Chapter 1

The Payload-Range class

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.

1.1 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 1.1.

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

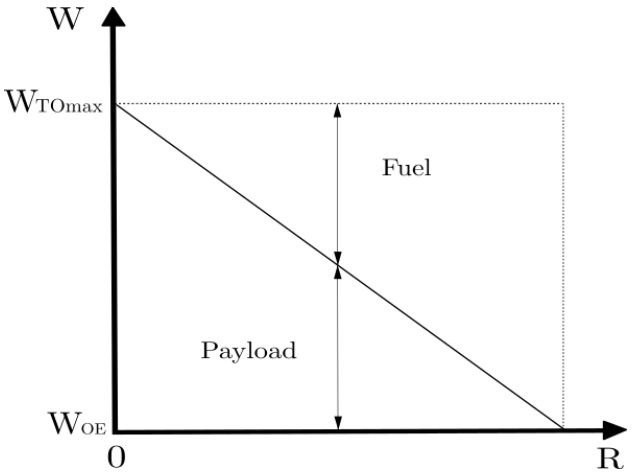

Figure 1.1: Payload-Range for a completely flexible aircraft

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 1.2 which 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 max 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.

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.

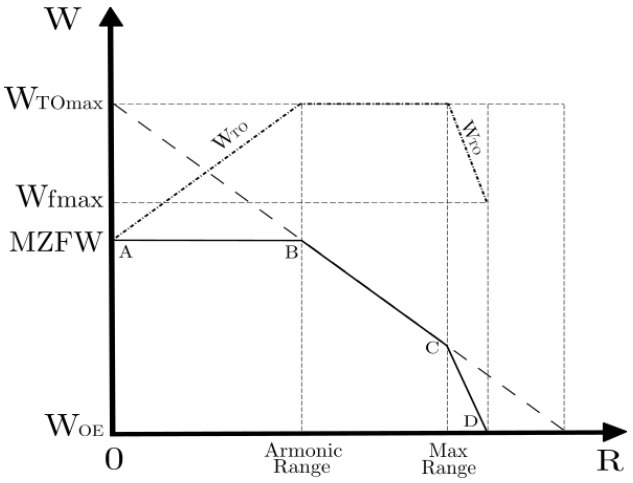

Figure 1.2: Payload-Range with MZFW limitation

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

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 kg, SFC in $\frac{lb}{hp \cdot \mathrm{h}}$ and SFCJ in $\frac{lb}{lb \cdot \mathrm{h}}$.

$$R = 603.5 \cdot \left(\frac{\eta_p}{SFC}\right)_{cruise} \cdot \left(\frac{L}{D}\right)_{cruise} \cdot \ln\left(\frac{W_i}{W_f}\right) \qquad \text{propeller aircraft} \quad (1.1)$$

$$R = \left(\frac{V}{SFCJ}\right)_{cruise} \cdot \left(\frac{L}{D}\right)_{cruise} \cdot \ln\left(\frac{W_i}{W_f}\right) \qquad \text{jet aircraft} \quad (1.2)$$

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.

$$W_{TO,max} = W_{OE} + W_{payoad,max} + W_{fuel} \quad (1.3)$$

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.

$$W_{fuel} = (1 - M_{ff}) \cdot W_{TO} \quad (1.4)$$

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

$$M_{ff} = \frac{W_1}{W_{TO}} \cdot \frac{W_2}{W_1} \cdot \frac{W_3}{W_2} \cdot \frac{W_4}{W_3} \cdot \frac{W_5}{W_4} \cdot \frac{W_6}{W_5} \cdot \frac{W_7}{W_6} \cdot \frac{W_8}{W_7} \cdot \frac{W_9}{W_8} \cdot \frac{W_{10}}{W_9} \cdot \frac{W_{fianl}}{W_{10}} \quad (1.5)$$

From this point on, it's necessary to have a table, like the one of table 1.1, 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.

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 (1.6).

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 1.1: Suggested fuel fractions

$$W_{TO,max} = W_{OE} + W_{payload} + W_{fuel,max} \tag{1.6}$$

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.

$$W_{TO} = W_{OE} + W_{fuel,max} \tag{1.7}$$

This new WTO 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.

1.2 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 1.2 data, a database which collects all data from table 1.1, 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 (1.3), (1.6) and (1.7), for each method, and of (1.4), for all of them, is performed in order to set up the following calculations and to obtain the unknown value of the payload in

`maxTakeOffMass`	Maximum take-off mass
`sweepHalfChordEquivalent`	Equivalent wing sweep angle at half chord
`surface`	Wing surface
`cd0`	Wing cD0
`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 1.2: Input data

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 (1.1) or (1.2) 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 1.1 and read the related line values.

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, also known as *sequence*, 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 1.3: 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 1.2 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.

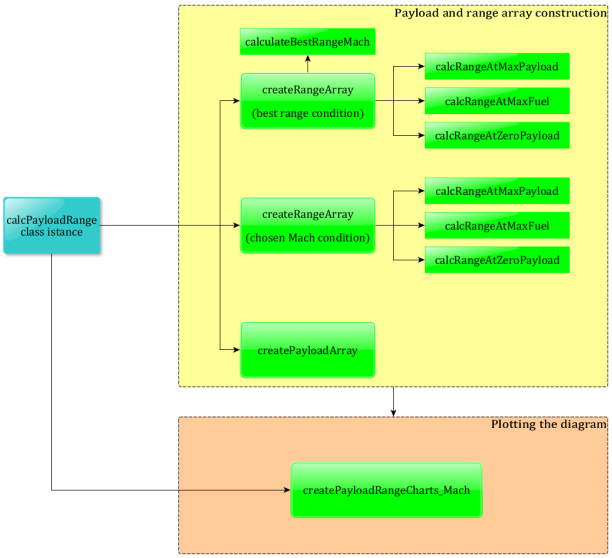

Figure 1.3: Payload-Range java class flowchart for best range and chosen Mach conditions comparison

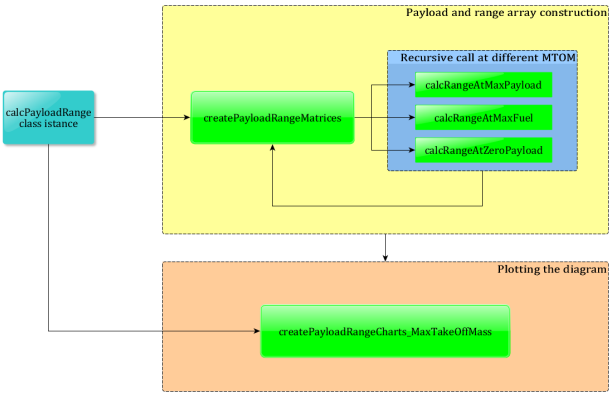

Figure 1.4: Payload-Range java class flowchart for maximum take-off mass parameterization

1.3 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 182 000 L, allowing for a 5000 nautical mile range with a typical 452 passengers payload, and an increased maximum take-off weight of 340 000 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 ??.

```
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("-----------------------------");
    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 1.1: 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 **??**.

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

```
//-----------------------------------------------------
// 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 1.2: 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.

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

Listing 1.3: 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, 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.

```
    // ----------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 1.4: 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 1.5: 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 1.6: 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 1.7: 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 1.8: 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.

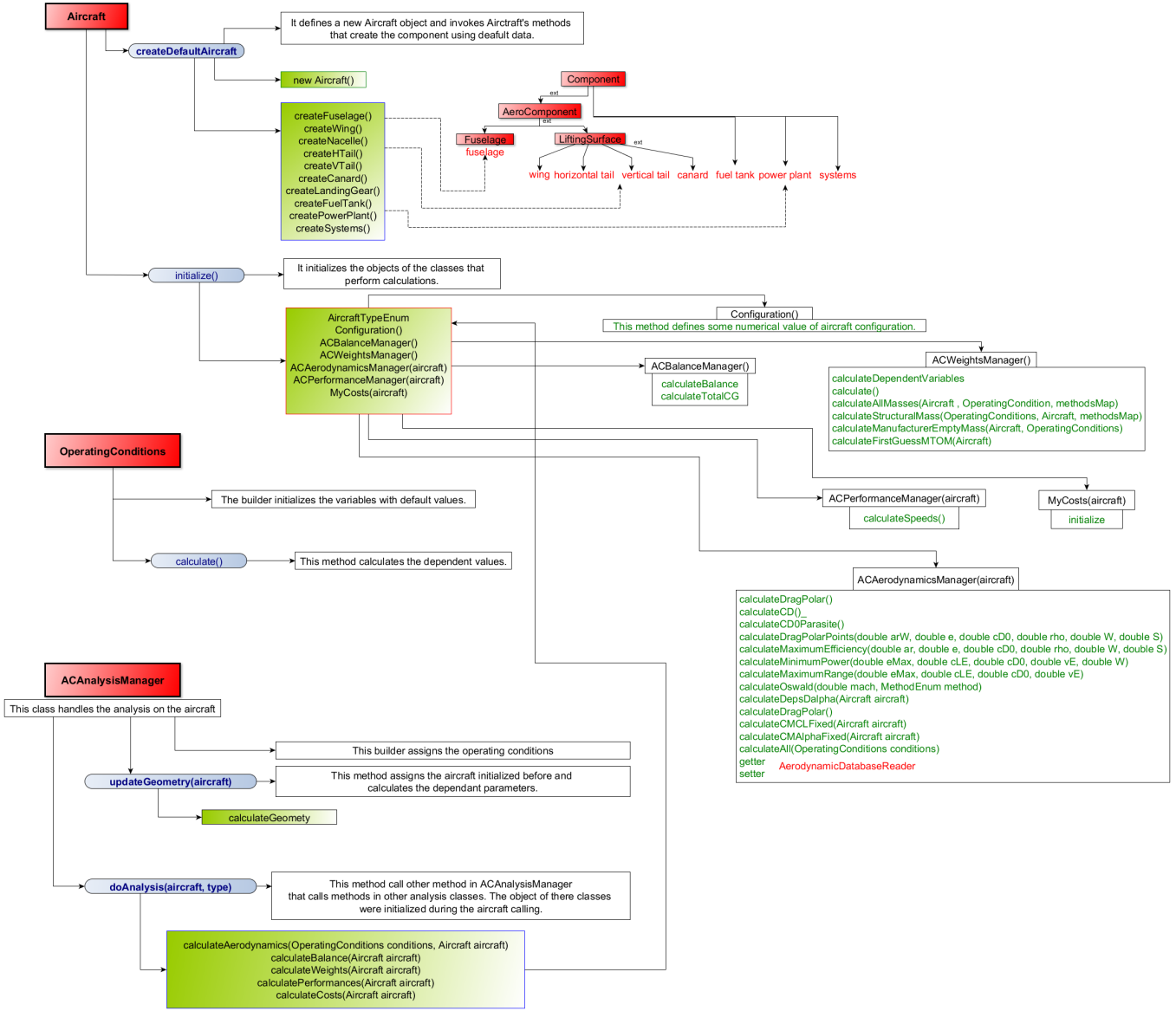

Figure 1.5: `Aircraft`, `OperatingConditions` and `ACAnalysisManager` flowchart

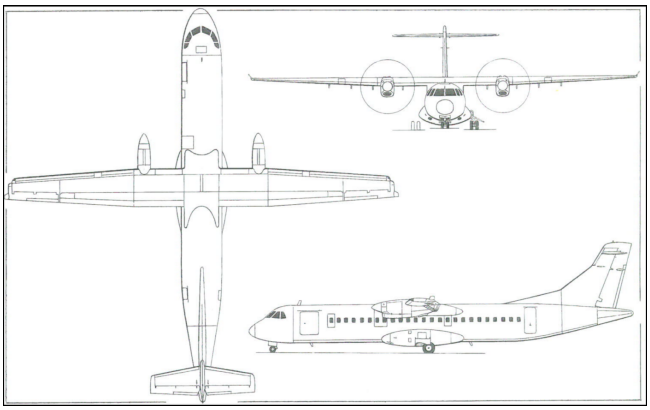

Figure 1.6: ATR-72 views – Jane's All the World's Aircraft 2004-2005

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	23 063,579 kg
`operatingEmptyMass`	operating empty mass	12 935,579 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 1.4: ATR-72 input data

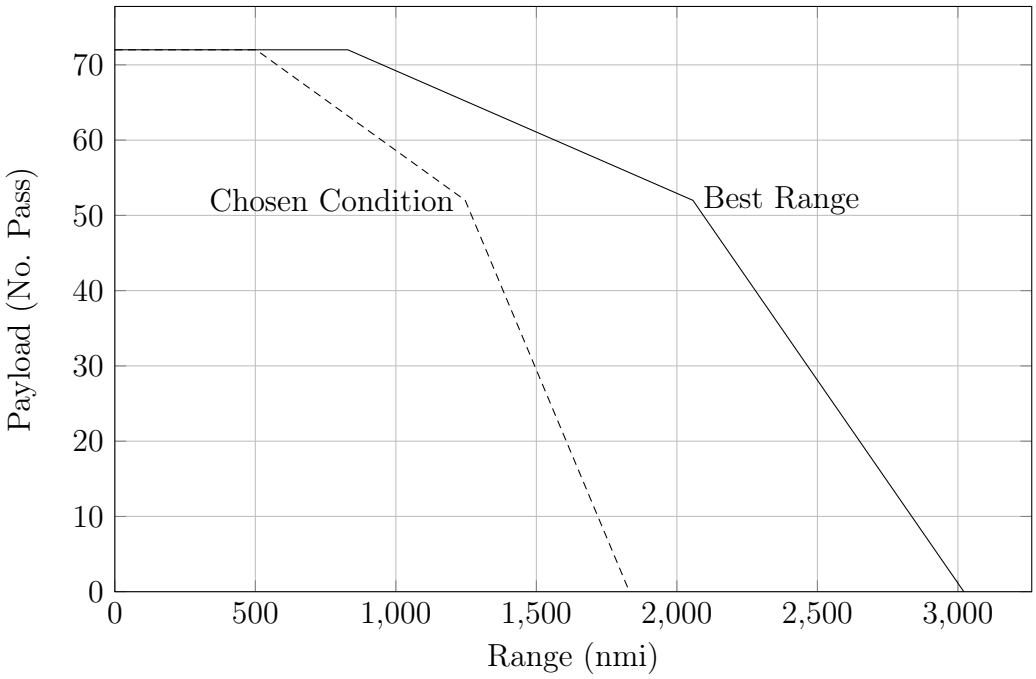

Figure 1.7: ATR-72 Payload-Range - Chosen Mach number and best range comparison

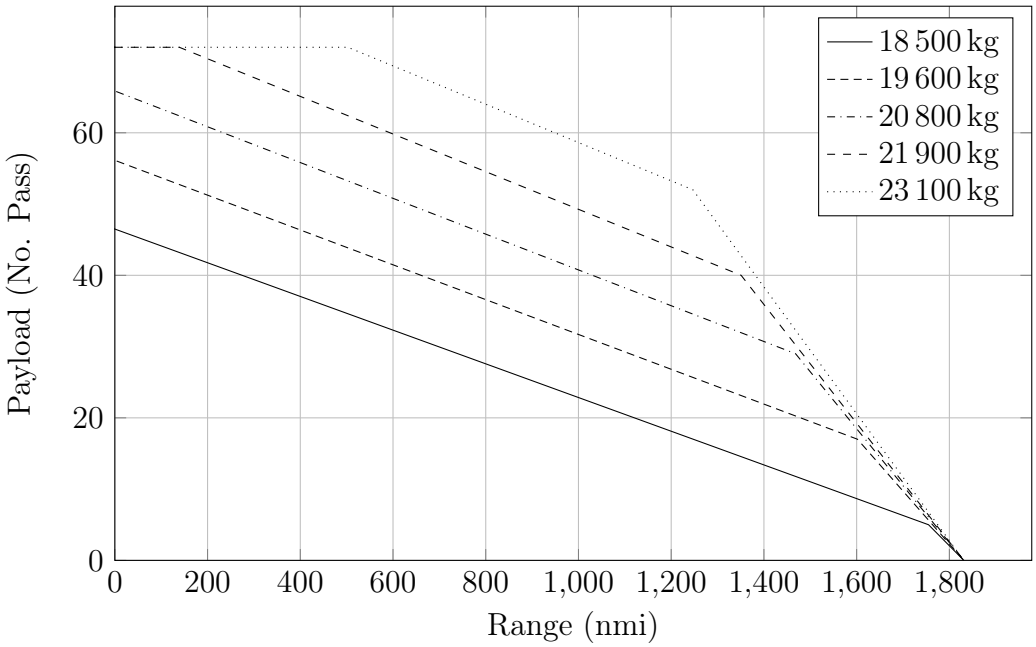

Figure 1.8: ATR-72 Payload-Range - Maximum take-off mass parameterization

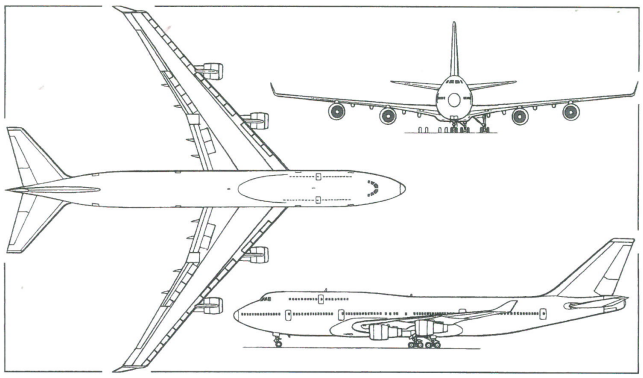

Figure 1.9: B747-100B views – Jane's All the World's Aircraft 2004-2005

Variables	**Description**	**Values**
`altitude`	Cruise altitude	11 000 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	354 991,506 kg
`operatingEmptyMass`	operating empty mass	153 131,986 kg
`maxFuelMass`	maximum fuel mass	147 409,520 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 1.5: B747-100B input data

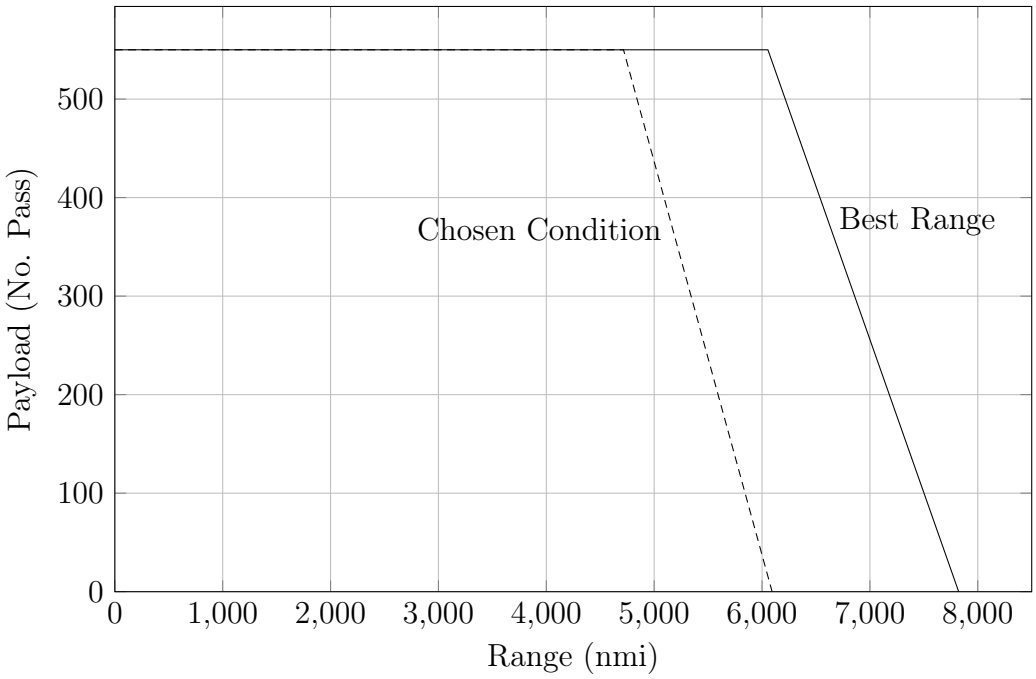

Figure 1.10: ATR-72 Payload-Range - Chosen Mach number and best range comparison

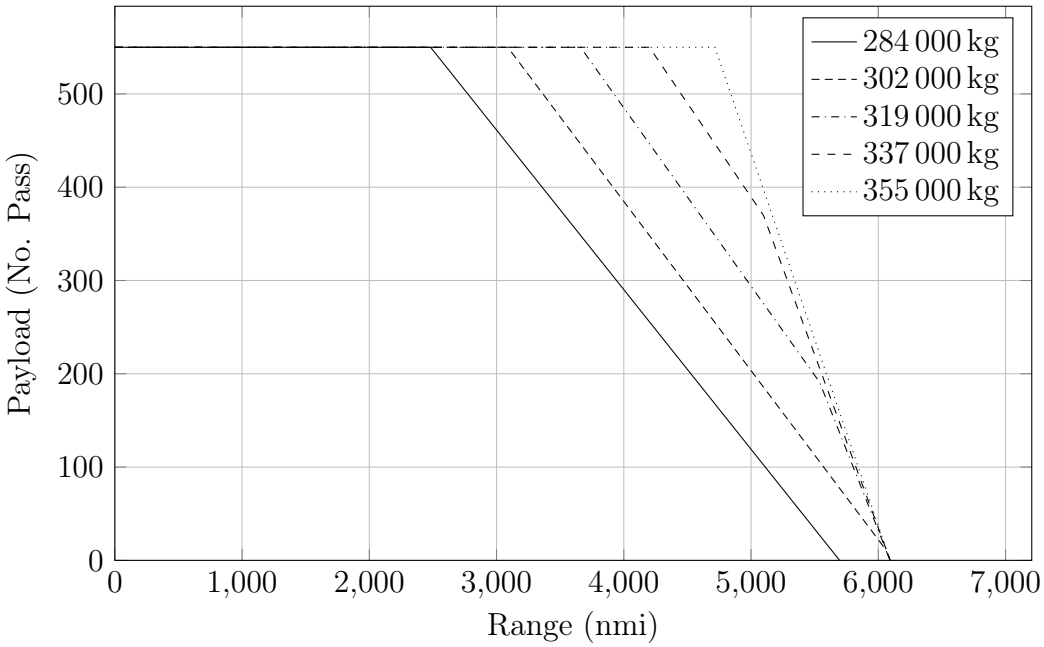

Figure 1.11: ATR-72 Payload-Range - Maximum take-off mass parameterization