'	46	'non_stand_atmos_clb'	76	'modified_fuselage_length__tr'
17	'pressure_drag_crz'	47	'lift_to_drag_crz'	77	'Kvs_Landing'
18	'induced_drag_crz'	48	'modified_landing_weight__tr'	78	'Cz_max_LD_'
19	'fin_size_'	49	'landing_gear_mass_'	79	'tofl_1'
20	'wing_wetted_area_'	50	'induced_drag_cth'	80	'time_crz'
21	'tail_wetted_area_'	51	'gravity_acc_clb'	81	'sound_velocity_cth'
22	'modified_tail_lever_arm__tr'	52	'fuselage_mass_'	82	'modified_fuselage_diameter__tr'
23	'nac_wetted_area_'	53	'friction_drag_cth'	83	'fus_fuel_ratio_'
24	'modified_mean_cruise_mass_(	54	'fin_mass_'	84	'cruise_thrust_1'
25	'fric_drag_factor_'	55	'Mach_stall_to_'	85	'climb_rate_1'
26	'fin_wetted_area_'	56	'MWE_factor_'	86	'app_speed_1'
27	'modified_drag_factor_crz_tr'	57	'Kvs_Take_Off'		
28	'top_of_climb_mass_cth'	58	'secured_Mach_to'		
29	'one_pax_weight_'	59	'scc_1_tr'		
30	'non_stand_atmos_cth'	60	'drag_factor_cth'		

Y-Lables (Models)							
1	'Fwing'	31	'wAht'	61	'Mwing'	91	'cx_cth'
2	'tuc'	32	'lfus'	62	'mass_clb'	92	'lod_clb'
3	'Awing'	33	'dnac'	63	'Mht'	93	'Mach_clb'
4	'KczmaxTO'	34	'wAnac'	64	'Msys'	94	'cx0_clb'
5	'Aref'	35	'LDW'	65	'vsnd_crz'	95	'cx_clb'
6	'disa_crz'	36	'MTOW'	66	'Ksfc'	96	'cxc_clb'
7	'alt_crz'	37	'Kcx0'	67	'cxc_cth'	97	'cxi_clb'
8	'Pamb_crz'	38	'wAvt'	68	'disa_clb'	98	'cz_clb'
9	'Tamb_crz'	39	'cx_crz'	69	'alt_clb'	99	'rho_to'
10	'g_crz'	40	'cx0_crz'	70	'Pamb_clb'	100	'rho_cth'
11	'span'	41	'mass_cth'	71	'Tamb_clb'	101	'rho_clb'
12	'ar'	42	'Wpax'	72	'lod_crz'	102	'KczmaxLD'
13	'czmax_TO'	43	'disa_cth'	73	'OWE'	103	'Fuel'
14	'phi'	44	'alt_cth'	74	'Mgear'	104	'vsnd_clb'
15	'Vht'	45	'Pamb_cth'	75	'cxi_cth'	105	'Fn_to'
16	'Lref'	46	'Tamb_cth'	76	'g_clb'	106	'Fn_cth'
17	'Mchar'	47	'g_cth'	77	'Mfus'	107	'Fn_clb'
18	'Kcxp'	48	'Mfurn'	78	'cx0_cth'	108	'Kmto'
19	'mass_crz'	49	'Npax'	79	'Mvt'	109	'Kmcr'
20	'Mach_crz'	50	'ne'	80	'Mach_stall_to'	110	'Kmcl'
21	'cz_crz'	51	'wAfus'	81	'Kmwe'	111	'lod_cth'
22	'Kind'	52	'disa_to'	82	'kvs_TO'	112	'alt_app'
23	'Vvt'	53	'alt_to'	83	'Mach_to'	113	'g_app'
24	'Lev'	54	'Pamb_to'	84	'BPR'	114	'NpaxFront'
25	'Leh'	55	'Tamb_to'	85	'FNslst'	115	'kvs_LD'
26	'Aht'	56	'Mach_cth'	86	'MWE'	116	'czmax_LD'
27	'cxc_crz'	57	'cz_cth'	87	'Mop'	117	'tofl'
28	'cxi_crz'	58	'g_to'	88	'Mprop'	118	'RA_time'
29	'Avt'	59	'dfus'	89	'RA'	119	'vsnd_cth'
30	'wAwing'	60	'PL'	90	'sfc'	120	'Naisle'
						121	'Kff'
						122	'kfn_cth'
						123	'vz_clb'
						124	'vapp'

Figure A- 24: X and Y axis labels for the incidence matrix in Figure A- 23

Variable flow model for first SCC: 1

Feedback number: 1, Number of modified models: 4

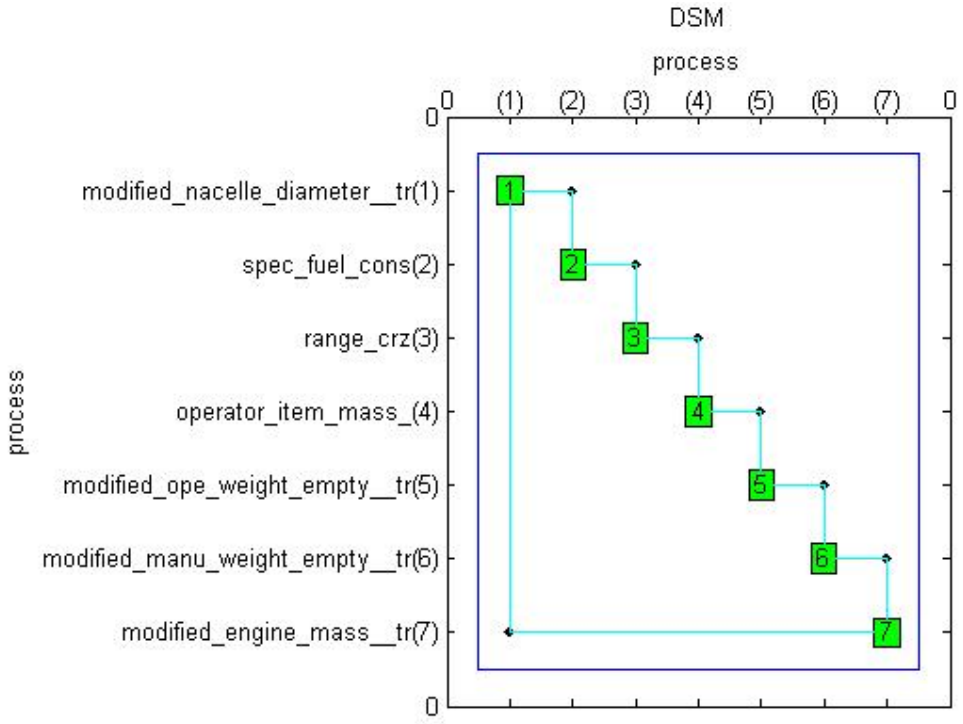


Figure A- 25: Design Structure Matrix of the first SCC (Variable flow model-1) for Case-3 of USMAC test case

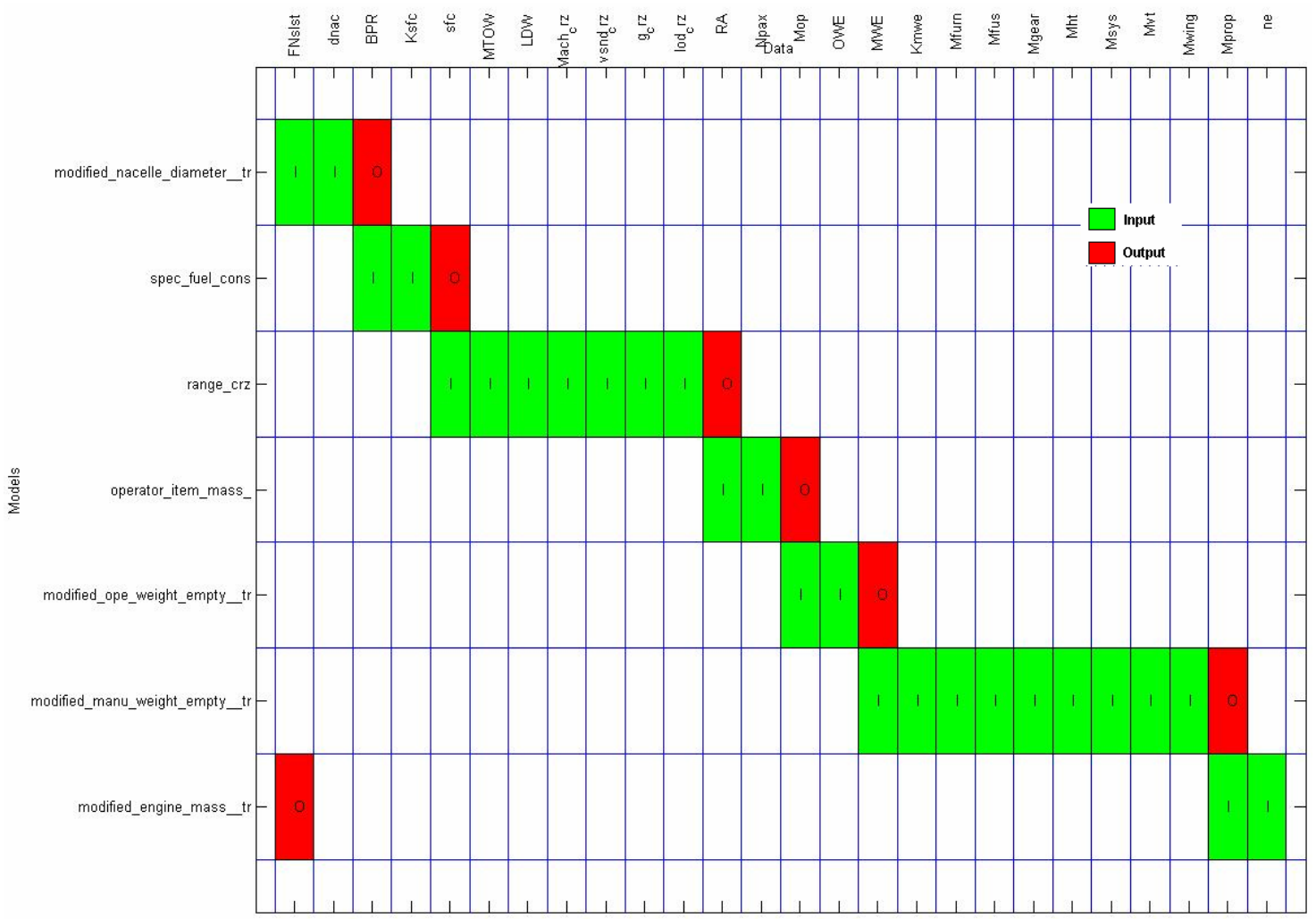


Figure A- 26: Incidence Matrix of the first SCC (Variable flow model-1) for Case-3 of USMAC test case

Variable flow model for first SCC: 2

Feedback number: 1, Number of modified models: 5

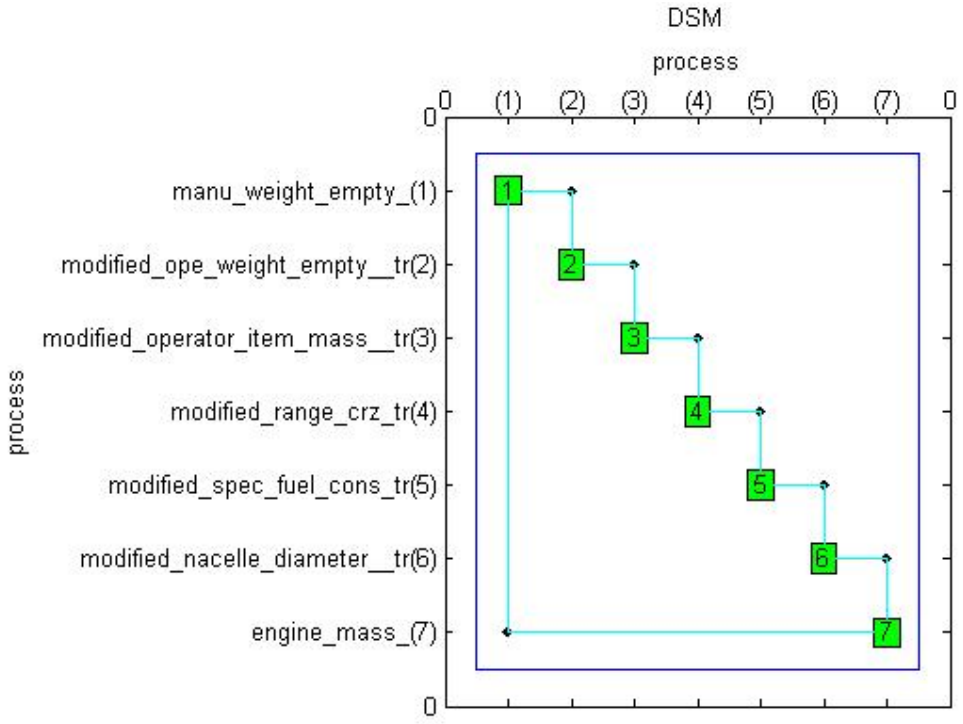


Figure A- 27: Design Structure Matrix of the first SCC (Variable flow model-2) for Case-3 of USMAC test case

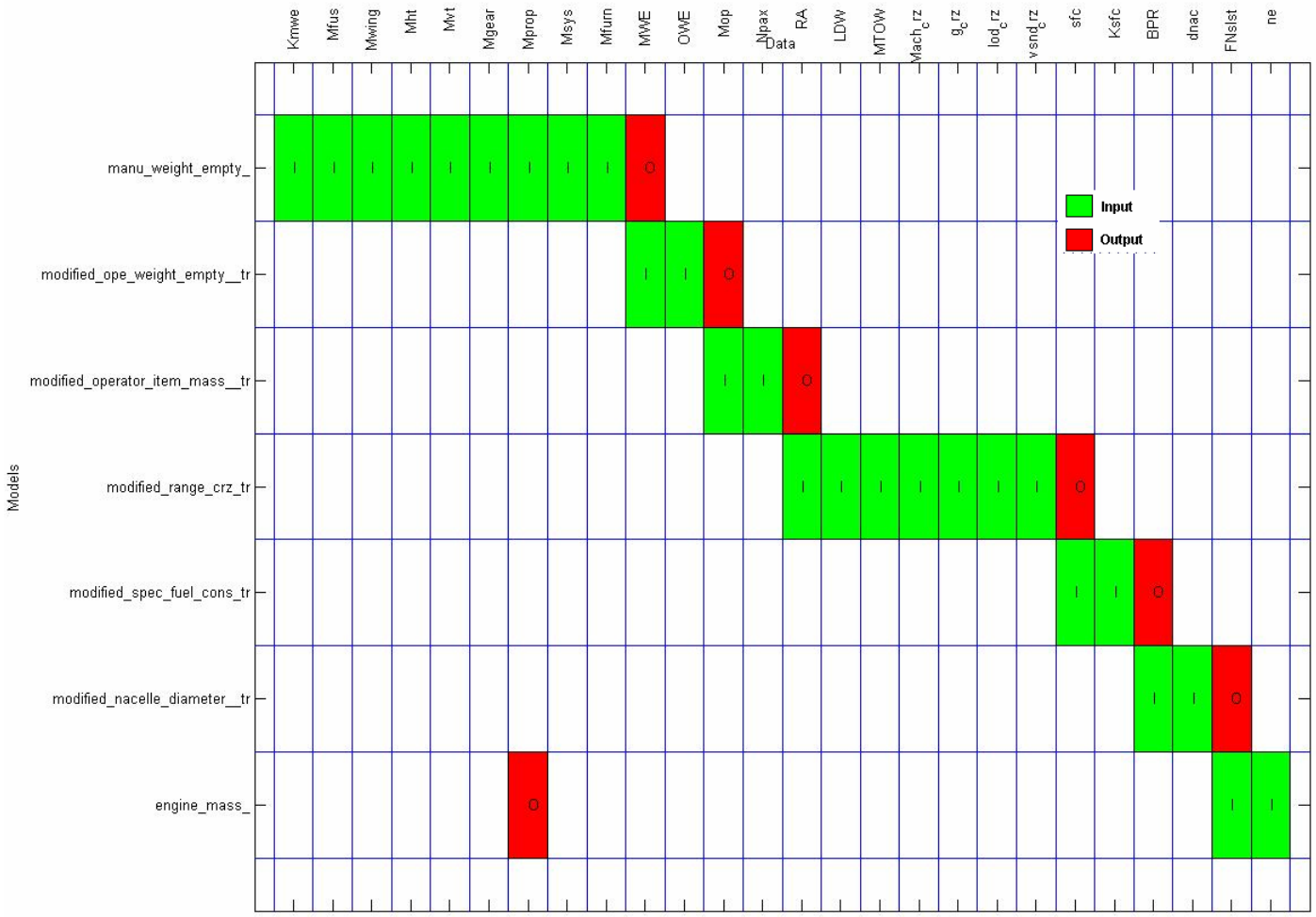


Figure A- 28: Incidence Matrix of the first SCC (Variable flow model-2) for Case-3 of USMAC test case

Variable flow model for second SCC: 1

Feedback number: 1, Number of modified models: 3

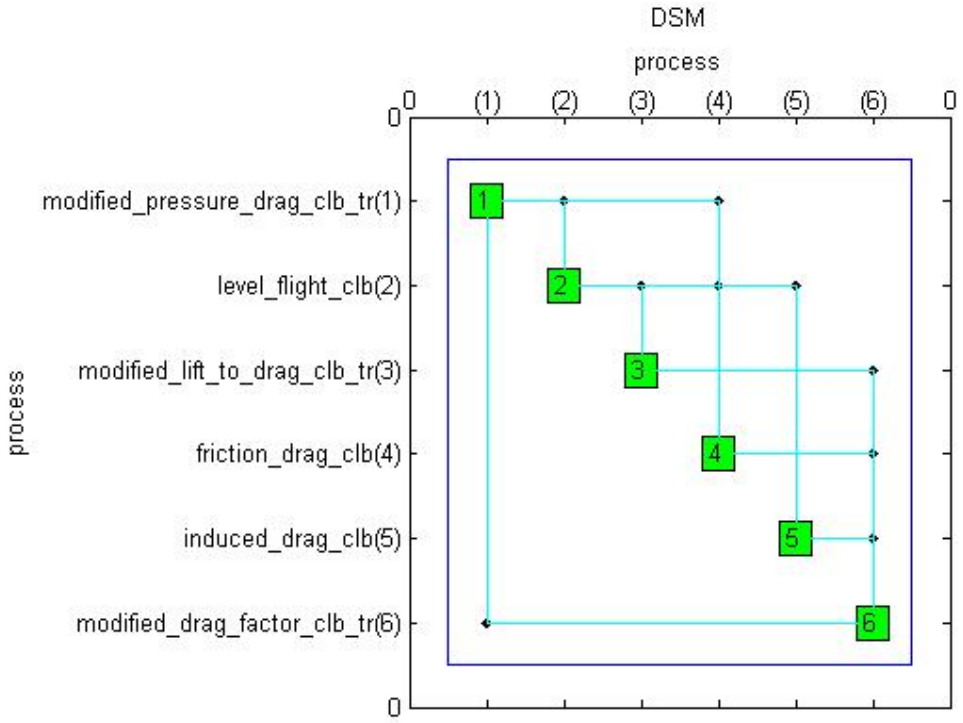


Figure A- 29: Design Structure Matrix of the second SCC (Variable flow model-1) for Case-3 of USMAC test case

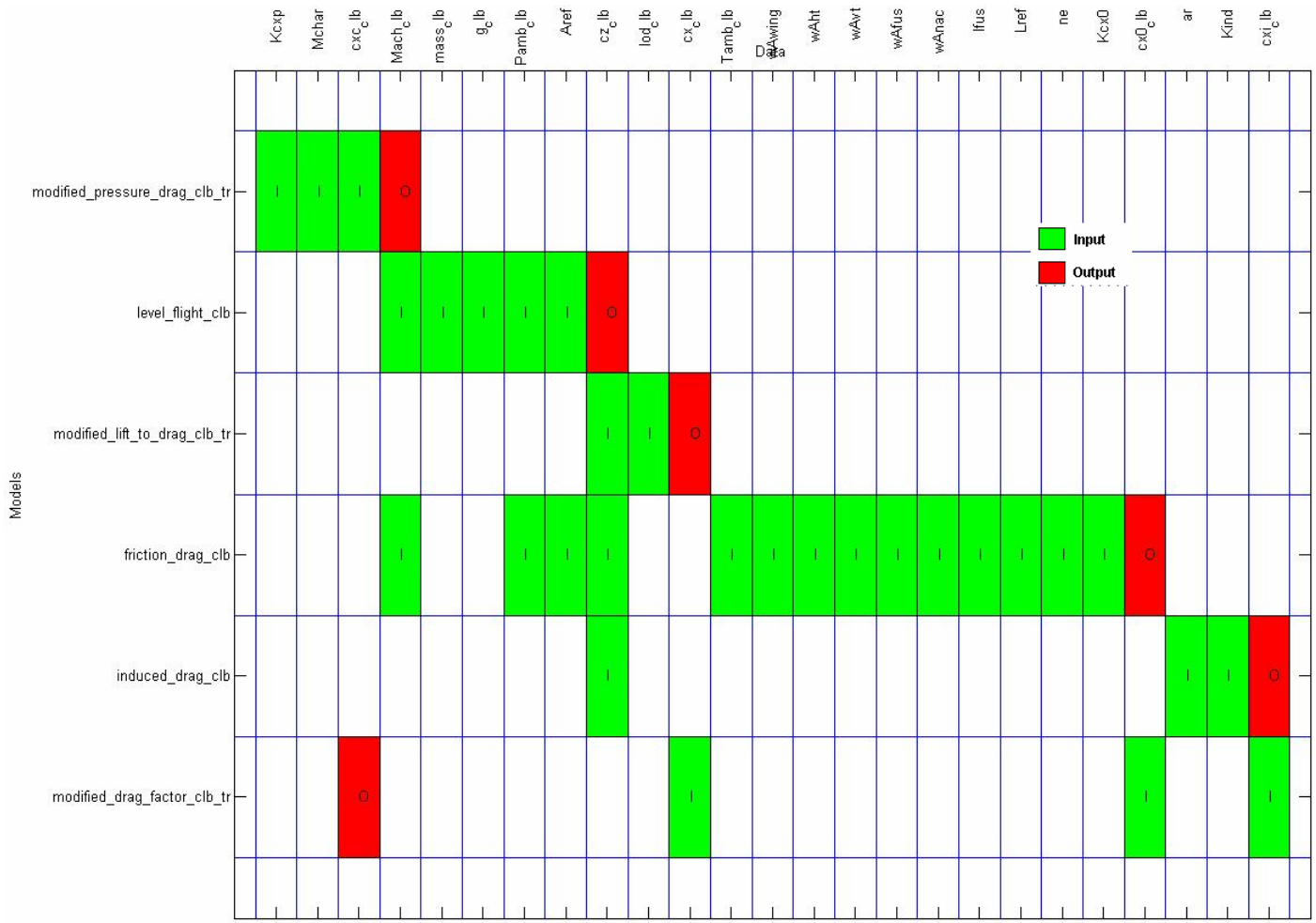


Figure A- 30: Incidence Matrix of the second SCC (Variable flow model-1) for Case-3 of USMAC test case

Variable flow model for second SCC: 2

Feedback number: 1, Number of modified models: 4

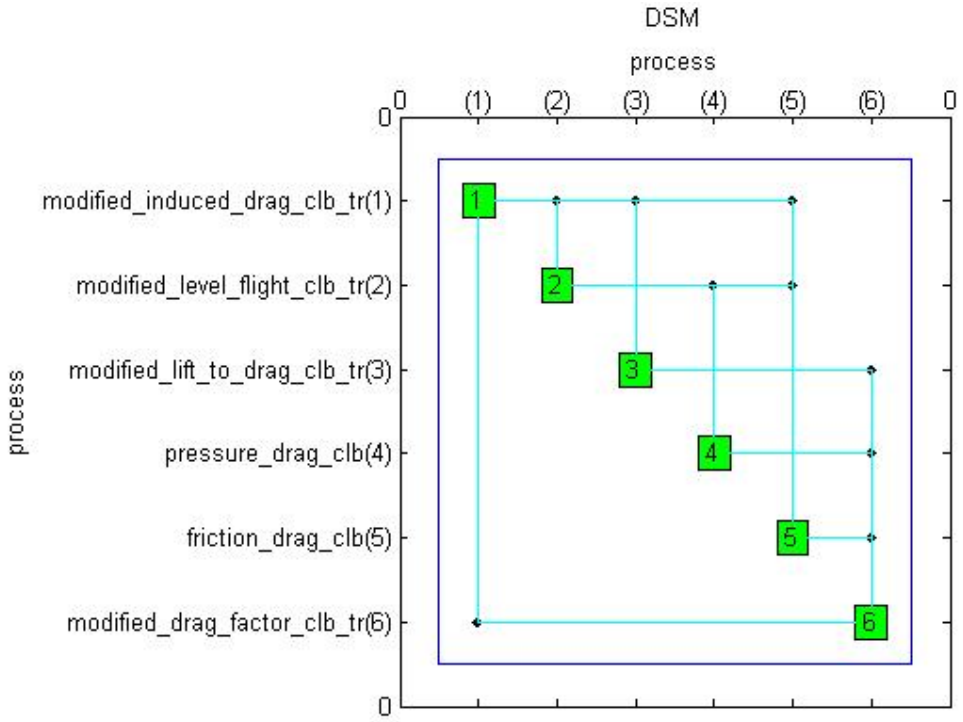


Figure A- 31: Design Structure Matrix of the second SCC (Variable flow model-2) for Case-3 of USMAC test case

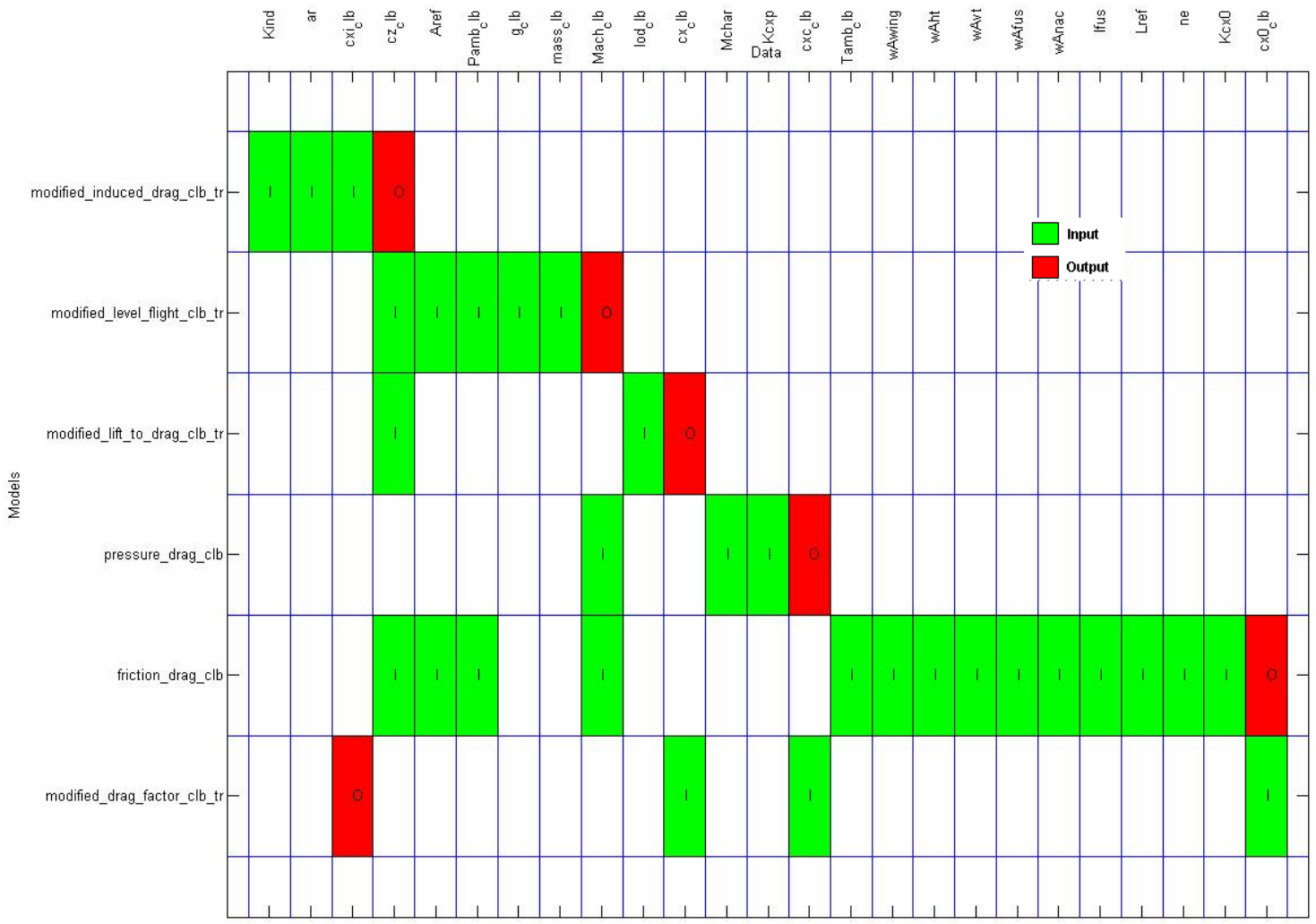


Figure A- 32: Incidence Matrix of the second SCC (Variable flow model-2) for Case-3 of USMAC test case

Variable flow model for second SCC: 3

Feedback number: 1, Number of modified models: 2

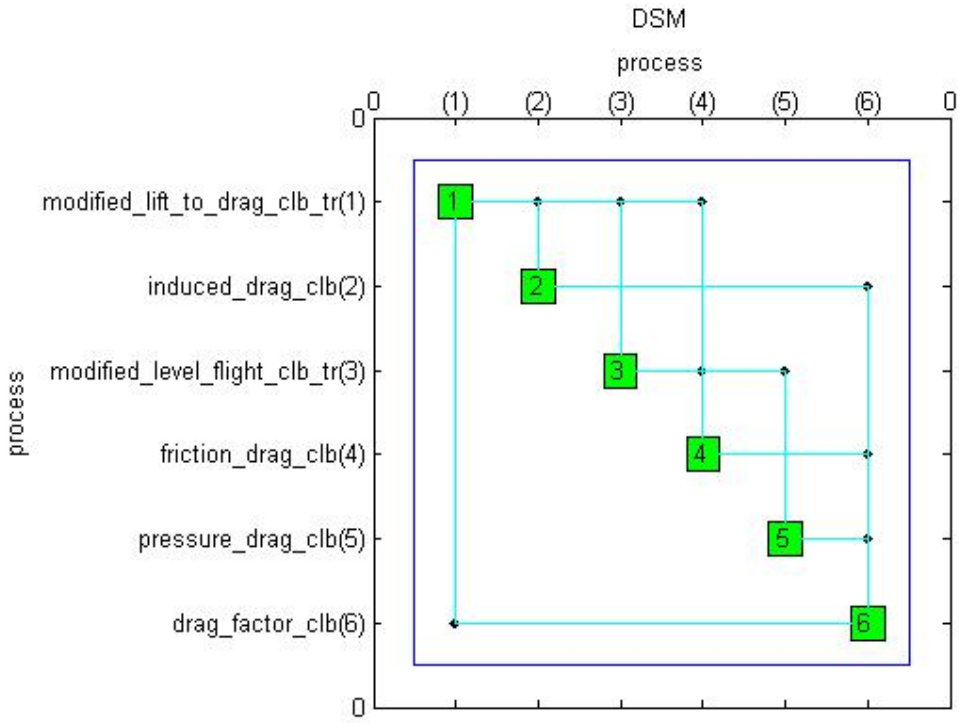


Figure A- 33: Design Structure Matrix of the second SCC (Variable flow model-3) for Case-3 of USMAC test case

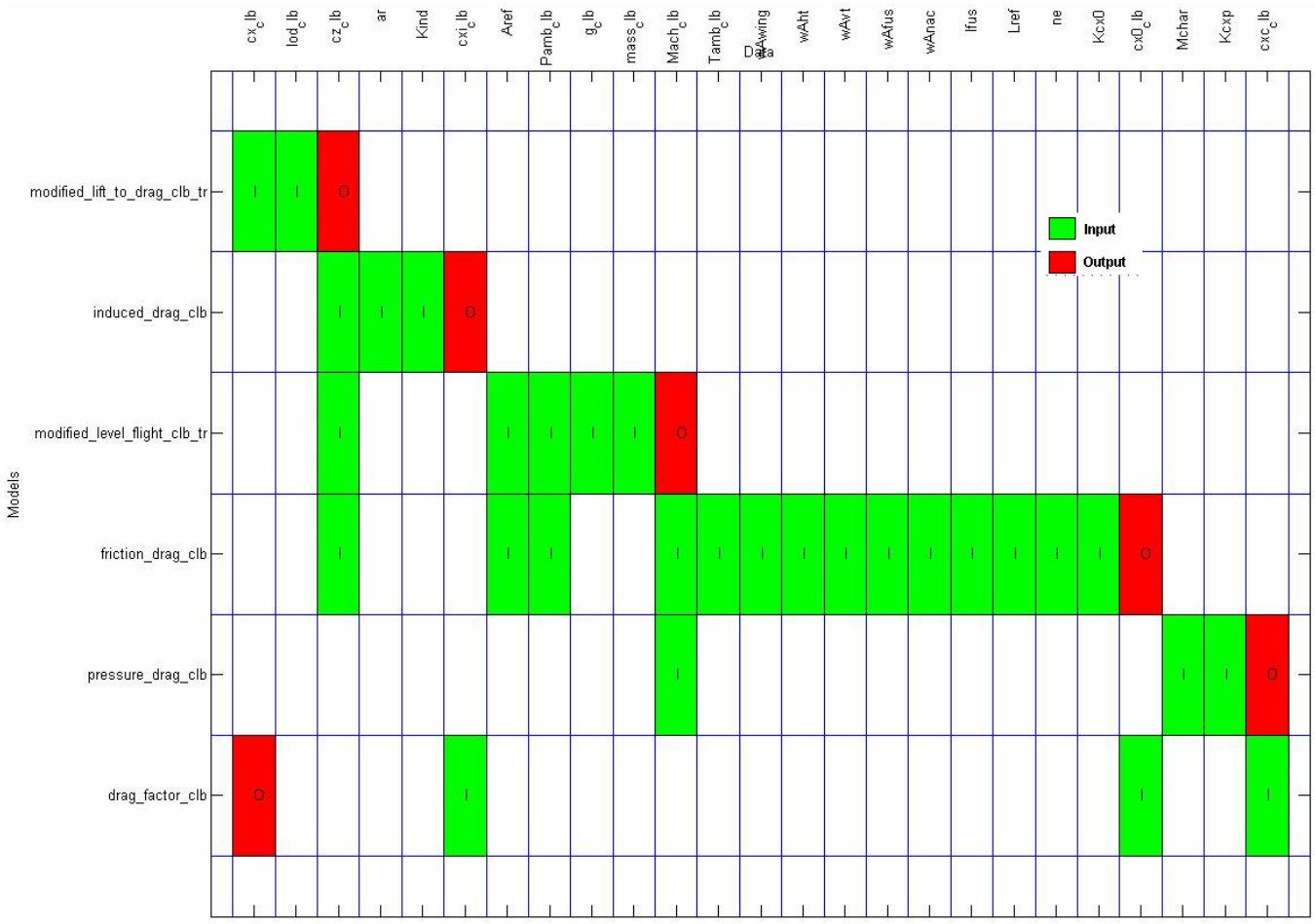


Figure A- 34: Incidence Matrix of the second SCC (Variable flow model-3) for Case-3 of USMAC test case

VI. Architecture of CWMD

This section contains the architectural, interface and design (Implementation) description of the CWMD.

i. Functional top-level decomposition

Functional top-level decomposition diagram in Figure A- 35 indicates the four main functions of CWMD and its interaction with the graphical user interface. 'Objects' in the figure corresponds to data, model, subprocess and study.

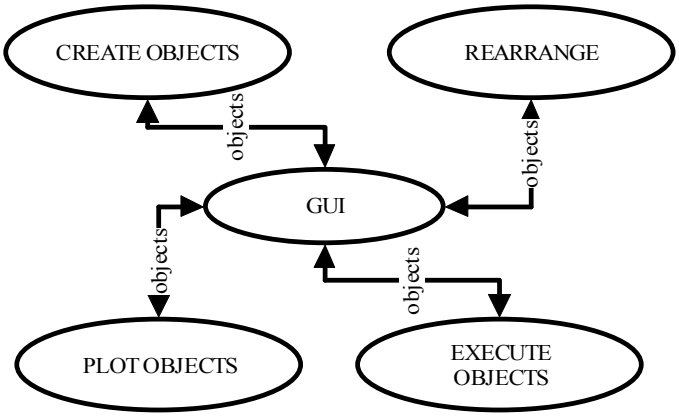


Figure A- 35 Top level functional flow systems diagram

ii. Functional application mapping

The functional top-level decomposition with its internal structure is shown in Figure A-36. The purpose is to show the first level structure of application components and how functional packages are allocated to components.

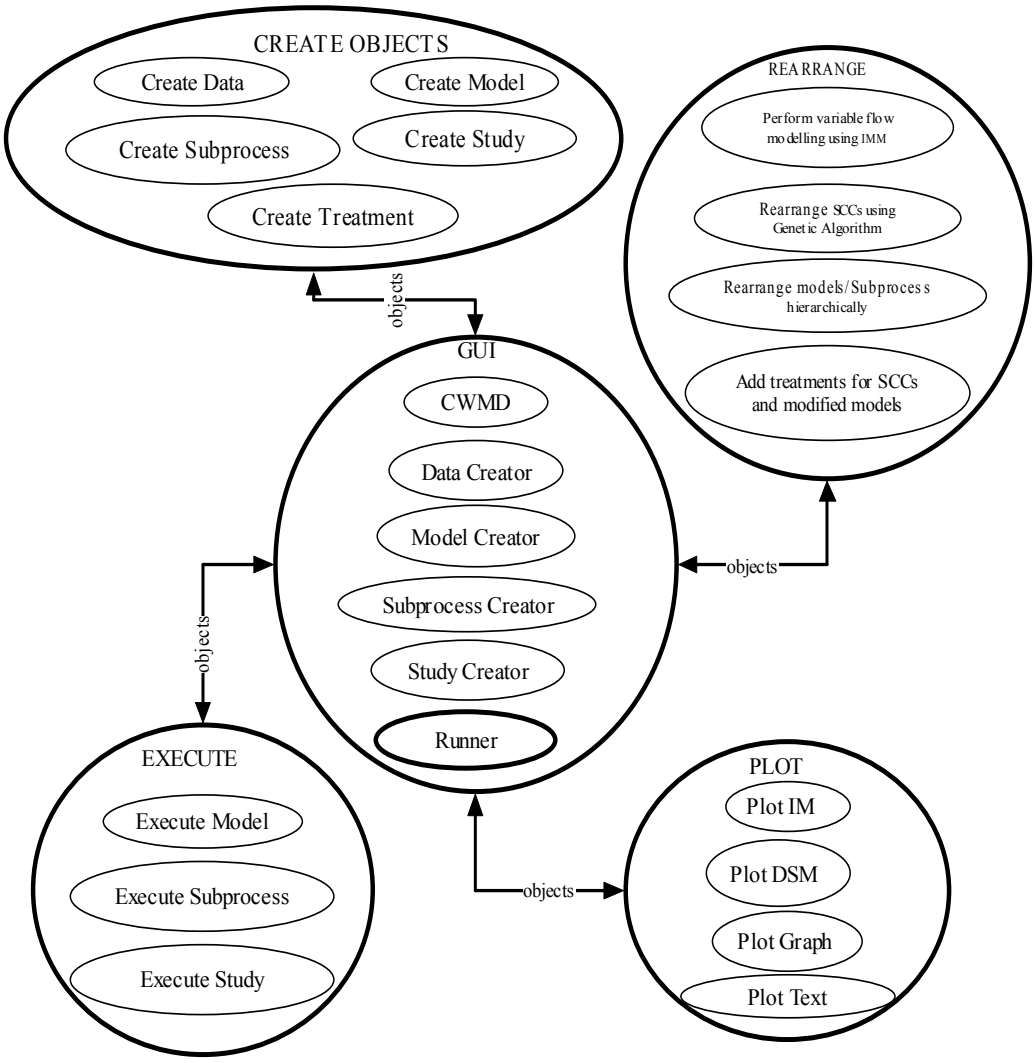


Figure A- 36 Functional top-level decomposition with its internal structure

iii. Product breakdown structure (PBS)

The hierarchical tree structure for the components is given below. The function of each component is self explanatory.

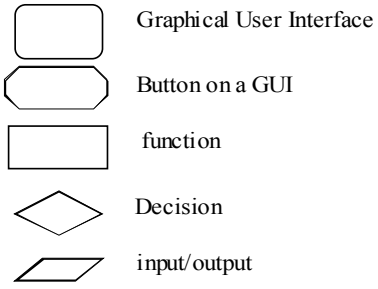
1. ***GUI***
   - ***1.1. CWMD GUI (***Figure A- 38***)***
   - ***1.2. DATA CREATOR (***Figure A- 39***)***
   - ***1.3. MODEL CREATOR (***Figure A- 40***)***
   - ***1.4. SUBPROCESS CREATOR (***Figure A- 41***)***
   - ***1.5. STUDY CREATOR (***Figure A- 42***)***

*1.6. **RUNNER (***Figure A- 43*)***

2. ***CREATE OBJECTS***
   - *2.1. **CREATE DATA (***Figure A- 44*)***
   - *2.2. **CREATE MODEL (***Figure A- 45*)***
   - *2.3. **CREATE SUBPROCESS (***Figure A- 46*)***
   - *2.4. **CREATE STUDY (***Figure A- 47*)***
   - *2.5. **CREATE TREATMENT (***Figure A- 48*)***
3. ***REARRANGE (***Figure A- 49*)***
   - *3.1. **PERFORM VARAIBLE FLOW MODELLING USING IMM (***Figure A- 50*)***
   - *3.2. **REARRANGE SCCS USING GENETIC ALGORITHM (***Figure A- 51*)***
   - *3.3. **REARRANGE MODELS/SPS HEIRARCHICALLY (***Figure A- 52*)***
4. ***PLOT MD/SP***
   - *4.1. **PLOT INCIDENCE MATRIX (***Figure A- 53*)***
   - *4.2. **PLOT DESIGN STRUCTURE MATRIX (***Figure A- 54*)***
   - *4.3. **PLOT GRAPH (***Figure A- 55*)***
   - *4.4. **PLOT TEXT (***Figure A- 56*)***
5. ***EXECUTE OBJECTS***
   - *5.1. **EXECUTE MODEL (***Figure A- 57*)***
   - *5.2. **EXECUTE SUBPROCESS (***Figure A- 58*)***
   - *5.3. **EXECUTE STUDY (***Figure A- 59*)***

The flow diagram for each component is shown in Figure A- 38 till Figure A- 59. A detailed flow diagram which combines all the component level flow diagrams is represented in Figure A- 37.

Symbols used in the Flow diagrams



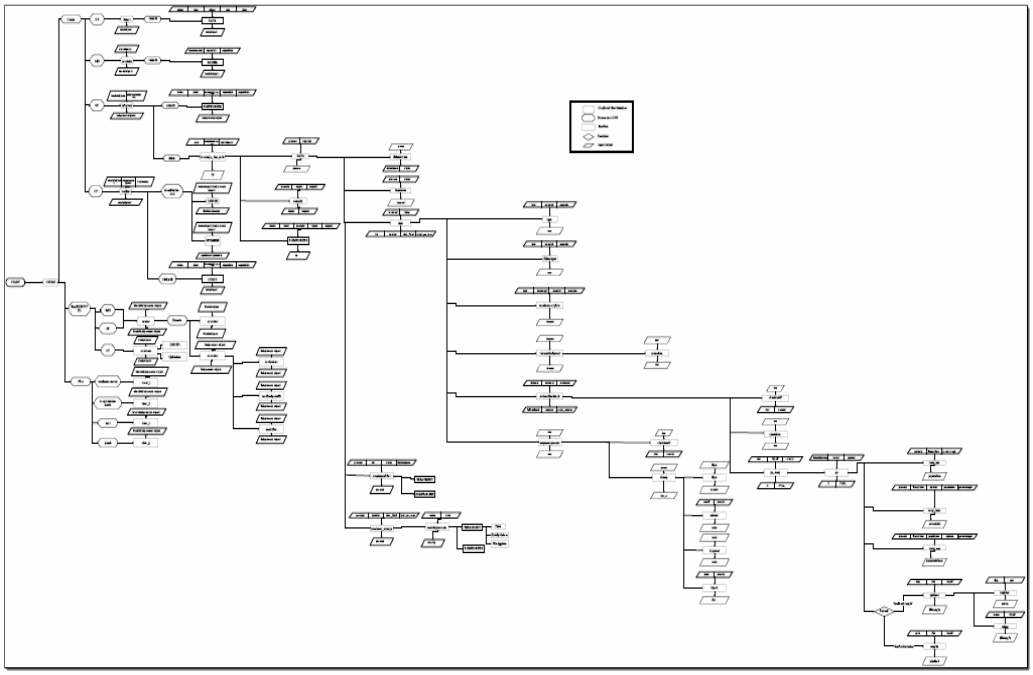


Figure A- 37 Functional flow systems diagram (attached sheet provides a zoomed view)

iv. Architecture Breakdown Diagrams

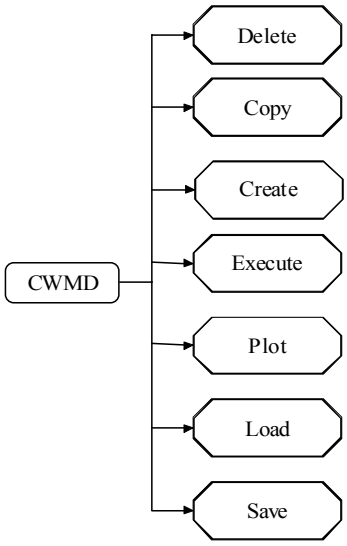


Figure A- 38 Flow diagram for CWMD GUI (Component 1.1)

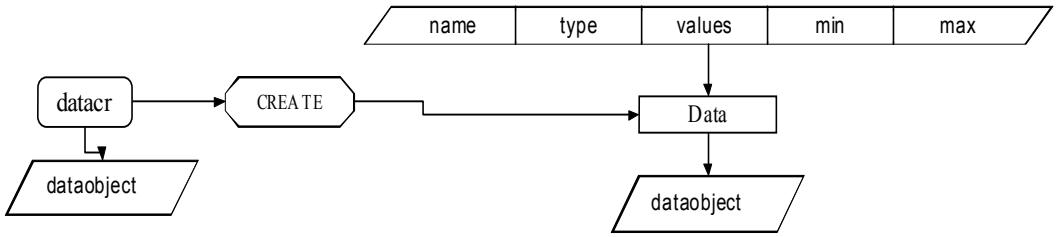


Figure A- 39 Flow diagram for 'data creator' GUI (Component 1.2)

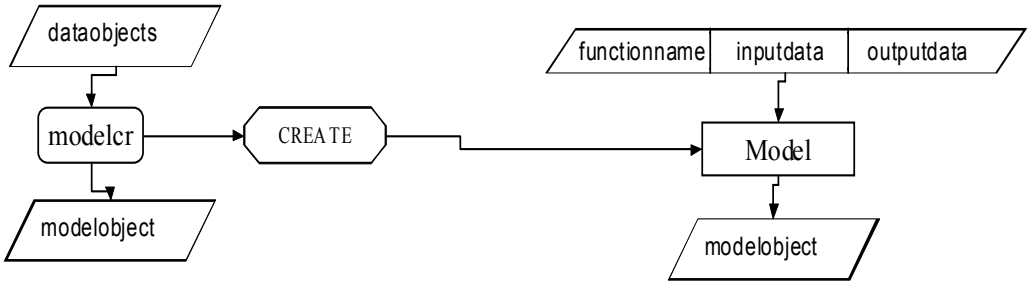


Figure A- 40 Flow diagram for 'model creator' GUI (Component 1.3)

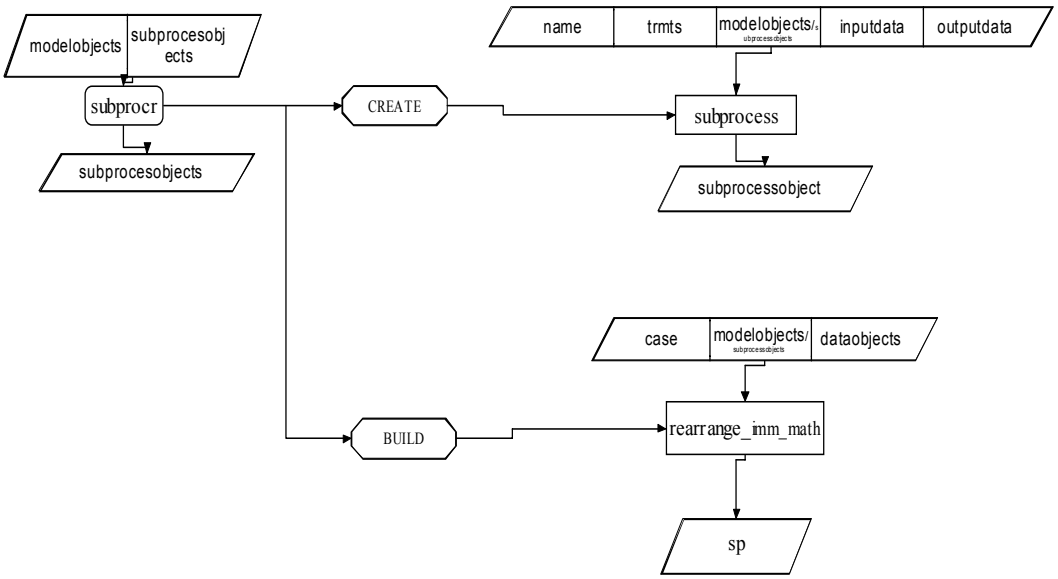


Figure A- 41 Flow diagram for 'subprocess creator' GUI (Component 1.4)

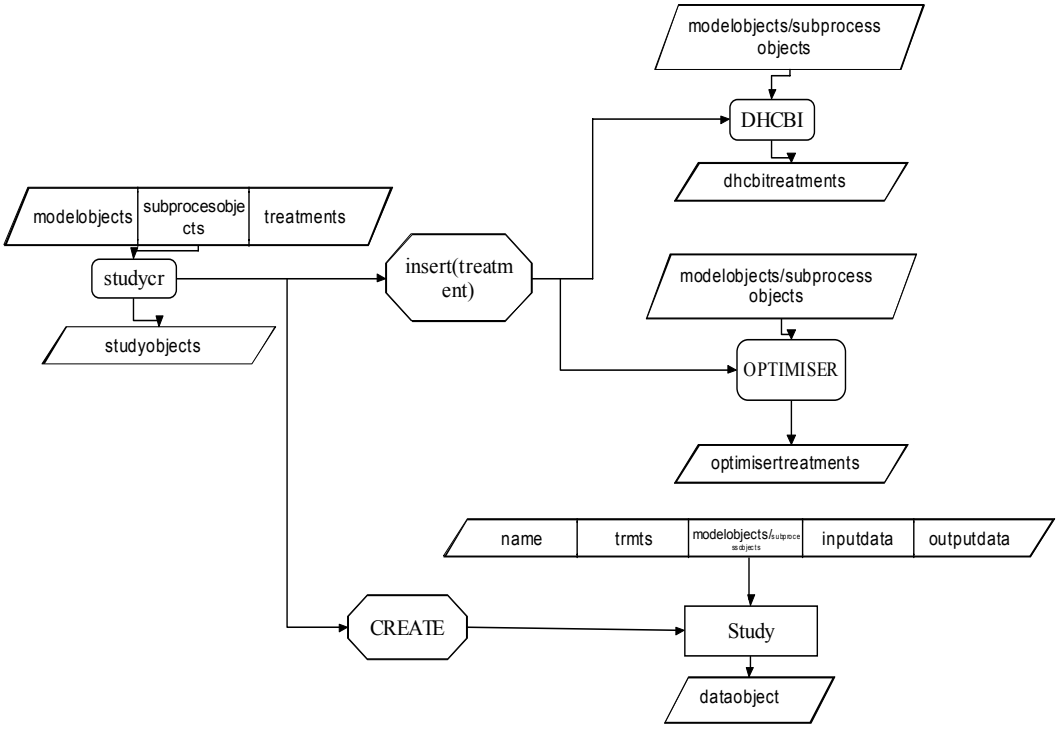


Figure A- 42 Flow diagram for 'study creator' GUI (Component 1.5)

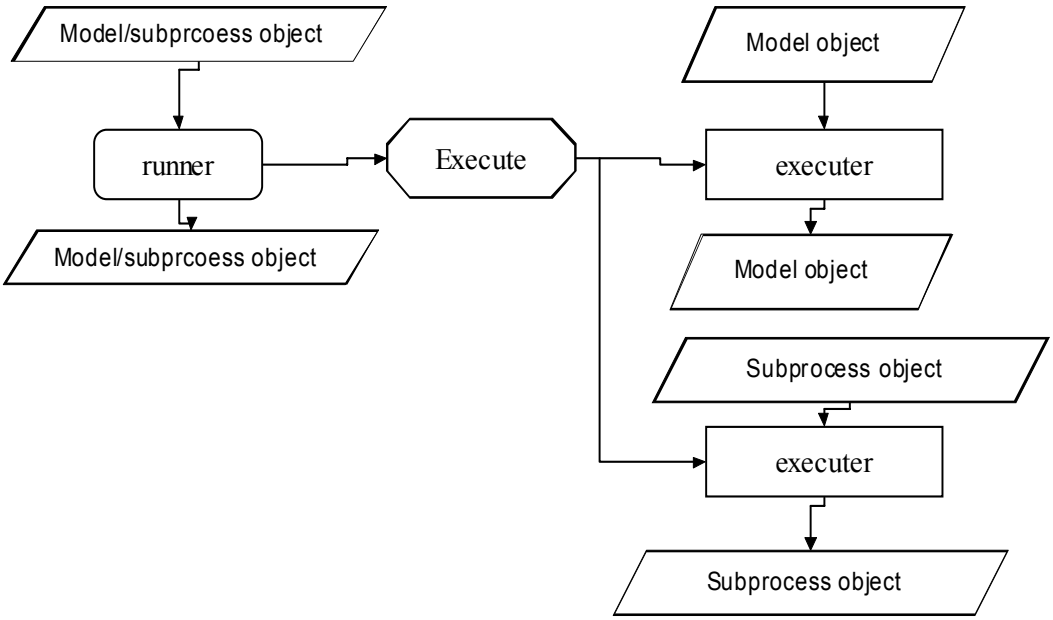


Figure A- 43 Flow diagram for 'runner' GUI (Component 1.6)

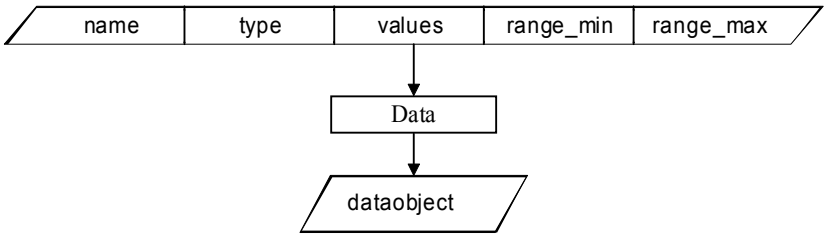


Figure A- 44 Flow diagram for creating data object creator (Component 2.1)

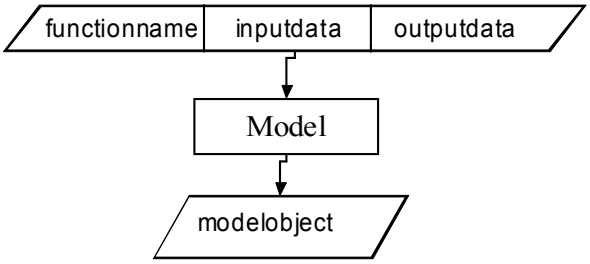


Figure A- 45Flow diagram for creating model object (Component 2.2)

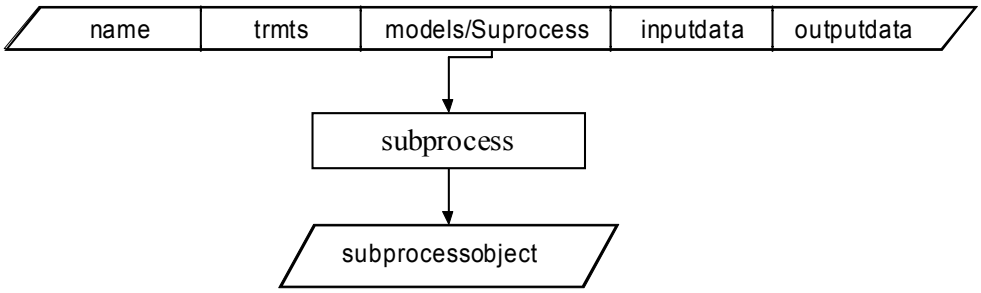


Figure A- 46 Flow diagram for creating subprocess object (Component 2.3)

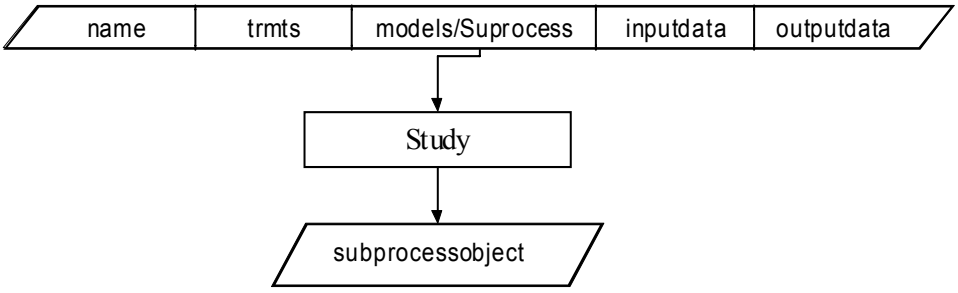


Figure A- 47 Flow diagram for creating study object (Component 2.4)

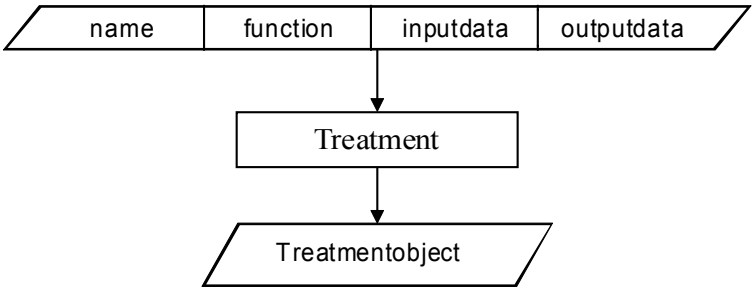


Figure A- 48 Flow diagram for creating treatment object (Component 2.5)

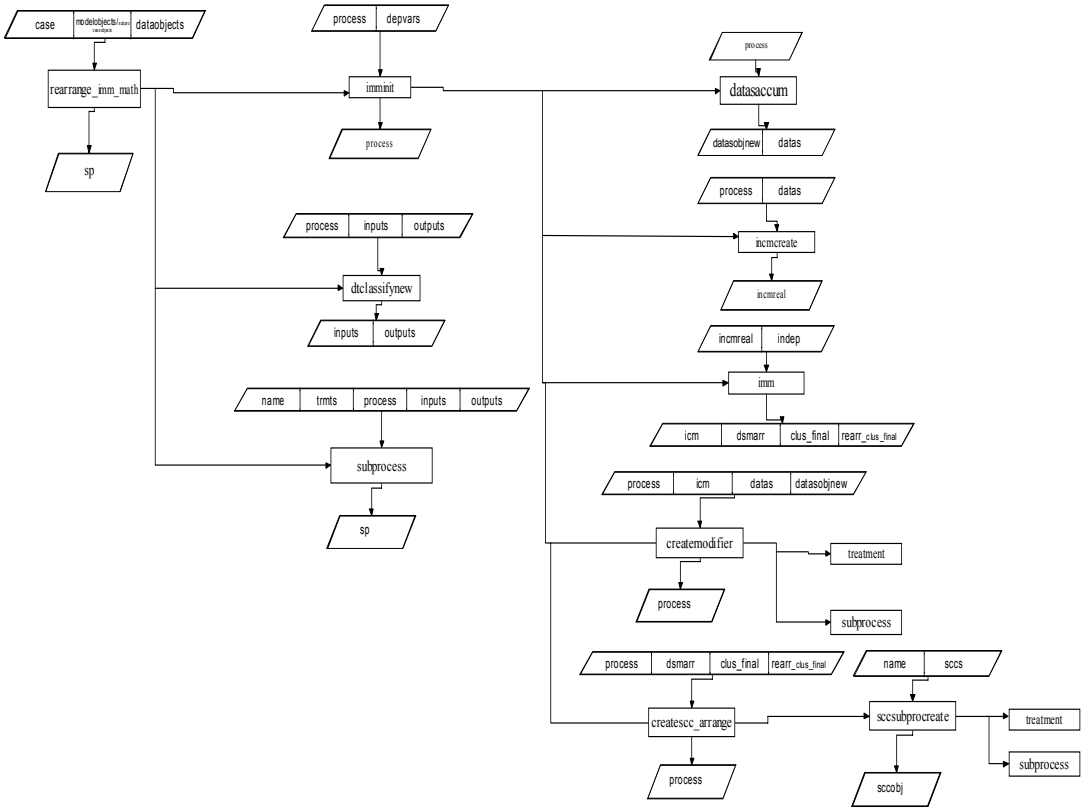


Figure A- 49 Flow diagram for rearrangement (Component 3)

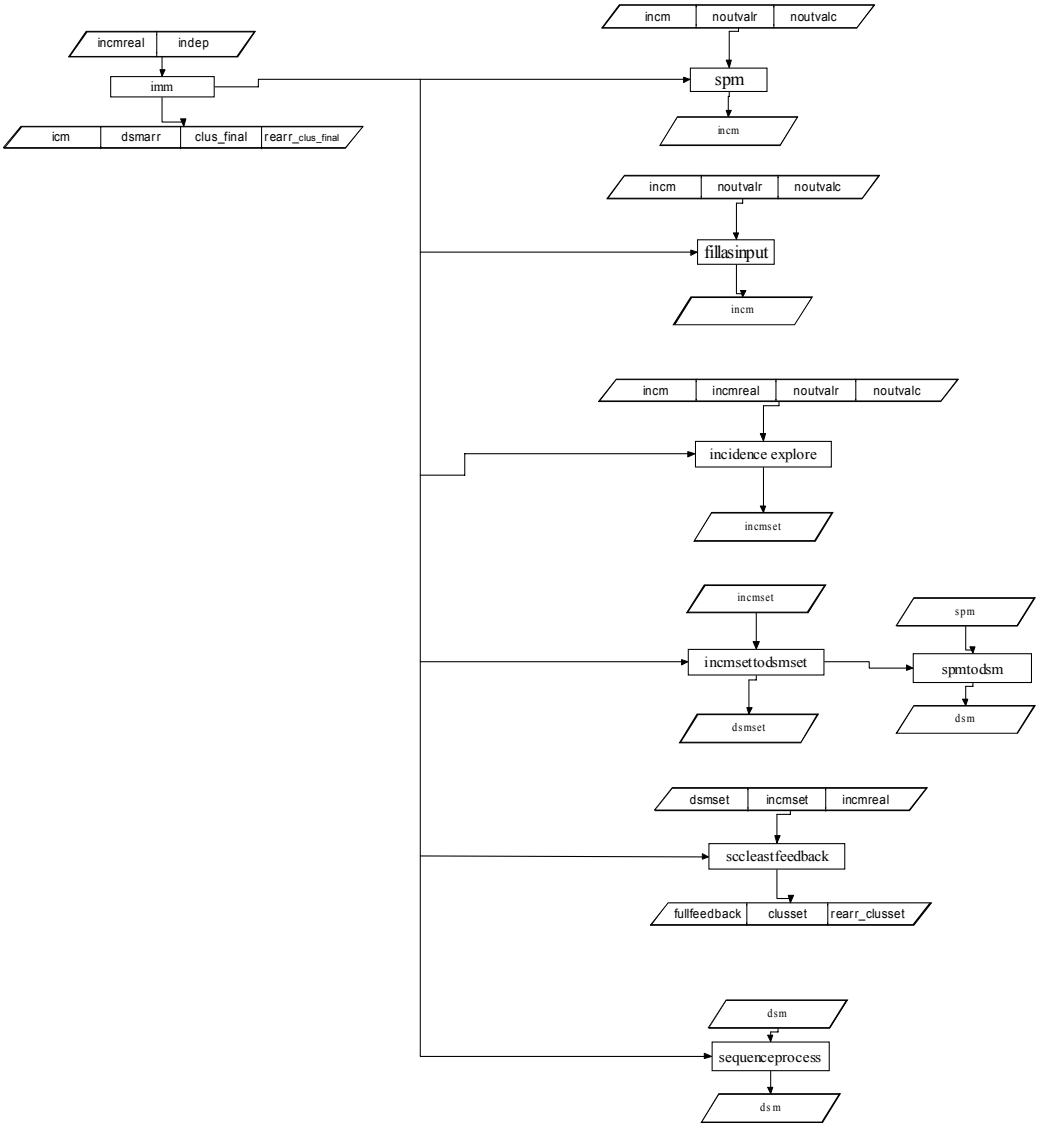


Figure A- 50 Flow diagram for variable flow modelling using incidence matrix method(Component 3.1)

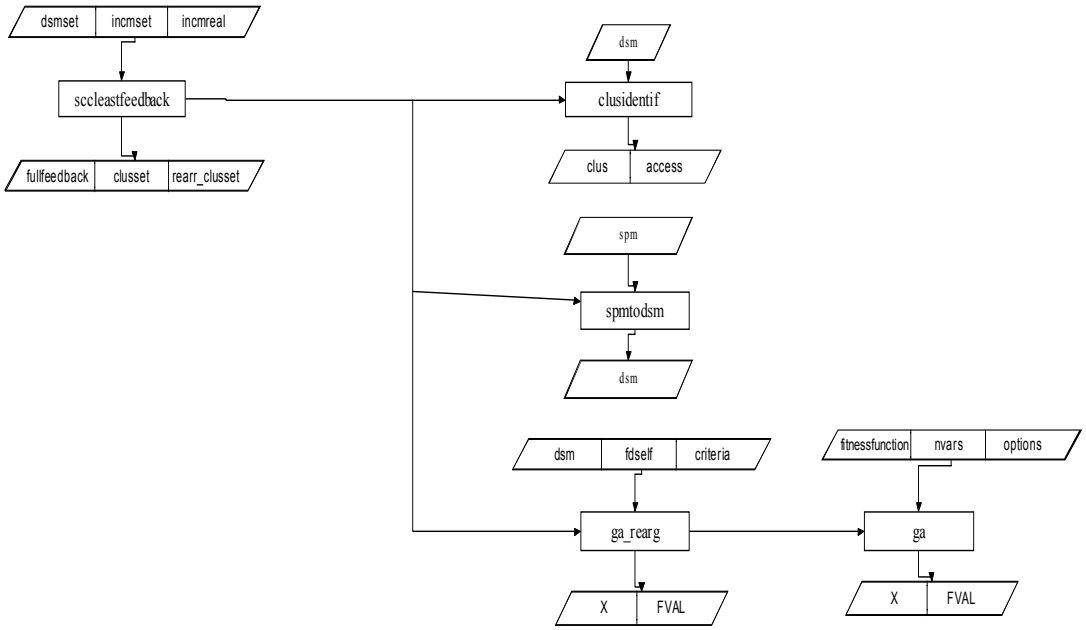


Figure A- 51 Flow diagram for rearrangement of SCCs by applying genetic algorithm (Component 3.2)

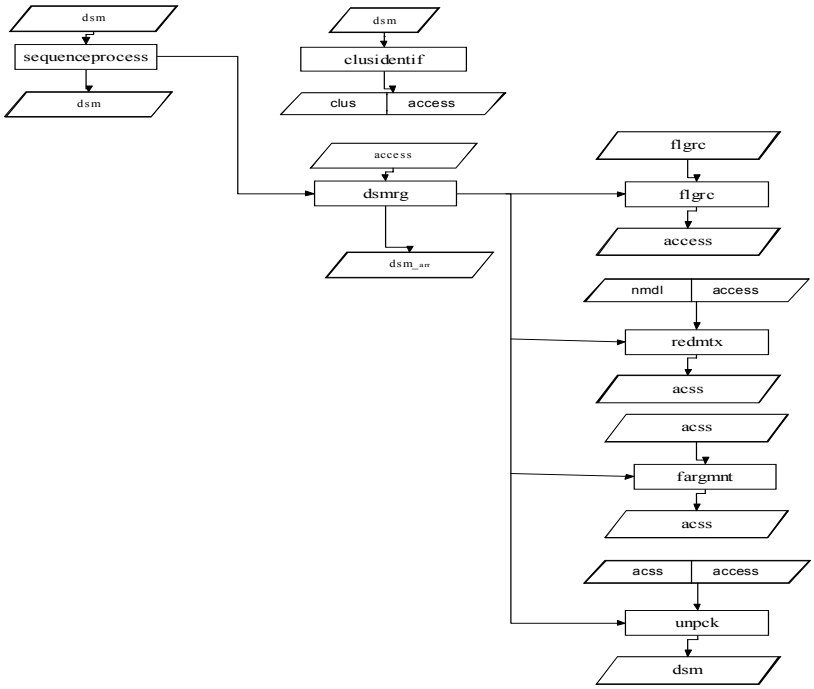


Figure A- 52 Flow diagram for rearing models/SPs hierarchically (Component 3.3)

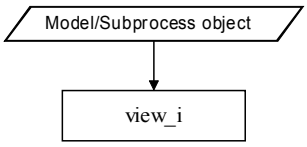


Figure A- 53 Flow diagram for representing MD/SP in an incidence matrix (Component 4.1)

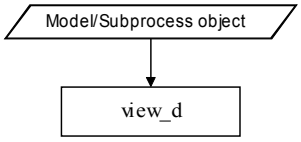


Figure A- 54 Flow diagram for representing MD/SP in a DSM (Component 4.2)

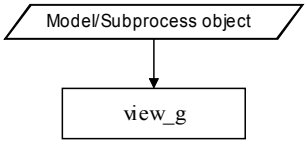


Figure A- 55 Flow diagram for representing MD/SP in a graph format (Component 4.3)

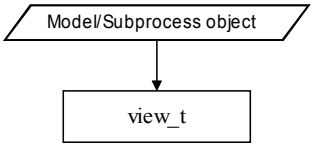


Figure A- 56 Flow diagram for representing MD/SP in a text GUI (Component 4.4)

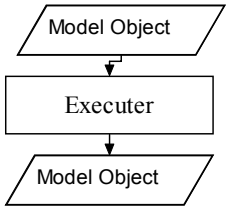


Figure A- 57 Flow diagram for executing model objects (Component 5.1)

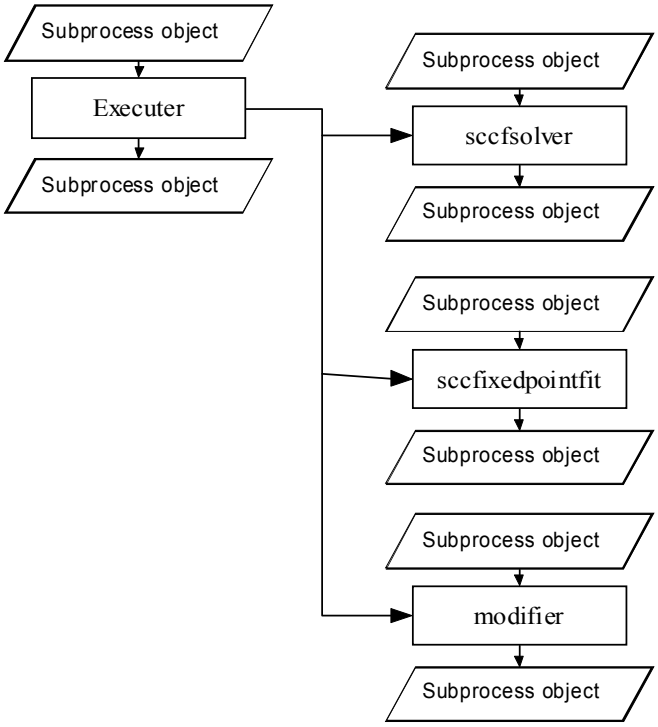


Figure A- 58 Flow diagram for executing subprocess objects (Component 5.2)

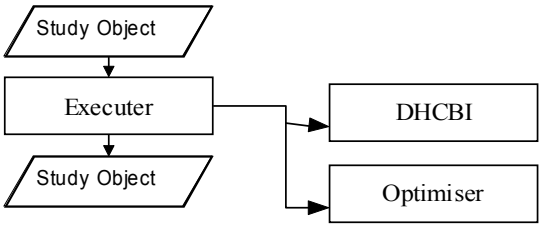


Figure A- 59 Flow diagram for executing study objects (Component 5.3)

v. Description of components of CWMD

The description of each component given in the Figure A- 38 till Figure A- 59 is given in the Table A- 3.

Table A- 3 Description of components of CWMD

Component ID	***Short Title***	***Description***	***Covered Function***
1. 1 (***Figure A-38***)	CWMD GUI	This is the main graphical user interface. There are options to create, execute, plot, delete and copy Objects (data, model, subprocess, study and treatments). The objects created can be loaded and saved using 'Load' and 'SAVE' buttons.	CWMD

Component ID	*Short Title*	*Description*	*Covered Function*
1. 2(***Figure A-39***)	DATA CREATOR	The function of this interface is to create data objects. The inputs to create the data (name, value, type, range) has to be provided by the user. This GUI can be activated from 'CWMD' window .	dater
1. 3(***Figure A-40***)	MODEL CREATOR	The function of this interface is to create model objects. The data objects loaded in the CWMD GUI will be the inputs to this GUI. The name of the model object to be created, its inputs and outputs has to be provided by the user. This GUI can be activated from 'CWMD' window .	modeller
1. 3(***Figure A-41***)	SUBPROCESS CREATOR	The function of this interface is to create subprocess objects. The model and subprocess objects loaded in the CWMD GUI will be the inputs to this GUI. The subprocess is created based on the models/sub processes and the independent variables selected by the user. This GUI can be activated from 'CWMD' window .	subprocr
1. 4(***Figure A-42***)	STUDY CREATOR	The function of this interface is to create study objects. The model and subprocess objects loaded in the CWMD GUI will be the inputs to this GUI. The study object is created based on the models/subprocesses and the treatment objects selected by the user. This GUI can be activated from 'CWMD' window . ▪	studycr
1. 5(***Figure A-43***)	RUNNER	The function of this interface is to execute the model and subprocess objects. The model/subprocess selected from the CWMD GUI will be the inputs to this GUI. This GUI can be activated from 'CWMD' window .	runner
2.1(***Figure A-44***)	CREATE DATA	Creates data object.	Data
2.2(***Figure A-45***)	CREATE MODEL	Creates data object.	model
2.3(***Figure A-46***)	CREATE SUBPROCESS	Creates subprocess object.	subprocess
2.4 (***Figure A-47***)	CREATE STUDY	Creates study object.	study
2.5(***Figure A-48***)	CREATE TREATMENT	Creates treatment object.	treatment
3(***Figure A- 49***)	REARRANGE	The main calling function for computational process plan Functions called imminit dtclassifynew subprocess(Component ID 2.3)	Rearrange_imm_math
3(***Figure A- 49***)	REARRANGE imminit	Inititalises the incm matrix and creates the incmreal matrix Functions called datasaccum incmcreate imm(Component ID 3.1)	imminit
3(***Figure A- 49***)	REARRANGE Imminit datasaccum	Creates a list of variables in the models and subprocess	datasaccum

Component ID	*Short Title*	*Description*	*Covered Function*
3(***Figure A- 49***)	REARRANGE Imminit incmcreate	Creates the foundation matrix of the models/subprocess in the process.	incmcreate
3(***Figure A- 49***)	REARRANGE Imminit createmodifier	Adds mathematical treatments for the subprocess/models which has its inputs and outputs modified and creates a subprocess Functions called Subprocess	createmodifier
3(***Figure A- 49***)	REARRANGE Imminit o createscc_arrange	Adds mathematical treatments for the subprocess/models which are part of the SCC, and creates a subprocess. Functions called Subprocess Sccsubprocreate	createscc_arrange
3(***Figure A- 49***)	REARRANGE dtclassifynew	Creates a cell array of the inputs and outputs for the group of models\subprocess given as inputs to 'dtclassify'	dtclassifynew
3.1(***Figure A-50***)	PERFORM VARAIBLE FLOW MODELLING USING IMM	Performs variable flow modelling based on incidence matrix method. The final incidence matrix, list of SCCs, the arrangement of SCCs etc are produced as outputs. Functions called indepcheck spm fillasinput incidence_explore incmsettodsmset sccleastfeedback(componenet ID 3.2) sequenceprocess(componenet ID 3.3)	imm
3.1(***Figure A-50***)	PERFORM VARIBALE FLOW MODELLING USING IMM - indepcheck	Checks whether incm will lead to violation of the constraints and leads to an unsolvable system.	indepcheck
3.1(***Figure A-50***)	PERFORM VARIBALE FLOW MODELLING USING IMM -spm	Conducts the incidence matrix method on the incm matrix provided.	spm
3.1(***Figure A-50***)	PERFORM VARIBALE FLOW MODELLING USING IMM -fillasinput	If the system is under constrained then this function will make certain variables to be independent and in parallel making sure that none of the constraints are violated.	fillasinput
3.1(***Figure A-50***)	PERFORM VARIBALE FLOW MODELLING USING IMM -incidence_explore	If the system is under constrained then this function will make certain variables to be independent and in parallel making sure that none of the constraints are violated. Functions called fillasinput spm *findchanval* (finds the rows of incm where the models associated variable are modified) incidence_explore	incidence_explore

Component ID	*Short Title*	*Description*	*Covered Function*
3.1(***Figure A-50***)	PERFORM VARIBALE FLOW MODELLING USING IMM -incmsettodsmset	If the system is under constrained then this function will make certain variables to be independent and in parallel ensuring that none of the constraints are violated. Functions called *Spmtodsm* (converts IM to DSM)	incmsettodsmset
3.2(***Figure A-51***)	REARRANGE SCCS USING GENETIC ALGORITHM	Rearranges the elements of SCCs using genetic algorithm, based on the criteria('feedback length' or 'feedback number') provided by the user Functions called clusidentif spmtodsm ga_rearg	sccleastfeedback
3.2(***Figure A-51***)	REARRANGE SCCS USING GENETIC ALGORITHM -clusidentif	Identifies the SCCs in the dsm provided and give as output the list of elements in each SCC	clusidentif
3.2(***Figure A-51***)	REARRANGE SCCS USING GENETIC ALGORITHM - ga_rearg	Identifies the SCCs in the dsm provided and give as output the list of elements in each SCC Functions called ga- Matlab function optimtest numfdb rearg_cross rearg_mut rearg_pop	ga_rearg
3.2(***Figure A-51***)	REARRANGE SCCS USING GENETIC ALGORITHM - ga_rearg - optimtest	Calculates the feedback length based on the dsm and the order in which dsm has to be rearranged. Functions called magicdsm fdblng	optimtest
3.2(***Figure A-51***)	REARRANGE SCCS USING GENETIC ALGORITHM - ga_rearg - optimtest -magicdsm	Creates a rearranged DSM(mdsm) out of the new arrangement of disciplines according to input from disp	ga_rearg
3.2(***Figure A-51***)	REARRANGE SCCS USING GENETIC ALGORITHM - ga_rearg - optimtest -fdblng	Finds feedback length given a DSM using the equation sum(DM(i,j)(i-j)),i=2:n,j=1:i-1	fdblng
3.2(***Figure A-51***)	REARRANGE SCCS USING GENETIC ALGORITHM - ga_rearg - rearg_cross	Position based crossover for rearranging the dsm	rearg_cross
3.2(***Figure A-51***)	REARRANGE SCCS USING GENETIC ALGORITHM - ga_rearg - rearg_mut	Mutation for rearranging the dsm	rearg_mut

Component ID	*Short Title*	*Description*	*Covered Function*
3.2(***Figure A-51***)	REARRANGE SCCS USING GENETIC ALGORITHM - ga_rearg - rearg_pop	Creates population given the GenomeLength	rearg_pop
3.2(***Figure A-51***)	REARRANGE SCCS USING GENETIC ALGORITHM - ga_rearg - numfdb	Calculates the feedback number based on the dsm and the order in which dsm has to be rearranged.	numfdb
3.3(***Figure A-52***)	REARRANGE MODELS/SPS HEIRARCHICALLY	Rearrangement of the DSM based on graph theory Functions called *clusidentif*(refer component 3.2) dsmrg	sequenceprocess
3.3(***Figure A-52***)	REARRANGE MODELS/SPS HEIRARCHICALLY -dsmrg	Rearrangement of the accessibility matrix based on graph theory Functions called flgrc redmtx fargmt unpck	dsmrg
3.3(***Figure A-52***)	REARRANGE MODELS/SPS HEIRARCHICALLY -dsmrg -flgrc	Inserts flags for the rows and columns of accessibility matrix which are identical.	flgrc
3.3(***Figure A-52***)	REARRANGE MODELS/SPS HEIRARCHICALLY -dsmrg -redmtx	The rows and columns of the acces matrix which are identical are collapsed into single rows and columns.	redmtx
3.3(***Figure A-52***)	REARRANGE MODELS/SPS HEIRARCHICALLY -dsmrg - fargmt	Final rearrangement based on graph theory	fargmt
3.3(***Figure A-52***)	REARRANGE MODELS/SPS HEIRARCHICALLY -dsmrg - unpck	Final rearrangement based on graph theory	unpck
4.1 (***Figure A-53***)	PLOT INCIDENCE MATRIX	Plots the subprocess which is given as input, as an incidence matrix. Functions called form_icm form_x_tckl form_y_tckl graph_matrix	view_i
4.2(***Figure A-54***)	PLOT DESIGN STRUCTURE MATRIX	Plots the subprocess which is given as input, as a design structure matrix. Functions called settingp dsmatrix	view_d

Component ID	*Short Title*	*Description*	*Covered Function*
		intercnt graph_matrix	
4.3(***Figure A-55***)	PLOT GRAPH	Plots the subprocess which is given as input, as a graph. Functions called views graph_to_dot dot_to_graph my_setdiff graph_draw textoval ellipse textbox make_layout arrow1	view_gi
4.4 (***Figure A-56***)	PLOT TEXT	Plots subprocess and models in the tabular format	view_t
5.1(***Figure A-57***)	EXECUTE MODEL	Executes model.	executer
5.2(***Figure A-58***)	EXECUTE SUBPROCESS	Executes subprocess. Functions called *Valuefillpre*- the results obtained after execution is updated for all data objects(inputs and outputs of the subprocess) embedded in the subprocess sccfsolver Sccfixedpoint modifier	executer
5.2(***Figure A-59***)	EXECUTE SUBPROCESS -sccfsolver	Solving SCC using the fsolve function of MATLAB. For details on FSOLVE function refer the user manual for MATLAB. Functions called *Fsolve*- Matlab function *Findguess*- finds the variables which are part of the feedback loops *Invrscc*- the objective function for fsolve	sccfsolver
5.2(***Figure A-59***)	EXECUTE SUBPROCESS -sccfixedpoint	Solving SCC using fixed point iteration. Functions called *Settingp*- converts the attributes of 'models' into a cell array *Dsmatrix*-Finds the strongly connected components(SCC) *dsmvarfind*- finds the variables which are part of the feedback loops *intercnt*-mapping inputs and outputs of models	sccfixedpoint
5.2(***Figure A-59***)	EXECUTE SUBPROCESS -modifier	Solving SCC using the fsolve function of MATLAB. For details on FSOLVE function refer the user manual for MATLAB. Functions called *Fsolve*- Matlab function *Findunk*- checking variables which are modified *Invrs*- objective function for fsolve	modifier
5.3(***Figure A-59***)	EXECUTE STUDY	Executes study. Functions called *Valuefillpre*- the results obtained after execution is updated for all data objects(inputs and outputs of the subprocess) embedded in the study DHCBI Optimiser	executer