Analysis of a flight mechanics simulator.

Fredrik Helgesson

Department of Aeronautical and Vehicle Engineering
KTH-Royal Institute of Technology
SE 100 44 Stockholm, Sweden
2019-09-18

Abstract: Aircraft design is an act of art requiring dedication and careful work to ensure good results. An essential tool in that work is a flight mechanics simulator. Such simulators are often built up of modules/models that are executed in a sequential order in each time iteration. This project aims to analyze potential improvements to the model execution order based on the dependency structure of one such simulator. The analysis method Design Structure Matrix (DSM), was used to define/map the dependencies and then Binary Linear Programming (BLP) was utilized to find five new potentially improved model orders to minimize the number of feedbacks from one iteration to the next one. Those five proposed execution orders were next compared and evaluated. The result is a model order that reduce the number of models receiving feedbacks from the previous iteration from 13 to 6, with insignificant changes in the precision of the simulator.

Keywords: Flight mechanics simulator, Dependency analysis, Design Structure Matrix, DSM, Binary Linear Programming

I. Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>Angle of attack</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Sideslip angle</td>
</tr>
<tr>
<td>$M$</td>
<td>Mach number</td>
</tr>
<tr>
<td>$m$</td>
<td>Actual A/C mass</td>
</tr>
<tr>
<td>$\theta$</td>
<td>Pitch angle</td>
</tr>
<tr>
<td>$\phi$</td>
<td>Roll angle</td>
</tr>
<tr>
<td>$\psi$</td>
<td>Yaw angle</td>
</tr>
<tr>
<td>$q$</td>
<td>Pitch rate</td>
</tr>
<tr>
<td>$r$</td>
<td>Yaw rate</td>
</tr>
<tr>
<td>$\dot{q}$</td>
<td>Pitch acceleration</td>
</tr>
<tr>
<td>$\ddot{q}$</td>
<td>Pitch acceleration</td>
</tr>
<tr>
<td>$\dot{r}$</td>
<td>Yaw acceleration</td>
</tr>
<tr>
<td>$\ddot{r}$</td>
<td>Yaw acceleration</td>
</tr>
<tr>
<td>$n_z$</td>
<td>Load factor in z-dir.</td>
</tr>
<tr>
<td>$C_N$</td>
<td>Normal force coef.</td>
</tr>
<tr>
<td>$C_T$</td>
<td>Tangential force coef.</td>
</tr>
<tr>
<td>$C_C$</td>
<td>Side force coef.</td>
</tr>
<tr>
<td>$C_{MX}$</td>
<td>Roll moment coef.</td>
</tr>
<tr>
<td>$C_{MY}$</td>
<td>Pitch moment coef.</td>
</tr>
<tr>
<td>$C_{MZ}$</td>
<td>Yaw moment coef.</td>
</tr>
</tbody>
</table>

1 MSc. Aerospace Engineering student
II. Introduction

Designing a modern high performance aircraft is not trivial. The success and economical effectiveness in developing such an aircraft relies heavily on the use of digital simulation tools, where the flight mechanics simulator is one of the most important. It is fed with parameters describing the aircraft, subsystems, the environment etc., and then simulates the motion and predicts the behavior of the aircraft. This means that even before the actual aircraft has flown for the first time one knows quite well how it will behave, which significantly lowers the risks and expenses of the development program. The flight mechanics simulator is also a very useful tool for evaluating modifications of existing aircraft and to investigate accidents and other incidents. One flight mechanics simulator software program is ARES (Aircraft Rigid-body Engineering Simulation), which is used at Saab. ARES has, for example, been utilized for developing the Gripen aircraft. The ARES software is developed and designed in-house at Saab and has been updated and extended several times during its operational life. The high level of detail in the aircraft modelling in ARES leads to simulations giving very useful and accurate predictions. ARES is built up of several models, each describing a particular system or set of equations, to effectively handle the complexity and wealth of details, see Figure 1.

1. ARES

The software architecture of the simulator consists of three types of building blocks, the SHELL, the generic aircraft models and the specific aircraft models. The SHELL is the spine of the simulator and handles the “administrative” tasks, such as the user interface and sending the right signals to the right places. This part of the simulator is very general and could be used for many kinds of simulations depending on the other models connected. The generic aircraft models are a block containing all the aircraft independent models, for example the atmosphere model (ATMOS) and some of the pilot models (PILOT). Finally, the last building block consists of the models that are specific to the aircraft simulated. Those models are for example the aerodynamics model (AERO), engine model (ENG) and fuel system model (FUEL).

The development of ARES has been organic and signals, as well as models, have been added when deemed necessary to increase the usefulness and precision of the simulation tool. The aim of this project is to analyze the interaction between the models and especially the sequence they are executed in. The execution order may influence the simulation results as some of the models have to use input data calculated in the previous iteration, and depending on the execution order the models that get input from previous iterations differ. The iteration frequency of ARES is 120 HZ, nevertheless for quick maneuvers and dynamic flight states there might be room for improvement to increase the precision. After the current status has been mapped, the work will continue with investigating potential improvements, applying those, and then evaluate the effect of the modifications.

III. Theory

This section presents the theoretical background of the work, which is presented in the following sections.

A. Dependence analysis

When doing a simulation in ARES each iteration represents a discrete point in time and the current iteration is dependent on the time history, which establishes a loop dependency between the iterations. In each iteration the equations of motion are integrated and the models are executed with the new state values as input. Analyzing the dependencies between the models, basically the information flow within each iteration, is necessary to understand what couplings there will be between the iterations. The couplings between the iterations depends on the execution order of the models within each iteration, as will be further discussed below.
B. Design Structure Matrix, DSM

The dependencies between tasks/statements/models of a system can be visually displayed by drawing a graph where the vertices are representing tasks, and arrows drawn between the vertices are representing the dependencies, as in Figure 1. However, for large systems this type of visual representation gets hard to interpret and to ease the analysis of such systems the design structure matrix is a good tool. The Design Structure Matrix, DSM, is an analysis method using the adjacency matrix and was developed, and first described by Donald V. Steward [1] in 1981. Since then, DSM has been applied and used in a range of different fields, all from product development, organizational management to project planning [2], but also for software analysis, e.g. [3] and [4]. The strengths of the DSM are the compact format in which large systems of dependencies can easily be overviewed and that mathematical matrix operations can be applied to analyze and rearrange the DSM. As in the adjacency matrix the dependencies between the tasks are marked with a mark in the position in the matrix where the two tasks intersect. Let \( S = [s_{ij}] \), where \( S \) is the tasks, and the DSM is a square matrix of size \( N \times N \). Further, \( s_{ij} \) \( R \) \( s_{ij} \) where \( i, j \) are integers and \( 1 \leq i, j \leq N \), and \( R \) represents a one-way/binary relation between the tasks \( s_i \) and \( s_j \), which is shown in the DSM matrix as \( s_{ij} = 1 \). In the case of undirected arrows between the vertices the DSM will become symmetric around the diagonal, but when the edges have a direction it becomes asymmetric, see Figure 2. A mark in position \( (i,j) \) in the DSM means that there exists an edge from task \( i \) to task \( j \). Therefore the marks in row \( i \) tells which inputs are needed and the marks in column \( i \) tells which models that use the output from task \( i \). For example in Figure 2, task D get its input from task A, B and F, and the output from task D is used by task B. Assume that the tasks A-H are models in ARES. To execute model A, input from model E is required, but model E has not been executed yet and the previous iteration output from model E must be used as input to model A. The mark \( (A,E) \) is therefore called a feedback mark just like all superdiagonal marks (marks above the diagonal).

The first step in the DSM method is obviously to translate the system to be analyzed to a DSM matrix, hence finding all the dependencies between the tasks, which in itself can be a demanding task for a large system. The DSM matrix is commonly used for uncovering potential improvements, e.g. finding a better sequence for a new, or already existing system, as ARES. A literature review was performed and concluded that several methods exists for analyzing DSM:s. The DSM created for ARES will be a process DSM, also called activity-based DSM. This type of DSM is part of the category time-based DSM:s. In a time-based DSM the tasks do not exist simultaneously, but rather in a sequential order, and for analyzing such DSM:s partitioning and sequencing are particularly useful, [4]. The goal of the DSM analysis of ARES is to find a model execution order where all models get as fresh data as possible, hence as few feedback marks as possible. However, if one, or several, tasks/models have two-way connections (being dependent on each other) it will not be possible to find an order that eliminates all the feedback marks.

2. Partitioning

Partitioning is the process of transforming an arbitrary DSM to triangular, or block diagonal form. This can be done in various ways. Some examples of partitioning algorithms can be found in [5], [6] and [7]. As a first step in the analysis the two methods the “Reachability method” described by Warfield in [5] and “the Path Search method” described by Gebala and Eppinger in [6] were used.

3. Sequencing

The act of sequencing is finding an optimal sequence/execution order within a DSM, with respect to an objective. The objective can be to minimize the number of feedbacks, as in this analysis, but other objectives are also possible, e.g. finding a sequence minimizing the distance of the feedback marks to the diagonal. In contrast to partitioning, which solely analyses the information available in the DSM, sequencing is an optimization problem that can be hard to solve for large DSM:s.
C. Problem description

As mentioned in the previous section the objective for this analysis is to minimize the number of feedback marks. A DSM with size \( n \times n \) has \( n! \) possible solutions, a DSM of size 10 will have \( 10! = 3.63 \times 10^6 \) possible sequences and the number of feasible solutions grows quickly with the DSM size, e.g. \( 15! = 1.31 \times 10^{12} \) and \( 20! = 2.43 \times 10^{18} \). Although the solution space is finite it grows quickly and an enumeration of all feasible solutions will only be feasible for small problems. The “objective function” (minimum number of feedback marks) is not continuous and there are no derivatives/ gradients to direct the search for the optimum. This restricts the possible solution methods quite drastically. Further, according to [8], the problem can be formulated as a Quadratic Assignment Problem, QAP. QAP:s are well-known combinatorial optimization problems and are classified as NP-complete. [8]

Computational complexity theory was developed to classify problems based on how hard they are to solve, or rather, how much time they require to be solved. Problems classified as P can be solved in polynomial time, which, in general, means that they are “easy”, as the time it takes to solve a problem of class P can be described as a polynomial, \( n^k \), where \( n \) is the size of the input and \( k \) is a constant defined by the problem. NP problems, on the other hand, are hard to solve as there exists no polynomial time algorithm, to the best knowledge of today, to solve these problems. If a NP problem can be transformed in polynomial time, to any other NP-problem it is said to be NP-complete, which is considered to be the hardest problems in NP. This means that if a polynomial time solution algorithm is found for any NP-complete problem, that algorithm could be used for all NP-problems. An example of a NP-complete problem well known to most people is the Sudoku puzzle. [9] Except of being hard to solve even for small sizes (usually 9x9) it also illustrates another defining property of NP problems; if given a solution it can be verified to be correct in polynomial time. This classification of problems builds on a worst-case analysis, which not necessarily is a good measure for the practical solvability of a problem. Some problems have very “bad” worst-case running times, but those worst-cases are very rare and most instances of the problem can be solved effectively even though it is predicted to be hard to solve. Similarly, some very large problems in P might not be feasible to solve, even though they are classified as “easy”, as they would require more time than the age of universe to be solved. [10]

Many problems in operations research and computer science, such as linear integer programming and many combinatorial optimization problems are NP-complete. However, the researchers and professionals faced with those problems have bravely accepted the challenge, and been able to develop solution methods also for NP-complete problems. Those are mainly heuristic methods, which means that instead of only searching for the optimal solution, one is mainly looking for very good feasible solutions. Some NP-complete problems are not possible, or at least very computational heavy to solve by any deterministic optimization method. Then heuristic methods are useful as they can give a “good enough” solution in a reasonable time, [11]. The most used sequencing method used in the literature concerning DSM:s are genetic algorithms, GA. GA is a heuristic method inspired from the natural evolution theory. There also exists a few other approaches in the literature, for example [12], which is a hybrid optimization method combining some heuristic rules together with solving small instances of the problem exactly with Binary Linear Programming, BLP.

4. Genetic Algorithm (GA)

In the DSM literature, genetic algorithms, GA, is by far the most popular sequencing method. Its most prominent features is that it cooperates both global and local search of the solution space and that several feasible solutions are evaluated in parallel. As the name suggests, the basic idea of GA is to mimic the natural evolution process. An initial population of feasible solutions are randomly generated. The initial population is generation zero, and will eventually evolve to a new generation by passing through the three genetic operations selection, crossover and mutation. Each feasible solution is called a chromosome and each gene in the chromosome corresponds to a task in the DSM. [13]

The selection operator performs the “survival of the fittest” and thereby increases the fitness of the population in each iteration. Crossover is exploring the search space globally while mutation performs local search, as well as making sure that genetic combinations that are exterminated by selection still have a chance to prove their fitness, [8]. When the solution has started to converge, the diversity in the population is quite low. The importance of mutation then grows as it increase the diversity and reduces the risk that the GA get stuck in a local optimum. [13]

Each of the genetic operators has its special function and together they create a powerful tool for finding near optimum solutions to complex combinatorial problems. Even though the search is controlled by probabilities, one should not regard it as a simple random search. The GA are governed by the fundamental theorems of schemata and the Building Block Theorem, BBT [13]. If \( V = \{0, 1, *\} \) is the alphabet used for encoding a chromosome, \([0,1]\) are the
values each of the genes can adopt and * is considered as a wildcard that can take any of the two values. A 7-bit chromosome can then be written as $A = 1101001$. This chromosome can be represented by several schemata e.g. $H_1 = 11****1$ or $H_2 = **01***$. The schema $H_1$ has a defining length $\delta(H_1) = 7$ and an order $\omega(H_1) = 3$, and similarly $\delta(H_2) = 2$ and $\omega(H_2) = 2$. According to the theory, schemata with above average fitness will survive and increase its appearance in the coming generations. Schemata with short defining length and high fitness values are referred to as building blocks and will have higher survival rate. The building blocks bring much of the efficiency to the GA:s as they have a high survival rate and much of the evolution effort is to combine different building blocks instead of randomly test arbitrary sequences of genes. For a more rigorous description of the building block hypothesis one may refer to [13].

5. Binary linear programming

Binary linear programming, BLP, is a special case of mixed-integer linear programming, MILP, where all variables are restricted to adopt binary values, $z \in \{0,1\}$, regularly called “decision variables” as they can be interpreted as “1 = yes” and “0 = no”. As mentioned in the previous section, Problem description, the sequencing is classified as NP-complete and no polynomial time algorithm exists to solve the BLP exactly. Which in theory means that only small problems can be solved optimally. However, the border between a “small” and “large” problem is vague and depends on several factors e.g. the time and processor power available. The formulation of the optimization also influences how efficiently it can be solved, [10].

The branch and bound method

The Matlab function intlinprog was used as the BLP solver. This function uses the “branch and bound”, BB, algorithm for solving mixed-integer linear programming problems. BB is the most used method for solving NP-hard/complete problems and utilize a “divide and conquer” strategy, the problem is divided in smaller instances that can be effectively solved, “conquered”. The BB algorithm is an enumerative relaxation algorithm, enumerative because it will enumerate solutions to find its way towards the goal and this is done partly by relaxing the initial problem by disregarding the integer constraints, as a standard linear programming problem is much easier to solve. By fixing the value of more and more variables, branches from the initial point will be created. The branches leading to infeasible solutions, or solutions that are worse than the so far best known solution are pruned, and thus reducing the number of solutions to enumerate. In that way, large portions of the solution space can be discarded from further analysis as it is known that the optimal solution cannot be found there, [10]. The effectiveness of the search for the optimal solution, is very reliant on the lower boundaries that are used to decide if pruning should be performed. This is where the relaxation comes in, by solving the relaxed problem an absolute lower bound can be found for the current branch as $x_i^{PL} \leq x_i^{PL}$ always will be true ($x_i^*$ is the optimal value for branch $i$). However, the relaxed solution may not be very close to the best integer solution for that branch and in intlinprog some different cutting plane methods are implemented to give a better lower bound. The cutting plane methods basically adds more linear constraints that forces the optimal solution to be less fractional, i.e. closer to an integer solution and hence giving a tighter lower bound, [14].

D. Evaluation metrics

To objectively evaluate of the suggested model order improvements three statistical metrics where utilized.

6. Mean Absolute Error, MAE

MAE is calculated by Eq. 6 and gives a mean difference/error between two sets of data points. $\bar{\gamma}$ is the set of data to be compared to the reference data $\gamma$ and $n$ is the number of sample points in the sets. The strength of MEA is that it is easy to interpret as it scales linearly with the errors calculated for each pair of points, [15].

7. Root Mean Square Error, RMSE

When calculating the RMSE, Eq. 7, the errors are squared and does not scale linearly with the size of the errors for each pair of samples. This means that a large error will contribute relatively more to the total error than a smaller error, which makes RMSE harder to interpret than MAE. However, the relation $MAE \leq RMSE$ [15] always holds so by comparing the two metrics some useful information can be obtained. If the values of the two metrics are close to each other that means that the mean error is distributed relatively evenly along the signal, but if the RMSE is much larger than the MAE that is an indication that regions with large discrepancies between the reference and comparison signal exists.
8. Cross-Correlation, CC

CC is a tool widely used within all fields of signal processing to find the similarity of two signals, as a function of time-lag. Imagine two signals that should be compared are plotted together in a graph, the reference signal is then kept stationary while the other is slid along the time-axis and the overlapping area of the two signals is computed. When the overlap is maximum the two signals are the most similar and it is expressed as a shift in time. The cross-correlation of a discrete-time signal is given by Eq. 8. [16] \( R_{xy} \) is the cross-correlation between the two discrete time series \( x[l] \) and \( y[l] \), \( \otimes \) is the symbol for correlation and \( n \) is known as the lag. \( R_{xy}[l] \) is calculated for each sample point between \(-N \leq l \leq N\) where \( N \) is the number of sample points in the signal and the resulting time-lag, \( l' \), between the signals is found where \( R_{xy} \) is maximum, \( \max(R_{xy}[l]) = R_{xy}[l'] \). [16] CC was used to uncover possible time-shifts in the variables compared between the different ARES versions due to the changed model orders.

\[
MAE = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i| \\
RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}}
\]

\( R_{xy}[l] = x[l] \otimes y[l] = \sum_{-N}^{N} x[k] \ast y[k + l] \)

IV. Method

E. DSM generation

When generating the DSM matrix for ARES each model was assumed to be one task and the input and output signals for each model were used to map the dependencies. So according to the DSM theory the model ENGINE get an input signal from the model ATMOS and hence the element value in row ENGINE and column ATMOS was set to 1. As the total number of models was rather large (> 100 pcs) a Matlab script was written to automatically generate the DSM from a list of all the signals (~20000) processed within ARES. However, a great portion of the models are “sub models” contained within other models and only communicate with other submodels under the same main model. Those submodels were therefore collapsed with their respective main models and the number of models could then be reduced to 31. To have a base/reference value for the following analysis it was important that the generated DSM incorporated the current execution order. When examining the current computation order it was realized that not all models could be reordered without major changes to the software (ARES). Those models, where therefore excluded from the reordering analysis, so that the final DSM to be reordered consisted of 21 models. In the Appendix, Figure 3 the generated DSM can be seen, the shaded area highlights the models that was subject to the reordering.

For the models 1-3 both the rows and columns are empty, which means that they have no dependencies. However, these three models have user commands as input and the output is only information to the user, and thus there are no couplings to other models.

F. DSM partitioning

Both the partitioning methods the reachability method and the path search method were implemented in Matlab and used to partition the DSM. However, the results did not show much improvement and it was considered necessary to do a proper sequencing as well.

G. DSM sequencing

Two solution methods were used for sequencing the ARES DSM, a heuristic simple genetic algorithm and a deterministic Binary Linear Programming approach.

9. Genetic algorithm

A simple GA, SGA, was coded in Matlab and used for sequencing the DSM. The selection operator used in the GA developed for this paper uses tournament selection, see [8], [17] and [18], where the tournament size is determined
by the user parameter \( q \). Crossover happens with probability \( p_c \), usually \( \sim 0.6 - 0.9 \) [8] and the crossover operator used was the position based crossover (PBX) [19] also called independent position crossover (IPX) [17]. Shift order mutation was selected for the mutation operator [8], [20]. Mutation occurs with probability \( p_m \).

The termination criteria used is described in [18]. The Convergence Ration, \( CR \), Eq. 1, is calculated from the best fitness value in the current generation, \( F_{\text{best}} \), and the average fitness of the population, \( F_{\text{avg}} \). The algorithm is terminated when \( CR \geq CT \), where \( CT \) is a user-defined Convergence Threshold, suggested in [18] to be 0.9-0.95.

\[
CR = \frac{F_{\text{best}}}{F_{\text{avg}}}
\]  

The GA has four user-input parameters, namely \( q \) (the tournament size), \( p_c \) (probability of crossover), \( p_m \) (probability of mutation) and \( CT \) (Convergence Threshold). Those parameters where determined with guidance from literature [8], [18], [19], [20] and then fine-tuned by calibration to DSM:s with known optimal sequencing found in literature. The final parameter values where set to be \( q = 8 \), \( p_c = 0.8 \), \( p_m = 0.0014 \) and \( CT = 0.9 \). The actual number of generations varied dependent on the size of the DSM block being sequenced, larger blocks required more generations to converge to a stable solution, but most of the runs were performed with 300-1500 generations. The population size was calculated as in [17]. The results are presented below in the Suggested improvements section together with the results from the Binary Linear Programming.

10. Binary Linear Programming

The BLP minimization formulation, eq. 2-5, proposed in [12] had been tested for DSM:s with up to 40 tasks, but after some trials by the author of this paper it was realized that DSM:s with > 100 tasks could effectively be solved. Therefore the heuristic rules were disregarded and the BLP formulation alone was used.

\[
\min_{z} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} z_{ij}
\]

s.t

\[
z_{ij} + z_{ji} = 1 \quad \text{for } 1 \leq i < j \leq n
\]

\[
z_{ij} + z_{ik} + z_{kj} \leq 2 \quad \text{for } i < j, i < k, \text{ and } j \neq k
\]

\[
z_{ij} \in \{0, 1\} \quad \text{for } 1 \leq i, j \leq n, i \neq j
\]

\( z \) is a square matrix with the same size as the DSM (=d), containing all the binary decision variables that should be determined by the optimization. \( z_{ij} = 1 \) if task \( i \) is performed before task \( j \), and otherwise \( z_{ij} = 0 \). The objective function Eq. 2 is simple but clever. Value should be added to the objective function only for superdiagonal marks, which might not seem obvious at a first glance. However, if \( d_{ij} = 1 \) task \( i \) has information dependency on task \( j \), which means that there is only a superdiagonal mark if task \( i \) precedes task \( j \) which also implies that \( z_{ij} = 1 \) and the mark will be counted, as \( d_{ij} \cdot z_{ij} = 1 \). The first constraint, Eq. 3, makes sure that each task occurs once as only one of \( z_{ij} \) and \( z_{ji} \) can be equal to 1 at the same time. The second constraint, Eq. 4, makes sure that each solution is feasible as \( z_{ij} = 1 \), task \( i \) precede task \( j \) and \( z_{ij} = 1 \), task \( j \) precedes task \( k \), then task \( i \) must also precede task \( k \), \( z_{ki} = 0 \). The last constraint, Eq. 5, ensures that \( z_{ij} \) is binary, hence either 1 or 0. To solve the BLP problem the Matlab function \texttt{intlinprog} used, which relies on the branch and bound method described above.

V. Suggested improvements and evaluation

H. Partitioning

The results from the path search method showed very little improvement. The original DSM had 13 feedback marks which was reduced to 12 feedback marks with the path search reordering. The reachability method showed more promising results, but was still only able to reduce the feedback marks from 13 to 9. Because of this, none of these two methods where used for more than the initial sequencing.

I. Sequencing

The number of feedback marks was further reduced with sequencing. The optimal sequence found by the BLP has 5 feedback marks compared to 13 for the currently used order, which is a reduction by 62%. Also the other sequencing
method, the genetic algorithm were able to find the optimal solution with 5 feedback marks. However, as can be seen in the Appendix, Table 2 the BLP is approximately 10 times faster than the GA approach, \( \approx 1.6 \) s for the BLP compared to \( \approx 14 \) s for the GA.

**J. Suggested model order improvements**

As showed in the two previous sections improvements regarding the number of feedback marks can be made. However, to develop a sensible, improved model order for evaluation, it is also important to take into account what signals are part of the feedback marks, i.e. what signals taking their value from the previous iteration. Signals of potentially very dynamic nature are more important to have as fresh as possible, e.g. forces and moment signals, whereas some signals only change slowly (compared to the 120 Hz iteration frequency), e.g. fuel quantity, or not at all (constants) and might therefore be accepted as feedbacks. Because of this it would not necessarily improve ARES to implement the optimal (in the sense of minimum feedback marks) model order found, as that could put a few very dynamic signals as feedbacks instead of a number of slowly changing original ones.

To determine if a signal should be regarded as dynamic or not, the available ARES documentation together with a portion of engineering judgement was used. 5 improvement proposals of the model order were developed and evaluated. Several improvement proposals were used, as the dynamics of the signals was hard to quantify in a reliable way and the exact response of ARES could not be foreseen. To help develop the improvement proposals some models where locked to their original places in the sequence by applying a penalty in the sequencing algorithm for moving them. In that way it was possible to guide/force the sequencing algorithm to fix certain signal paths in a desired order. In the Appendix, Table 3, the feedbacks of the original execution order and the five improvement proposals can be seen. The improvement proposals was implemented in ARES one by one, by making appropriate changes in the ARES source code, written in FORTRAN.

**K. Evaluation**

As reference for the evaluation of the potential improvements the current ARES version released for general use at Saab was used, referred to as the official ARES version or the reference. The evaluation process aimed to find the improvement proposal that matched best with the official ARES version. A set of 5 maneuverers was developed to be the basis of the comparison, the maneuverers were:

- **Pulse**: The simulation starting from a trimmed unaccelerated wings-level flight and then performing a double pulse in pitch, roll and yaw and an increase followed by a reduction in engine thrust. This maneuverer is not used in real flying but gives the aircraft rotations around all axes and induce interesting dynamics in the simulation.

- **Landing**: Starting established on final approach for landing, extending the landing gear, performing a landing and applying brakes during the ground roll. This maneuverer simulates the dynamics caused by extending the landing gear, the ground effect and low speed/high angle of attack flying.

- **Landing 2**: This maneuverer is the same as the Landing maneuverer but during the ground roll a few turns are performed and then a main wheel tire puncture is simulated. This maneuverer additionally tests some of the dynamics that occurs during ground operations.

- **Supersonic**: Initiated in a trimmed subsonic unaccelerated wings-level state, the engine thrust is increased to maximum and the aircraft accelerated to supersonic speed where the thrust is reduced and the airbrakes extended, which leads to a deceleration back to subsonic speed. It tests the behavior in the problematic transonic region and the effects of extending the airbrakes.

- **Chute**: Starting in a steep climb which develops into a deep stall, then recovery is performed with the aid of the pitch moment chute (only fitted to test aircrafts). The stall induces dynamic aerodynamic forces/moments and the chute deployment also creates large, rapid forces/moments which makes this maneuverer interesting to use for comparison.

21 variables were selected to be compared between the different ARES versions; \( \alpha, \beta, \theta, \phi, \psi, m, M, alt, n_p, p, q, r, \dot{p}, \dot{q}, \dot{r}, CN, CT, CC, CMX, CMY, CMZ \). These variables was all considered important for describing the
motion and state of the aircraft, hence relevant to compare between the different ARES versions. Additionally, some timing comparisons was performed where 10 simulations of each maneuver and each ARES version was timed and the average was calculated and compared.

11. Comparison
The comparison was based on the simulations of the five maneuvers presented above, and the three metrics MAE, RMSE and CC. The results was plotted as bar plots similar to Figure 5 and Figure 6 in the Appendix, where the discrepancies between the reference and different model orders were visualized. The Mod:s were ranked for each maneuver according to their conformity (MAE values) to the reference, 1 was given to the Mod with the best fit and 5 to worst. By summing those points for each Mod the model order with the best overall fit to the reference could be found, see Table 1.

L. Evaluation results
Generally, the magnitudes of the discrepancies were small, less than 1 % difference between the official ARES and the modified versions. For the chute maneuver the largest differences were present, an example is shown in Appendix Figure 4 which shows a 1.5% error for $\alpha$.

Table 1 Ranking of the Mod:s for each maneuver, 1=best fit and 5=worst fit.

<table>
<thead>
<tr>
<th></th>
<th>Pulse</th>
<th>Landing</th>
<th>Landing 2</th>
<th>Supersonic</th>
<th>Chute</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mod 1</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>17</td>
</tr>
<tr>
<td>Mod 2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Mod 3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>Mod 4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td>Mod 5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>18</td>
</tr>
</tbody>
</table>

M. Discussion of the evaluation
To enhance the understanding of the different model orders, an investigation of the different versions and their differences was performed. For the maneuvers pulse, landing, landing 2 and supersonic the ARES version Mod 2 was superior, with almost no discrepancy at all for any variable, see Table 1. Mod 4 had the worst fit, while Mod 5 was quite good for some variables but bad at others. Mod 1 and 3 performed moderate and was following each other close for most of the variables, although Mod 1 was a little worse than Mod 3. For the chute maneuver the results was a bit different, Mod 5 showed the best fit to the reference while Mod 2 was a bit behind but still second best. Mod 1 and 3 still follow each other but performed worst, Mod 4 was only a little better than Mod 1 and 3.

When tracing those results back to the actual feedbacks some conclusions could be drawn, however, all behaviors could not be pinpointed due to the intricate modelling involved. The fact that Mod 2 had the best fit to the reference for most of the maneuvers can be attributed to that 4 of its 6 feedbacks are also found in the original model order, see Table 3. The other two feedbacks contain relatively “slow” signals, fuel quantities (FUELQ_*) and hydraulic pump rotor speed(N_HR). Mod 4 has the aerodynamic forces and moments as feedback signals which probably leads to the bad fit to the reference. The total forces and moments are common feedback signals for Mod 1 and 3, which explains why they are following each other so tightly. The behavior of Mod 5 is harder to pair with any specific feedback, but one reason could be that four models in Mod 5 have feedbacks from the Incidence model, which includes state variables such as e.g. $\alpha$, $\beta$ and $M$. Anyhow, Mod 5 is performing best on the chute maneuver, but why not Mod 2 is best on this maneuver as well could not be established. Mod 1, 3 and 4 have feedbacks from the Forandmom model and for the chute maneuver this is probably the reason for poorer performance from these Mod:s, due to the very quick and large forces and moments produced by the deep stall and pitch moment chute.

The CC analysis revealed that time-shifts occurred in some of the variables, but the number, and actual variables, differed between the different Mod:s. Just as for the MAE/RMSE analysis the magnitude of the shifts was small (~1-10 iterations) and their cause hard to determine. Mod 2 did not have any variable that was shifted and Mod 1 and 3 were behaving very similar, but there is no clear pattern as they for some variables are lagged and for other advanced.
For pulse, landing, landing 2 and supersonic Mod 5 also had both lagged and advanced variables, for chute it only had four variables that lagged one iteration each. The majority of the variables that time-shifted in Mod 4 was lagged except for the chute maneuver, where Mod 4 only had advancing shifts.

The results from the execution time comparison can be seen in the Appendix, Table 4. Only small differences was observed between the different simulations and hence was the “Time” simulation added, see Table 4. Time is simulating a 1000s cruise flight with the sole purpose of magnifying execution time differences between the ARES versions. The results from the “Time” simulations showed very similar results and the differences between them cannot unambiguously be related to the differences in the model order without further analysis.

VI. Discussion

Solving a NP-complete problem depends on several factors, mainly its size and type (e.g. decisional or functional problem). Exact methods/algorithms can, as have been discussed above, only be used for relatively small problem sizes, while for larger instances approximate methods is mostly used [21]. Sequencing the model order of the current ARES appeared to be sufficiently small to be effectively solved exact. However, as this was not known from the beginning, a heuristic method where also developed and tested, namely the genetic algorithm. Even if the GA did not perform as good as the BLP it still had a reasonable execution time and close to optimal, or optimal result, and was kept as a comparison to the BLP. The differences between the modified ARES versions showed to be very small.

Attempts to compare the modified ARES output directly to flight test data was made, but no meaningful conclusions could be drawn. This was mainly because it is very hard to model exactly the same conditions as those the real test aircraft was experiencing, which is necessary to find valid conclusions. The official ARES version have been tested and verified against the real aircraft and was therefore considered as the best feasible comparison reference.

The cross-correlation metric, CC, was used to find time shifts in the variables between the different ARES versions, but it showed to be misleading sometimes. If a part of the signal are shifted but the majority of the signals are not, the time-shift will be underestimated at some parts of the signal and overestimated at other parts as the CC tries to find the best fit for the entire signal. This behavior was especially observed in the comparison between the two maneuvers Landing and Landing 2.

A central question to be asked is whether the improved model order should be applied or not, as it might induce errors that will show up later. However, as part of this work is to make a foundation for future modifications and extensions of ARES it is important to have a well-defined initial starting point that can be explained by logical arguments. Further, as not all possible maneuvers and actions of ARES have been evaluated the suggested model order should in theory be advantageous over the current order, as more models get fresh data.

VII. Conclusions

After the dependency study was performed, the DSM matrix for ARES could be created. It was then sequenced both by a Binary Linear Programming approach and a Genetic Algorithm to find the optimal model order. Based on the optimal model order found five improved model orders was suggested. The influence of changing the model execution order of the output of ARES simulations was small. By analyzing the small differences that was found, a model order, superior to the original, could be established. This improvement proposal gave almost exactly the same result as the official ARES, for the variables analyzed, but reduced the number of feedback marks from 13 to 6, a reduction by 54 %. To further increase the confidence in the proposed model order it may be advisable to test more maneuvers and/or variables and compare to the official ARES. A possible continuation of this work could be to make a deeper analysis of the signals being part of the feedbacks by quantifying the “dynamicness” of all signals and minimizing the “dynamicness” of the feedbacks, rather than the number. One aspect that have not been considered during this project is to find sets of models that could be executed simultaneously. That could potentially decrease the execution time for simulations. Another possible performance enhancer could be to run different models in different frequencies. As discussed earlier in the report, some signals are more dynamic than others and the iteration frequency have to be decided by the most demanding models/signals, but that frequency might be too high for other models. If different models could be executed with different frequencies that would reduce the processor capacity needed for simulations. As an example, the 120 Hz frequency is needed for the most dynamic models (Aero, Forandmom, etc), while e.g. the Fuel model might only need to have a update frequency of 4 Hz, so instead of running it each of the 120 iterations per
second it could be executed once every 30th iteration. One step even further is to not only considering the models, but the signals alone and split up models so that the dynamic signals are one part updated with higher frequency than the part with not so dynamic signals.

**Acknowledgments**

To start with, I would like to sincerely thank my company supervisor Alexandra Oprea for the opportunity and for all help along the way. Further I want to thank Andreas Nilsson, and the rest of the team at Flight mechanics and Performance for always assisting me in the best possible way and for making me feel welcome to Saab. Last, but not least, thank you Ulf Ringertz for being my examiner at KTH. I could not have wished for a better way to finish my studies than here at Saab. It has not only been a master thesis project, but also an eye-opener to what possibilities there are for an aspiring aeronautical engineer.

**References**


Appendix

Table 2 Execution time ($t_{ex}$) and $F^*$ comparison of the different sequencing methods.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>BLP</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{ex}$ [s]</td>
<td>1.58</td>
<td>13.87</td>
</tr>
<tr>
<td>$\sigma_{ex}$ [s]</td>
<td>0.05</td>
<td>7.27</td>
</tr>
<tr>
<td>$F^*$</td>
<td>5</td>
<td>5.35</td>
</tr>
</tbody>
</table>

Table 3 Feedback marks in original execution order and in the proposed improvement orders (“Mod 1-5”)

Figure 3 DSM for official ARES. Each number in the first column and row corresponds to a model in ARES. The shaded area are the models that take part in the sequencing.
Table 4  Average execution times, $t_{ex}$, and standard deviation, $\sigma_{t_{ex}}$, for the different maneuvers and ARES versions.

<table>
<thead>
<tr>
<th>Version</th>
<th>Measurement</th>
<th>Pulse</th>
<th>Landing</th>
<th>Landing 2</th>
<th>Supersonic</th>
<th>Chute</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>$t_{ex}$ [s]</td>
<td>1.980</td>
<td>5.966</td>
<td>5.772</td>
<td>3.642</td>
<td>4.715</td>
<td>37.942</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{t_{ex}}$ [s]</td>
<td>0.12</td>
<td>0.03</td>
<td>0.29</td>
<td>0.15</td>
<td>0.24</td>
<td>0.31</td>
</tr>
<tr>
<td>Mod 1</td>
<td>$t_{ex}$ [s]</td>
<td>1.857</td>
<td>5.861</td>
<td>6.074</td>
<td>3.528</td>
<td>4.974</td>
<td>38.735</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{t_{ex}}$ [s]</td>
<td>0.06</td>
<td>0.31</td>
<td>0.20</td>
<td>0.15</td>
<td>0.22</td>
<td>2.30</td>
</tr>
<tr>
<td>Mod 2</td>
<td>$t_{ex}$ [s]</td>
<td>1.863</td>
<td>5.684</td>
<td>5.804</td>
<td>3.610</td>
<td>4.685</td>
<td>38.469</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{t_{ex}}$ [s]</td>
<td>0.08</td>
<td>0.15</td>
<td>0.30</td>
<td>0.13</td>
<td>0.20</td>
<td>0.85</td>
</tr>
<tr>
<td>Mod 3</td>
<td>$t_{ex}$ [s]</td>
<td>2.018</td>
<td>5.693</td>
<td>5.900</td>
<td>3.881</td>
<td>4.592</td>
<td>37.836</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{t_{ex}}$ [s]</td>
<td>0.08</td>
<td>0.14</td>
<td>0.13</td>
<td>0.16</td>
<td>0.17</td>
<td>0.98</td>
</tr>
<tr>
<td>Mod 4</td>
<td>$t_{ex}$ [s]</td>
<td>1.948</td>
<td>5.816</td>
<td>6.039</td>
<td>3.781</td>
<td>4.796</td>
<td>39.175</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{t_{ex}}$ [s]</td>
<td>0.07</td>
<td>0.19</td>
<td>0.11</td>
<td>0.13</td>
<td>0.28</td>
<td>0.98</td>
</tr>
<tr>
<td>Mod 5</td>
<td>$t_{ex}$ [s]</td>
<td>1.965</td>
<td>5.710</td>
<td>5.972</td>
<td>3.613</td>
<td>4.68</td>
<td>38.560</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{t_{ex}}$ [s]</td>
<td>0.10</td>
<td>0.10</td>
<td>0.07</td>
<td>0.13</td>
<td>0.22</td>
<td>1.33</td>
</tr>
</tbody>
</table>

Figure 4  The variation of angel of attack, $\alpha$, for all Mod:s and the official ARES for the chute maneuver. This was among the largest discrepancies ($\pm \sim 0.5^\circ$) found for all variables and maneuvers.
Figure 5  Bar plot for the Normalized MAE metric for the pulse maneuver. Mod 2 show no discrepancy at all while Mod 4 is worst for all variables.

Figure 6  Bar plot of the offsets from the pulse maneuver. For Mod 2 no offsets are detected but for the other Mods one iteration plus or minus can be seen.