UNIVERSITY OF TURKISH AERONAUTICAL

ASSOCIATION

FACULTY OF AERONAUTICS AND ASTRONAUTICS

SENIOR DESIGN PROJECT

Department	Astronautical Engineering
Subject	Optimization of a Liquid Rocket Engine Results Using Optimization Algorithms and Rocket Propulsion Analysis Software
Name and Surname	Binnur DORUK Mert SEVER
Student ID	130122014 130122037
Supervisor	Dr. Ceyhun TOLA

Preface

In this work, performance results of a liquid rocket engine modeled in Rocket Propulsion Analysis commercial software is optimized using coupled Matlab - Python scripts and ModeFRONTIER software. Values of oxidizer-fuel ratio and chamber pressure are optimized to yield highest possible specific impulse value to maximize the efficiency of the engine.

This design project has been an excellent and rewarding experience. We have been able to work with our professor and we are sure that will be able to help me with working on academic areas in future. We appreciate to dear Dr. Ceyhun TOLA to support this study and guide to us through all this project.

Ankara, 2018

Binnur DORUK & Mert SEVER

Summary

The senior design project which is explained in this report is done under the guidance of Dr. Ceyhun Tola in University of Turkish Aeronautical Association (UTAA) in Ankara at the second term of 2017-2018 academic year. In this project, brute force algorithm and multi-objective genetic algorithm methods (MOGA) are used and compared to eaxh other to optimize specific impulse (vac) in a short period of time. MATLAB, modeFRONTIER, Python and Rocket Propulsion Analysis (RPA) programs are used to accomplish the optimization process with MOGA solver. On the other hand, only Python and RPA programs are used. To apply the brute force method, a Python script is prepared and to apply the MOGA method, a flow chart in modeFRONTIER commercial software environment is constructed. Finally efficiency and applicability of these methods are compared.

This study reveals that scripting RPA software in assistance with another coding environment such as Python of Matlab is quite usefull for optimization processes since this process saves lots of time during the design phase of the rocket motors.

Table of Content

Table of Figures

1. Introduction

A rocket (from Italian rocchetto "bobbin")[1] is a missile, spacecraft, aircraft or other vehicle that obtains thrust from a rocket engine. Rocket engine exhaust is formed entirely from propellant carried within the rocket before use.[2]

Rocket motors work by activity and response and push rockets forward simply by expelling their exhaust the other way at fast, and this property makes rockets can also work in the vacuum. Rockets Works more efficently in space than atmosphere. Rockets can generate much more power than airbreathing engines and the result of that large accelerations occurs. As a result of these huge accelerations, the control of the rocket will be more difficult. The rocket needs more accurate control. To control the rocket rely on momentum, airfoils, auxiliary reaction engines, gimballed thrust, momentum wheels, deflection of the exhaust stream, propellant flow, spin, and/or gravity.

Figure 1: Sample rocket picture[3]

Solid Propellant Rocket

A solid-propellant rocket is a propulsion system using solid propellants (fuel/oxidizer). The earliest rockets were solid-fuel rockets powered by gunpowder; they were used in warfare by the Chinese, Indians, Mongols and Persians, as early as the 13th century.[4]

Liquid Propellant Rocket

A liquid-propellant rocket engine uses liquid fuel and oxidizers to produce thrust. Liquids are desirable because their reasonably high density allows the volume of the propellant tanks to be relatively low, and it is possible to use lightweight centrifugal turbopumps to pump the propellant from the tanks into the combustion chamber, which means that the propellants can be kept under low pressure. This permits the use of low-mass propellant tanks, resulting in a high mass ratio for the rocket.[5]

2. Optimization Methods and Analysis Softwares

In this senior design project, specific impulse (vac) is optimized by changing oxidizer-fuel ratio and chamber pressure. Brute force and MOGA methods are used to accomplish the optimization processes. These methods are operated on Python, MATLAB and modeFRONTIER environments. After these programs defined the inputs values, RPA program takes them and solves the function by using these inputs to find the value of specific impulse (vac).

2.1 Optimization Methods

2.1.1 Brute Force Algorithm

Brute force is a direct approach to fathom a problem based on expressions and definitions of the concepts included of the problem. It is considered as one of the most straightforward approach to apply and is beneficial for solving small-size instances of a problem.

2.1.2 Multi Objective Genetic Algorithm

Multi objective formulations are veridical patterns for many complicated optimization problems in engineering. In many problems, objectives under consideration can conflict with each other and single objective algorithm remains incapable, because of single objective algorithm results cannot be improper all objectives in problem. In such circumstances, MOGA is necessary to find acceptable results for all objects.

2.2 Analysis Softwares

2.2.1 Rocket Propulsion Analysis (RPA)

For a long time, MS-DOS or UNIX programs were used to determine the performance of the rocket motors by rocket propulsion experts, researchers and students. Despite these programs are still adequate for many cases, new users are not preferring them in general. First reason of this situation is old-fashioned interfaces of programs that have poor usability. The second one is absence of active development of them. Users begin a quest for new programs to make performance prediction of rocket engines.

Rocket Propulsion Analysis (RPA) is a preferable software in this stage. It is a multi-platform analysis tool intended for use in conceptual and preliminary design.

RPA determines rocket's thermodynamic properties, obtains chemical equilibrium composition of combustion products and predicts the theoretical rocket performance by using a few engine parameters such as combustion chamber pressure, used propellant components, and nozzle parameters.

A strong, demonstrated, and industry-acknowledged Gibbs free energy minimization approach is utilized to get the combustion composition. RPA can fulfill analysis of nozzle flows with shifting and frozen chemical equilibrium, nozzle performance with respect to overexpansion and flow separation, altitude performance, throttled engine performance. Optimization of propellant components mixture ratio for maximum specific impulse of bipropellant systems, estimation of test, that is actual, nozzle performance, nested analysis and thrust chamber thermal analysis are other operations that can done by using RPA. Nested analysis is related to four independent variables;

1. Component ratio
2. Chamber pressure
3. Nozzle inlet conditions
4. Nozzle exit conditions

RPA has expandable thermodynamic data library based on NASA Glenn thermodynamic database. Determination of combustion chamber size for given chamber pressure, propellant mass flow rate, or throat diameter and designing parboiling nozzle contour or truncated ideal nozzle contour (TIC) using two-dimensional (axisymmetric) method of characteristics can done by using RPA.

RPA differs from other tools in three main points;

1. Own implementation of thermodynamic analysis, whereas many other tools use CEA
2. Availability of development libraries
3. Intended for conceptual and preliminary design
    - Steady-state analysis
    - Simplified model initialization with minimum of input parameters
    - Wide usage of semi-empirical relations and coefficients

Again, three main points that RPA is usually used in them;

1. Conceptual and preliminary design of rocket engines and propulsion systems
2. Education and academic research
3. Library for development of commercial software

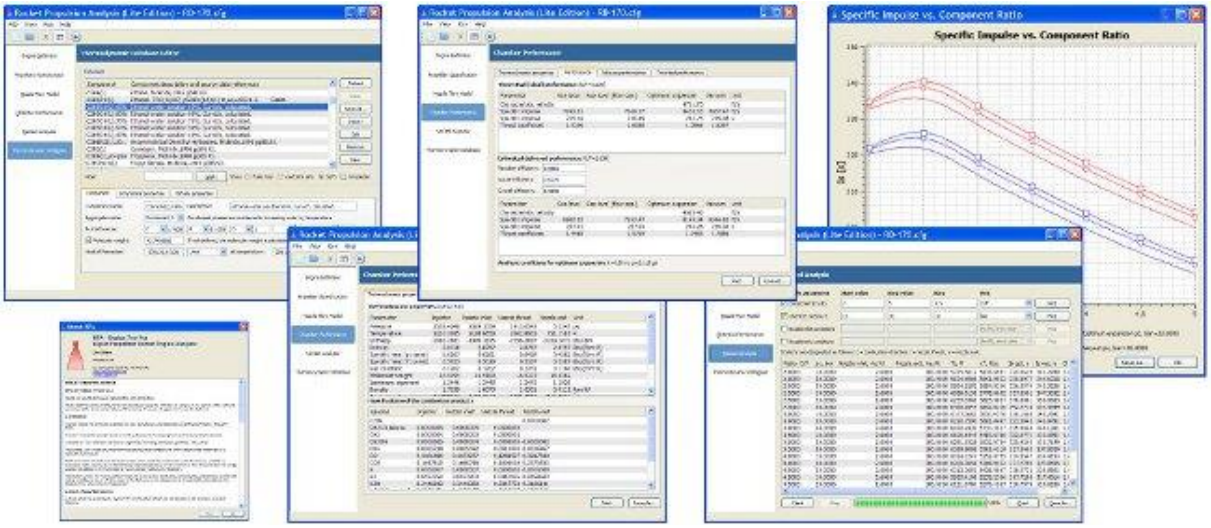

Figure 2 : Interface of Rocket Propulsion Analysis Program[6]]

2.2.2 Python

Coding is the foundation of computing. Over the years people, that interested in coding, try different programming languages.

Python's underlying improvement was initiated by Guido van Rossum in the late 1980s. Today, it is produced by the Python Software Foundation. Since Python is a multiparadigm dialect, Python software engineers can achieve their missions utilizing diverse styles of programming; imperative, object oriented, reflective or functional. Python can be utilized as a part of Web improvement, numeric programming, game development, serial port access and more.

Python is a broadly useful programming language which can be utilized for a wide assortment of utilizations.

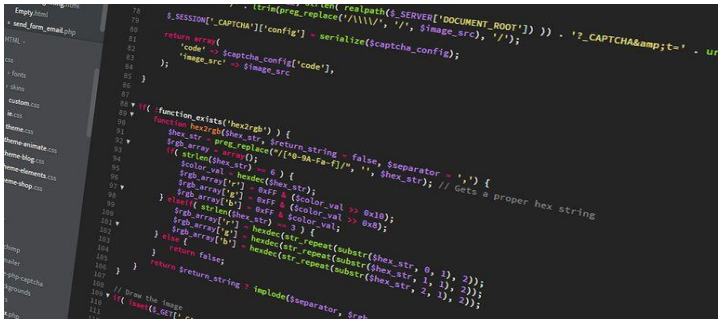

Figure 3 : A sample of Python compiler[7]

2.2.3 MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects, which together represent the state-of-the-art in software for matrix computation.

It has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to *learn* and *apply* specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others. [8]

2.2.4 ModeFRONTIER

modeFRONTIER was developed as a spin-off of the European research project on "Design Optimization", called FRONTIER. The project started in 1996. In 1999, following the end of the project, ESTECO was founded with the aim of further developing a commercial version of the FRONTIER software, which later became modeFRONTIER. In 2000, the commercial software, FRONTIER 2.0 was released (following FRONTIER 1.0, released in 1998). In 2002, the 2.5 version is released under the new name, modeFRONTIER, still in use today. modeFRONTIER is a Multidisciplinary Design Optimization (MDO) platform that is a field of engineering that uses optimization methods to solve design problems incorporating a number of disciplines. Its workflow based environment, and multi-objective optimization algorithms are used for streamlining the engineering design process to cut time and cost while obtaining improved results. [9]

3. Optimization

3.1 Optimization with Brute Force Algorithm

In this project, the RPA program is tried to be used more efficiently. To use RPA in an efficient manner Python is used. Python has a capability of run the programs. The RPA.exe is called several times by the code that is written in Python environment. The .cfg files are called by RPA and datas are gotten by RPA from .cfg files. By this way, changing the datas which are on .cfg file is achievable using Python.

Purpose of this project is optimization in a short amount of time. The best solution of specific impulse(vac) without running RPA several times by user is tried to find. Using the Python script developed within the content of this project, several internal ballistic performance analysis are performed using RPA software without using the interface of it and outputs are written on a file. The cfg file is located in the folder where RPA was installed. Finally, outputs, that are specific impulse(vac) results by every input, are checked by the in house Python scirpt and the best design is determined. Figure 4 summarizes the flow scheme of the stated optimization systematic

Figure 4:Flow chart of the optimization process used for the Brute ForceMethod

Details of the Python Script

Initially, some packages should be imported from Python library to be used.

To call and run the program on command line, begin with the import shown in Fig. 5:

```
import os
from subprocess import call
```

Figure 5 : Import to run program

After importing the libraries that program needs, the directory should be shown to computer to run the RPA.exe and to open example file which the program change data from it. The example file is moved into the RPA.exe folder so it is not need to show another directory for it. With read command the Python reads the .cfg file.

```
os.chdir("C:\\Program Files\\RPA\\2.3\\standard")
template = open('aestus.cfg', 'r+')
read_template = template.read()
```

Figure 6 :Opening directory of RPA and Aestus.cfg

At the beginning of the script, initial and limit values of the O/F ratio and the chamber pressure are asked as inputs .

```
print("enter the start point of o/f ratio value: ")
offsetR = float(input())
print("enter the finish point of o/f ratio value: ")
limitR = float(input())
stepR =0.1
print("enter the start point of chamber pressure value: ")
offsetT = float(input())
initial=offsetT
print("enter the finish point of chamber pressure value: ")
limitT = float(input())
stepT = 1
```

Figure 7:Getting inputs from user

The initial and the final points are also specified. In this portion, a loop is created to give the O/F ratio and the chamber pressure data from user to cfg file. To check all of the situations, two inputs are specified and for this reason 2 nested loops shown in Fig. 8 are created.

```
while offsetR <= limitR:
    cfg_file = read_template[:]
    cfg_file = cfg_file.replace('change_me', str(offsetR))
    while offsetT <= limitT:
        temp_file = cfg_file[:]
        temp_file = temp_file.replace('me_change', str(offsetT))
        output = open('input' + str(count) + '.cfg', 'w+')
        output.write(temp_file)
        output.close()
        mylistR.append(offsetR)
        mylistT.append(offsetT)
        os.mkdir("output" + str(count))
        call(["rpac.exe", "-i", "input" + str(count) + ".cfg", "-o", "output" + str(count)])
        offsetT=offsetT+ stepT
        count += 1
    offsetR=offsetR+ stepR
    offsetT=initial
    move=move+1
```

Figure 8 : Loop

In order to further explain the duty of these loops, following example is prepared. If the O/F ratio values are {1,2,3} and chamber pressure values are {8,9}, the loops are worked as in Fig. 9.

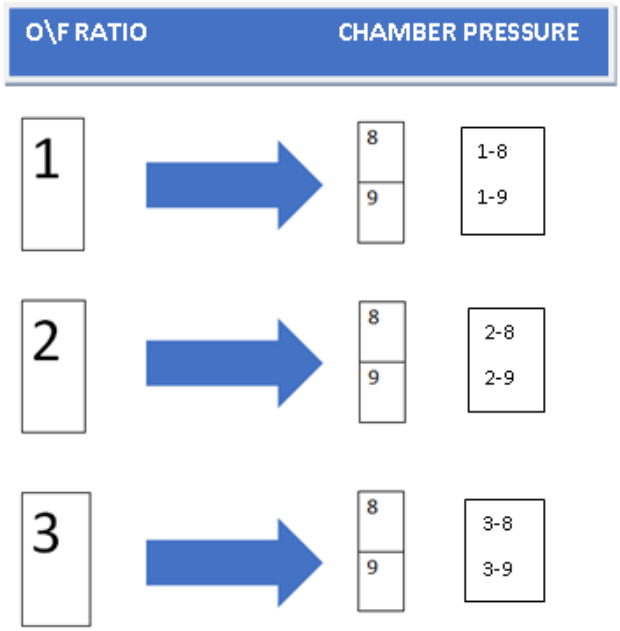

Figure 9 :Algorithm of optimization

Inside the loop, the script also replaces the data and write new values into the new input files. First, it replaces O/F ratio (Fig. 10) then keeps file with the replaced value and enter the other loop to change the chamber pressure (Fig. 11). It gives every chamber pressure value for the first O/F value and writes it in a different .cfg file (Fig. 12). After creating input files, the program starts to create the output folders and files. It gives a directory and creates a new folder after that writes the outputs for each operation (Fig. 13).

```
cfg_file = read_template[:]
cfg_file = cfg_file.replace('change_me', str(offsetR))
```

Figure 10 :Changing O/F ratio

```
temp_file = cfg_file[:]
temp_file = temp_file.replace('me_change', str(offsetT))
```

Figure 11 :Changing chamber pressure

```
output = open('input' + str(count) + '.cfg', 'w+')
output.write(temp_file)
output.close()
```

Figure 12 :Creating a new input file

```
os.mkdir("output" + str(count))
call(["rpac.exe", "-i", "input" + str(count) + ".cfg", "-o", "output" + str(count)])
```

Figure 13 :Creating an output directory and output file

With these operations, outputs for all situations and all inputs which are changed by Python are determined. Finally, the inputs belonging to the most efficienct design is selected by the script. To find out the maximum specific impulse (vac), a new loop is also created and all of theoutput files are checked. The script opens every output files and then searchs the specific impulse values. The numbers read from the output files are recorded into the memory and are compared with the next sequentially. If the next one is bigger the program registers the new number into its memory as the biggest one, if the next one is smaller memory remained the same. After these operations, highest value of specific impulse(vac) is selected by the Python script.

```
f = open("C:\\Program Files\\RPA\\2.3\\standard\\output" + str(counter) + "\\console_info.log")
strs = f.readlines()
```

Figure 14 :Opening and reading the output files

```
for z in strs:
    n = z.find("Specific impulse (vac):")
    if n > -1:
        x = float(z[25:-6])
        mylistOutput.append(x)
        break
if biggest <= x:
    biggest = x
    posindex = counter
counter += 1
f.close()
```

Figure 15 :Finding the highest specific impulse(vac) value

3.2 Optimization with MOGA-II Solver

modeFrontier commercial optimization software is used to accomplish the same optimization process with Multi Objective Genetic Algorithm Method. The modeFRONTIER has several optimization techniques and in the content of this wok MOGA-II solver is preferred since genetic algorithm methods converges much more faster than the brute force algorithms. Also in this method, writing parameters in harddisk is not expected which makes calculation faster.

Sequence of the optimization process is listed as follows:

- Computer runs modeFRONTIER
- ModeFRONTIER gives expected input parameters to cfg file by using Matlab
- MATLAB runs the script in command line and keeps variables
- ModeFRONTIER gets result variable (ISP(vac)) from Matlab.

Figure 16 illustrates the flowchart of the modeFRONTIER optimization process.

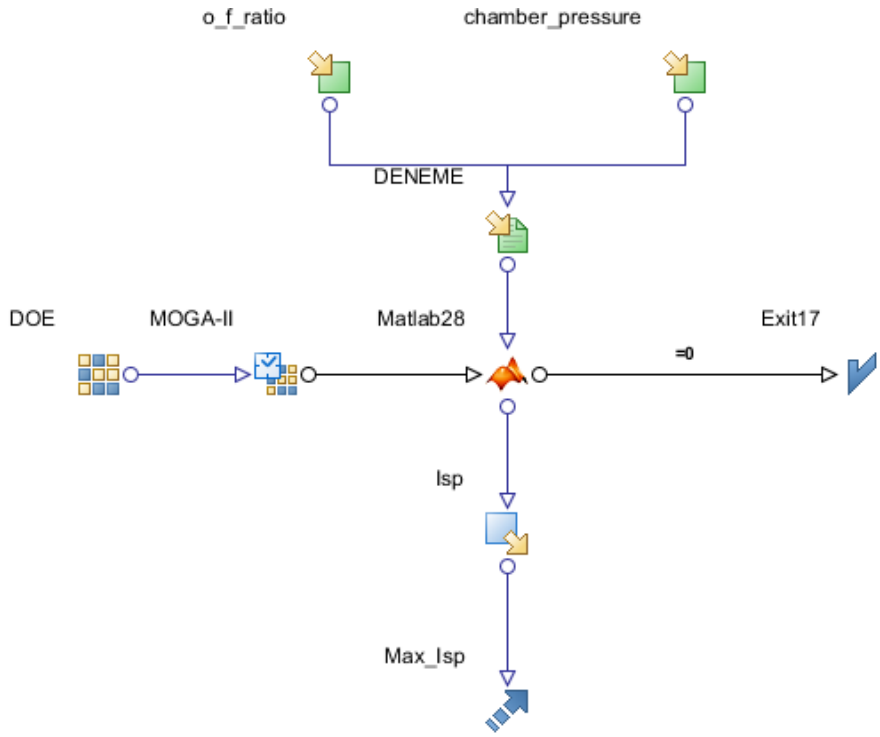

Figure 16 :Workflow tree of modeFRONTIER

In the software environment boundries of the design variables are defined. After that, locations of the oxidizer-fuel ratio and chamber pressure input rows are specified to the software (Figure 17).

```
import os
from subprocess import call
os.chdir("C:\\Program Files\\RPA\\2.3\\standard")
template = open('aestus.cfg', 'r+')
read_template = template.read()

offsetR = [redacted]

offsetT = [redacted]


count=1
cfg_file = read_template[:]
cfg_file = cfg_file.replace('change_me', str(offsetR))
temp_file = cfg_file[:]
temp_file = temp_file.replace('me_change', str(offsetT))
output = open('input' + str(count) + '.cfg', 'w+')
output.write(temp_file)
output.close()

call(["rpac.exe", "-i", "input" + str(count) + ".cfg", "-o", "output" + str(count)])
```

Figure 17 :Python script of project

In the figure, red and green parts represent the locations of the oxidizer-fuel ratio the chamber pressure values respectively. After that process modeFRONTIER assigns initial values for design variables and updates them considering the internal ballistic performance analysis results.

After locating the input parameters now programme uses the MATLAB to run deneme.py file in command line.

```
clear all
clc
[status,result]=system('deneme.py');
k=strfind(result,'Specific impulse (vac):    ')
B=result(k+24:(k+34));
Isp=str2num(B)
```

Figure 18 :MATLAB script of project

With this command Matlab runs the RPA programme using the command line environment and after the solution, it opens the output file of the RPA, finds the specific impulse (vac) value and transmists it to the modeFRONTIER. Thus, ModeFRONTIER maximizes the specific impulse(vac) value.

4. Results

Within the content of this work the liquid rocket engine is optimized in terms of specific impulse value using two different methods which are brute force and multi objective generic algorithms. Rocket Propulsion Analysis (RPA) commercial software is used to calculate the internal ballistic performance properties.

During the optimization process, chamber pressure is bounded between 5 bar to 50 bar and oxidizer-fuel ratio is bounded between 0.5 to 10. Solution process takes approximately 40 minutes for brute force method and takes approximately 5 minutes for genetic algorithm method which means MOGA-II solver is quite efficient.

Since modeFRONTIER has a graphical user interface, illustration of the results is much more easier than the bute force method. Using the brute force method it is only possible to find the design variables providing the highest specific impulse (vac) value (see Fig. 21) but it is possible to construct a detailed graph summarizing the feasible designs and highlighting the optimum values of oxidizer-fuel ratio and chamber pressure values with modeFRONTIER environment (See Fig. 25).

Inputs and Results of the Brute Force Algorithm

When the following inputs shown in Fig. 19 are inserted into the in house Python script, output values and files shown in Fig. 20 and Fig. 21 are determined sequentially.

```
C:\Python27>python RPA.py
enter the start point of o/f ratio value:
0.5
enter the finish point of o/f ratio value:
10
enter the start point of chamber pressure value:
5
enter the finish point of chamber pressure value:
50
```

Figure 19 :Input values of brute force method

```
Please set up the reasonable frozen equilibrium model.
Failed Could not find nozzle section equilibrium
 Initiating Max SI Search  ----------
Biggest SI (Vac) is on 920th test
```

Figure 20 :Output of brute force method

```
        d_lnV_d_lnT:      1.0000150
        d_lnV_d_lnp:     -1.0000003
Molecular weight, M:     25.2580919
Molecular weight, MW:    25.2580919
            Cp (eq):       43.22547       J/(mol K)
                            1.71135       kJ/(kg K)
            Cv (eq):       34.91076       J/(mol K)
                            1.38216       kJ/(kg K)
              gamma:       1.2381707
                  k:       1.2381702
                  R:       329.18053        J/(kg K)
                  a:       732.57940            m/s
   Specific volume:        99.01458          m^3/kg
                rho:         0.01010          kg/m^3
            C-phase:       0.0000000     Condensed phase mass fraction
           Velocity:     3321.40995            m/s
       Mach number:       4.5338566
     Pressure ratio:     1142.19457
Specific impulse (vac):  3451.90876            m/s
Specific impulse (opt):  3321.40995            m/s
Thrust coeff. (vac):        2.01421
Thrust coeff. (opt):        1.93807
```

Figure 21 :Output script of 920.th output file

According to the brute force algorithm, the optimum design parameters providing approximately 3451.909s specific impulse (vac) are a chamber pressure value of 50 bar with an oxidizer-fuel ratio of 2.4 (see Fig. 22 and 23).

```
combustionChamberConditions :
{
  pressure :
  {
    value=50.0;
    unit = "bar";
  };
};
```

Figure 22 :Input value of chamber pressure for maximum ISP (vac) value

```
propellant :
{
  components :
  {
    ratio :
    {
      value =2.4;
      unit = "O/F";
    };
```

Figure 23 :Input value of oxidizer-fuel ratio for maximum ISP (vac) value

Inputs and Results of the MOGA-II

When the following inputs shown in Fig. 24 are inserted into the modeFRONTIER, graphical results illustrated in Fig. 25 are determined.

Input Variable

	Name	Variable Type	Value	Expression	Distribution	Scale	Lower Bound	Upper Bound
0	o_f_ratio	Variable	0.0		None	0	0.5	10.0
1	chamber_pressure	Variable	0.0		None	0	5.0	50.0

Figure 24 :Input values of MOGA

According to graphical results of the genetic algorithm method, the optimum design parameters providing approximately 3451.918s specific impulse (vac) are a chamber pressure value of 50 bar with an oxidizer-fuel ratio of 2.403.

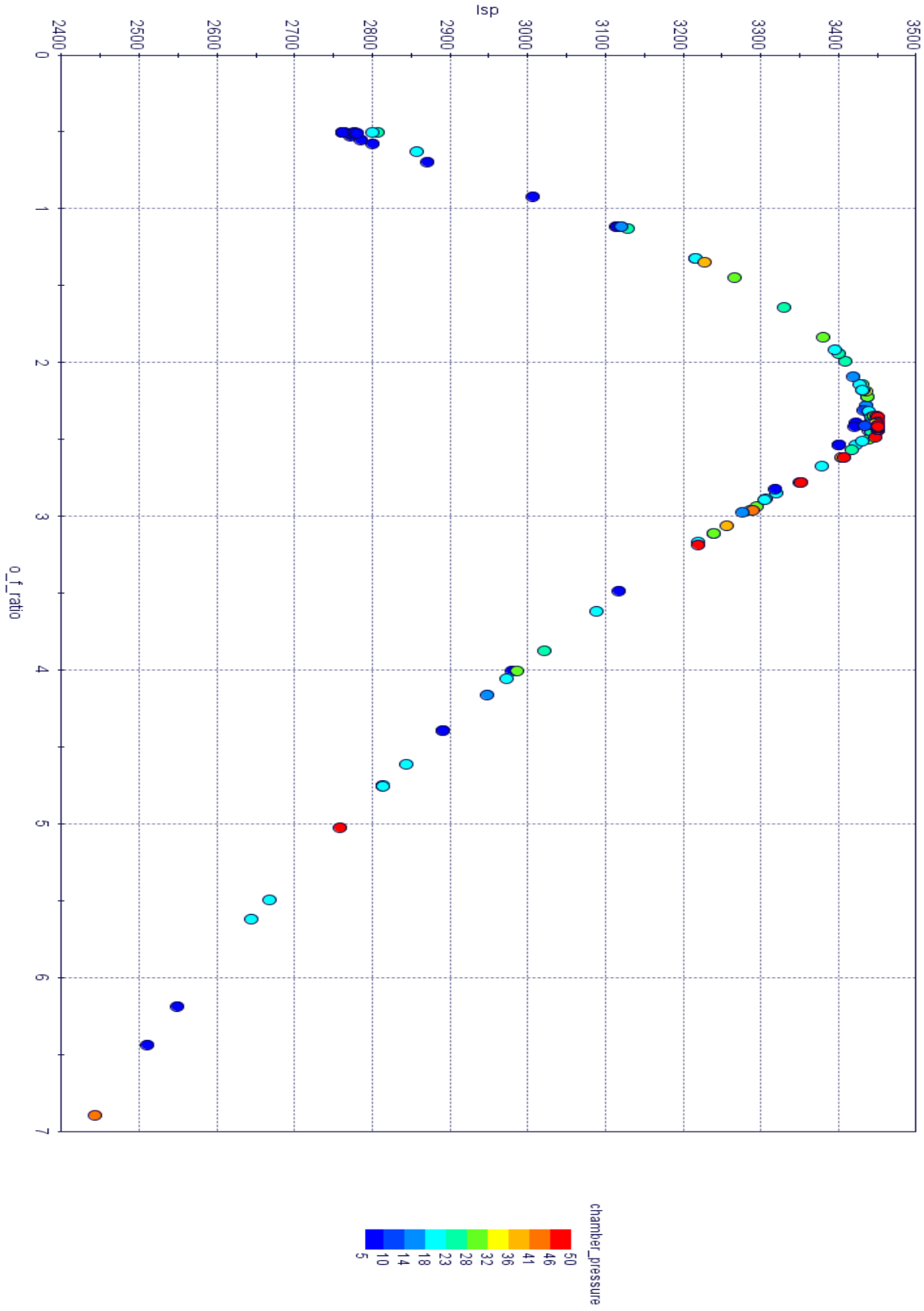

Figure 25 :Output graph of MOGA method

5. Conclusion

Purpose of this project is development a Python script using a specific optimization method for liquid rocket engine design and optimization. In this way, variation of the specific impulse (vac) with respect to the design parameters (oxidizer-fuel ratio and chamber pressure) are examined using 3D graphical outputs.

Throughout the optimization processes the most efficient chamber pressure and oxidizer-fuel ratio values are determined using both an in house Python script that uses brute force algorithm and modeFRONTIER commercial optimization software that uses genetic algorithm method.

The in house Python script uses brute force algorithm method to optimize the Rocket Propulsion Analysis (RPA) software results, varying the design parameters and evaluating the output files of the RPA software. Finally, the code creates a graph depending on two input values and outputs.

The modeFRONTIER uses the multi-objective genetic algorithm method for optimization and the program also takes two design variables and provides graphical results for the variation of the output parameter which is specific impulse(vac).

Comparison of the different methods showed that solution process takes approximately 40 minutes for brute force method and takes approximately 5 minutes for genetic algorithm method which means MOGA-II solver is quite efficient. On the other hand, the accuracy of the in house Python script and MOGA-II solver are almost same since they approximately determined the same oxidizer-fuel ratio and chamber pressure values for an optimum specific impulse(vac) value as a result of the optimization processes.

As a future work, its planned to further improve the in house Python script in three different ways. The first thing is, the program writes the inputs on the hard disk which slows down the optimization process, so running the program without writing inputs on hard disk with using an API will be more efficient. The second thing is the program only takes two design variables.

Increment of the design variable number will provide more detailed results about the design process. The final issue that is planned to improve is that, program will give a chance to user choosing an output type.

With this method user can use the program for optimizing not only specific impulse(vac) but also other things like chamber temperature.

On the other hand, modeFRONTIER part of the project is also planned to be improved by increasing the number of design variables. In this way, the modeFRONTIER chart will be more extensive.

6. Acknowledgements

With this project, Rocket Propulsion and programming courses that we took from our university is helpful for us. The programming course (JAVA) helped us to write a program on Python environment and the Rocket Propulsion course helped us on understanding the idea of RPA program.

Overall, the senior design project at University of Turkish Aeronautical Association has been a success. We could not be more thankful.

7. References

1. *Bernhard, Jim (1 January 2007). Porcupine, Picayune, & Post: How Newspapers Get Their Names. University of Missouri Press. p. 126. ISBN 9780826266019. Archived from the original on 19 November 2017. Retrieved 28 May 2016.*
2. *Sutton, George P.; Biblarz, Oscar (2001). Rocket Propulsion Elements. John Wiley & Sons. ISBN 9780471326427. Archived from the original on 12 January 2014. Retrieved 28 May 2016.*
3. https://bilgihanem.com/wp-content/uploads/2018/01/roket-turleri-nelerdir-960x480.jpg
4. http://www.spacetravel.com/reports/LockMart_And_ATK_Athena_Launch_Vehicles_Selected_As_A_NASA_Launch_Services_Provider_999.html
5. http://www.wikizero.net/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvTGlxdWlkLXByb3BlbGxhbnRfcm9ja2V0
6. http://w.lpre.de
7. http://maxpixel.freegreatpicture.com/Web-Web-Developer-Coding-Development-Code-944499
8. http://cimss.ssec.wisc.edu/wxwise/class/aos340/spr00/whatismatlab.htm
9. http://www.wikizero.net/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvTW9kZUZST05USUVS
10. http://w.lpre.de/docs/index.htm
11. https://stackoverflow.com/questions/8575062/how-to-show-matplotlib-plots-in-python
12. https://stackoverflow.com/questions/tagged/numpy
13. https://stackoverflow.com/questions/3345202/getting-user-input
14. https://stackoverflow.com/questions/16574898/how-to-load-ttf-file-in-matplotlib-using-mpl-rcparams/16574948#16574948