Università degli Studi di Napoli Federico II

Scuola Politecnica e delle Scienze di Base

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE
Corso di Laurea in Ingegneria Aerospaziale

Tesi di Laurea Magistrale
in
Ingegneria Aerospaziale ed Astronautica

Development of a Java Framework for Parametric Aircraft Design

The Performance Analysis Module

Candidato:
Vittorio Trifari
Matricola M53/411

Relatori:
Prof. Fabrizio Nicolosi
Prof. Agostino De Marco

Anno Accademico 2014/2015

Abstract

This thesis has the main purpose of providing a comprehensive overview about the development, in Java, of a software dedicated to the preliminary design of an aircraft, focusing on the performance analysis module.

The point of view from which this subject will be observed expects first to define methodologies and theoretical aspects necessary for the examined performance calculation for then show, in more detail, the implementation of the latter within the software; this will be seen both from the point of view of the developer, through a detailed explanation of the architecture of the different calculation modules, both of a potential user developer, by showing some commented examples of use supplied with graphical and numerical results suitable, among other things, also to the validation of calculations performed.

Sommario

Questo lavoro di tesi ha lo scopo principale di fornire una panoramica esaustiva circa lo sviluppo, in ambiente Java, di un software dedicato al progetto preliminare di un aeromobile, concentrandosi sull'aspetto dell'analisi delle performance.

Il punto di vista con il quale verrà affrontato questo argomento prevede dapprima di inquadrare le metodologie e gli aspetti teorici necessari per la stima delle performance in esame per poi mostrare, più nel dettaglio, l'implementazione di questi ultimi all'interno del software; ciò sia dal punto di vista dello sviluppatore, attraverso la spiegazione dettagliata dell'architettura dei vari moduli di calcolo, sia dal punto di vista di un potenziale utente, tramite alcuni esempi d'uso commentati e corredati di risultati grafici e numerici atti, tra l'altro, anche alla validazione dei calcoli svolti.

Contents

Chapter 1

Software overview

Chapter 2

Payload-Range

The Payload-Range diagram is a very important resource in aircrafts design because it allows to illustrate an aircraft operational limits by showing the trade-off relationship between the payload that can be carried and the range that can be flown; moreover, in synergy with the D.O.C. diagram, it's very useful to the future aircraft customers to have an idea of expected profits derived by their investment.

Payload-Range is also a key feature of the design requirements so it's possible to state that its relevance is felt through all the preliminary and conceptual design phases.

2.1 Theoretical background

To better understand the theoretical background behind this diagram, the first step is to focus the attention upon the link between the range and the fuel consumption, which is influenced by aircraft weight through Breguet formulas.

It's possible to imagine, at first, an aircraft in which payload and fuel weights can be managed at will, so that, in this completely flexible aircraft, the more the required range is, the less is the payload in order to carry more fuel at a specified maximum take-off weight; this leads to the diagram in figure 3.1.

Because the payload is carried in the fuselage while the fuel is carried in wing tanks, the more the payload, the more the weight in the fuselage; this means that wings are incrementally more bending loaded so that a heavier structure is required. However, in this way the operating empty weight grows up limiting the payload the aircraft can carry, at that maximum take-off and fuel weights, for that range; the consequence of this is that, in particular for long range missions, the payload is too low, providing low profits.

In order to avoid this situation it's possible to decide, preventively, the maximum payload weight the aircraft can carry in fuselage so that the maximum bending load for wings and, in consequence their structural weight, is set; this particular weight is named maximum zero fuel weight (MZFW).

As a result of this, the previous diagram changes in the one reported in Figure 3.2 which

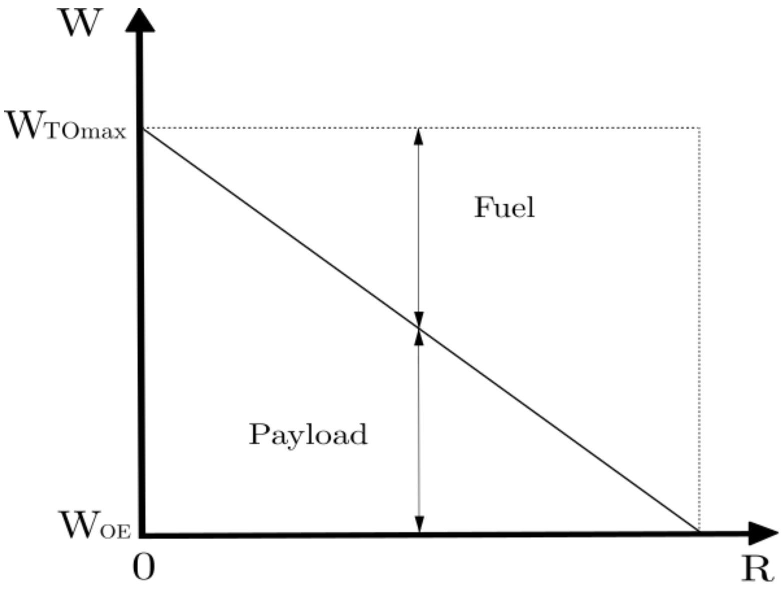

Figure 2.1 Payload-Range for a completely flexible aircraft

starts with the maximum zero fuel weight and keep it constant until the maximum take-off weight is reached. From this point on, the far the aircraft have to go, the more the fuel it needs, but, since the maximum take-off weight is set, the only way to increase fuel weight is to reduce the payload; this condition is represented by the second segment of the diagram and, along this, the aircraft weight is always equal to the maximum take-off weight. With further increase of the required range for the aircraft, the fuel tanks maximum capacity will be reached prevent to store more fuel; the only solution at this problem is to put the additional fuel into the fuselage, reducing, consequently, the payload weight in order to respect the maximum zero fuel weight limitation. Along this last segment the payload weight decreases more rapidly with increasing range reaching, ideally, the value of zero at which the flight mission is useless; moreover the aircraft weight is not equal to the maximum take-off weight anymore but to a lower one.

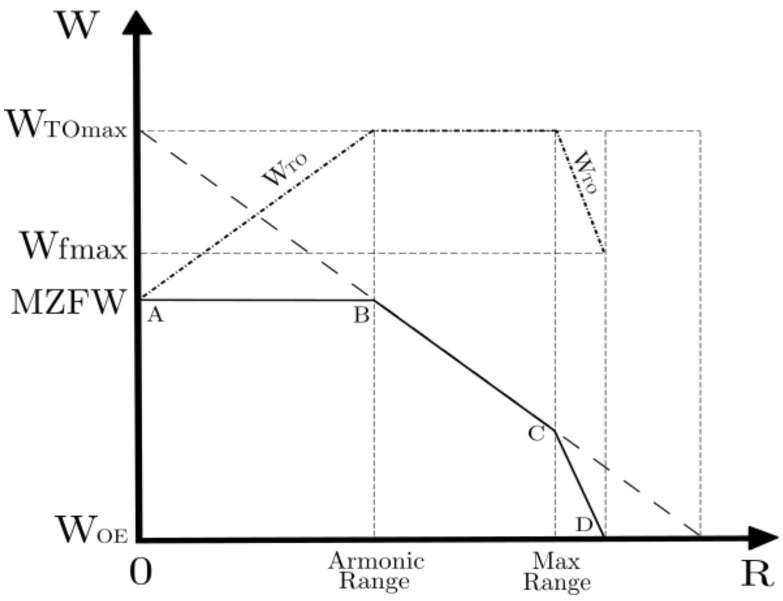

Figure 2.2 Payload-Range with MZFW limitation

Now that the theory behind the Payload-Range is explained, it's interesting to see how to build up the diagram. For this purpose at least the preliminary weight estimation has to be done in order to know the value of the following weights.

- $\mathrm{W_{TO}}$
- $\mathrm{W_{OE}}$
- $\mathrm{W_{Payload}}$

Furthermore four fundamental couples of payload and range values have to be defined:

- Point A, in which the payload is the maximum allowed and the range is zero.
- Point B, in which the maximum range, with maximum take-off weight and maximum payload, is reached. This range value is called Armoinc Range.
- Point C, in which the fuel tanks full capacity is reached with a payload lower than the maximum one and still the maximum take-off weight.
- Point D, in which the take-off weight decreases because of the reduction of payload to zero in order to store more fuel in the fuselage.

Since point A is very simple to obtain, it's more interesting to analyze how to calculate points B, C and D coordinates.

For point B it's necessary to focus the attention upon Breguet formulas in order to calculate the maximum range the aircraft can fly at maximum take-off weight with maximum payload; these are different for jet or propeller aircraft and can be written as follow, with R in km, SFC in $\frac{\mathrm{lb}}{\mathrm{hp \cdot h}}$ and SFCJ in $\frac{\mathrm{lb}}{\mathrm{lb \cdot h}}$.

$$R = 603.5 \cdot \left(\frac{\eta_p}{SFC}\right)_{cruise} \cdot \left(\frac{L}{D}\right)_{cruise} \cdot \ln\left(\frac{W_i}{W_f}\right) \qquad \text{propeller aircraft} \tag{2.1}$$

$$R = \left(\frac{V}{SFCJ}\right)_{cruise} \cdot \left(\frac{L}{D}\right)_{cruise} \cdot \ln\left(\frac{W_i}{W_f}\right) \qquad \text{jet aircraft} \tag{2.2}$$

Knowing the specific fuel consumption, the aerodynamic and propeller efficiency, or speed for jet aircraft, in cruise condition, the only unknown left for calculate the range is the weight ratio. In order to calculate this it's necessary to start by evaluating the amount of fuel the airplane can take on board, at maximum take-off weight and maximum payload, with the following formula.

$$W_{TO,max} = W_{OE} + W_{payoad,max} + W_{fuel} \tag{2.3}$$

Once the used fuel weight is known, it's possible to use the fuel fraction method from the weight estimation design phase in order to find the total weight ratio across the entire aircraft mission.

$$W_{fuel} = \left(1 - M_{ff}\right) \cdot W_{TO} \tag{2.4}$$

M_{ff} is the fuel fraction of the entire mission profile given by the following expression.

$$M_{ff} = \frac{W_1}{W_{TO}} \cdot \frac{W_2}{W_1} \cdot \frac{W_3}{W_2} \cdot \frac{W_4}{W_3} \cdot \frac{W_5}{W_4} \cdot \frac{W_6}{W_5} \cdot \frac{W_7}{W_6} \cdot \frac{W_8}{W_7} \cdot \frac{W_9}{W_8} \cdot \frac{W_{10}}{W_9} \cdot \frac{W_{fianl}}{W_{10}} \tag{2.5}$$

From this point on, it's necessary to have a table, like the one of table 2.1, in which all weight ratios can be found statistically except for cruise, alternate cruise and loiter phases; these three unknown ratios are those that build up the Breguet weight ratio required by the range formula.

Airplane type	**Start, Warm-up**	**Taxi**	**Take-off**	**Brief descent**	**Brief climb**	**Landing, Taxi and Shutdown**
Homebuilts	0,998	0,998	0,998	0,995	0,995	0,995
Single Engine	0,995	0,997	0,998	0,992	0,993	0,993
Twin Engine	0,992	0,996	0,996	0,990	0,992	0,992
Agricultural	0,996	0,995	0,996	0,998	0,999	0,998
Business Jets	0,990	0,995	0,995	0,980	0,990	0,992
Regional TBP's	0,990	0,995	0,995	0,985	0,985	0,995
Transport Jets	0,990	0,990	0,995	0,980	0,990	0,992
Mil. Trainers	0,990	0,990	0,990	0,980	0,990	0,995
Fighters	0,990	0,990	0,990	0,960-0,900	0,990	0,995
Mil.Patrol,Bmb and Trspt	0,990	0,990	0,995	0,980	0,990	0,992
Flying boats,Amph. and Floats	0,992	0,990	0,996	0,985	0,990	0,990
Supersonic Cruise	0,990	0,995	0,995	0,920-0,870	0,985	0,992

Table 2.1 Suggested fuel fractions

For the evaluation of point C, similar steps have to be followed with the difference that, in this case, the fuel is the maximum that the wing tanks can store and the payload is the unknown of the (2.6).

$$W_{TO,max} = W_{OE} + W_{payload} + W_{fuel,max} \tag{2.6}$$

From this equation it's possible to calculate point C ordinate, while for the range abscissa the same procedure used for point B range has to be followed with the difference that now it's necessary to use the fuel weight relative to the maximum fuel tank capacity which is known from the wing fuel tank design.

Finally for point D, the payload is set at zero so that is possible to evaluate the WTO, relative to maximum fuel weight with no passengers, from the following equation.

$$W_{TO} = W_{OE} + W_{fuel,max} \tag{2.7}$$

This new take-off weight generates a lower M_{ff}, from fuel fraction equation, which leads to a bigger weight ratio to be used in Breguet formulas with the result of a bigger range.

2.2 Java class architecture

In this paragraph the implementation in the JPAD software of the Payload-Range diagram, through the `calcPayloadRange` Java class, is presented. The idea has been to create a dedicated Java class which is demanded of the calculation of the four couple of range and payload values presented before; moreover it has to confront the user chosen Mach number condition with the best range one and to parametrize the analysis at different maximum take-off weight in order to help users making design decisions about different version of the same aircraft.

The class core consists of the following three principal methods which have to evaluate points B, C and D coordinates using the procedure explained before.

- `calcRangeAtMaxPayload`
- `calcRangeAtMaxFuel`
- `calcRangeAtZeroPayload`

Each of these requires in input the table 2.2 data, a database which collects all data from table 2.1, named *FuelFractions.h5*, and an engine database, for turboprop or turbofan, in which all specific fuel consumption values, at given Mach number and altitude, are stored.

From this point on, the evaluation of (2.3), (2.6) and (2.7), for each method, and of (2.4), for all of them, is performed in order to set up the following calculations and to obtain the unknown value of the payload in point C case; after that the fuel fractions database is read in order to obtain all known data necessary to calculate the required weight ratio to be used in (2.1) or (2.2) depending on the aircraft engine type. It's important to highlight that the database reading process is made up so that it can recognize all different kind of aircraft from table 2.1 and read the related line values.

`maxTakeOffMass`	Maximum take-off mass
`sweepHalfChordEquivalent`	Equivalent wing sweep angle at half chord
`surface`	Wing surface
`cd0`	Wing c_{D0}
`oswald`	Wing oswald factor
`cl`	The current lift coefficient in cruise configuration
`ar`	Wing aspect ratio
`tcMax`	Mean maximum thickness of the wing
`byPassRatio`	Engine by-pass ratio (when needed)
`eta`	Propeller efficiency (when needed)
`altitude`	Cruise altitude
`currentMach`	The actual Mach number during cruise
`isBestRange`	A boolean variable that is true if the evaluation of the best range condition is performed, otherwise it's false

Table 2.2 Input data

Since each method has to compare the chosen Mach number condition with the best range one, the boolean variable presented before is used in order to distinguish between different application cases; in this way, when the user specify the aircraft engine type and the boolean variable, the method reads the specific fuel consumption value, at given Mach number and altitude, from the related engine database and performs the calculation of the lift and drag wing coefficients which are then used to obtain the aerodynamic efficiency value. At this point is possible to calculate the range that represents point B, C or D abscissa.

It should be noted that, in case of a true value of the boolean variable, the evaluation of the lift and drag wing coefficients, as well as the best range Mach number that replaces the user chosen one, is performed by calculating the parabolic drag polar characteristic points and choosing those that maximizes the range (point E for the turboprop and point A for the turbofan); whereas for a false value of the boolean variable, the lift coefficient is calculated by using the current flight condition angle of attack into the linear part of the wing lift curve, while the drag coefficient is evaluated from the aircraft total drag polar taking also into account potential wave drag sources.

Now that all points coordinates are known, the two following methods are demanded to build up abscissas and ordinates values arrays to be used in plotting the diagram.

- `createRangeArray`
- `createPayloadArray`

They use the Java `List` interface to generate an ordered collection of values which are then populated with the following values.

Range Array	Payload Array
0,0	Maximum payload, in number of passengers
`calcRangeAtMaxPayload` output	Maximum payload, in number of passengers
`calcRangeAtMaxFuel` output	Payload calculated in `calcRangeAtMaxFuel`
`calcRangeAtZeroPayload` output	0,0

Table 2.3 Payload and range array components

Finally the two arrays are used as input, together with another one from the best range condition analysis, for the last method that has to plot the diagram; this is called `createPayloadRangeCharts_Mach` and uses the `JFreeChart` Java library to generate a .png output image into the output folder of the software and is also able to create a .tikz version of the diagram to be used in LaTeX.

As said before, this class finds another purpose in parameterizing the diagram in different maximum take-off weight conditions. To do that, the three core methods are used again inside a new one, named `createPayloadRangeMatrices`, which implements a recursive calculation of the three points coordinates at different maximum take-off mass conditions generated by decreasing the original mass by 5% until it reaches a total decrease of 20%. This new method requires the same input data reported in table 2.2 except for the maximum take-off mass which is generated inside the method itself.

The output matrices are then used in a plotting method, equivalent to previous one but named `createPayloadRangeCharts_MaxTakeOffMass`, which generates the .png and .tikz output images by receiving as input two matrices and not two arrays as before.

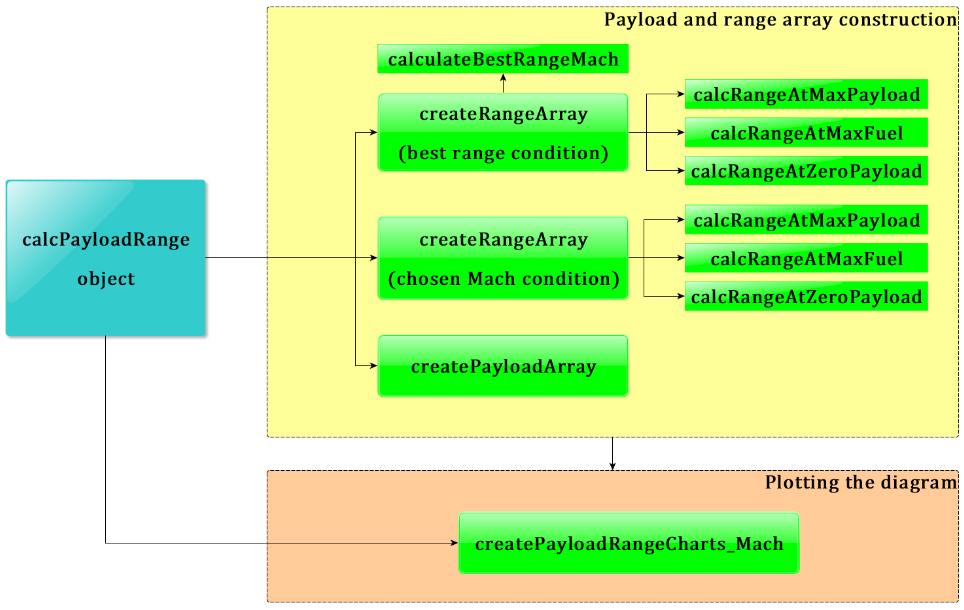

Figure 2.3 Payload-Range java class flowchart for best range and chosen Mach conditions comparison

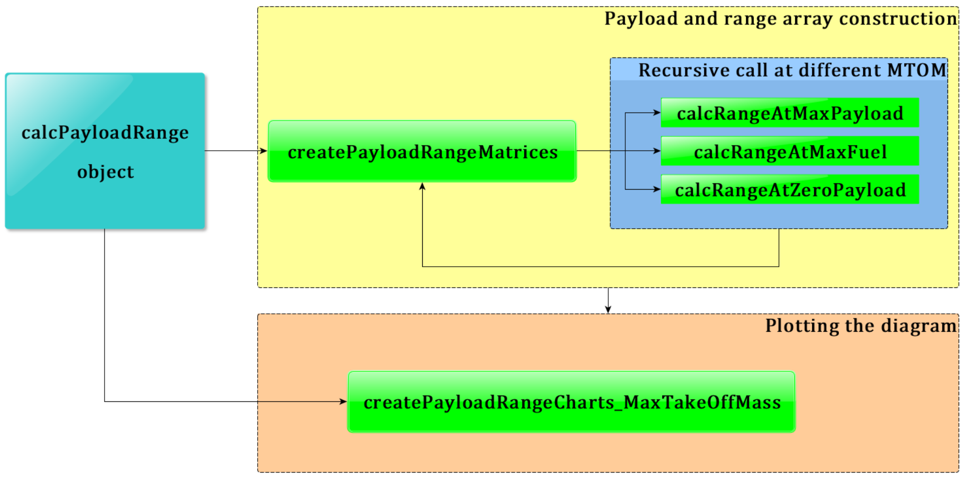
Payload and range array construction
Recursive call at different MTOM
calcPayloadRange object
createPayloadRangeMatrices
calcRangeAtMaxPayload
calcRangeAtMaxFuel
calcRangeAtZeroPayload
Plotting the diagram
createPayloadRangeCharts_MaxTakeOffMass

Figure 2.4 Payload-Range java class flowchart for maximum take-off mass parameterization

2.3 Case study: ATR-72 and B747-100B

With the purpose of validating calculations presented before, two case studies have been taken into account; the first one is on the ATR-72 and the second one on the B747-100B.

The ATR-72, made by the French-Italian aircraft manufacturer ATR, is a stretched variant of the original ATR-42 that entered in service in 1989; it's powered by two turboprop engine and it's used typically as a regional airliner. The main purpose of its design was to increase the seating capacity throughout the stretching of the fuselage by 4.5m, an increased wingspan, more powerful engines and an increased fuel capacity by approximately 10%.

The Boeing 747 is a wide-body commercial jet airliner and cargo aircraft, often referred to by its original nickname, Jumbo Jet, or Queen of the Skies. Its distinctive hump upper deck along the forward part of the aircraft makes it among the world's most recognizable aircraft, and it was the first wide-body produced. Manufactured by Boeing's Commercial Airplane unit in the United States, the original version of the 747 had two and a half times greater capacity than the Boeing 707, one of the common large commercial aircraft of the 1960s. The four-engine 747 uses a double deck configuration for part of its length and it's available in passenger and other versions. Boeing designed the 747's hump-like upper deck to serve as a first class lounge or extra seating, and to allow the aircraft to be easily converted to a cargo carrier by removing seats and installing a front cargo door. The 747-100B model was developed from the 747-100SR, using its stronger airframe and landing gear design; the type had an increased fuel capacity of 182000 L, allowing for a 5000 nautical mile range with a typical 452 passengers payload, and an increased maximum take-off weight of 340000 kg was offered; unlike the original 747-100, the 747-100B was offered with Pratt & Whitney JT9D-7A, General Electric CF6-50, or Rolls-Royce RB211-524 turbofan engines.

The two presented tests share the same architecture so that a general guideline can be followed; this strategy has been designed with the purpose of making a general procedure in order to simplify user's work.

For more clarity, listings of the two test types are reported, as an exemplificative practice, only in the case of the ATR-72 turboprop aircraft.

First of all it's necessary to set up all default folders in order to make them achievable from any location inside the software; this is done with the static method `initWorkingDirectoryTree` of the `MyConfiguration` class located in `JPADConfigs` package. In this way, every time the user wants to point at a specific folder, like the input or output directory, it's necessary only to call the static method `getDir`, also from `MyConfiguration` class, which reads the folder name, from a dedicated enumeration, and associate it with the related directory from a `HashMap` named `mapPaths`.

The second step is to read the engine and the fuel fractions databases with the related java class reader; this particular class type uses the super class `DatabaseReader`, which implements the use of `MyHDFReader` class, to access an .h5 dataset values. For more information about how to build and use an HDF database, see Appendix A.

```java
public class PayloadRange_Test_TP{
  //----------------------------------------------------
  // MAIN:
  public static void main(String[] args) throws
          HDF5LibraryException, NullPointerException{
    System.out.println("----------------------------");
    System.out.println("PayloadRangeCalc_Test :: main");
    System.out.println("----------------------------");
    //------------------------------------------------
    // Assign all default folders
    MyConfiguration.initWorkingDirectoryTree();
    //------------------------------------------------
    // Setup database(s)
    String databaseFolderPath = MyConfiguration.getDir(FoldersEnum.DATABASE_DIR);
    String aerodynamicDatabaseFileName = "Aerodynamic_Database_Ultimate.h5";
    String fuelFractionDatabaseFileName = "FuelFractions.h5";
    AerodynamicDatabaseReader aeroDatabaseReader = new AerodynamicDatabaseReader(
            databaseFolderPath,
            aerodynamicDatabaseFileName
            );
    FuelFractionDatabaseReader fuelFractionReader = new FuelFractionDatabaseReader(
            databaseFolderPath,
            fuelFractionDatabaseFileName
            );
```

Listing 2.1 Excerpt of the ATR-72 Payload-Range test - preliminary steps

Now that all required resources are set up, following steps are to create the aircraft model, assign its operating conditions and perform all necessary analysis like the aerodynamic one. This is done throughout the use of the following three fundamental classes, which modes of operation are explained in the flowchart of figure 3.5.

- `Aircraft` class
- `OperatingConditions` class
- `ACAnalysisManager` class

```java
//---------------------------------------------------
// Operating Condition / Aircraft / AnalysisManager
// (geometry calculations)
OperatingConditions theCondition = new OperatingConditions();

Aircraft aircraft = Aircraft.createDefaultAircraft("ATR-72");

aircraft.get_theAerodynamics().set_aerodynamicDatabaseReader(aeroDatabaseReader);
aircraft.get_theFuelTank().setFuelFractionDatabase(fuelFractionReader);
aircraft.set_name("ATR-72");
aircraft.get_wing().set_theCurrentAirfoil(new MyAirfoil(
        aircraft.get_wing(),
        0.5)
         );

ACAnalysisManager theAnalysis = new ACAnalysisManager(theCondition);
theAnalysis.updateGeometry(aircraft);

//---------------------------------------------------
// Set the CoG(Bypass the Balance analysis allowing
// to perform Aerodynamic analysis only)
CenterOfGravity cgMTOM = new CenterOfGravity();

// x_cg in body-ref.-frame
cgMTOM.set_xBRF(Amount.valueOf(12.0, SI.METER));
cgMTOM.set_yBRF(Amount.valueOf(0.0, SI.METER));
cgMTOM.set_zBRF(Amount.valueOf(2.3, SI.METER));

aircraft.get_theBalance().set_cgMTOM(cgMTOM);
aircraft.get_HTail().calculateArms(aircraft);
aircraft.get_VTail().calculateArms(aircraft);

theAnalysis.doAnalysis(aircraft, AnalysisTypeEnum.AERODYNAMIC);
```

Listing 2.2 Excerpt of the ATR-72 Payload-Range test - Aircraft, OperatingConditions and ACAnalysisManager setup

All data required by the `calcPayloadRange` class are now ready to be used and it's possible to create an istance of this class accessing, in this way, to all its methods, which have been explained in the previous paragraph.

```java
//---------------------------------------------------
// Creating the Calculator Object
PayloadRangeCalc test = new PayloadRangeCalc(
            theCondition,
            aircraft,
            AirplaneType.TURBOPROP_REGIONAL
            );
```

Listing 2.3 Excerpt of the ATR-72 Payload-Range test - Payload-Range class istance creation

The first check that is implemented has the purpose of inspect wether or not the chosen cruise Mach number is bigger than the crest critical one; in this case a warning message is launched in order to inform the user of the situation. The method demanded of this is `checkCriticalMach` which accepts in input a given Mach number, compares it with the one calculated with Kroo method [5], starting from the cruise lift coefficient, and return a boolean variable that is true only if the chosen Mach number is bigger than the crest critical one indeed.

```java
    // ----------CRITICAL MACH NUMBER CHECK-------------
    boolean check = test.checkCriticalMach(theCondition.get_machCurrent() );

    if (check)
      System.out.println("\n\n------------------------"
      +"\nCurrent Mach lower then critical Mach number"
      +"\nCurrent Mach = " + theCondition.get_machCurrent()
      +"\nCritical Mach = " + test.getCriticalMach()
      +"\n\n\t CHECK PASSED -> PROCEDING TO CALCULATION "
      +"\n\n"
      +"-----------------------------------------------");
    else{
      System.err.println("\n\n------------------------"
      +"\nCurrent Mach bigger then critical Mach number"
      +"\nCurrent Mach = " + theCondition.get_machCurrent()
      +"\nCritical Mach = " + test.getCriticalMach()
      +"\n\n\t CHECK NOT PASSED -> WARNING!!! "
      +"\n\n"
      +"--------------------------------------");
    }
```

Listing 2.4 Excerpt of the ATR-72 Payload-Range test - critical Mach number check

```
---------------------------------------------------------
Current Mach is lower then critical Mach number.
Current Mach = 0.43
Critical Mach = 0.6659791543529567

   CHECK PASSED --> PROCEDING TO CALCULATION
---------------------------------------------------------
```

Listing 2.5 Excerpt of the ATR-72 Payload-Range test results - critical Mach number check

```
---------------------------------------------------------
Current Mach is lower then critical Mach number.
Current Mach = 0.84
Critical Mach = 0.8490814607347361

   CHECK PASSED --> PROCEDING TO CALCULATION
---------------------------------------------------------
```

Listing 2.6 Excerpt of the B747-100B Payload-Range test results - critical Mach number check

At this point, if the user wants to compare the chosen Mach number condition with the best range one, all that it's needed is to invoke the `createRangeArray` method, for both the operative conditions, and the `createPayloadArray` one; after that the diagram is generated from the method `createPayloadRangeCharts_Mach`.

```
// ----------------BEST RANGE CASE-------------------
System.out.println();
System.out.println("-------BEST RANGE CASE--------");
List<Amount<Length>> vRange_BR = test
        .createRangeArray(
                test.getMaxTakeOffMass(),
                test.getSweepHalfChordEquivalent(),
                test.getSurface(),
                test.getCd0(),
                test.getOswald(),
                aircraft.get_theAerodynamics().getcLE(),
                test.getAr(),
                test.getTcMax(),
                test.setByPassRatio(0.0),
                test.getEta(),
                test.getAltitude(),
                test.calculateBestRangeMach(
                        EngineTypeEnum.TURBOPROP,
                        test.getSurface(),
                        test.getAr(),
                        test.getOswald(),
                        test.getCd0(),
                        test.getAltitude()),
                true);
// -------------USER CURRENT MACH-------------------
System.out.println();
System.out.println("------CURRENT MACH CASE-------");
List<Amount<Length>> vRange_CM = test
        .createRangeArray(
                test.getMaxTakeOffMass(),
                test.getSweepHalfChordEquivalent(),
                test.getSurface(),
                test.getCd0(),
                test.getOswald(),
                test.getCl(),
                test.getAr(),
                test.getTcMax(),
                test.setByPassRatio(0.0),
                test.getEta(),
                test.getAltitude(),
                test.getCurrentMach(),
                false);
// -------------------PLOTTING----------------------
// Mach parameterization:
List<Double> vPayload = test.createPayloadArray();
test.createPayloadRangeCharts_Mach(
        vRange_BR,
                vRange_CM,
                vPayload,
                test.calculateBestRangeMach(
                        EngineTypeEnum.TURBOPROP,
                        test.getSurface(),
                        test.getAr(),
                        test.getOswald(),
                        test.getCd0(),
                        test.getAltitude()),
                test.getCurrentMach());
    }
    //--------------------------------------------------
    // END OF THE TEST
  }
```

Listing 2.7 Excerpt of the ATR-72 Payload-Range test - Payload-Range diagram evaluation and plot

Otherwise, if it's the maximum take-off mass parameterization the wanted analysis, the required call is for `createPayloadRangeMatrices` method followed by the plotting method `createPayloadRangeCharts_MaxTakeOffMass`.

```
        // -----------------MTOM PARAMETERIZATION----------
        test.createPayloadRangeMatrices(
                test.getSweepHalfChordEquivalent(),
                test.getSurface(),
                test.getCd0(),
                test.getOswald(),
                test.getCl(),
                test.getAr(),
                test.getTcMax(),
                test.setByPassRatio(0.0),
                test.getEta(),
                test.getAltitude(),
                test.getCurrentMach(),
                false
                );
        // ------------------PLOTTING---------------------
        // MTOM parameterization:
        test.createPayloadRangeCharts_MaxTakeOffMass(
                test.getRangeMatrix(),
                test.getPayloadMatrix()
                );
    }
  }
```

Listing 2.8 Excerpt of the ATR-72 Payload-Range test - maximum take-off mass parameterization

In conclusion, following pages shows a summary of input data used for each analyzed aircraft, together with their related results in terms of Payload-Range diagrams for both the chosen Mach number and best range comparison and max take-off mass parameterization.

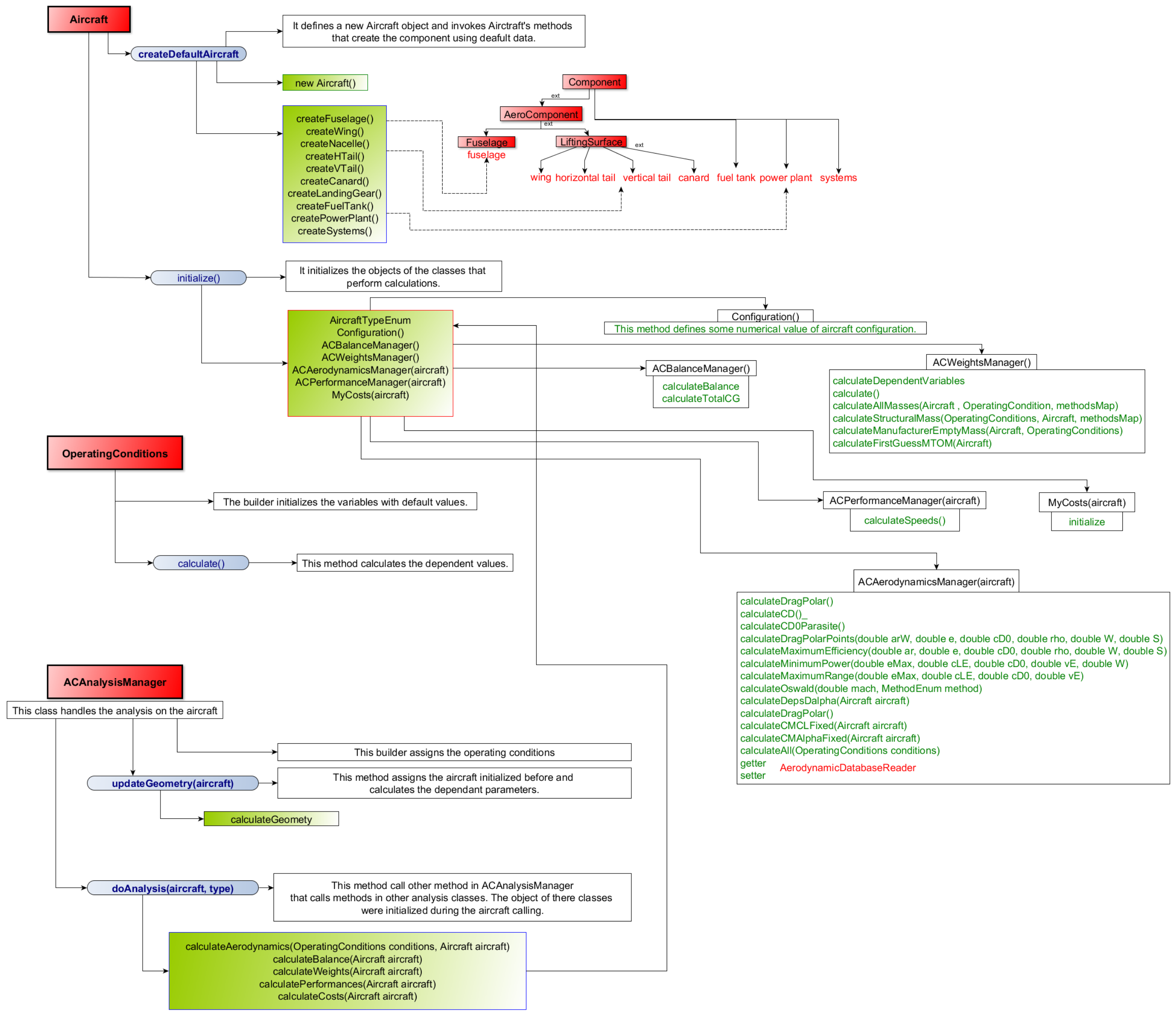
Aircraft
createDefaultAircraft
It defines a new Aircraft object and invokes Airctraft's methods that create the component using deafult data.
new Aircraft()
createFuselage()
createWing()
createNacelle()
createHTail()
createVTail()
createCanard()
createLandingGear()
createFuelTank()
createPowerPlant()
createSystems()
Component
ext
AeroComponent
ext
Fuselage
LiftingSurface
ext
fuselage
wing
horizontal tail
vertical tail
canard
fuel tank
power plant
systems
initialize()
It initializes the objects of the classes that perform calculations.
AircraftTypeEnum
Configuration()
ACBalanceManager()
ACWeightsManager()
ACAerodynamicsManager(aircraft)
ACPerformanceManager(aircraft)
MyCosts(aircraft)
Configuration()
This method defines some numerical value of aircraft configuration.
ACBalanceManager()
calculateBalance
calculateTotalCG
ACWeightsManager()
calculateDependentVariables
calculate()
calculateAllMasses(Aircraft , OperatingCondition, methodsMap)
calculateStructuralMass(OperatingConditions, Aircraft, methodsMap)
calculateManufacturerEmptyMass(Aircraft, OperatingConditions)
calculateFirstGuessMTOM(Aircraft)
OperatingConditions
The builder initializes the variables with default values.
calculate()
This method calculates the dependent values.
ACPerformanceManager(aircraft)
calculateSpeeds()
MyCosts(aircraft)
initialize
ACAerodynamicsManager(aircraft)
calculateDragPolar()
calculateCD()_
calculateCD0Parasite()
calculateDragPolarPoints(double arW, double e, double cD0, double rho, double W, double S)
calculateMaximumEfficiency(double ar, double e, double cD0, double rho, double W, double S)
calculateMinimumPower(double eMax, double cLE, double cD0, double vE, double W)
calculateMaximumRange(double eMax, double cLE, double cD0, double vE)
calculateOswald(double mach, MethodEnum method)
calculateDepsDalpha(Aircraft aircraft)
calculateDragPolar()
calculateCMCLFixed(Aircraft aircraft)
calculateCMAlphaFixed(Aircraft aircraft)
calculateAll(OperatingConditions conditions)
getter
setter
AerodynamicDatabaseReader
ACAnalysisManager
This class handles the analysis on the aircraft
This builder assigns the operating conditions
updateGeometry(aircraft)
This method assigns the aircraft initialized before and calculates the dependant parameters.
calculateGeomety
doAnalysis(aircraft, type)
This method call other method in ACAnalysisManager that calls methods in other analysis classes. The object of there classes were initialized during the aircraft calling.
calculateAerodynamics(OperatingConditions conditions, Aircraft aircraft)
calculateBalance(Aircraft aircraft)
calculateWeights(Aircraft aircraft)
calculatePerformances(Aircraft aircraft)
calculateCosts(Aircraft aircraft)

Figure 2.5 `Aircraft`, `OperatingConditions` and `ACAnalysisManager` flowchart

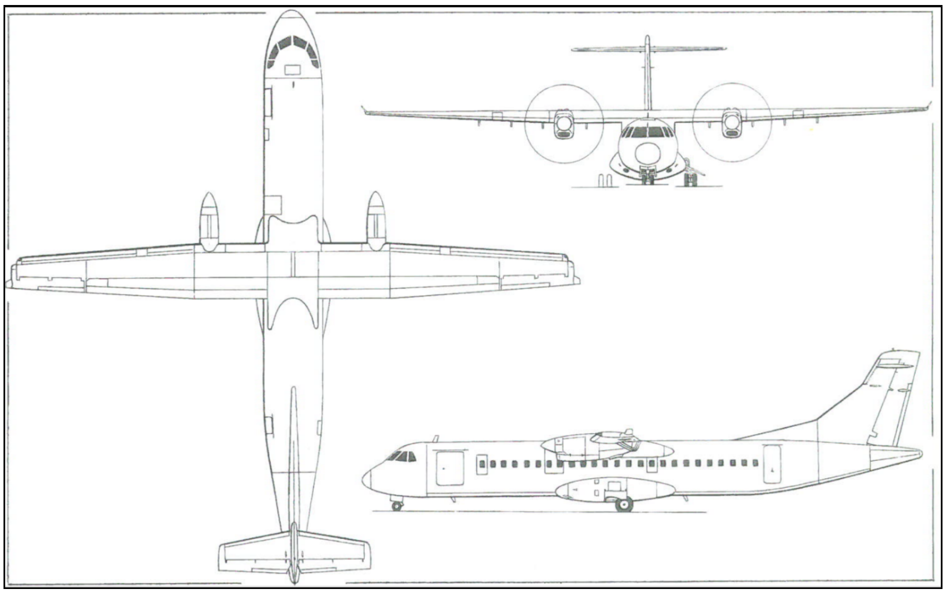

Figure 2.6 ATR-72 views – Jane's All the World's Aircraft 2004-2005

Variables	**Description**	**Values**
`altitude`	Cruise altitude	6000 m
`machCurrent`	The user chosen Mach number	0,43
`surface`	Wing surface	61 m^2
`taperRatioEquivalent`	Taper ratio of the equivalent wing	0,545
`maxTakeOffMass`	maximum take-off mass	23063,579 kg
`operatingEmptyMass`	operating empty mass	12935,579 kg
`maxFuelMass`	maximum fuel mass	5000,000 kg
`nPassMax`	maximum number of passengers	72
`sweepLEEquivalent`	Equivalent wing leading edge sweep angle	3,142 °
`oswald`	Wing oswald factor	0,7585
`byPassRatio`	Engine bypass ratio	0,0
`eta`	propeller efficiency	0,85
`tcMax`	Mean maximum thickness of the wing	0,1675
`cl`	The cruise lift coefficient	0,45
`ar`	Wing aspect ratio	12
`cd0`	Wing parasite drag coefficient	0,0317

Table 2.4 ATR-72 input data

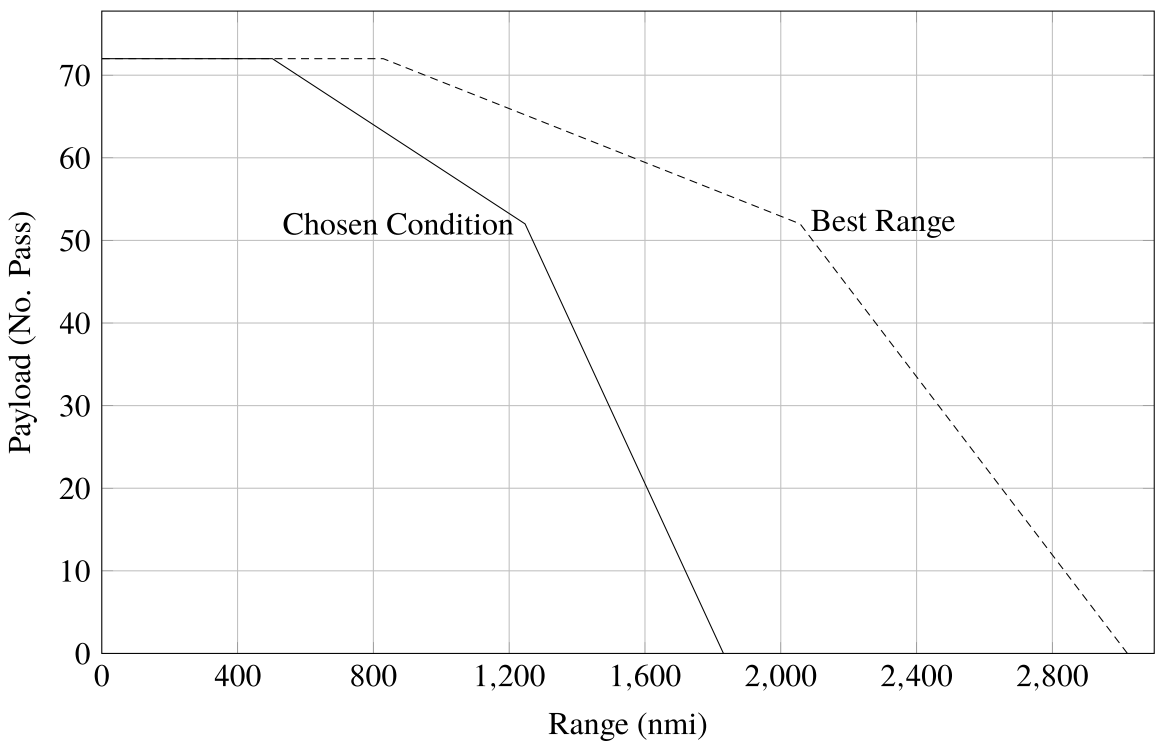

Figure 2.7 ATR-72 Payload-Range - Chosen Mach number and best range comparison

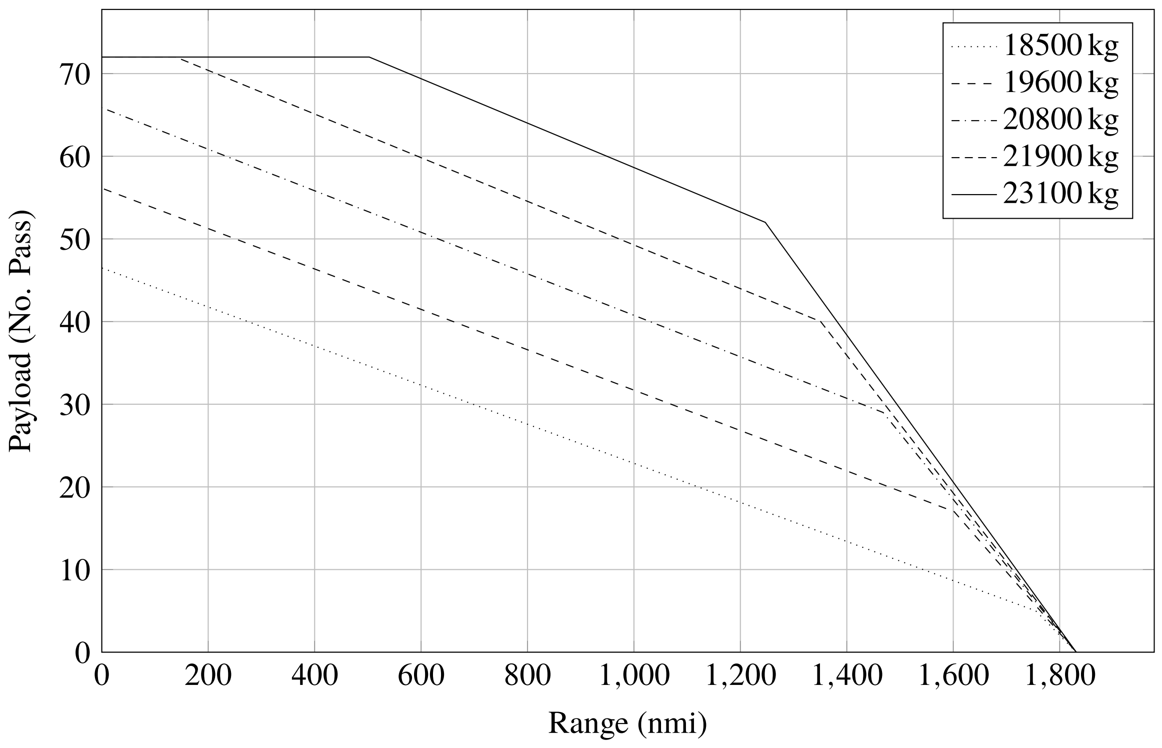

Figure 2.8 ATR-72 Payload-Range - Maximum take-off mass parameterization

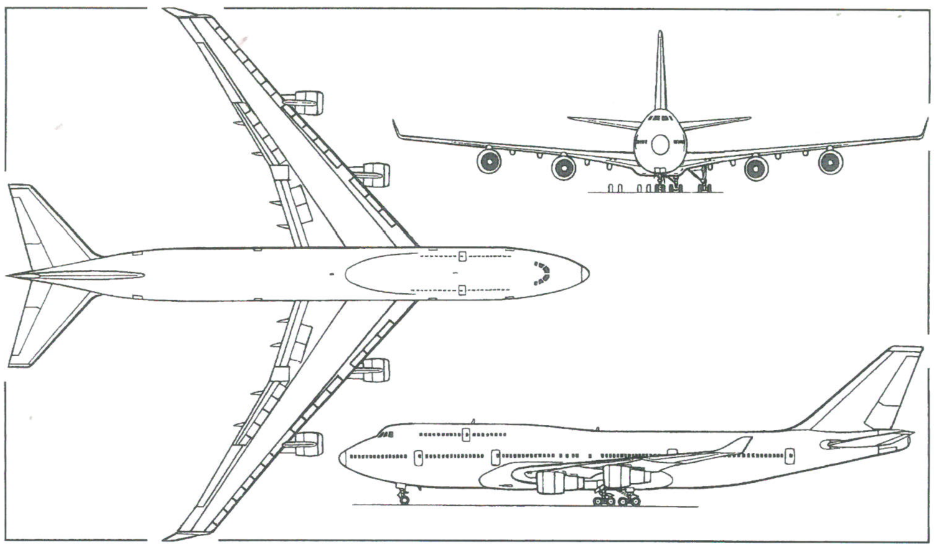

Figure 2.9 B747-100B views – Jane's All the World's Aircraft 2004-2005

Variables	Description	Values
`altitude`	Cruise altitude	11000 m
`machCurrent`	The user chosen Mach number	0,83
`surface`	Wing surface	511 m^2
`taperRatioEquivalent`	Taper ratio of the equivalent wing	0,284
`maxTakeOffMass`	maximum take-off mass	354991,506 kg
`operatingEmptyMass`	operating empty mass	153131,986 kg
`maxFuelMass`	maximum fuel mass	147409,520 kg
`nPassMax`	maximum number of passengers	550
`sweepLEEquivalent`	Equivalent wing leading edge sweep angle	38,429 °
`oswald`	Wing oswald factor	0,6277
`byPassRatio`	Engine bypass ratio	5,0
`eta`	propeller efficiency	0,0
`tcMax`	Mean maximum thickness of the wing	0,1292
`cl`	The cruise lift coefficient	0,45
`ar`	Wing aspect ratio	6,9
`cd0`	Wing parasite drag coefficient	0,0182

Table 2.5 B747-100B input data

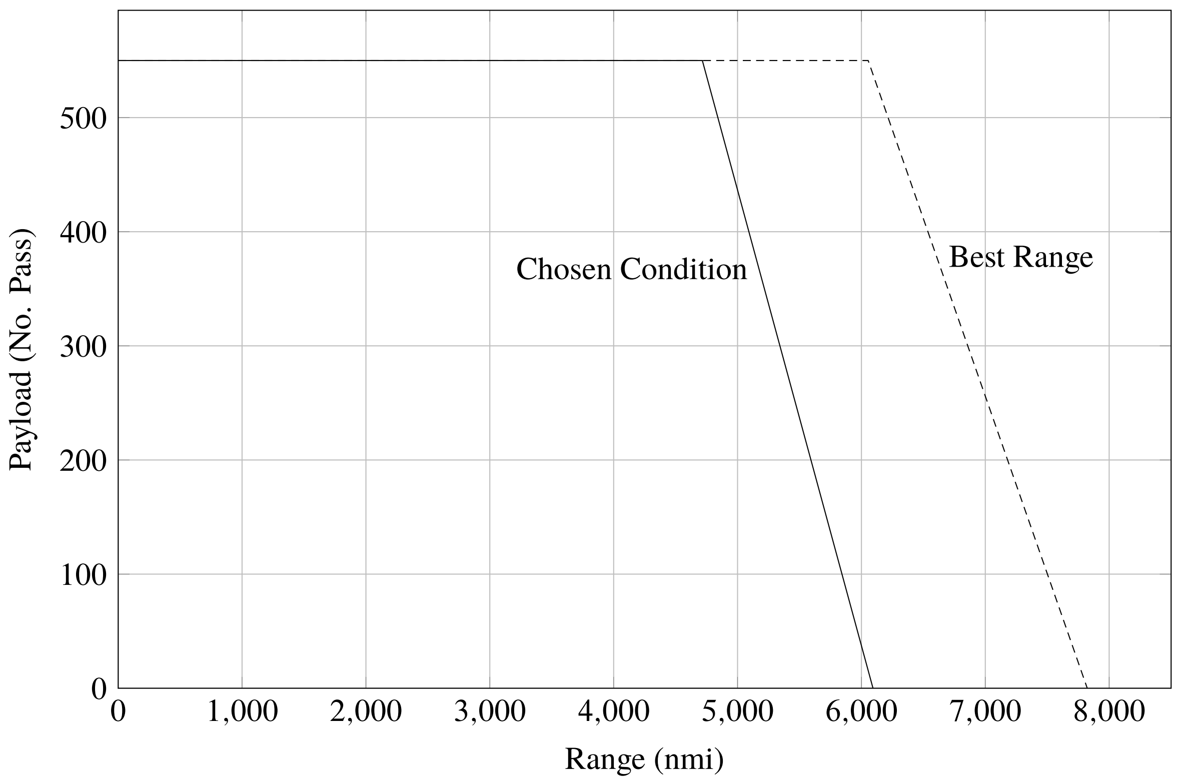

Figure 2.10 B747-100B Payload-Range - Chosen Mach number and best range comparison

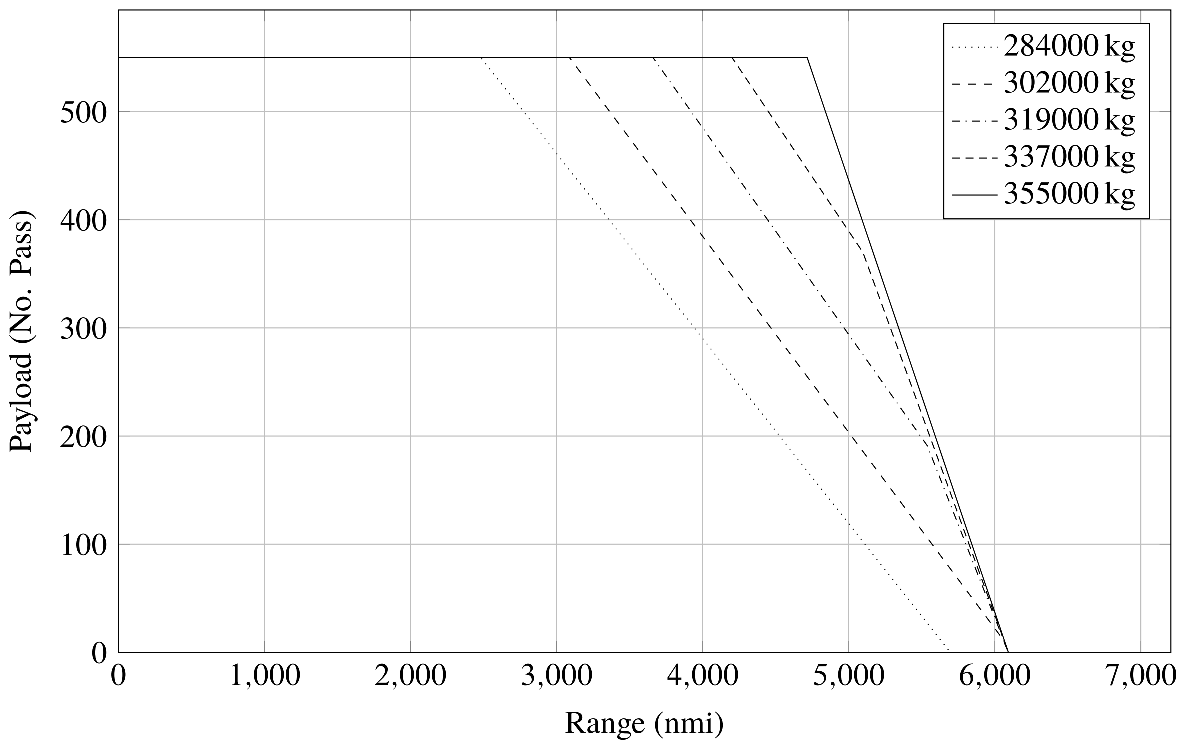

Figure 2.11 B747-100B Payload-Range - Maximum take-off mass parameterization

Variables	Values
Range (point A)	0 nmi
Payload (point A)	72 passengers
Range (point B)	502.587 nmi
Payload (point B)	72 passengers
Range (point C)	1246.827 nmi
Payload (point C)	52 passengers
Range (point D)	1831.171 nmi
Payload (point D)	0 passengers
C_L	0.421
C_D	0.0388
Efficiency	10.853
SFC	0.424

Variables	Values
Range (point A)	0 nmi
Payload (point A)	550 passengers
Range (point B)	4714.449 nmi
Payload (point B)	550 passengers
Range (point C)	4715.799 nmi
Payload (point C)	550 passengers
Range (point D)	6093.423 nmi
Payload (point D)	0 passengers
C_L	0.385
C_D	0.0296
Efficiency	13.00518
SFC	0.626

Table 2.6 Review of ATR-72 (left) and B747-100B (right) results at chosen Mach number conditions

Chapter 3

Specific Range

In this chapter an anlysis of the specific range is performed with the aim of obtain useful information about cruise performance.

In particular, starting from generalized performance evaluation and interfacing them with the fuel consumption, the final objective will be to define the specific range as function of Mach number, obtaining the so called *cruise grid chart* which is a very important tool for pilots because it allows to choose the correct speed, during cruising phase, in order to follow some mission objectives like minimum fuel consumption or a fast cruise.

3.1 Theoretical background

The first step that has to be done in order to obtain the *cruise grid chart* is to define generalized performance in terms of thrust and drag. The *generalized* attribute given to these quantities stands for the fact that they are independents from altitude and this result is reached through the parameter δ which represents a ratio between the total pressure at compressor inlet and the standard pressure at sea level.

Regarding the thrust, the generalized version can be obtained by dividing it by δ as shown below.

$$\frac{T}{\delta} = \frac{T_f}{\delta} - M \cdot a_0 \cdot \frac{\sqrt{\theta \cdot m_a}}{\delta} \tag{3.1}$$

where

- T, is the net thust
- T_f, is the gross thrust
- M, is the mach number
- a_0, is the sound speed at sea level

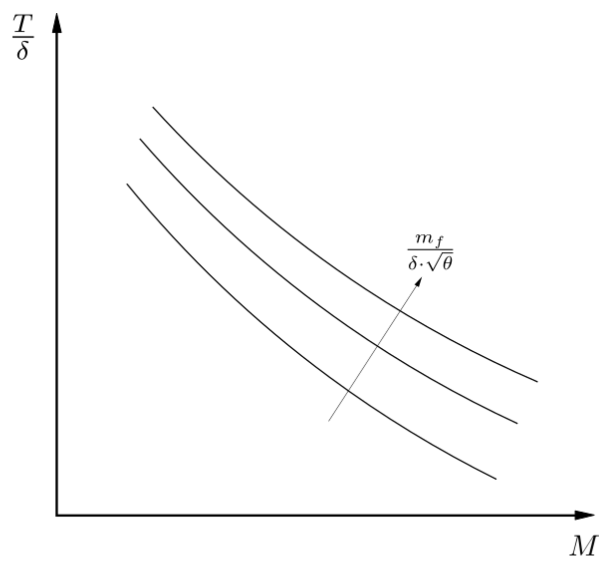

Figure 3.1 Qualitative trend of the generalized thrust v.s. Mach number parameterized in generalized fuel flow rate

- $\dfrac{\sqrt{\theta}\cdot m_a}{\delta}$, is the genralized air flow rate which is function of the generalized fuel flow rate given by $\dfrac{m_f}{\delta\cdot\sqrt{\theta}}$

so that the genralized thrust results as a function of Mach number and generalized fuel flow rate.

$$\frac{T}{\delta} = \frac{T_f}{\delta}\cdot f\left(\frac{m_f}{\delta\cdot\sqrt{\theta}}\right) \tag{3.2}$$

As shown in figure 3.1 the generalized thrust decreases with Mach number, at given fuel flow rate, and grows with the latter, at given Mach number. This because if the Mach number grows at fixed fuel flow rate, the air flow rate grows reducing the thrust; otherwise, if fuel flow rate grows at fixed Mach number, air flow rate is lower giving more thrust.

With this function it's possible to correlate generalized thrust to hourly fuel consumption which is a main keypoint in building the cruise grid chart of an endurance based aircraft such as UAV.

Since transport aircrafts rely more on range performance, it's necessary to obtain the same relationship between generalized thurst and fuel consumption referred to the generalized specific range indicated with $\delta\bar{s}$.

Dividing the generalized fuel flow rate, which is dimesionally equal to $\frac{\text{kg}}{\text{s}}$, by a velocity, the result has a dimension of $\frac{\text{m}}{\text{kg}}$ that represents the reciprocal of the specific range. In this way it's possible to state the following relation.

$$\frac{m_f}{\delta\cdot\sqrt{\theta}} = \frac{M\cdot a_0}{\delta\bar{s}} \tag{3.3}$$

As expectd from the relation 3.3 the thrust has a trend which is the inverse of the previous

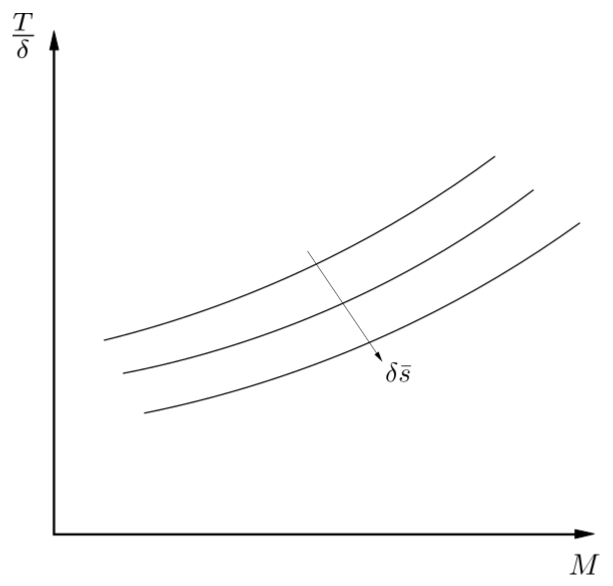

Figure 3.2 Qualitative trend of the generalized thrust v.s. Mach number parameterized in generalized specific range

parameterization. In fact now, for a given Mach number, if the pilot wants to go farther he has to decrease the thrust in order to reduce the fuel consumption; otherwise, at a given distance to reach, the pilot has to increase the thrust in order to make the Mach number grows.

Since the thrust has always to be compared with the drag in order to evaluate if the aircraft can fly in a specific cruise condition without loosing speed and altitude, it's necessary to obtain a generalized drag trend as well.

This ai very similar to the drag trend, as can be seen from figure 3.3, and it depends from aircraft weight as well; but, since these have to be generalized quantities, the weight is a generalized weight too.

By the overlap of figure 3.2 and figure 3.3 charts, it's easy to note that the best Mach number, for a given generalized weight, is located at the intersection of the two curves as reported in

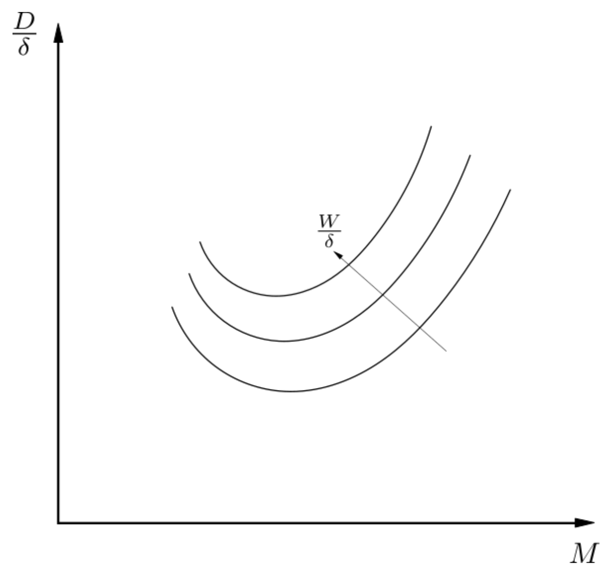

Figure 3.3 Qualitative trend of the generalized drag v.s. Mach number parameterized in generalized weight

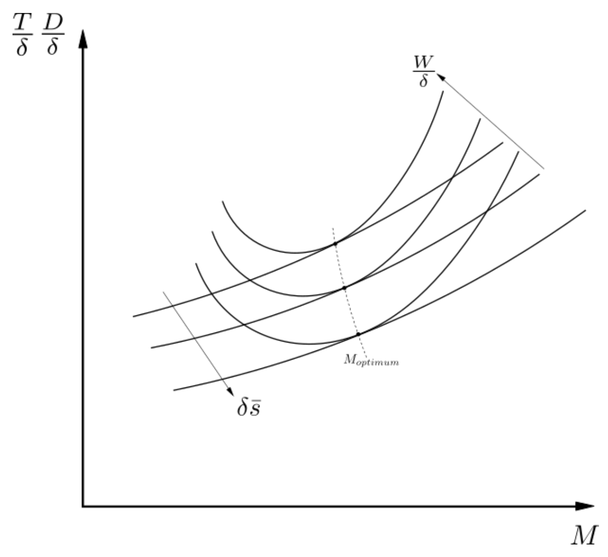

Figure 3.4 Comparison between generalized drag and generalized thrust as functions of Mach number

figure 3.4. In fact, in order to obtain a bigger specific range at fixed weight, the generalized drag would be higher than the generalized thrust; otherwise, if the pilot wants to fly faster at given weight, he has to increase thrust so that the specific range will decrease due to the increasing fuel consumption.

Since during the cruise phase the aircraft weight decreases continuously, the pilot has to gain altitude in order to leave the generalized weight unchanged; this explains why during cruise the aircraft continues to climb.

The specific range can also be connected to Breguet formulas, reported at 2.1 and 2.2, as it can be obtained by dividng the autonomy factor, $A.F.$, by the aircraft weight; in particular the autonomy factor, groups three main aircraft efficiency and can be written as follow.

$$A.F. = \frac{\eta_p}{SFC} \cdot \frac{L}{D} \qquad \text{propeller aircraft} \tag{3.4}$$

$$A.F. = \frac{V}{SFCJ} \cdot \frac{L}{D} \qquad \text{jet aircraft} \tag{3.5}$$

where

- η_p, is propeller efficiency
- SFC, is related to propulsive efficiency
- $\frac{L}{D}$, is the aerodynamic efficiency

At given generalized weight and generalized specific range, the optimum Mach is known as explained before and so the autonomy factor can be calculated by multiply $\frac{W}{\delta}$ and $\delta\bar{s}$. Repeating this operation for different generalized weight conditions, allows to define the autonomy factor trend as function of the generalized weight in which each point of the chart is related to an optimum Mach number for the specific range.

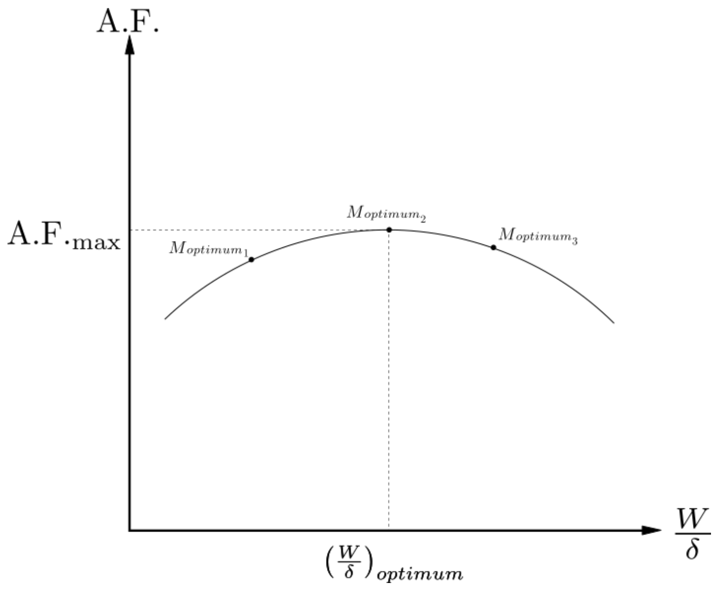

Figure 3.5 Autonomy factor trend as function of the generalized weight

As can be seen from figure 3.5 the autonomy factor has a maximum at a specific generalized weight which is the one that the pilot should maintain during the cruise phase.

If the altitude is fixed, and so δ is constant, the chart in figure 3.5 can be seen as function of Mach number for a given aircraft weight; at this point, knowing that the autonomy factor leads to the specific range if divided by the aircraft weight, it's possible to define the specific range trend as funcion of the Mach number parameterized in aircraft weight.

The chart so obtained in figure 3.6 is the one upon which the cruise grid is defined; on the latter, in fact, four lines are drawn each of which is related to a precise mission objecive.

It's important to highlight that, on long distances, the maximum distance line is not often followed during the cruise because it is tied to a low speed which adversely affects the total flight time increasing the D.O.C.; in order to avoid this condition, pilots prefer to follow the long range line which has only 1% of penalty on the specific range but, at the same time, allows to fly at significantly higher speed with benefits on flight time and, as a result, on the D.O.C.

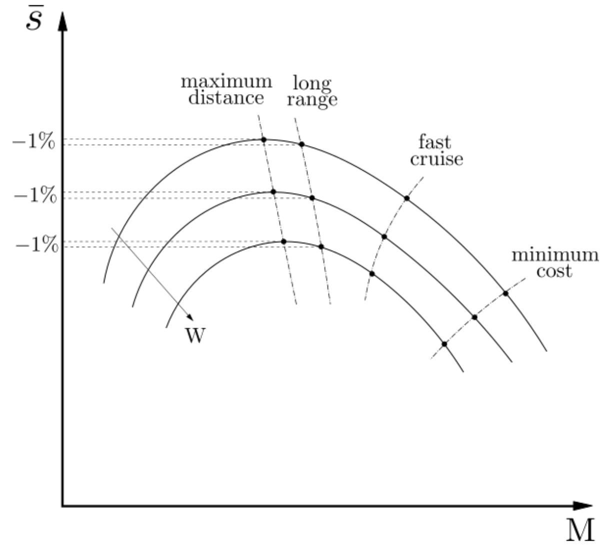

Figure 3.6 Specific range as function of the Mach number parameterized in aricraft weight

3.2 Java class architecture

3.3 Case study: ATR-72 and B747-100B

Chapter 4

High lift devices effects

Chapter 5

Take-off performance

Appendices

Appendix A

HDF DATABSE CREATION AND READING

A.1 Creation of a database using MATLAB

Creation and mangment of an HDF Dataset are very important to handle because they allow to generate resources which are required by a lot of analysis; for example by using this datasets it's possible to implement a new engine type allowing the analysis of the preformance of a new aircraft. This feature has not been implemented inside JPAD with the purpose of being able to generate the required resources independently.

First of all it's necessary to have curves of the database that has to be digitalized; then, with the use of sotware like *PlotDigitizer*, it's possible to acquire them using a finite number of points chosen by the user. The output of this procedure is a *.csv* file containing all the copule of points which have been used to digitalize the specific curve.

Now the matlab code comes in play to manage these data and to generate the digitalized curves and the HDF dataset. In the example reported there are four curves defined by points through *PlotDigitizer* which have, firstly, been imported in MATLAB generating four *.mat* files; at this point the code interpolates curves points with cubic splines in order to have more points to plot for each curve.

Finally curves are plotted and the HDF Dataset is populated by using *h5create* and *h5write*; in particular curves points, abscissas and parameterization values are attached to the h5 file through these commands.

Listing A.1 MATLAB script for creating the HDF Database

```
clc; close all; clear all;

%% Import data
DeltaAlphaCLmax_vs_LambdaLE_dy1p2 =
     importdata('DeltaAlphaCLmax_vs_LambdaLE_dy1p2.mat');
DeltaAlphaCLmax_vs_LambdaLE_dy2p0 =
     importdata('DeltaAlphaCLmax_vs_LambdaLE_dy2p0.mat');
```

```
DeltaAlphaCLmax_vs_LambdaLE_dy3p0 =
    importdata('DeltaAlphaCLmax_vs_LambdaLE_dy3p0.mat');
DeltaAlphaCLmax_vs_LambdaLE_dy4p0 =
    importdata('DeltaAlphaCLmax_vs_LambdaLE_dy4p0.mat');

nPoints = 30;
lambdaLEVector_deg = transpose(linspace(0, 40, nPoints));

%% dy/c = 1.2
smoothingParameter = 0.999999;
DAlphaVsLambdaLESplineStatic_Dy1p2 = csaps( ...
    DeltaAlphaCLmax_vs_LambdaLE_dy1p2(:,1), ...
    DeltaAlphaCLmax_vs_LambdaLE_dy1p2(:,2), ...
    smoothingParameter);
DAlphaVsLambdaLEStatic_Dy1p2 = ppval( ...
    DAlphaVsLambdaLESplineStatic_Dy1p2, ...
    lambdaLEVector_deg);

%% dy/c = 2.0
smoothingParameter = 0.999999;
DAlphaVsLambdaLESplineStatic_Dy2p0 = csaps( ...
    DeltaAlphaCLmax_vs_LambdaLE_dy2p0(:,1), ...
    DeltaAlphaCLmax_vs_LambdaLE_dy2p0(:,2), ...
    smoothingParameter);
DAlphaVsLambdaLEStatic_Dy2p0 = ppval( ...
    DAlphaVsLambdaLESplineStatic_Dy2p0, ...
    lambdaLEVector_deg);

%% dy/c = 3.0
smoothingParameter =0.999999;
DAlphaVsLambdaLESplineStatic_Dy3p0 = csaps( ...
    DeltaAlphaCLmax_vs_LambdaLE_dy3p0(:,1), ...
    DeltaAlphaCLmax_vs_LambdaLE_dy3p0(:,2), ...
    smoothingParameter);
DAlphaVsLambdaLEStatic_Dy3p0 = ppval( ...
    DAlphaVsLambdaLESplineStatic_Dy3p0, ...
    lambdaLEVector_deg);

%% dy/c = 4.0
smoothingParameter = 0.999999;
DAlphaVsLambdaLESplineStatic_Dy4p0 = csaps( ...
    DeltaAlphaCLmax_vs_LambdaLE_dy4p0(:,1), ...
    DeltaAlphaCLmax_vs_LambdaLE_dy4p0(:,2), ...
    smoothingParameter);
DAlphaVsLambdaLEStatic_Dy4p0 = ppval( ...
    DAlphaVsLambdaLESplineStatic_Dy4p0, ...
    lambdaLEVector_deg);

%% Plots
figure(1)
plot (lambdaLEVector_deg, DAlphaVsLambdaLEStatic_Dy1p2, '-*b' ... , ...);
hold on;
plot (lambdaLEVector_deg, DAlphaVsLambdaLEStatic_Dy2p0, '-b' ... , ...);
plot (lambdaLEVector_deg, DAlphaVsLambdaLEStatic_Dy3p0, '*b' ... , ...);
plot (lambdaLEVector_deg, DAlphaVsLambdaLEStatic_Dy4p0, 'b' ... , ...);
xlabel('\Lambda_{le} (deg)'); ylabel('\Delta\alpha_{C_{L,max}}');
title('Angle of attack increment for wing maximum lift in subsonic flight');
legend('\Delta y/c = 1.2', '\Delta y/c = 2.0', '\Delta y/c = 3.0','\Delta y/c =
    4.0');
axis([0 50 0 9]);
grid on;

%% preparing output to HDF
% dy/c
dyVector = [1.2;2.0;3.0;4.0];
%columns --> curves
myData = [ ...
        DAlphaVsLambdaLEStatic_Dy1p2, ...
        DAlphaVsLambdaLEStatic_Dy2p0, ...
        DAlphaVsLambdaLEStatic_Dy3p0, ...
        DAlphaVsLambdaLEStatic_Dy4p0];

hdfFileName = 'DAlphaVsLambdaLEVsDy.h5';
if ( exist(hdfFileName, 'file') )
    fprintf('file %s exists, deleting and creating a new one\n', hdfFileName);
    delete(hdfFileName)
else
    fprintf('Creating new file %s\n', hdfFileName);
end
% Dataset: data
h5create(hdfFileName, '/DAlphaVsLambdaLEVsDy/data', size(myData'));
h5write(hdfFileName, '/DAlphaVsLambdaLEVsDy/data', myData');
% Dataset: var_0
h5create(hdfFileName, '/DAlphaVsLambdaLEVsDy/var_0', size(dyVector'));
h5write(hdfFileName, '/DAlphaVsLambdaLEVsDy/var_0', dyVector');
% Dataset: var_1
h5create(hdfFileName, '/DAlphaVsLambdaLEVsDy/var_1', size(lambdaLEVector_deg'));
h5write(hdfFileName, '/DAlphaVsLambdaLEVsDy/var_1', lambdaLEVector_deg');
```

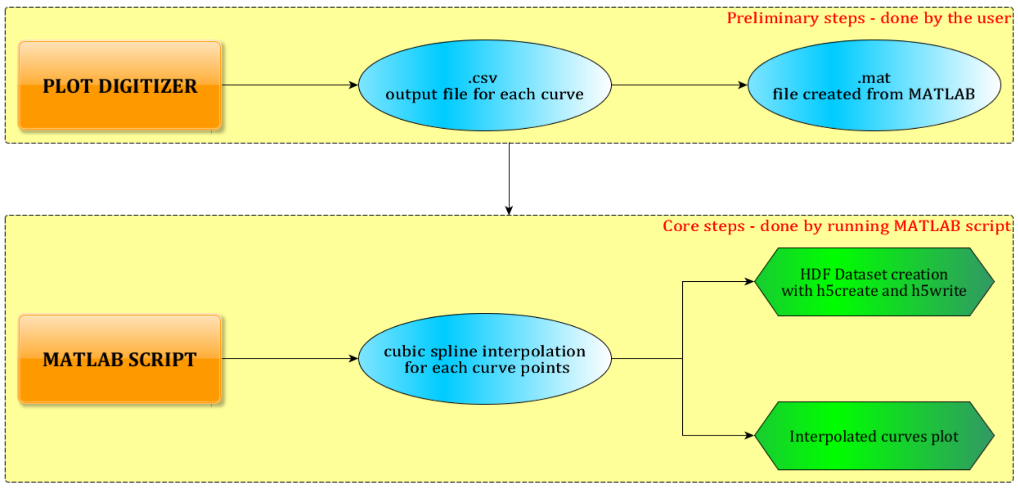
Preliminary steps - done by the user
PLOT DIGITIZER
.csv output file for each curve
.mat file created from MATLAB
Core steps - done by running MATLAB script
MATLAB SCRIPT
cubic spline interpolation for each curve points
HDF Dataset creation with h5create and h5write
Interpolated curves plot

Figure A.1 Flowchart of an HDF Database creation

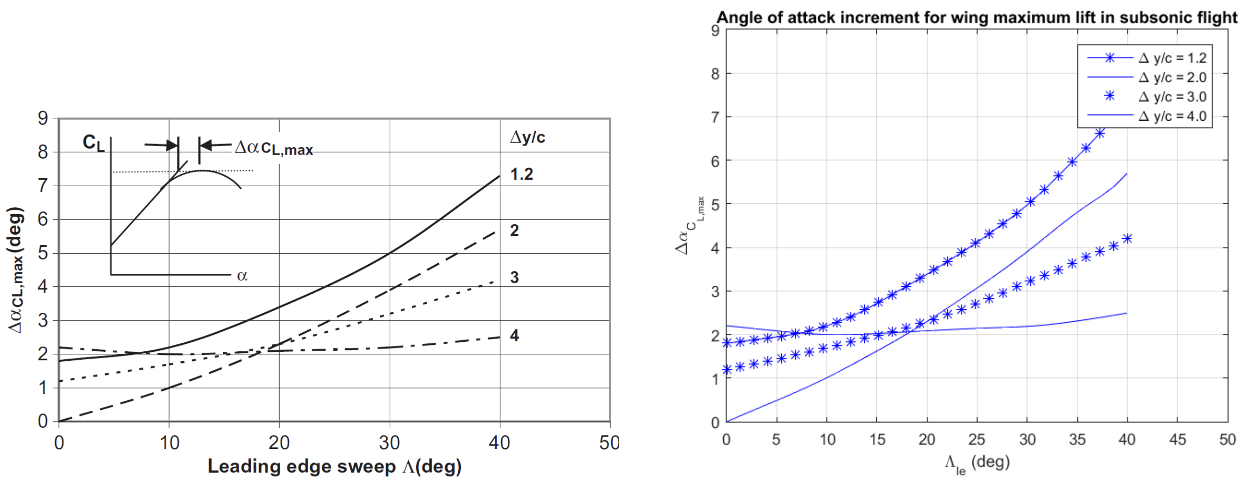

Figure A.2 Comparison between the initial graph and the digitized one

A.2 Reading data from an HDF database in JPAD

After creating the database, this has to be read in order to obtain the required data; inside JPAD this operation can be done defining a specific class, which extends the abstract class `DatabaseReader` that is designed for HDF dataset reading.

This son class has a specific structure which main key points can be summarized in the following ones:

1. Creation of variables, in number equal to the function to be interpolated, using the type `MyInterpolatingFunction`
2. Creation of variables for all values that are wanted to be read from the interpolating functions
3. Creation of a builder that accepts the folder path string and the file name string of the database. This builder has to launch the interpolating method for all functions contained into the database by using `MyInterpolatingFunction` methods.
4. Creation of a getter method for each of the variables allocated at point 2 in order to obtain values from interpolated functions by giving in input the required parameters

In particular the class `MyInterpolatingFunction` implements methods for a spline, bicubic and tricubic data interpolation as well as three methods for extracting a specific value from each of the previou interpolated curve.

The following listing describes, with an example upon the aerodynamic database, how the reader class should be build following the previous steps.

Listing A.2 DatabaseReader son class creation

```
public class AerodynamicDatabaseReader extends DatabaseReader {
// STEP 1:
  private MyInterpolatingFunction
          c_m0_b_k2_minus_k1_vs_FFR,
          ar_v_eff_c2_vs_Z_h_over_b_v_x_ac_h_v_over_c_bar_v;
```

```java
// STEP 2:
  double cM0_b_k2_minus_k1, ar_v_eff_c2;
// STEP 3:
  public AerodynamicDatabaseReader(String databaseFolderPath, String
    databaseFileName) {
    super(databaseFolderPath, databaseFileName);

    c_m0_b_k2_minus_k1_vs_FFR =
            database.interpolate1DFromDatasetFunction(
                    "(C_m0_b)_k2_minus_k1_vs_FFR"
            );
    ar_v_eff_c2_vs_Z_h_over_b_v_x_ac_h_v_over_c_bar_v =
            database.interpolate2DFromDatasetFunction(
                    "(AR_v_eff)_c2_vs_Z_h_over_b_v_(x_ac_h--v_over_c_bar_v)"
                    );
  }
// STEP 4:
  public double get_C_m0_b_k2_minus_k1_vs_FFR(double length, double diameter) {
    return c_m0_b_k2_minus_k1_vs_FFR.value(length/diameter);
  }
// STEP 4:
  public double get_AR_v_eff_c2_vs_Z_h_over_b_v_x_ac_h_v_over_c_bar_v(double zH,
    double bV, double xACHV, double cV) {
    return ar_v_eff_c2_vs_Z_h_over_b_v_x_ac_h_v_over_c_bar_v.value(zH/bV, xACHV/cV);
  }
}
```

Bibliography

[1] L. Attanasio. «Development of a Java Application for Parametric Aircraft Design». Master thesis. University of Naples "Federico II", 2014.

[2] Various Authors. *ATR 72*. URL: https://en.wikipedia.org/wiki/ATR_72.

[3] Various Authors. *Boeing 747*. URL: https://en.wikipedia.org/wiki/Boeing_747.

[4] Various Authors. *eXtensible Markup Language (XML)*. URL: https://en.wikipedia.org/wiki/XML.

[5] S. Ciornei. *Mach number, relative thickness, sweep and lift coefficient of the wing - An empirical investigation of parameters and equations*. Paper. 31.05.2005.

[6] Oracle Corporation. *List*. URL: https://docs.oracle.com/javase/8/docs/api/java/util/List.html.

[7] Jean-Marie Dautelle. *Jscience*. URL: http://jscience.org/.

[8] E. De Bono. «Prestazioni di decollo e atterraggio. Calcolo approssimato e verifica sperimentale.» Thesis. University of Naples "Federico II", 2011.

[9] D. et al Gilbert. *JFreeChart*. URL: http://www.jfree.org/jfreechart/.

[10] The HDF Group. *HDF*. URL: https://www.hdfgroup.org/products/java/hdf-java-html/javadocs/ncsa/hdf/hdf5lib/H5.html.

[11] D. Howe. *Aircraft conceptual design synthesis*. Aerospace Series. Professional Engineering Publishing, 2000.

[12] J. Huwaldt. *1976 Standard Atmosphere Program*. URL: http://thehuwaldtfamily.org/java/Applications/StdAtmosphere/StdAtmosphere.html.

[13] P. Jackson and FRAeS. *Jane's All the World's Aircraft*. Jane's All the World's Aircraft. Jane's Information Group, 2004-2005.

[14] L.R. Jenkinson, D. Rhodes, and P. Simpkin. *Civil jet aircraft design*. AIAA education series. Arnold, 1999.

[15] K. Kawaguchi. *Args4j*. URL: http://args4j.kohsuke.org/.

[16] I. Kroo and R. Shevell. *Aircraft Design: Synthesis and Analysis*. 1999.

[17] A. Lausetti. *Decollo e atterramento di aeroplani, idrovolanti trasportati*. Levrotto & Bella Editrice S.a.s., 1992.

[18] B. W. McCormick. *Aerodynamics, Aeronautics, and Flight Mechanics*. John Wiley & Sons, 1979.

[19] L.M. Nicolai and G. Carichner. *Fundamentals of Aircraft and Airship Design*. AIAA education series v. 1. American Institute of Aeronautics and Astronautics, 2010.

[20] F. Nicolosi and G. Paduano. *Behind ADAS*. 2011.

[21] E. Obert. *Aerodynamic Design of Transport Aircraft*. IOS Press, 2009.

[22] D.P. Raymer. *Aircraft Design: A Conceptual Approach*. 1992.

[23] D.P. Raymer. *Aircraft Design / RDS-Student: A Conceptual Approach*. AIAA Education Series. Amer Inst of Aeronautics &, 2013.

[24] M.H. Sadraey. *Aircraft Design: A Systems Engineering Approach*. Aerospace Series. Wiley, 2012.

[25] P.M. Sforza. *Commercial Airplane Design Principles*. Elsevier Science, 2014.

[26] E. Torenbeek. *Advanced Aircraft Design: Conceptual Design, Technology and Optimization of Subsonic Civil Airplanes*. Aerospace Series. Wiley, 2013.

[27] E. Torenbeek. *Optimum Cruise Performance of Subsonic Transport Aircraft*. Delft University Press, 1998.

[28] E. Torenbeek. *Synthesis of Subsonic Aircraft Design*. 1976.

[29] E. Torenbeek. *Synthesis of Subsonic Airplane Design: An Introduction to the Preliminary Design of Subsonic General Aviation and Transport Aircraft, with Emphasis on Layout, Aerodynamic Design, Propulsion and Performance*. Springer, 1982.

[30] A.D. Young. *The Aerodynamic Characteristics of Flaps*. Technical Report. 1947.

Glossary

Direct Operative Cost The totality of aircraft costs directly connected to the aircraft flight. It can be seen as the amount of money necessary to carry 1 ton of payload upon 1 km.

Hierarchical Data Format A set of file formats (HDF4, HDF5) designed to store and organize large amounts of data.

Java Program toolchain for Aircraft Design Collection of libraries and classes with the aim of providing complete aircraft preliminary design analyses through the use of several semi-empirical formulas tested against experimental data.

List The java.util.List interface is a subtype of the java.util.Collection interface. It represents an ordered list of objects, meaning you can access the elements of a List in a specific order, and by an index too. You can also add the same element more than once to a List.

Parsing Parsing or syntactic analysis is the process of analysing a string of symbols, either in natural language or in computer languages, conforming to the rules of a formal grammar.

Portable Network Graphics A raster graphics file format that supports lossless data compression. PNG was created as an improved, non-patented replacement for Graphics Interchange Format (GIF), and is the most used lossless image compression format on the Internet.

Table a collection that associates an ordered pair of keys, called a row key and a column key, with a single value. A table may be sparse, with only a small fraction of row key / column key pairs possessing a corresponding value.

TikZ ist kein Zeichenprogramm TikZ is a set of higher-level macros that use PGF.

User developer the term refers to the developer which will use a method without being interested in how the method performs the required action. This is the case of a utility method: the developer is the one who writes the method, while the user developer is who uses that method to accomplish some action which requires the functionality provided by the utility method. It has to be noticed that the user developer and the developer can be the same person.

Acronyms

DOC	Direct Operative Cost.
HDF	Hierarchical Data Format.
JPAD	Java Program toolchain for Aircraft Design.
MZFW	Maximum Zero Fuel Weight.
PNG	Portable Network Graphics.
SFC	Specific Fuel Consumption.
SFCJ	Jet Specific Fuel Consumption.
TikZ	TikZ ist kein Zeichenprogramm.
UAV	Unmanned Aerial Vehicle.

List of symbols

$\mathcal{A}\!\!\!\mathbf{R}$ aspect ratio.

$(\,)_{cruise}$ quantity related to cruise condition.

η_p propeller efficiency.

$(\,)_{\mathbf{f}}$ quantity related to flaps.

$(\,)_{\mathbf{LG}}$ quantity related to the landing gear.

b span.

$C_{\mathbf{D}}$ aerodynamic drag coefficient.

$C_{\mathbf{L}}$ aerodynamic lift coefficient.

$C_{\mathbf{D0}}$ aerodynamic parasite drag coefficient.

D aerodynamic drag.

$\boldsymbol{g}$ gravitational acceleration.

$i_{\mathbf{W}}$ the angle between the wing root chord and the ACRF x-axis.

L aerodynamic lift.

M Mach number.

$M_{\mathbf{ff}}$ Fuel fraction over entire mission.

c chord.

Λ sweep.

λ taper ratio.

t thickness.

n load factor.

R	range in nmi or km.
S	surface.
T	thrust.
V	scalar velocity.
$W_{\mathbf{OE}}$	Operating Empty Weight.
$W_{\mathbf{TO}}$	Take Off Weight.
$W_{\mathbf{fuel}}$	Fuel Weight.
$W_{\mathbf{Payload}}$	Payload Weight.
W	weight, in N or lbf.
$\alpha_{\mathbf{W}}$	angle of attack referred to wing root chord.
ρ	air density.