POLITECNICO DI TORINO

Master degree in Automotive engineering

Master degree thesis

Refinement of a Matlab code for the representation of experimental data and characteristic parameters evaluation for a light-duty application Diesel engine.

Supervisors:

Prof. Stefano D'Ambrosio

Prof. Roberto Finesso

Candidate:

Gabriele Sanna

Index

1.Introduction

The experimental Data analysis and their representation, as well as their simulation, is becoming even more important nowadays in the analysis of any type of internal combustion engine, in order to reduce as much as possible their testing phase and contemporary to have a reliable way to estimate their main parameters behavior. This practice is leading to an overall reduction in terms of costs and time, due to the increasing reduction of the needs for physical testing on benches.
On the market, several software of simulation of the engine processes are already present (e.g. GT-Power), but they often need for a deep and detailed knowledge of all the main construction parameters to be given as input data.
This project has been developed in order to acquire, represent and elaborate all the main parameters of the engine as well as its internal processes evolution, knowing only the main geometrical construction parameters such as bore, stroke and valve timing.
This has been done building a 0-D model for the evaluation and reconstruction of the pressure cycle and of the heat release rate, implemented using Matlab.
The output data of the described model, can be then compared to the experimental ones, focusing in particular on the values of maximum pressure reached and on the accuracy of the combustion process modeling.

2.Outline of the given data set

In a Compression Ignition (CI) engine during the intake phase fresh air is absorbed from the ambient, the fuel is injected towards the end of the compression stroke, just before the desired ignition time. As the fuel is injected (as an atomized liquid) , inside the combustion chamber the temperature and pressure are already very high and so some portions of the mixed air-fuel spontaneously ignite. This occurs during a period of time in which the injection has already started, but not yet finished. This phenomenon is due to spontaneous chemical reactions occurring at the injection of the fuel, but they take some time to be completed so the fuel doesn't produce immediately a visible flame or an appreciable pressure rise. So for CI engine, we define as the "ignition" the moment at which the pressure starts to rise appreciably and a flame front appears, while for the short period in which spontaneous reactions take place before the ignition, we refer as "delay period".
In general fuels for Diesel engines have their ignition point at around 350°C, far below the Spark Ignition engines ones, they have an high Cetane number, in order to be very reactive to the conditions present inside the combustion chamber.
The combustion process itself is described using the Wiebe function, that depicts the value of the fraction of the mass of fuel burned at every instant along the combustion phase.
Different possible injection strategies are used in CI engines, in this model the data given show the usage of two (or also three) pilot injections before the main one, with the possibility to use two pot-injections after the main one.
It is worth to note that all of those injections events, theoretically behave in the same way described before, ever single ones with their own delay period. But since the conditions of pressure and temperature inside the chamber are different at every injection event, they contribute in different ways to the speed and amount of the combustion process.
This last described phenomena are being modeled, in this project, defining a multiple Wiebe function, in which every injection event has its own evolution and in the end they all sum up to give a better description of the whole process.
The pressure cycle and the temperature evolution, are described using a zero-dimensional model that considers the behavior of the fraction of burned mass, the heat exchange trough the walls, the variation of the volume of the combustion chamber, and the first law of thermodynamics.

2.1 Engine construction parameters

The given data set is relative to a 1956 cubic centimeters Turbo-Diesel engine, with a bore of 90.5 millimeters and a stroke of 82 millimeters.
The compression ratio is identified as 16.5, while the elongation value (crank radius/ connecting rod length) is 0.28.

2.2 Main given parameters

From the engine map:

N_ENGINE: engine set revolution speed [rpm];

P: Useful developed power [kW];

P_max: Maximum pressure inside the cylinder [bar];

APMAX: Angle of occurrence of the maximum pressure value [°ATDC];

BMEP: Brake mean effective pressure [bar];

TORQUE: Developer torque at engine shaft [Nm];

IMEP: Indicated mean effective pressure [bar];

H_i: Fuel lower heating value [KJ/Kg];

BSFC: Brake specific fuel consumption [g/kWh];

QFUEL_MG: Fuel mass flow rate per engine cycle [mg/c];

IVC: Inlet valve closure angle [°BTDC];

EVO: Exhaust valve opening angle [°ATDC];

In addition to these parameters, the AVL Concerto test files provide us also with:

$QPIL_i$: Injected fuel quantity for the pilot injection considered [mg];

AMAIN: Desired reference angle for the main injection event [°BTDC];

INJCRV_TIPIL$_i$COR_MP: Corrected start of energising time component value [us];

3.Graphical representation of data

3.1 Data acquisition

Data are given on an Excel file where every line represent a specific Rpm point and every column represents a specific computed quantity.
The Matlab function "*xlsread*" is used to acquire from those data from the Excel grid, and to subdivide them into a matrix relative to the numerical characters (N) and into one relative to the text characters (T).
Using a *for* cycle, numerical characters listed as *NaN* in the Excel file, are deleted in order to have an n-lines and m-columns matrix (this matrix is called A and it includes n sample points ans m evaluated parameters).
Sample points are characterized by an engine spin value (Rpm) and a mean effective pressure (bmep) value, respectively listed in the column number 1 and in the column number 2 of the A matrix; they are necessary to build another matrix using the "*meshgrid"* function, taking as input values the vectors containing rpm and bmep values sorted in increasing order through the function "*sort*".
Now it is possible to interpolate the values of every given parameter, on all the points of the built grid, using the function "*scatteredInterpolant*", that receives as input values the vectors Rpm, bmep, and the one relative to the investigated parameter, in addition to the interpolating mode ("*linear*").

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1	P254	ALPHA	ALPHA_d	N_dem_D	N_ENGINE	NOxCalc	P	p_me_unk	SPEED	T_dem_D	T_dem_E	T_IDLE	TORQUE	TSetD_K	V_ACT	FB_MASS	FB_mact	FB_MEAN	FB_TIME	FB_VAL	FB_Tout	APMAX1	APMAX2	APMAX3	APMAX4	IMEP1	IMEP2
2		%	%	rpm	rpm	g/kWh	kW	bar	rpm	Nm	Nm	Nm	Nm	Nm	km/h	g	kg/h	kg/h	s	kg/h	°C	deg	deg	deg	deg	bar	bar
3		0	0	1159	850	207,88	0	0	900	0	0	0	0	-0,3	0	4,43	0,4	**	40	0,41	23,91	3	2	3	3	1,11	1,45
4		4,7	4,7	1000	999	2,95	0,85	0,52	999	0	7,8	0	8,1	2,8	0	6,63	0,6	**	40	0,61	23,91	3	11	11	3	1,74	2,05
5		5,6	5,5	1000	999	1,99	1,66	1,02	999	0	15,6	0	15,8	8,9	0	8,2	0,74	**	40	0,76	23,91	13	12	12	12	2,32	2,68
6		7,7	7,7	1000	998	1,7	3,3	2,02	998	0	31,1	0	31,5	20,1	0	11,43	1,03	**	40	1,04	23,91	14	12	13	12	3,43	3,7
7		9,6	9,6	1000	998	1,61	4,94	3,04	998	0	46,7	0	47,3	30,8	0	14,93	1,34	**	40	1,34	23,91	13	11	13	13	4,35	4,75
8		11,7	11,7	1000	998	**	6,58	4,05	998	0	62,3	0	63	45,5	0	18,58	1,67	**	40	**	23,91	11	12	12	11	5,45	5,82
9		14,5	14,5	1000	998	1,96	8,23	5,06	998	0	77,8	0	78,7	63,5	0	22,26	2	**	40	2,01	23,91	11	11	12	11	6,69	6,82
10		17,9	17,9	1000	998	3,32	9,86	6,06	998	0	93,4	0	94,3	81	0	25,93	2,33	**	40	2,29	23,91	13	11	11	12	7,76	7,84
11		20,4	20,4	1000	998	4,13	11,49	7,06	998	0	109	0	109,9	98,3	0	29,93	2,69	**	40	2,66	23,91	11	11	11	12	8,9	8,91
12		23,3	23,3	1000	999	5,46	13,13	8,07	999	0	124,5	0	125,6	115,8	0	34,28	3,09	**	40	3,06	23,91	12	11	11	10	9,99	9,98
13		4,6	4,6	1250	1249	5,24	1,06	0,52	1249	0	7,8	0	8,1	1,9	0	8,31	0,75	**	40	0,75	23,91	2	2	1	2	2,08	2,09
14		5,7	5,7	1250	1249	1,99	2,05	1,01	1249	0	15,6	0	15,7	8,9	0	10,23	0,92	**	40	0,94	23,91	3	12	12	4	2,54	2,59
15		8,6	8,6	1250	1249	1,21	4,08	2,01	1249	0	31,1	0	31,2	22,2	0	14,34	1,29	**	40	1,33	23,91	13	12	12	13	3,5	3,65
16		12,1	12,1	1250	1248	**	6,15	3,02	1248	0	46,7	0	47	35,3	0	18,69	1,68	**	40	1,68	23,91	12	13	11	12	4,66	4,84
17		14,6	14,6	1250	1248	1,31	8,19	4,02	1248	0	62,3	0	62,6	51	0	23,19	2,09	**	40	2,03	23,91	11	13	11	13	5,91	5,95
18		17,4	17,4	1250	1249	1,36	10,22	5,02	1249	0	77,8	0	78,2	67,8	0	27,72	2,49	**	40	2,54	23,91	12	12	14	12	6,85	7,09
19		20,6	20,5	1250	1249	**	12,26	6,02	1249	0	93,4	0	93,8	84,7	0	32,08	2,89	**	40	2,89	23,91	11	12	13	11	7,96	8,02
20		23,5	23,5	1250	1249	1,89	14,3	7,02	1249	0	109	0	109,3	101,4	0	36,72	3,31	**	40	3,26	23,91	12	12	12	13	8,9	9,14
21		26,5	26,5	1250	1249	**	16,32	8,02	1249	0	124,5	0	124,8	118,2	0	41,35	3,72	**	40	**	23,91	13	13	13	12	10,11	10,37
22		29,3	29,3	1250	1249	2,95	18,36	9,02	1249	0	140,1	0	140,3	135,2	0	46,1	4,15	**	40	4,11	23,91	13	11	12	13	11,25	11,37
23		31,8	31,8	1250	1250	**	20,41	10,02	1250	0	155,7	0	156	152,5	0	50,79	4,57	**	40	**	23,92	12	12	12	12	12,2	12,37
24		33,4	33,4	1250	1250	5,67	22,51	11,05	1250	0	171,2	0	172	170,3	0	55,8	5,02	**	40	5,01	23,94	12	12	11	12	13,56	13,69
25		35,6	35,6	1250	1250	**	24,25	11,9	1250	0	186,8	0	185,2	185,2	0	60,21	5,42	**	40	**	23,97	11	11	11	12	14,28	14,75
26		41,3	41,3	1250	1251	8,64	28,56	14,01	1251	0	217,9	0	218,1	222,4	0	71,1	6,4	**	40	6,46	24	12	12	12	12	16,93	17,07
27		47,3	47,3	1250	1251	**	32,66	16,01	1251	0	249	0	249,3	258,4	0	85,13	7,66	**	40	**	24,03	12	12	12	12	19,41	19,21
28		98,7	98,7	1250	1252	**	34,34	16,83	1252	0	280,2	0	262	273,4	0	91,94	8,27	**	40	**	24,04	12	13	13	12	20,44	20,26
29		100	100	1250	1252	**	34,09	16,71	1252	0	0	0	260,1	271,1	0	91,83	8,26	**	40	**	24,05	13	13	12	12	20,1	20,04
30		5,3	5,3	1500	1499	6,51	1,27	0,52	1499	0	7,8	0	8,1	2,8	0	10,04	0,9	**	40	0,88	24,05	4	3	3	2	2,03	2,03
31		6,4	6,4	1500	1499	1,85	2,49	1,02	1499	0	15,6	0	15,9	9	0	12,44	1,12	**	40	**	24,05	3	3	3	3	2,63	2,56
32		9,9	9,9	1500	1499	1,05	4,92	2,01	1499	0	31,1	0	31,3	23,1	0	17,24	1,55	**	40	1,57	24,05	14	13	13	14	3,81	3,57
33		12,7	12,7	1500	1499	**	7,42	3,04	1499	0	46,7	0	47,3	38,5	0	22,35	2,01	**	40	2,04	24,06	13	13	13	14	4,82	4,77
34		15,6	15,6	1500	1499	0,71	9,86	4,03	1499	0	62,3	0	62,8	54,9	0	27,74	2,5	**	40	2,5	24,09	14	13	13	14	5,78	5,71

```
10      %% EXCEL DATA ACQUISITION
11
12 -    [N,T]=xlsread('A20DTH_BetalplusHW_EngineMapping.xls');
```

```
  %Matrix of data acquired from excel
for(i=1:size(N,2)) %definition of matrix columns

    m=1;

    for(j=1:size(N,1)) %definition of matrix rows

        A(m,i)=N(j,i);
        m=m+1;

    end

end
```

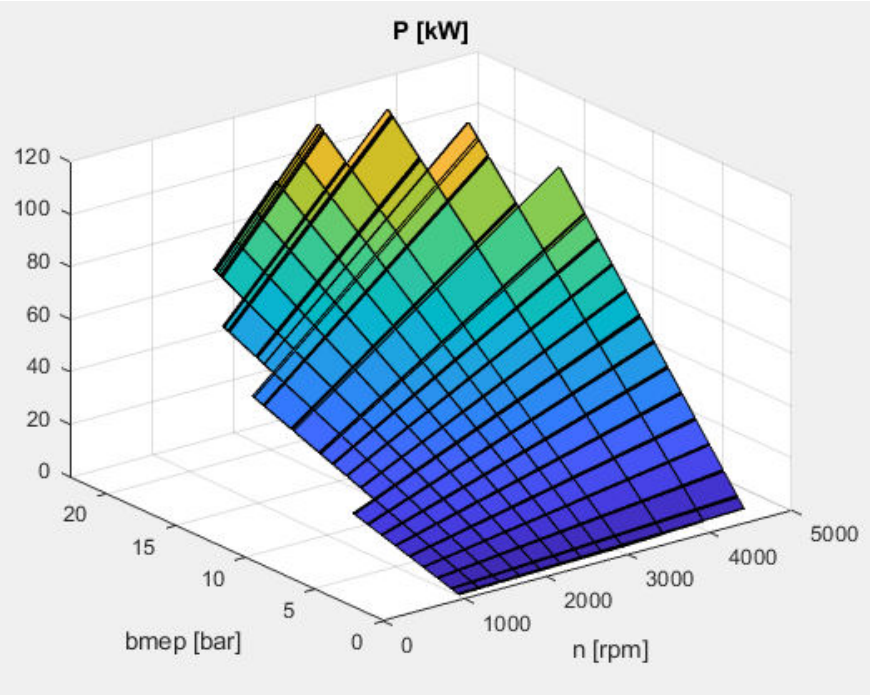
```

rpm=A(:,8);
bmep=A(:,7);

n_i=202; %suddivisioni della meshgrid

px=(max(rpm)-min(rpm))/(n_i-1);
py=(max(bmep)-min(bmep))/(n_i-1);

[Xq,Yq]=meshgrid(sort(rpm),sort(bmep));
```

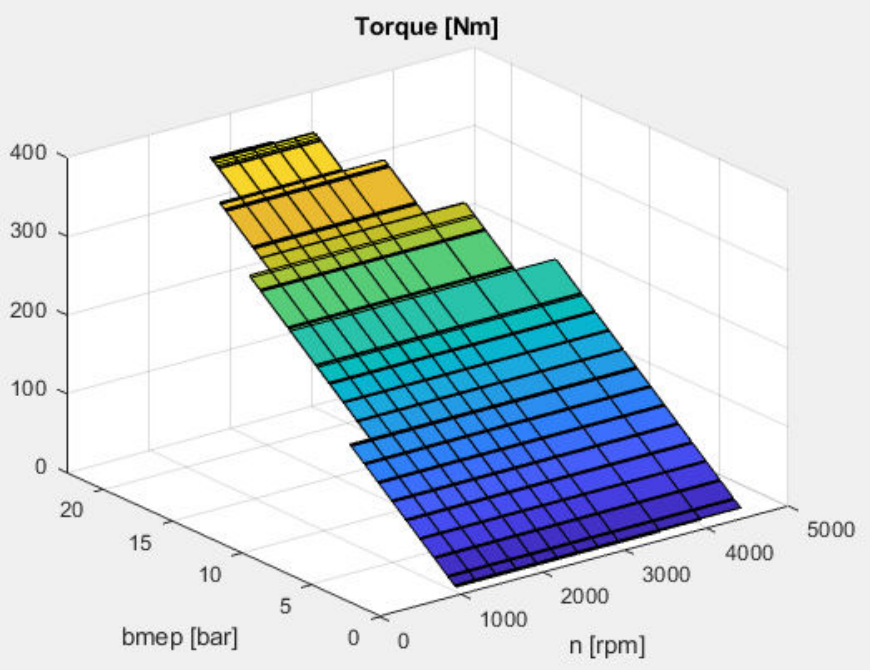
3.2 Representation of the listed plane

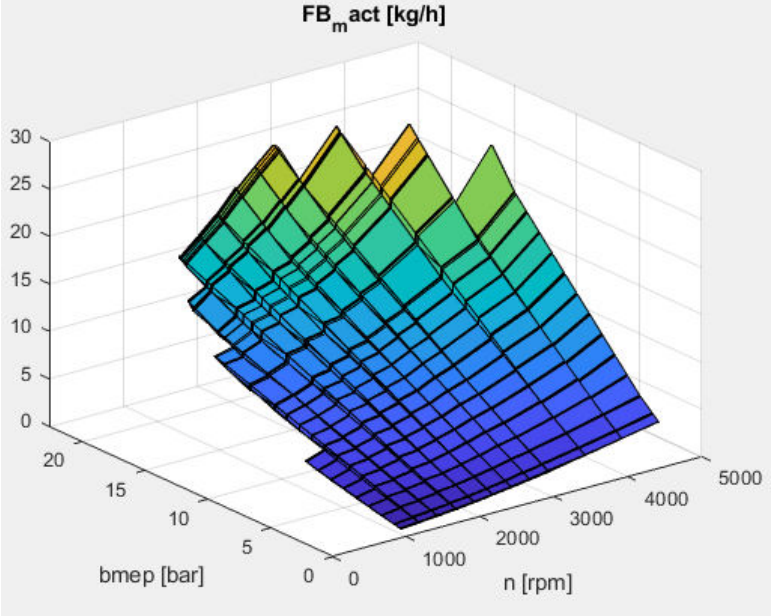
The following step is the graphical representation of all the parameters, using the "*surf*" function, to which X and Y coordinates of the grid are given, in addition to the values obtained by the previously mentioned function "*scatteredInterpolant*" as shown below.

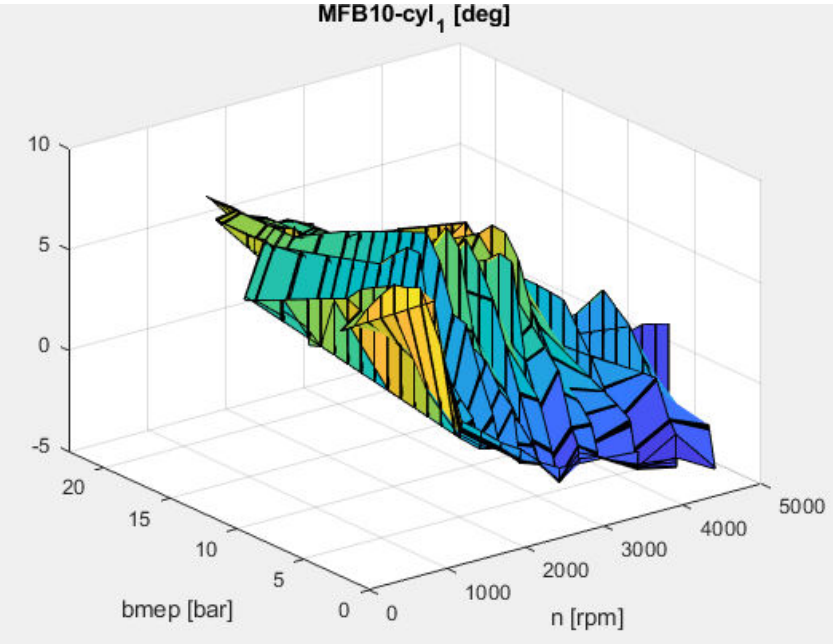
```
for(i=1:size(A,2))

    F=scatteredInterpolant(rpm,bmep,A(:,i),'linear','none');

    vq1=F(Xq,Yq);
    B(:,:,i)=vq1;
    ti=strcat(T(1,i),'[',T(2,i),']');

    figure
    surf(Xq,Yq,vq1);
    table(ti);
    xlabel('n [rpm]');
    ylabel('bmep [bar]');
    drawnow
end
```

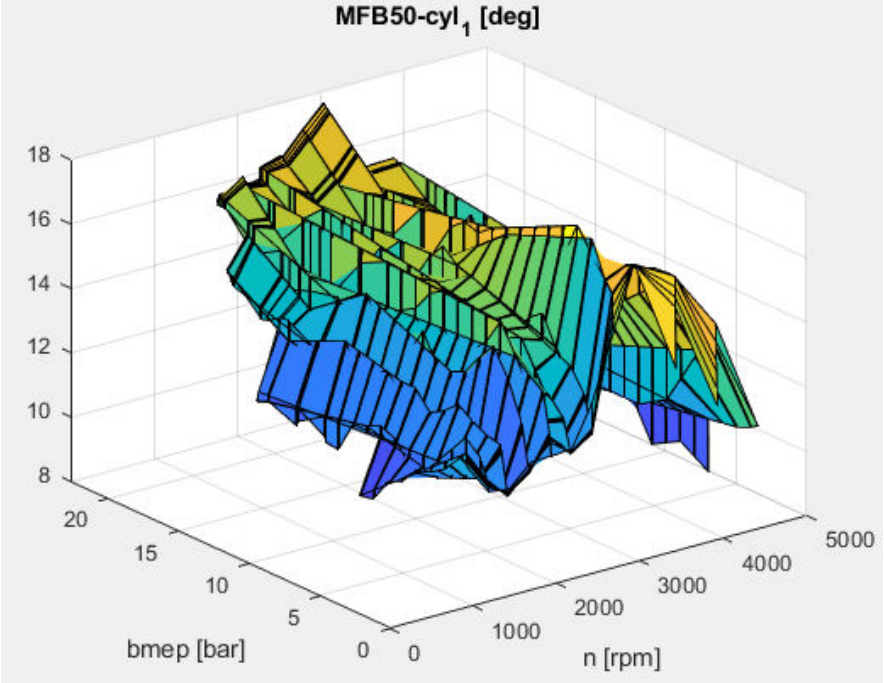
3.3 Graphical representation of the main parameters

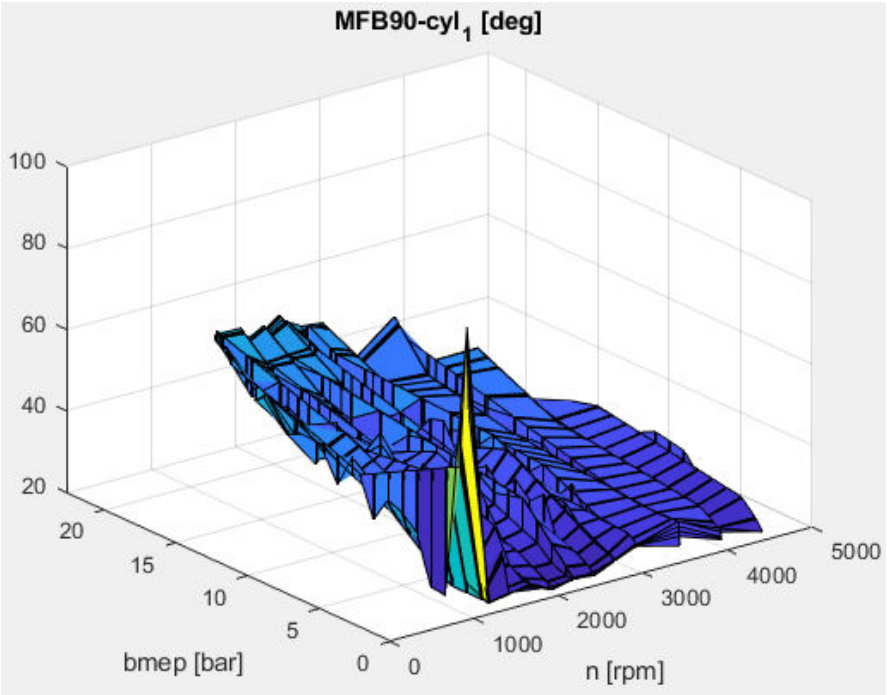


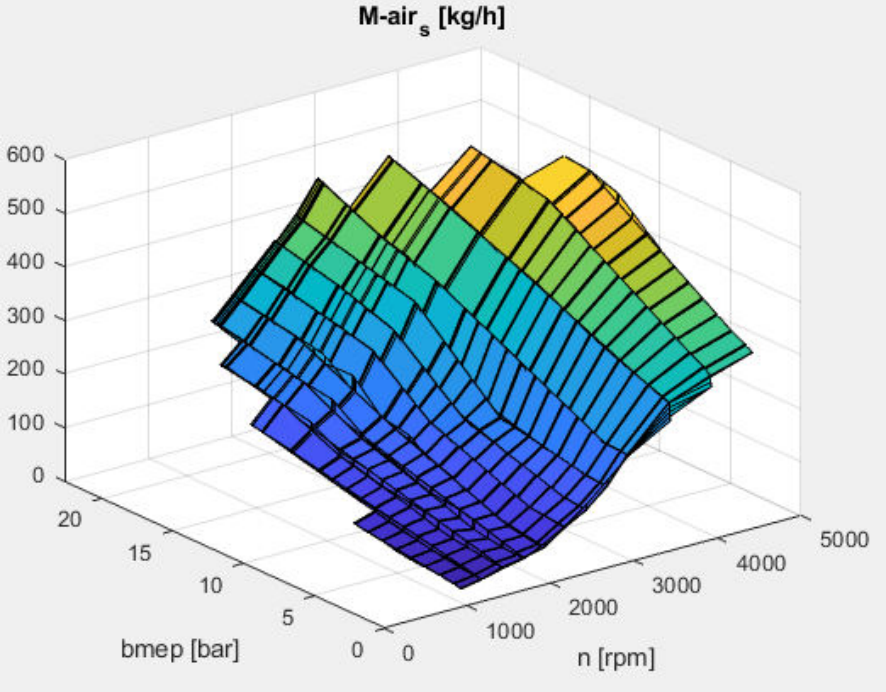


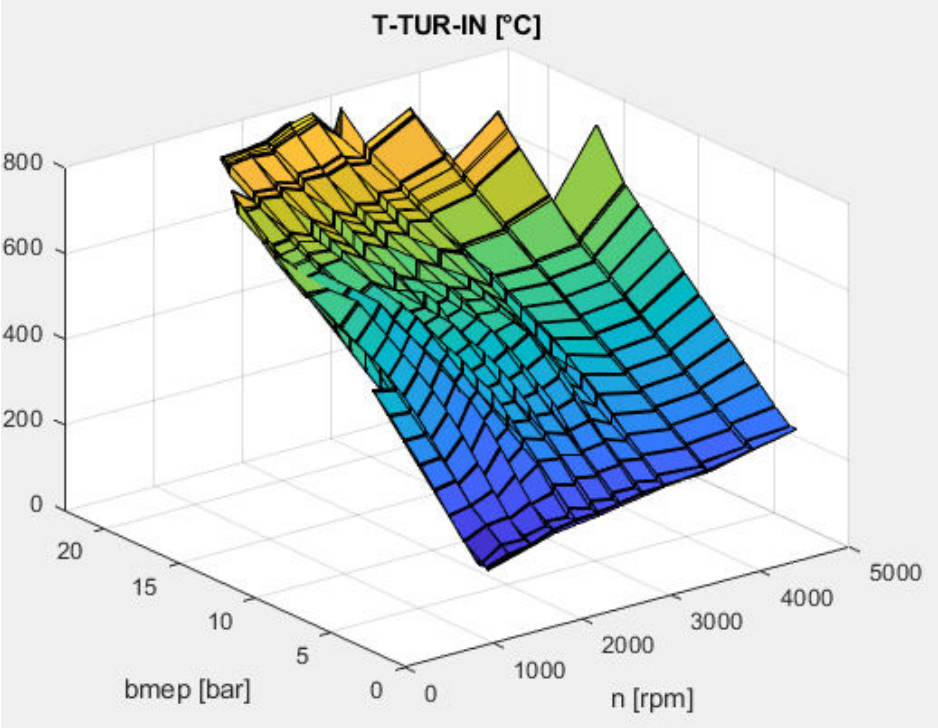

FB_m act [kg/h]
30
25
20
15
10
5
0
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]

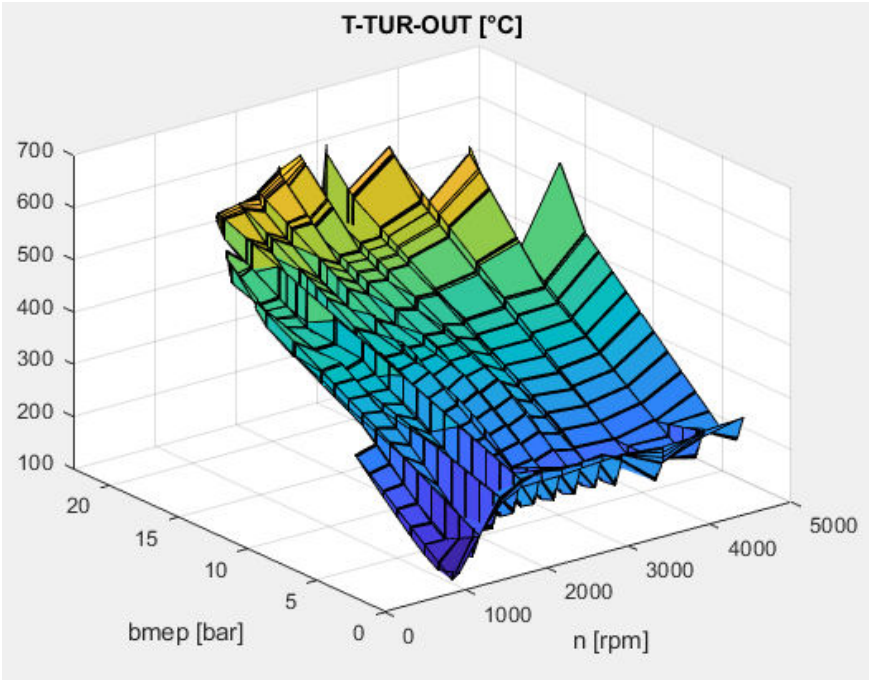

MFB10-cyl_1 [deg]
10
5
0
-5
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]

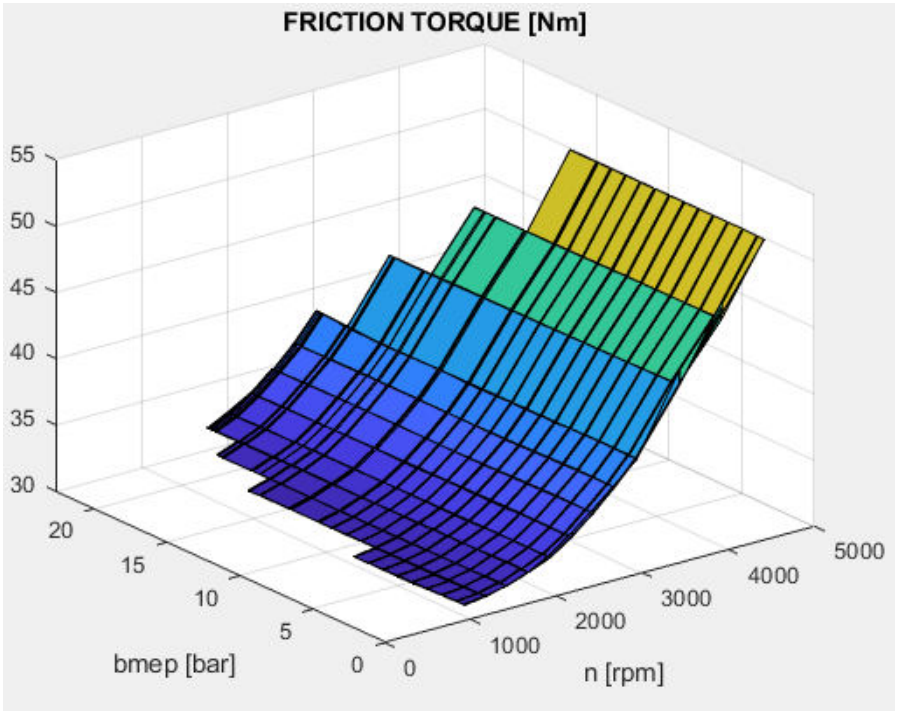

MFB50-cyl$_1$ [deg]
18
16
14
12
10
8
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]

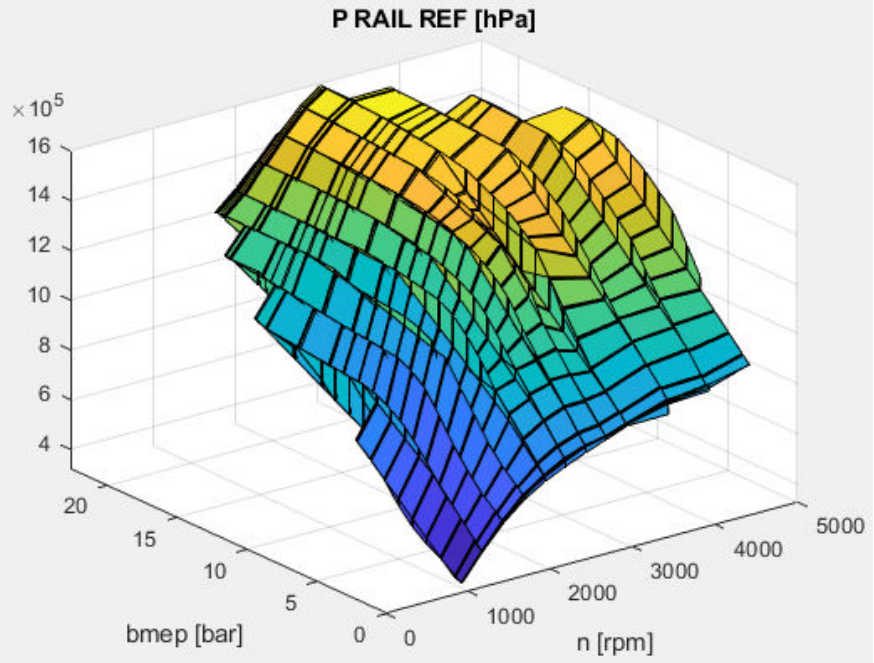

MFB90-cyl$_1$ [deg]
100
80
60
40
20
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]

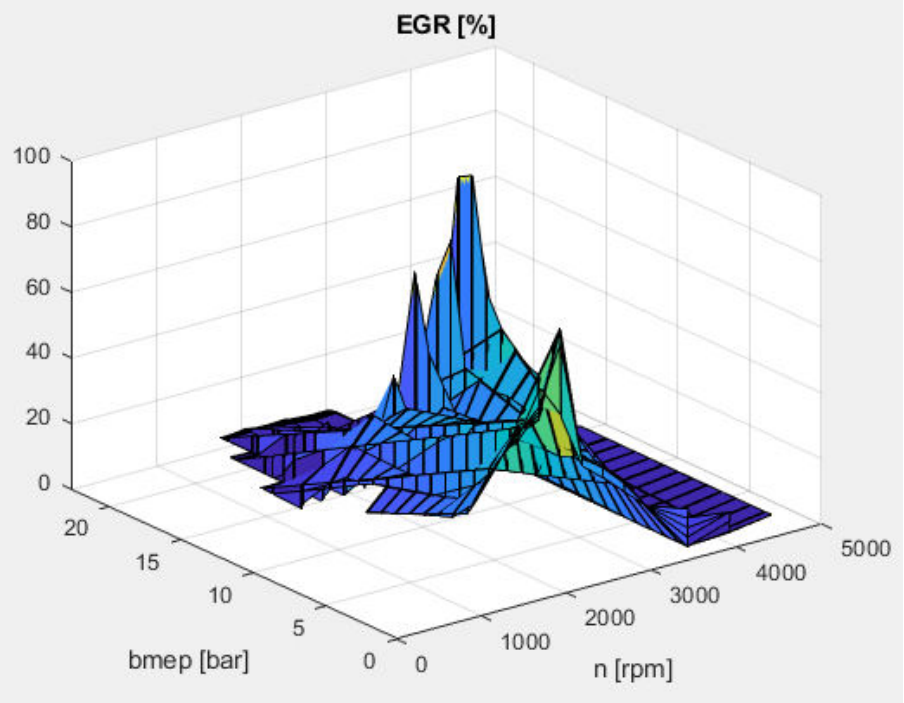

M-air$_s$ [kg/h]
600
500
400
300
200
100
0
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]

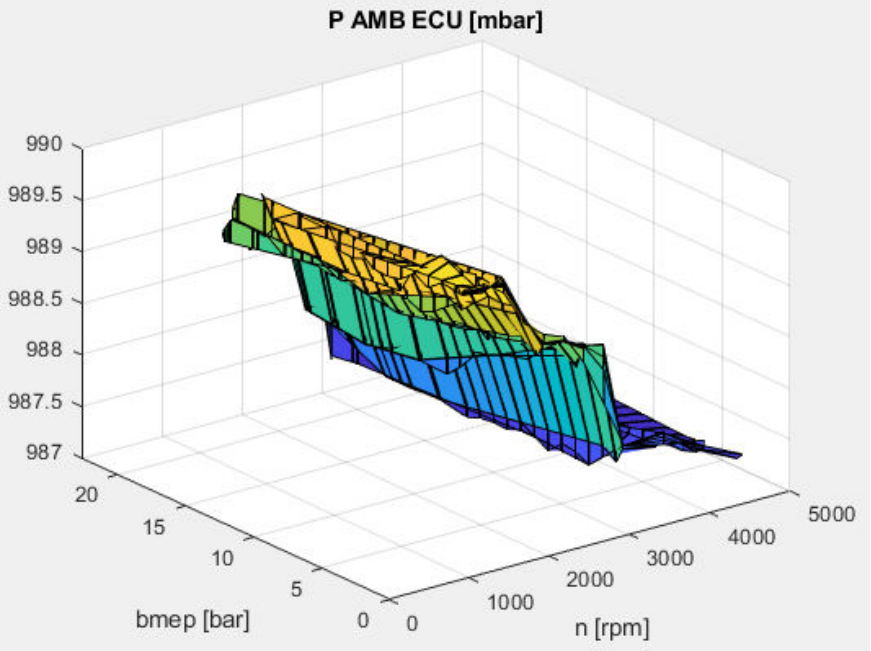

T-TUR-IN [°C]
800
600
400
200
0
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]

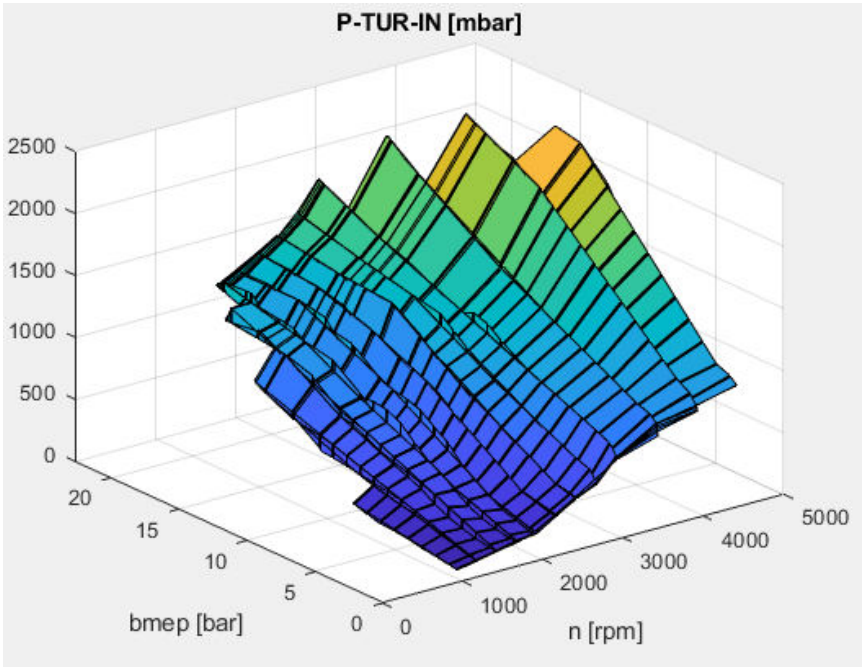

T-TUR-OUT [°C]
700
600
500
400
300
200
100
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]

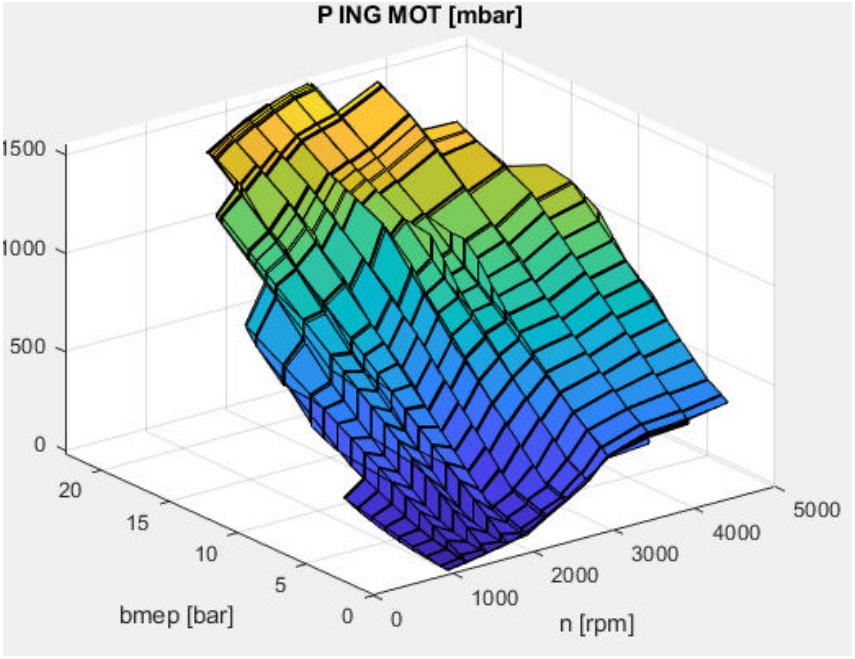

FRICTION TORQUE [Nm]
55
50
45
40
35
30
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]

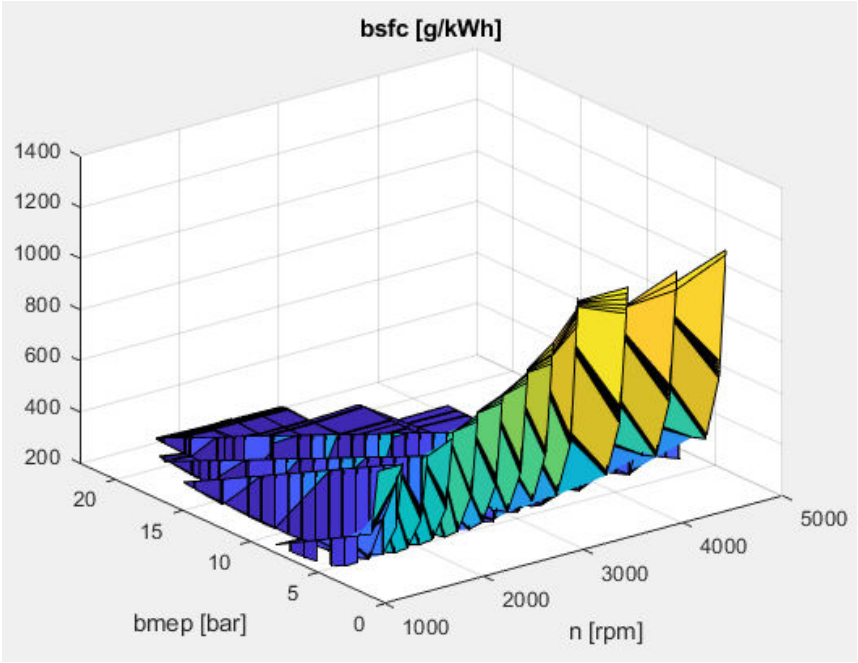

P RAIL REF [hPa]
×10^5
16
14
12
10
8
6
4
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]

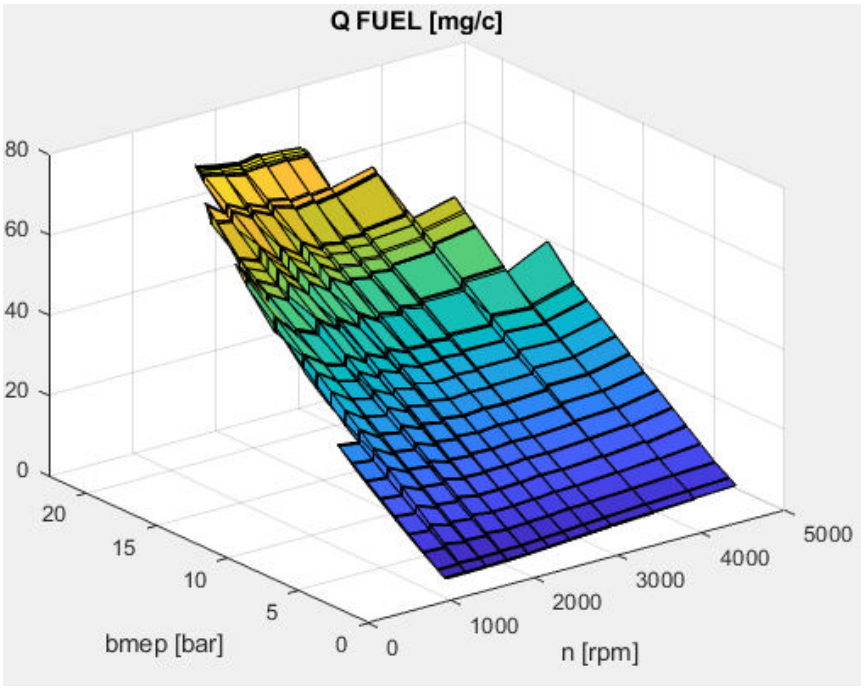

EGR [%]
100
80
60
40
20
0
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]

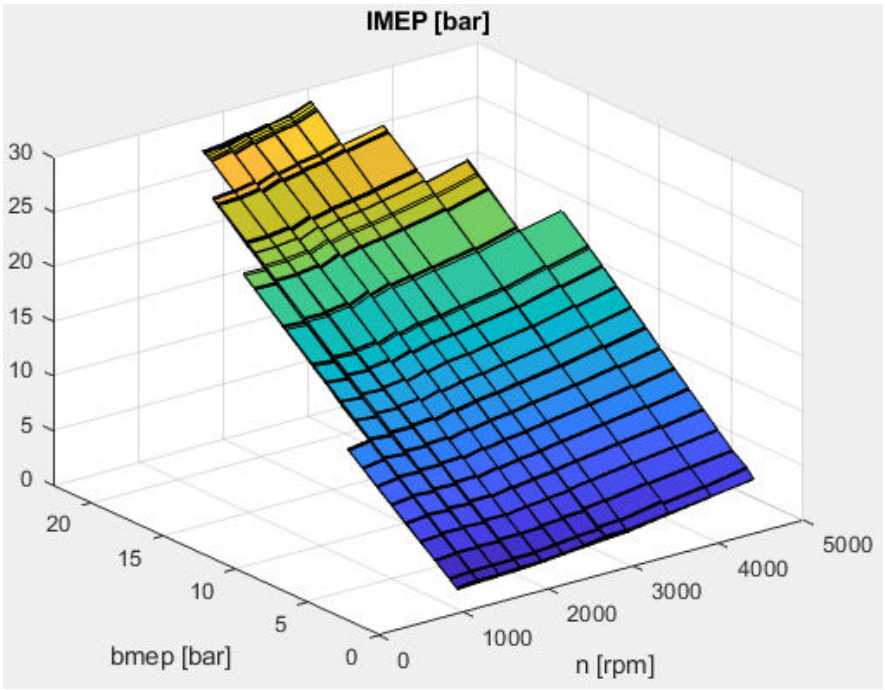

P AMB ECU [mbar]
990
989.5
989
988.5
988
987.5
987
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]

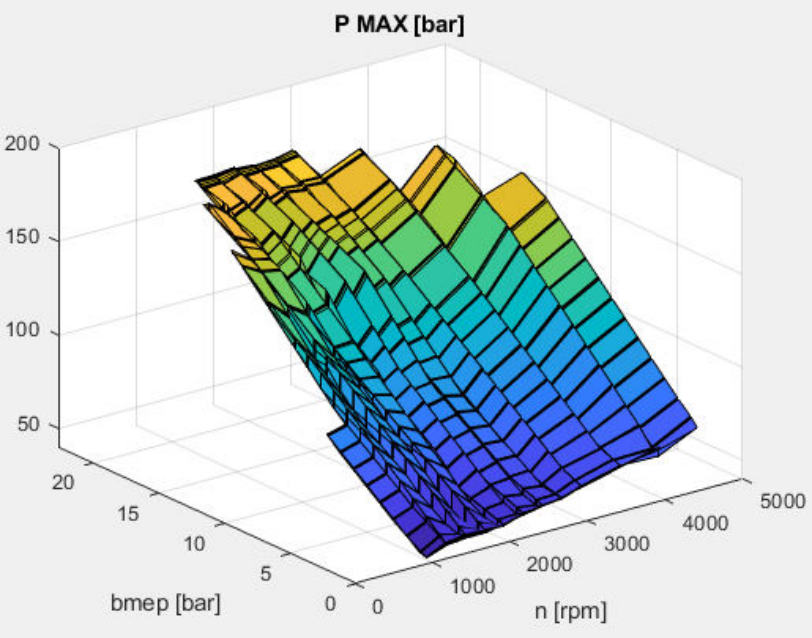

P-TUR-IN [mbar]
2500
2000
1500
1000
500
0
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]

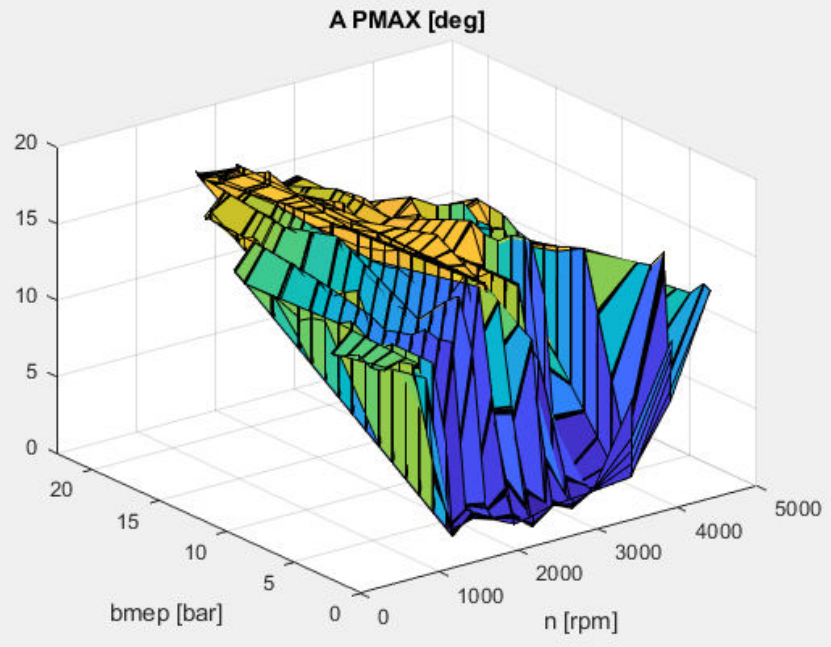

P ING MOT [mbar]
1500
1000
500
0
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]


bsfc [g/kWh]
1400
1200
1000
800
600
400
200
20
15
10
5
0
bmep [bar]
1000
2000
3000
4000
5000
n [rpm]


Q FUEL [mg/c]
80
60
40
20
0
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]


IMEP [bar]
30
25
20
15
10
5
0
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]

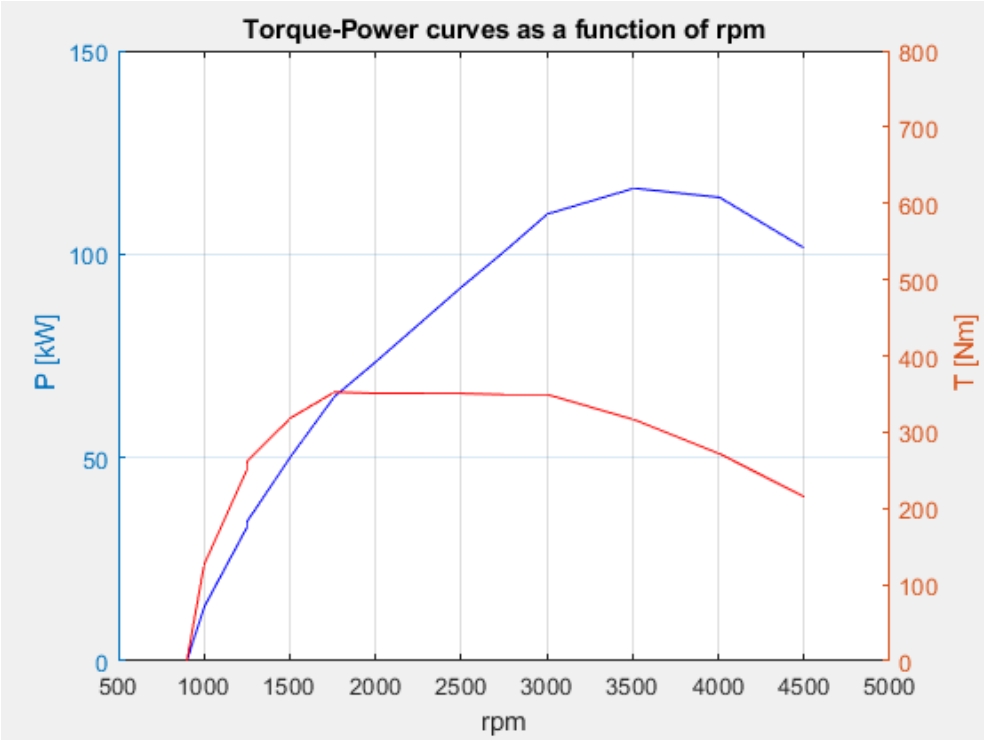

P MAX [bar]
200
150
100
50
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]


A PMAX [deg]
20
15
10
5
0
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]

3.4 Torque-Power curves

Hereby it is shown the script segment and the plot relative to the Torque-Power curves, where the maximum values for torque and power are considered, as a function of the rpm value.

```
297     % TORQUE-POWER CURVES
298
299     [rpm,index]=unique(rpm);
300
301     C=max(B(:,1:end,12));
302     P=max(B(:,1:end,6));
```



3.5 AVL Concerto data acquisition

An additional set of 133 samples, was given using the AVL Concerto Software. From that set of samples, information about the actual instantaneous pressure values, burning rate exploitation, compression and expansion adiabatic exponents and other relevant parameters, had been acquired. That acquisition had been performed using the Matlab tool named "*Catool.exe*" in which, using the script "*TAF_new_yixiN.m*", the specific sample was declared in addition to the sampling interval (set as one thousandth of the specific rpm value considered).
After that, the script converted all the considered cycles parameters into Matlab readable data.

```
%ifile = load_ifile('20081106_TdOff_IMA0_Csam_injmap.003',1);
file = uigetfile('*.*','PLEASE SELECT THE CORRESPONDING IFILE');
ifile = load_ifile(file);
P_raw = ifile.PCYL4.data;
Pman=ifile.PMAN4.data;
```

```
%COSTANTS
    x0 = 0.;
    xn = 720.;
    precs=10^-5;
    itertot= 100;
    A = inputdlg('please input hintv'); %sampling frequency interval
    hintv = str2num(cell2mat(A));
    rh=0.1;  %acquiring step
    nptmax=7201;
    nptciclo_p=7200;
    nintmax=720;
    nclmax=256;
    ncicliv=100;
    nintma2=722;
    nintmap1=721;
```

4.Efficiency, injected fuel and vacuum pressure model evaluation

The main engine efficiencies are calculated, then the useful efficiency, efficiency values
organic, limit efficiency and internal thermo-fluid dynamics efficiency. The filling coefficients are then calculated from the value of bsfc .

4.1 Useful efficiency

It defines the global efficiency of the engine, and so the ratio between useful power and anergy provided to the mechanism. It is computed starting from the useful power and the bsfc values:

$$\eta_u = \frac{P_u}{m_b \cdot n_{cyl} \cdot \frac{n_{mot}}{m} \cdot H_i}$$

$$\eta_{u,q} = \frac{1}{H_i \cdot bsfc \cdot 3.6}$$

Where "*m*" represent the crankshaft revolutions necessary to complete an engine cycle, and so it is equal to 2 for a 4-strokes engine.

4.2 Mechanical efficiency

It compares the useful work available at the crankshaft and the internal work done by the gas on the piston, and so it is representative of the energy losses due to frictions between mechanical components of the engine itself.
It is evaluated starting from the values of bmep and imep:

$$\eta_m = \frac{bmep}{imep}$$

4.3 Limit efficiency

This calculation has been made considering the geometrical compression ratio and the polytropic exponent relative to the expansion phase:

$$\eta_{\lim} = 1 - \frac{1}{\varepsilon^{y_{\exp}-1}}$$

The limit efficiency compares the maximum mechanical energy exploitable by the cycle with the one introduced in every cycle. It is then determined by thermodynamic laws (so it is a function of cycle parameters) and that's why it shows a limit value:

$$\eta_{\lim} = \frac{L_{i,\lim}}{m_b \cdot H_i}$$

4.4 Internal fluid-dynamic efficiency

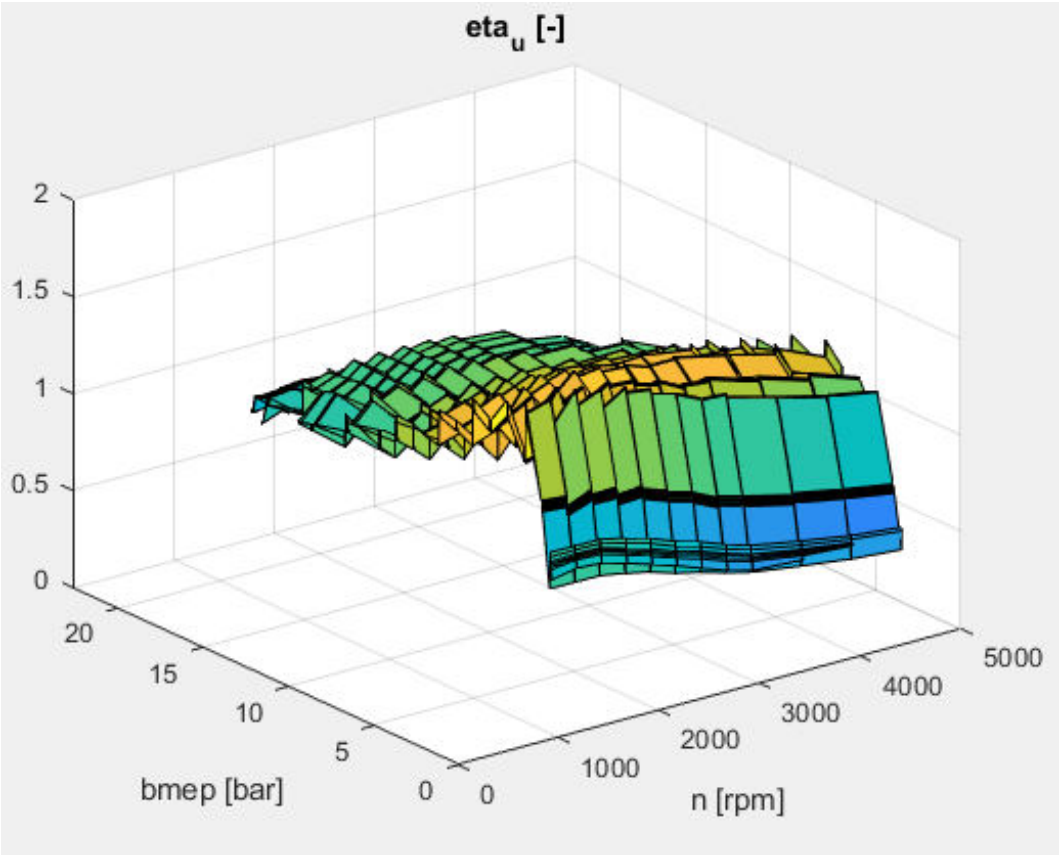
This calculation is performed starting from the values of the previously evaluated efficiencies:

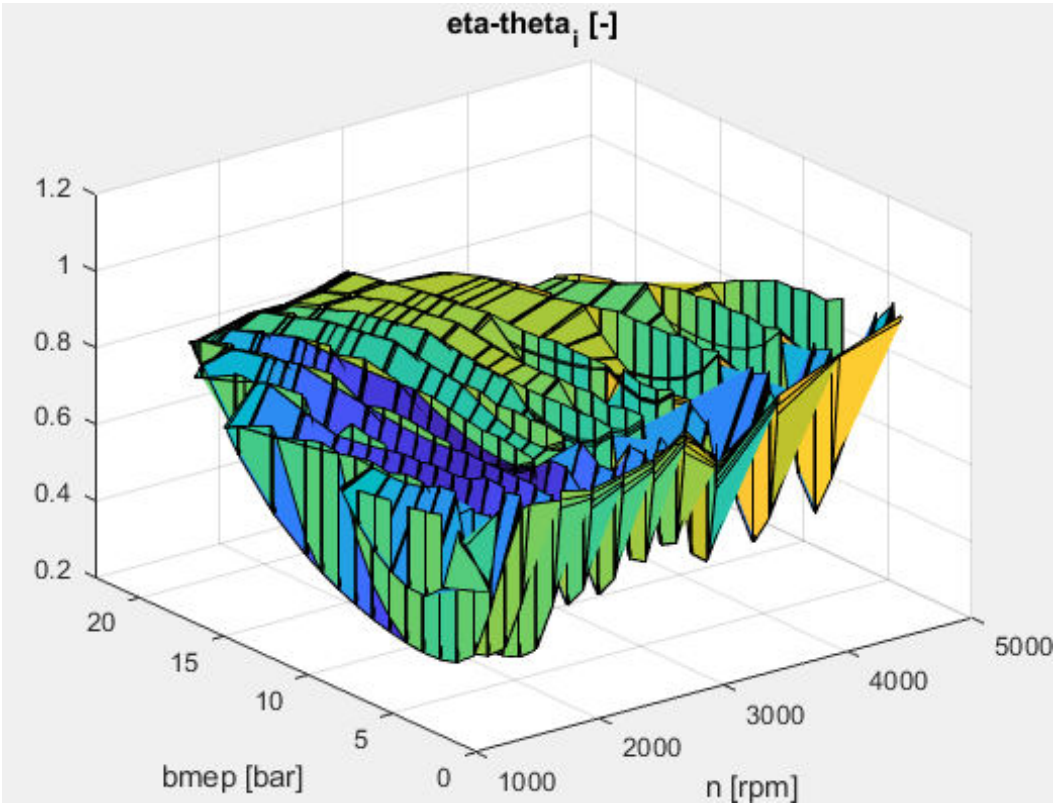
$$\eta_{\theta,i} = \frac{\eta_u}{\eta_m \cdot \eta_{\lim}}$$

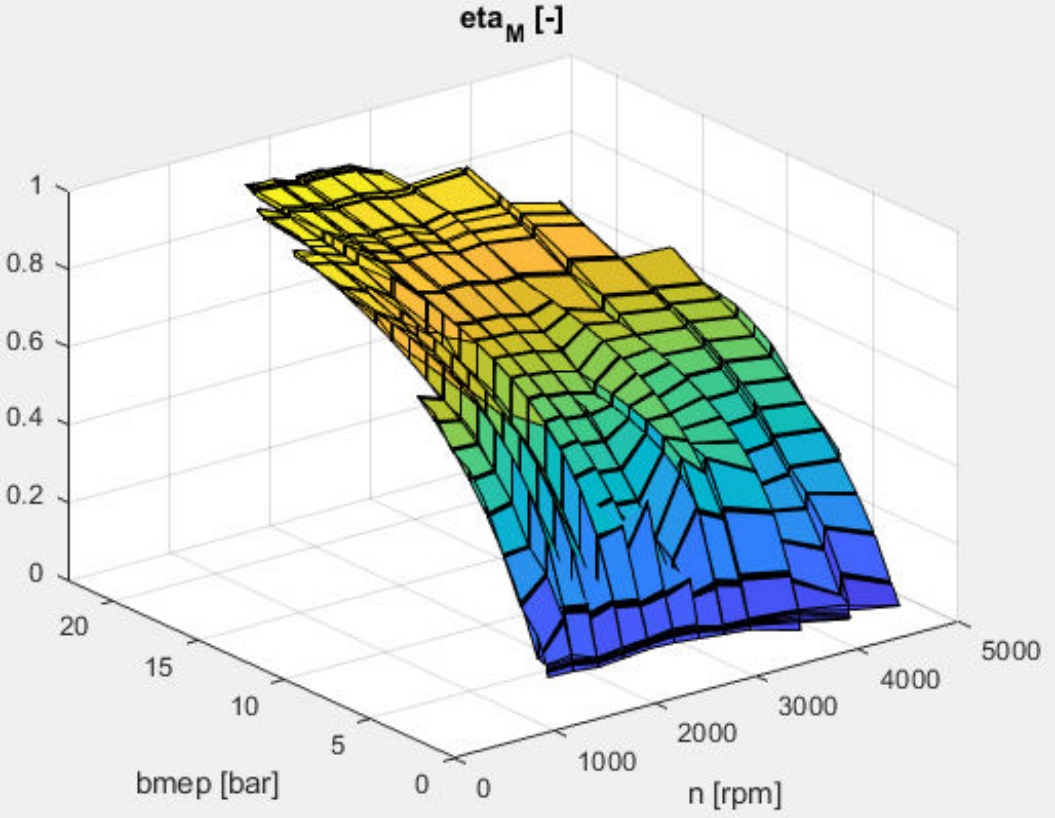
The internal thermo-dynamic efficiency compares the internal work with the total theoretical work that we could obtain without any mechanical losses (limit internal work):

$$\eta_{\vartheta,i} = \frac{L_i}{L_{i,\lim}}$$

```
86      %% EFFICIENCIES EVALUATION
87 -    r_c=16.5;
88 -    y_comp=1.3440;
89 -    y_e=1.3063;
90
91 -     eta_u=10^6*P_u./(4*FB_mact.*rpm/120*H_i);%useful efficiency evaluated from P_u and FB_mact
92
93 -    eta_u_obj=10^6*P_u./(4*QFUEL_MG.*rpm/120*H_i);%useful efficiency evaluated from P_u and QFUEL_MG
94
95 -    eta_u_q=1./(H_i*BSFC_g/1000/3600);%useful efficiency evaluated from declared bsfc
96
97 -    eta_m=bmep./imeph; %mechanical efficiency
98
99 -    eta_lim=1-1./(r_c.^(y_e-1));%limit efficiency evaluated using geometrical compression ratio and y_exp
```


eta$_u$ [-]
2
1.5
1
0.5
0
20
15
10
5
0
bmep [bar]
0
1000
2000
3000
4000
5000
n [rpm]


eta-theta$_i$ [-]
1.2
1
0.8
0.6
0.4
0.2
20
15
10
5
0
bmep [bar]
1000
2000
3000
4000
5000
n [rpm]



4.6 Filling coefficients

It is possible to calculate the filling coefficients in the various points of the dimensioned plane according to the reference conditions or the intake environment (upstream of the air filter) or the intake manifold (downstream of the intercooler). In both cases we proceed by calculating the air density in the reference conditions from which it is
It is possible to calculate the ideal quantity of air introduced in the reference conditions:

$$\rho_{air_{rif}} = \frac{p_{air_{rif}}}{R \cdot T_{air_{rif}}}$$

$$m_{air_{rif}} = \rho_{air_{rif}} \cdot \frac{iV}{4}$$

The quantity of air actually sucked in at each cycle is calculated as a function of the quantity of fuel injected and stoichiometric ratio:

$$m_{air_{eff}} = \alpha_{st} \cdot \lambda \cdot m_b$$

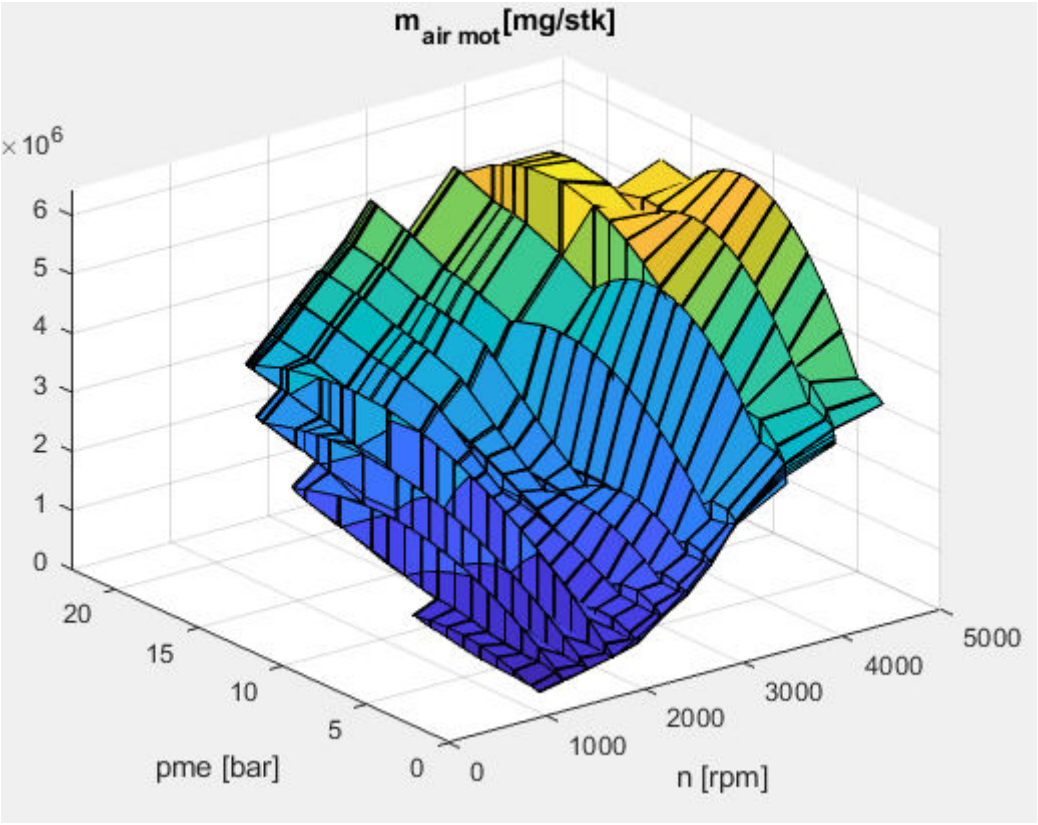
The filling coefficients can then be calculated as the ratio of the actual mass aspirated and the ideal mass calculated in the respective reference conditions:

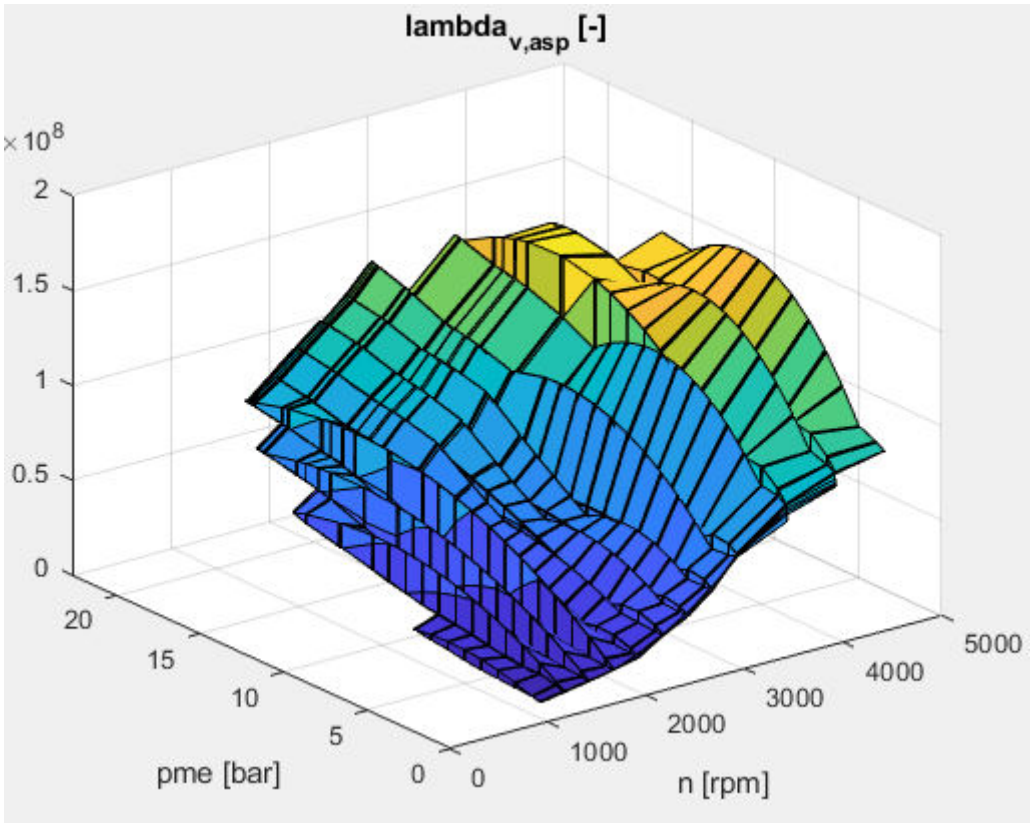
$$\lambda_v = \frac{m_{air_{eff}}}{m_{air_{id}}}$$

The filling coefficients are also calculated considering also the EGR mass reintroduced into the combustion chamber:

$$\lambda_v = \frac{m_{air_{eff}} + m_{EGR}}{m_{air_{id}}}$$

in both cases, both the environmental conditions of intake and the conditions in the intake manifold are considered.




lambda v,asp [-]
×10^8
2
1.5
1
0.5
0
20
15
10
5
0
pme [bar]
0
1000
2000
3000
4000
5000
n [rpm]

5.Definition of Wiebe curves parameters starting from pressure signals analysis

As mentioned before, in addition to the Excel file regarding all the parameters given as a function of Rpm and bmep, a set of 133 test signals acquisitions had been imported from AVL Concerto in Matlab using the Catool.exe instrument. Those experimental data had been processed in order to extrapolate a lot of useful parameters like the metrics of the fraction of mass burned, the heat release rate and so on. Then from the experimental data set we get also a measured curfe for xb (burned fraction). Two different scripts had been then developed to estimate (through the least squares method implemented using the function "*lsqcurve*") the values of the parameters a and m of a calculated Wiebe curve, referring in the first case to the whole xb curve itself, and in the second case only referring to the given metrics parameters.

5.1 Useful parameters estimation

After having declared the sample file, and the sampling frequency interval (as mentioned before), the program implements a single zone combustion model, and it requests to give as fixed parameters the Start Of Injection of the first pilot injection (in crank angle degrees before TDC), the quantity of fuel injected during this first pilot injection, and the total amount of fuel injected during a cycle, all those quantities are readable from the ECU and listed in the given sample test.
After that, design parameters, like bore, stoke, number of cylinder, exhaust valve opening and closure and so on, are acquired from the excel file "*A20DTR_Engine_Data-xlsx*" using the function "*xlsread*".
All those given fixed parameters are used to estimate hypothetical values for the combustion event main parameters, all evaluated as a function of reference crank angle degrees and of the acquiring step "*rh*".

```
table_engine = xlsread('A20DTR_Engine_Data.xlsx');
%% A20DTR engine data8

stroke =ifile.stroke;                    % in mm
bore =ifile.bore;                        % in mm
eps = ifile.compression_ratio;           % compression ratio
conrod = ifile.conrod_length;            % in mm
Ncyl = ifile.number_of_cylinders;        % number of cylinders

EVC = table_engine(6);          % exhaust valve closing angle after TDC
EVO = table_engine(7);          % exhaust valve open after TDC
Tw = table_engine(8);           % temperature of clyinder body in K

displ = pi * bore^2 /4 * stroke * 10^-9; %volume unitario

SOI=360-SOI_PIL1;%single zone approach starting point, should be equal to the earliest SOI
EOC=450;   %single zone approach ending point


teta_PMI_rad= pi-asin(0/(conrod-(stroke/2)));
teta_PMS_rad= asin(0/(-conrod-(stroke/2)));
sfas_deg_PMI= teta_PMI_rad*180/(pi-180);
sfas_deg_PMS= teta_PMS_rad*180/pi;
TDC_ASP =0 +sfas_deg_PMS;
TDC_ESP =360 +sfas_deg_PMS;
```

```
ip_fine_pol=    330/rh+ 1;
ip_fine_comb=   450/rh+1;
ip_ch_asp    = 198/rh+ 1;
ip_ap_sca=   (TDC_ESP+EVO)/rh+1;
ip_fine_elab= EOC/rh+1;
```

5.2 Metrics and HRR evaluation from acquired data

The single zone approach in then used, inside a *"for"* loop that comprises all the engine cycles carried out for the considered sample, to compute the value of Qnet_dot (the flow of heat exchanged), that in turns is used to calcuate a value for the end of combustion. This value is now function of the energy exchange process and not only an approximation carried out only using the crank angle degrees values as it was done previously when defining its hypothetical value.
Then the cumulative quantity of Qnet is computed through an integration of Qnet_dot carried out using the function "*cumtrapz*".
Qch is suddenly defined as the difference of values of Qnet along all the engine cycle for the sample considered, then it is used to define the matrix Xr of the fuel burned instantaneously, that will be the reference for the estimation of all the desired metrics of the burned fraction along the cycle. This estimation is done comparing the instantaneous value of Xr with a fixed value (the percentage of burned mass searched), and using a simple "*if*" cycle, it is carried out the index corresponding to the instant at which Xr becomes greater than that fixed value. That specific index point will identify the point in which we will find the requested fraction of mass burned inside the cylinder. This index value will be later interpolated with the crank angle degrees values vector, in order to have a reference value of the corresponding metric along the engine cycle only function of the crank angle position.

The script also carries out the values of the maximum pressure, Imep, and starting from the hypotheses made for the combustion event, also the exponents for the two polytropic transformations of compression and expansion.
In the end, using the hypothetical point for the end of combustion, the Heat Release Rate is evaluated as the ratio between Qnet_dot and Qnet, as well as its maximum value and the angle at which it takes place.

```
for j=1:ncicliv
 P=ZTAF(:,j);
dP=[0;diff(P)];
dV=[0;diff(V)];


q=[qPill qTot_fromECU-qPill];   %mm3 fuel quantity for each injection shot

tmp1(:,j)=(gamma/(gamma-1).*P*1e5.*dV+1/(gamma-1).*V.*dP*1e5)/CAstep;   %J/deg


[~,pointerSOI]=min(abs(CA-SOI));
[~,pointerEOC]=min(abs(CA-EOC));

Qnetdot(pointerSOI:pointerEOC,j)=tmp1(pointerSOI:pointerEOC,j);   %J/deg


%calcolo EOC%%%%%%%%%%%%

EOC_calc=calc_fine_comb(P,V,Qnetdot(:,j));

  EOC_m(j)=EOC_calc- TDC_ESP;
```

```
%% Qch by shifting and scaling
Qnet(:,j)=cumtrapz(CA,Qnetdot(:,j));

[~,pointermin]=min(Qnet(:,j));
[~,pointermax]=max(Qnet(:,j));
scalefac=sum(q)*ERHO/(Qnet(pointermax,j)-Qnet(pointermin,j));
Qch=zeros(length(CA),j);
Qch(pointermin:pointermax,j)=(Qnet(pointermin:pointermax,j)-Qnet(pointermin,j))*scalefac;
Qch(pointermax+1:end,j)=Qch(pointermax,j);
Xr(:,j)=Qch(:,j)/sum(q)/ERHO;
```

```
%MFB50
[~,pointer]=min(abs(Xr(:,j)-0.5));
if Xr(pointer,j)>0.5
    ind=[pointer-1 pointer];
elseif Xr(pointer,j)<0.5
    ind=[pointer pointer+1];
end
%MFB10
[~,pointer10]=min(abs(Xr(:,j)-0.1));
if Xr(pointer10,j)>0.1
    ind10=[pointer10-1 pointer10];
elseif Xr(pointer10,j)<0.1
    ind10=[pointer10 pointer10+1];
end
%MFB5
[~,pointer5]=min(abs(Xr(:,j)-0.05));
if Xr(pointer5,j)>0.05
    ind5=[pointer5-1 pointer5];
elseif Xr(pointer5,j)<0.05
    ind5=[pointer5 pointer5+1];
end
%MFB90
[~,pointer90]=min(abs(Xr(:,j)-0.9));
if Xr(pointer90,j)>0.9
    ind90=[pointer90-1 pointer90];
elseif Xr(pointer90,j)<0.9
    ind90=[pointer90 pointer90+1];
```

```
%%%%%%%%%%%% valutazione parametri statistici%%%%%%%%%%%%%%%%%%%%%%
MFB50(j)=interp1(Xr(ind,j),CA(ind),0.5)-360;
MFB5(j)=interp1(Xr(ind,j),CA(ind5),0.5)-360;
MFB10(j)=interp1(Xr(ind,j),CA(ind10),0.5)-360;
MFB90(j)=interp1(Xr(ind,j),CA(ind90),0.5)-360;
MFB1(j)=interp1(Xr(ind,j),CA(ind1),0.5)-360;
```

```
Pmax(j) = max(P);   %PRESSIONE MASSIMA
[~,pointerPmax]=max(P);
Apmax(j)=CA(pointerPmax)-360; %angolo pressione max
dp_dteta_max(j) =max(diff(P)/CAstep);
[~,pointerdp_dteta_max]=max(diff(P)/CAstep);
Adp_dteta_max(j)=CA(pointerdp_dteta_max)-360;

%%%%%%%%% IMEP %%%%%
IMEP(j)=sum(P(1:length(P)).*dV)/displ;

%%%%%%SOC%%%%%%%

m_comp= log(P(ip_ch_asp)/P(ip_fine_pol))/log(V(ip_fine_pol)/V(ip_ch_asp));
m_esp= log(P(pointerEOC)/P(ip_ap_sca))/log(V(ip_ap_sca)/V(pointerEOC));
```

```
415
416 -     ip_EOC= ((EOC_m(j)+TDC_ESP-TDC_ASP)*10)+1;
417
418 -   hrr(:,j)=Qnetdot(:,j)/Qnet(int64(ip_EOC),j);
419 -   hrr_max(j) = max(hrr(:,j));    %HRR MAX
420 -   [~,pointerHRRmax]=max(hrr(:,j));
421 -   AHRR_MAX(j)=CA(pointerHRRmax)-360; %angolo pressione max
```

5.3 Wiebe curve parameters estimation as a function of x_b

The main script TAF_new_yixiN had also implemented a subscript for the function "*stat2*" that receives as inputs all the values from the ECU such as the SOI of the pilot injection, the quantity of injected fuel and so on, and in turns it provides the ensemble values for Qnet, hrr, and xb.
In particular, as stated before, the ensemble curve of the burned fraction of mass inside the cylinder (xb_ens) had been set as a reference to model a MultiWiebe curve, only function of the parameters a and m, that could be carried out only giving as input values the quantities of fuel injected in the pilot and main injections (readable from ECU tables for each sample, but they can also be given as fixed data by the user) and the hypothetical values of combustion start and combustion duration estimated in the main script.
Each one of the four (3 pilot injections plus one MAIN injection) curves corresponds to the evolution of the fuel injected per each injection event, and then those curves are summed up to give a more precise representation of the evolution of the combustion process inside the cylinder.
At first, in the main script, all the Start Of Injection (SOI) values are acquired from the excel file, then from those values, the corresponding values of start of combustion (called theta_0) of each portion of fuel injected are estimated adding to the SOI value the Nozzle Opening Delay (NOD), and the Ignition Delay (ID) value, calculated for each injection event using the formula:

$$\tau_{id} = A \cdot p^{-n} \cdot \exp\left(\frac{E_A}{R \cdot T}\right)$$

with A and n fixed coefficients depending on the type of the engine and E_a the activation energy of the fuel considered, computed using:

$$E_A = \frac{618{,}840}{CN + 25}$$

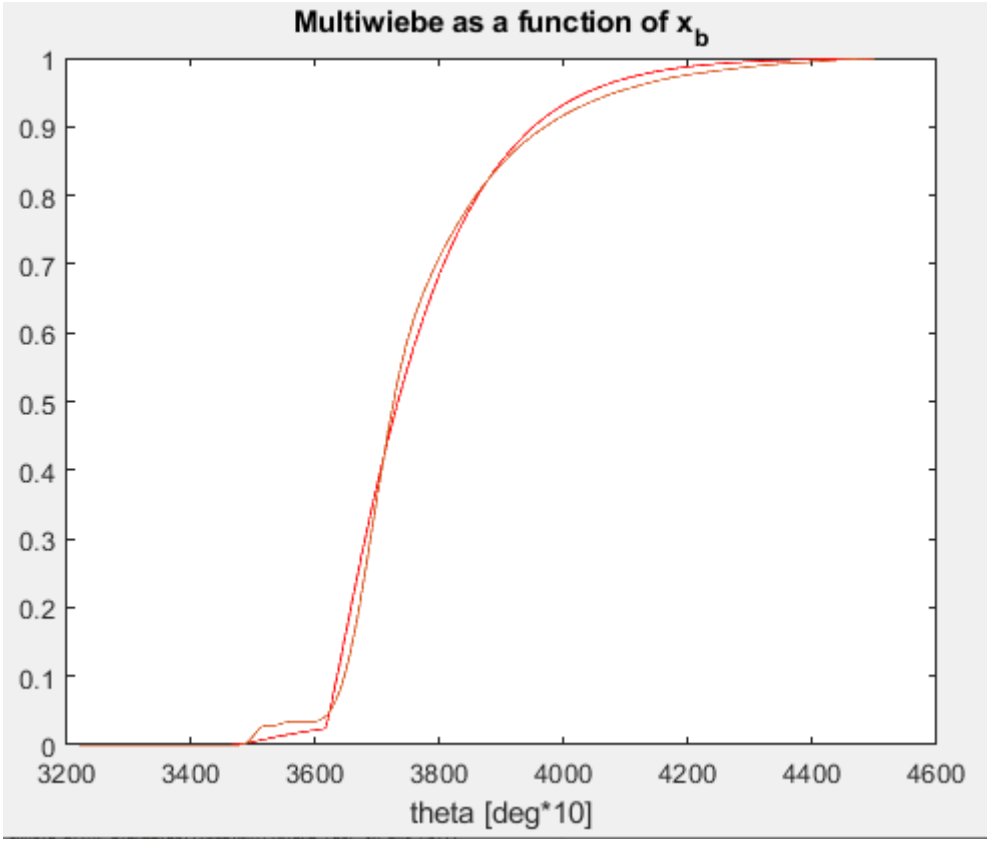
Where CN is the Cetane number of the fuel. The temperature is evaluated using the perfect gas law and the in-cylinder mass at the corresponding instant. So in the end the ID is converted from microseconds to crank angle degrees.

```
514      %%Calculation of ID for each teta0
515
516 -    NOD=250; %[us]
517
518 -    IVC=round(IVC_deg);
519 -    A=40.44;
520 -    n=1.19;
521 -    CN=45;
522 -    Ea=618.840/(CN+25);
523 -    R=8.13;
524 -    N_ID=n_rec;
525
526 -    p3=P(round(SOI_Pil3)*10)*0.986923;
527 -    m_tot_cyl3=(1.2*p_rel_mot_rec*V(IVC*10)/(287*T_coll_rec)/10);
528 -    T3=(P(round(SOI_Pil3)*10)*V(round(SOI_Pil3)*10))/(10*287*m_tot_cyl3);
529 -    ID_ms3=A*p3^-n*exp(Ea/(R*T3));
530 -    ID_deg3=ID_ms3*0.006*N_ID;
531
532 -    teta0_Pil3=SOI_Pil3+((NOD*10^-6)*(line(3)*360/60))+ID_deg3;
533 -    teta0_Pil3=double(teta0_Pil3);
```

Then after having set the initial condition values for all the parameters of the model defined, the function "*lsqcurvefit*" is used to find (using the least squares method) the values of the parameters a and m that best fits with the objective curve xb_ens imported from the main script.
After that, those parameters are used in each of the three Wiebe curves corresponding to each injection event, that compose the final MultiWiebe curve.
The combustion efficiency CE value is assumed equal to 1.
At the end of the script, the use of the matlab function "*rmse"* had beed performed to computed the mean square error from the calculated Multiwiebe curve and the experimental one.

```
function [a,m,Multiwiebe,delta_teta_0,mean_square_error_Multiwiebe,mean_square_error_Wiebe]= Wiebe02(jj,xb_ens,ip_comb,ip_fine_comb,ip_EOC

delta_teta_0=ip_EOC-ip_comb;
t=[round(SOI_PILfirst-30)*10:1:ip_fine_comb]';
xb_ens_cal=xb_ens(round(SOI_PILfirst-30)*10:ip_fine_comb);

f=@(B,t) 1*(((qPil3/m_b).*(1-exp((-B(1).*((t-(teta0_Pil3*10))./(delta_teta_0)).^(B(2)+1))))).*(t>teta0_Pil3*10)+... %pilot injection 1
    ((qPil2/m_b).*(1-exp((-B(1).*((t-(teta0_Pil2*10))./(delta_teta_0)).^(B(2)+1))))).*(t>teta0_Pil2*10)+...%pilot injection 2
    ((qPil1/m_b).*(1-exp((-B(1).*((t-(teta0_Pil1*10))./(delta_teta_0)).^(B(2)+1))))).*(t>teta0_Pil1*10)+...%pilot injection 3
    ((q_Main/m_b).*(1-exp((-B(1).*((t-(teta0_MAIN*10))./(delta_teta_0)).^(B(2)+1))))).*(t>teta0_MAIN*10)); %Main

B0=[0 0];
[B]=lsqcurvefit(f,B0,t,xb_ens_cal);

a=real(B(1));
m=real(B(2));
```

```
CE=1;
Multiwiebe=CE.*(((qPil3/m_b).*(1-exp((-a.*((t-((teta0_Pil3*10)))./(delta_teta_0)).^(m+1))))).*(t>teta0_Pil3*10)+... %pilot injection 1
     ((qPil2/m_b).*(1-exp((-a.*((t-((teta0_Pil2*10)))./(delta_teta_0)).^(m+1))))).*(t>teta0_Pil2*10)+...%pilot injection 2
     ((qPil1/m_b).*(1-exp((-a.*((t-((teta0_Pil1*10)))./(delta_teta_0)).^(m+1))))).*(t>teta0_Pil1*10)+...%pilot injection 3
     ((q_Main/m_b).*(1-exp((-a.*((t-((teta0_MAIN*10)))./(delta_teta_0)).^(m+1))))).*(t>teta0_MAIN*10)); %Main

multiwiebe_test=1.*(1-exp((-a.*(((t-teta0_first*10))./(delta_teta_0)).^(m+1)))).*(t>teta0_first*10);

 figure
 plot(t,Multiwiebe,'r')
 title('Multiwiebe as a function of x_b')
 xlabel('theta [deg*10]')
 hold on
 plot(t,xb_ens_cal)


mean_square_error_Multiwiebe=real(rmse(Multiwiebe,xb_ens_cal))
mean_square_error_Wiebe=real(rmse(multiwiebe_test,xb_ens_cal))

end
```

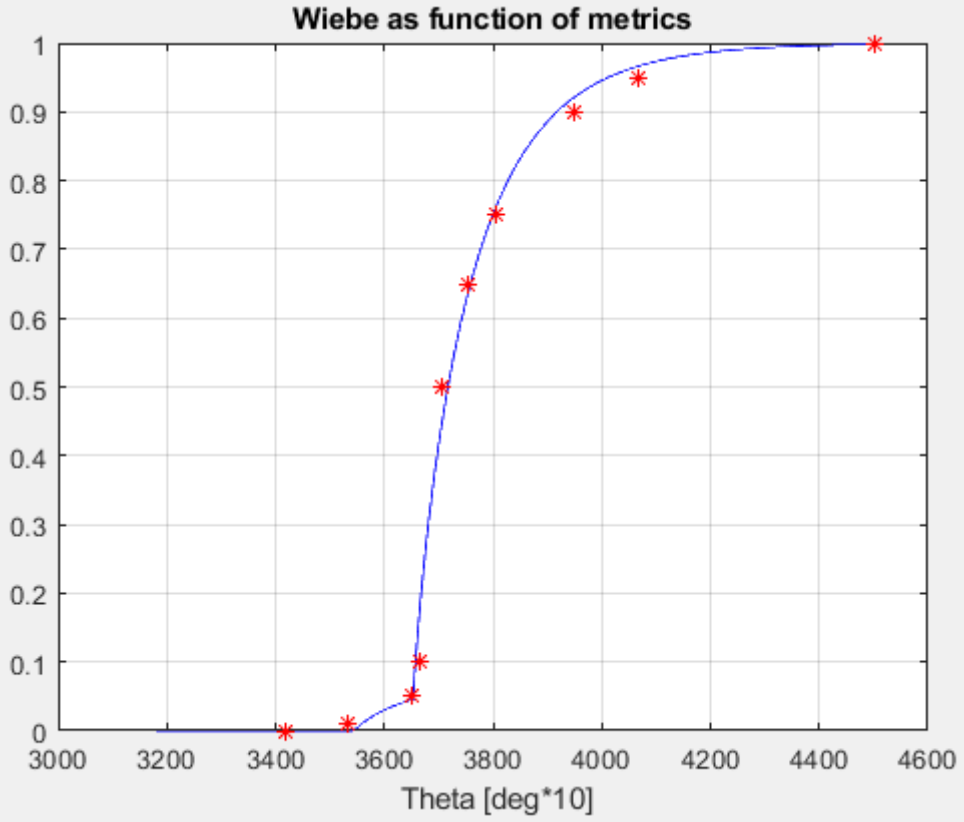


5.4 Wiebe curve parameters estimation as a function of the given metrics

The same structure had been applied to another subscript that aims to estimate again the values of the two parameters a and m, but in this case referring only to the given specific metrics values found in the main script.
Also in this case the least squares method is implemented, but now the comparison is made only between the computed Multiwiebe curve and the values of the experiental one at some fixed points (metrics) of interest.
The reference percentage values of all the metrics are imported, as well as the reference pointer values.
The vector containing all the initial conditions for the parameters is set and then the "*lsqcurvefit*" function is used.

```
function [am,mm,delta_teta_0m,Wiebe_metrics,Multiwiebe_metrics,mean_square_error_Multiwiebe_metrics,mean_square_error_Wiebe_metrics] = Wiebe_metr

mfb=[0 0.01 0.05 0.1 0.5 0.65 0.75 0.9 0.95 1];
t=[ip_comb-100,pointer1,pointer5,pointer10,pointer,pointer65,pointer75,pointer90,pointer95,ip_fine_comb];
xb_ens_cal=xb_ens(round(SOI_PILfirst-30)*10:ip_fine_comb);
delta_teta_0m=ip_EOC-ip_comb;

f=@(B,t) 1*(((qPil3/m_b).*(1-exp((-B(1).*((t-(teta0_Pil3*10))./(delta_teta_0m)).^(B(2)+1))))).*(t>teta0_Pil3*10)+... %pilot injection 1
    ((qPil2/m_b).*(1-exp((-B(1).*((t-(teta0_Pil2*10))./(delta_teta_0m)).^(B(2)+1))))).*(t>teta0_Pil2*10)+...%pilot injection 2
    ((qPill/m_b).*(1-exp((-B(1).*((t-(teta0_Pill*10))./(delta_teta_0m)).^(B(2)+1))))).*(t>teta0_Pill*10)+...%pilot injection 3
    ((q_Main/m_b).*(1-exp((-B(1).*((t-(teta0_MAIN*10))./(delta_teta_0m)).^(B(2)+1))))).*(t>teta0_MAIN*10)); %Main


B0=[0 0];
[B]=lsqcurvefit(f,B0,t,mfb);

am=real(B(1));
mm=real(B(2));
```

```
x=[round(SOI_PILfirst-30)*10:1:ip_fine_comb]';
Wiebe_metrics= (1-exp((-am.*((x-(teta0_first*10))./delta_teta_0m).^(mm+1)))).*(x>teta0_first*10);

CE=1;
Multiwiebe_metrics=CE.*(((qPil3/m_b).*(1-exp((-am.*((x-((teta0_Pil3*10)))./(delta_teta_0m)).^(mm+1))))).*(x>teta0_Pil3*10)+... %pilot injection 1
     ((qPil2/m_b).*(1-exp((-am.*((x-((teta0_Pil2*10)))./(delta_teta_0m)).^(mm+1))))).*(x>teta0_Pil2*10)+...%pilot injection 2
     ((qPill/m_b).*(1-exp((-am.*((x-((teta0_Pill*10)))./(delta_teta_0m)).^(mm+1))))).*(x>teta0_Pill*10)+...%pilot injection 3
     ((q_Main/m_b).*(1-exp((-am.*((x-((teta0_MAIN*10)))./(delta_teta_0m)).^(mm+1))))).*(x>teta0_MAIN*10)); %Main


figure
plot(x,Multiwiebe_metrics,'b')
title('Wiebe as function of metrics')
xlabel(' Theta [deg*10]')
axis([3000 4600 0 1])
grid on
hold on
```


Wiebe as function of metrics
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
3000
3200
3400
3600
3800
4000
4200
4400
4600
Theta [deg*10]

6.Pressure cycle reconstruction

To evaluate the pressure trend in the chamber it is necessary to apply the first principle of thermodynamics at the volume defined by the combustion chamber in the compression phase, combustion and expansion (in which there is a volume that can be considered closed, neglecting the gas leaks between the valve seats and at the piston-barrel interface).
At the same time, the combustion law defined by the Wiebe function must be considered in order to define the contribution of heat introduced by the fuel.

6.1 Description of the Matlab code

The code relating to the evaluation of the pressure in the chamber consists of the main code ("Main") and by two functions it calls (“Wiebe” and “Pressure_Woschni”).
The Main code reads through “xlsread” each row of the Excel file and extracts them variables useful for evaluating the pressure cycle (mfb, IVC, pcoll, Tcoll,…).
Then it first calls the “Wiebe” function which receives the values of mfb and as output returns the parameters of the Wiebe function that best fit the given mfb values and then the “Pressure_Woschni1” function which actually implements the models useful for evaluating the pressure trend in the chamber and to calculate the quantities to be compared with the values experimental provided (eg. pmax, pR, max, IMEPH…).

6.2 Combustion chamber surface evaluation

It is useful to calculate the lateral surfaces of the combustion chamber as a function of the angle of rotation of the crank "θ" later useful for calculating the heat exchange towards the outside.
In particular, the terms relating to plunger, head and barrel are considered:

$$_{piston} = \pi \cdot \frac{d^2}{4}$$

$$S_{head} = k_t \cdot \pi \cdot \frac{d^2}{4}$$

The term k_t= 1.1 takes into account the non-planarity of the head in the case of ignition engines commanded. For the calculation of the surface of the barrel as a function of the crank angle the axial displacement of the piston with respect to the TDC is considered:

$$x_\vartheta = \left[r \cdot \left(1 - \cos(\theta) + \frac{1}{\lambda} \cdot (1 - \cos(\beta)) \right) \right]$$

where the connecting rod angle β is computed using:

$$\beta = \pm\sqrt{\left(1 - \Lambda^2 \cdot \sin^2(\theta)\right)}$$

$$\Lambda = \frac{r}{l}$$

$$S_{cyl(\vartheta)} = \frac{\pi}{4} \cdot d^2 \cdot x_{(\vartheta)}$$

6.3 Evaluation of the chamber volume

The calculation of the volume of the chamber as a function of the crank angle is calculated starting from the bore, stroke and geometric compression ratio values.
The total unit volume is given by the sum of dead volume and unit displacement:

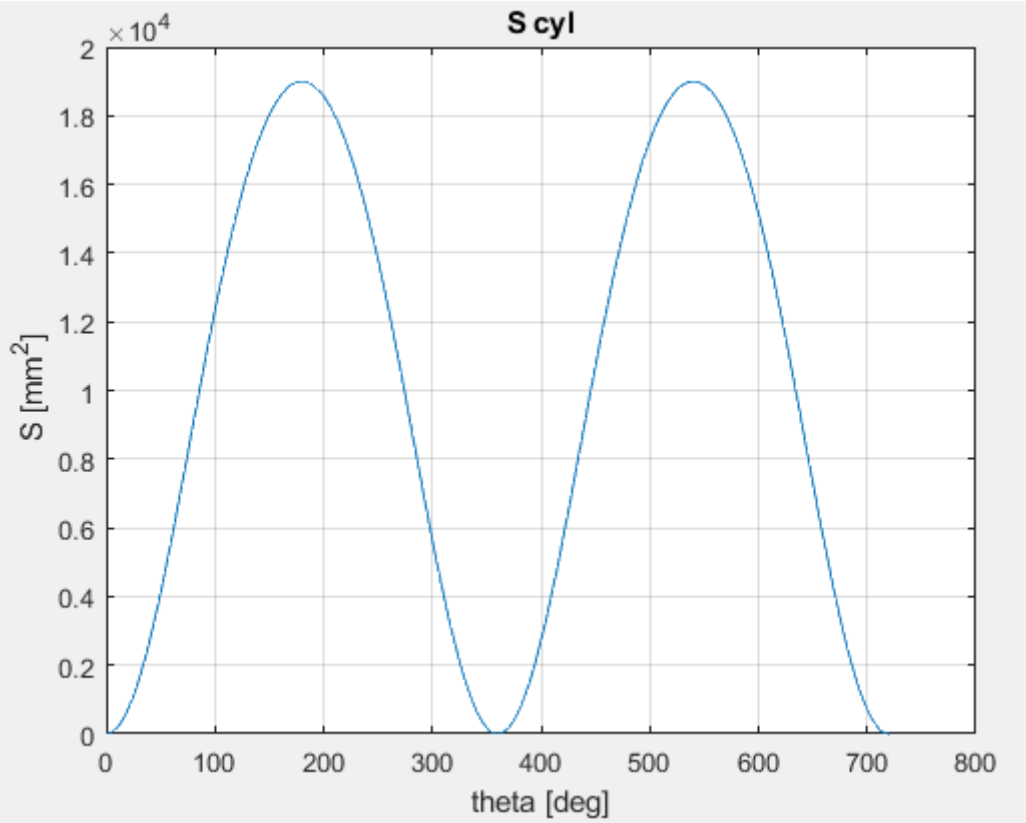
$$V_{tot,u} = V_{m,u} + V_u$$

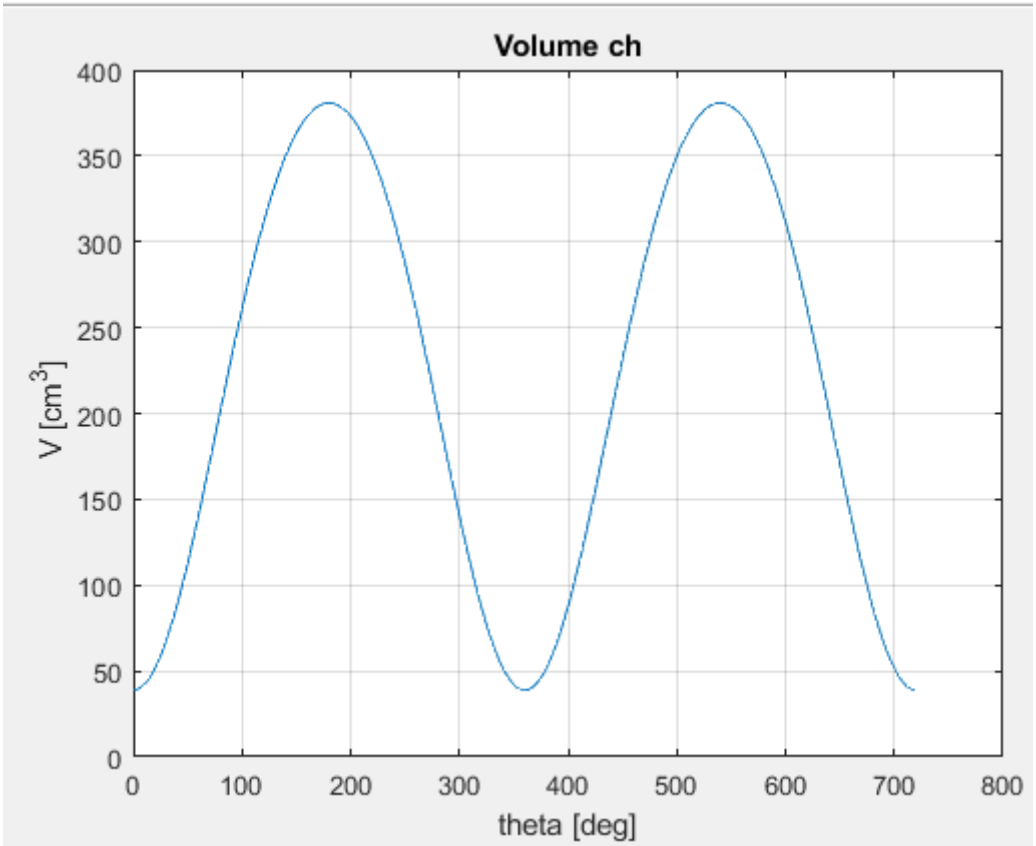
$$V_u = \frac{\pi}{4} \cdot d^2 \cdot c$$

$$V_{m,u} = \frac{V_u}{\varepsilon - 1}$$

The chamber volume is given as a function of the axial piston displacement and so as a function of θ:

$$V_{(\vartheta)} = V_m + S_{piston} \cdot x_{(\vartheta)}$$



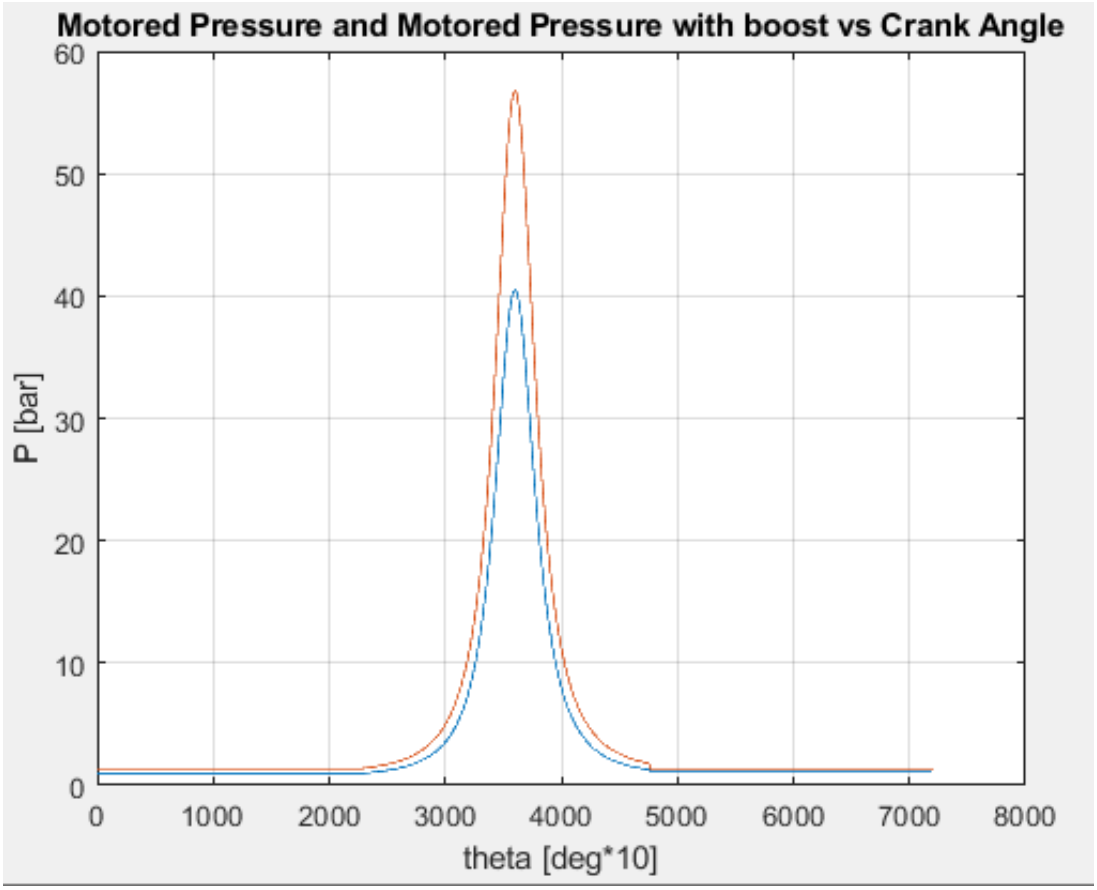


6.4 Motored cycle pressure evaluation

The motored cycle pressure computation is done under the hypothesis of polytropic compression and expansion, with relative exponents named y_{comp} and y_{exp}.
The motored pressure cycle and the motored pressure cycle with overboost, have been modeled. The difference between the two of them is that in the first one we only consider the pressure evolution inside the cylinder without combustion, with the incoming air at ambient conditions, while in the second one we also consider the contribution of the air compressor to the pressure of the incoming air charge.
The atmospheric pressure and the intake valve closure instant IVC had been taken as reference points, separating between the compression and the expansion phase.

$$\begin{cases} p_{motored,i} = p_{atm} \cdot \left(\frac{V_{IVC}}{V_i} \right)^{y_{comp}} & if -> i < i_{TDC}; \end{cases}$$

$$\begin{cases} p_{motored,i} = p_{TDC} \cdot \left(\frac{V_{TDC}}{V_i} \right)^{y_{exp}} & if -> i > i_{TDC}; \end{cases}$$



6.5 Acquisition of Wiebe function parameters

The exponents a and m of the Multi-Wiebe function, evaluated using the scripts presented before (the model works having as input either “a” and “m” from the “Wiebe_metrics” script or from the “Wiebe” one), are imported and used to model a sample Wiebe curve that will give all the instantaneous values of the fraction of mass burned, the reconstructed pressure will depend on.

```
% Wiebe functions

theta=[0.1:0.1:720];

multiwiebe=zeros;

for(i=(round(SOI_PILfirst-30)*10)+100:1:(round(SOI_PILfirst-30)*10)+100+round(size(Multiwiebe,[1]))-1)
    multiwiebe(i)=Multiwiebe(i-((round(SOI_PILfirst-30)*10)+100-1));
end
multiwiebe((round(SOI_PILfirst-30)*10)+100+round(size(Multiwiebe,[1])))=1;
for(i=(round(SOI_PILfirst-30)*10)+100+round(size(Multiwiebe,[1]))+1:1:7200)
    multiwiebe(i)=1;
end
```

6.6 Evaluation of the pressure inside the combustion chamber

Having defined a Wiebe function that models the combustion process, for every sampling instant the pressure inside the cylinder is computed using:

$$p_{i+1} = p_{i-1} + \frac{1}{V_i} \cdot \left[(k_i - 1) \cdot (dQ_{ch} + dQ_{ht}) - k_i \cdot p_i \cdot (V_{i+1} - V_{i-1}) \right]$$

Where the heat exchange parameter dQ_{ht} has bees computed using the Woschni heat exchange model that defines the thermal power exchanged with the cylinder walls using the formula:

$$S_{head} \cdot (T_i - T_{head}) + S_{piston} \cdot (T_i - T_{piston}) + S_{cylinder,i} \cdot (T_i - T_{cylinder}) \left[W \right]$$

With the global heat exchange parameter is defined as:

$$h_{Wh,i} = 3.26 \cdot d^{-0.26} \cdot p_i^{0.8} \cdot w_i^{0.8} \cdot T_i^{-0.55} \left[\frac{W}{m^2 \cdot K} \right]$$

In which w_i is the mean local speed of the gases estimated using the following formula:

$$w_i = C_1 \cdot v_m + C_2 \cdot \frac{V_i \cdot T_r}{p_r \cdot V_r} \cdot (p_i - p_{motored}) \left[\frac{m}{s} \right]$$

Considering as reference values of T_r, V_r, p_r the corresponding values at IVC instant, and for C_1 and C_2 respectively, 2.28 and $3.24*10^3$ during the combustion and expansion phase, and 2.28 and 0 during the compression phase.
Temperature values are estimated as:

T_{head}=550K;
T_{piston}=450K;
$T_{cylinder}$=430K;

V_m is the mean piston $\left(\frac{c}{2} \cdot n \right)$ speed while the temperature at the i-th instant is obtained using the perfect gas law:

$$T_i = \frac{p_i \cdot V_i}{m_{tot} \cdot R} \left[K \right]$$

The heat exchanged through the walls at every instant of the sampling is then given by:

$$dQ_{ht,i} = \frac{\dot{Q}_{ht,i}}{\omega} \cdot d\vartheta = \frac{\dot{Q}_{ht,i}}{\omega} \cdot (\vartheta_{i+1} - \vartheta_{i-1}) \left[J \right]$$

Being $\dot{Q}_{ht,i} = \frac{dQ_{ht,i}}{dt}$ and $\omega = \frac{d\vartheta}{dt}$

The chemical term of the heat release is given by:

$$dQ_{ch} = H_i \cdot dm_b$$

But since it is:

$$dm_b = m_b \cdot dx_b$$

we get:

$$dQ_{ch} = m_b \cdot H_i \cdot dx_b$$

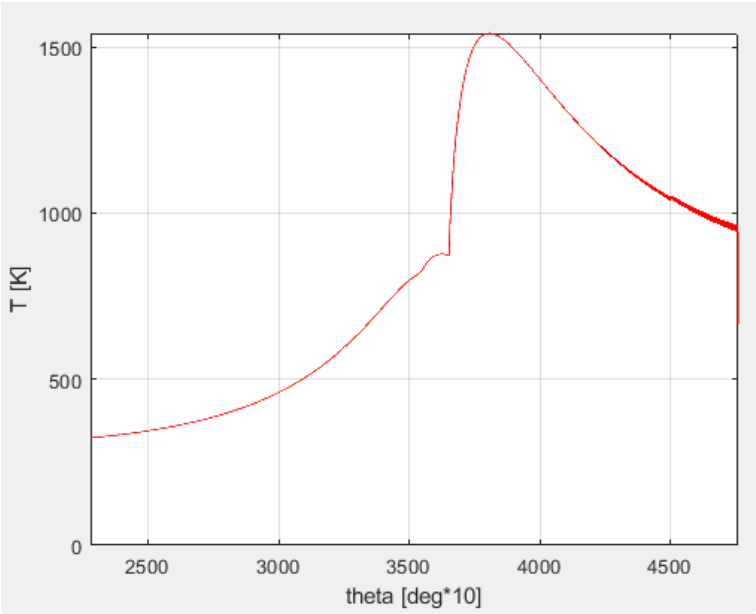
For the discretized system the pressure equation finally becomes:

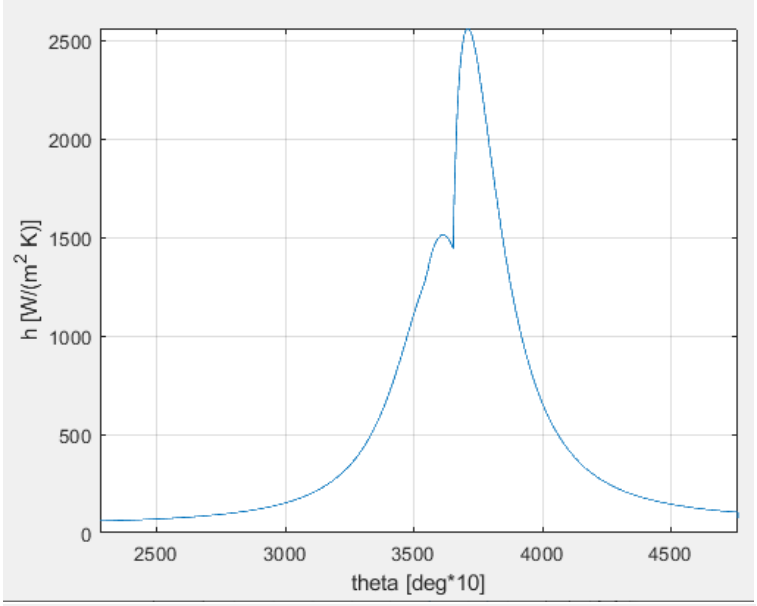
$$p_{i+1} = p_{i-1} + \frac{1}{V_i} \cdot \left[(k_i - 1) \cdot \left[(x_{b,i+1} - x_{b,i-1}) \cdot m_b \cdot H_i - k_i \cdot p_i \cdot (V_{i+1} - V_{i-1}) + \frac{\dot{Q}}{\omega} \cdot (\vartheta_{i+1} - \vartheta_{i-1}) \right] \right]$$

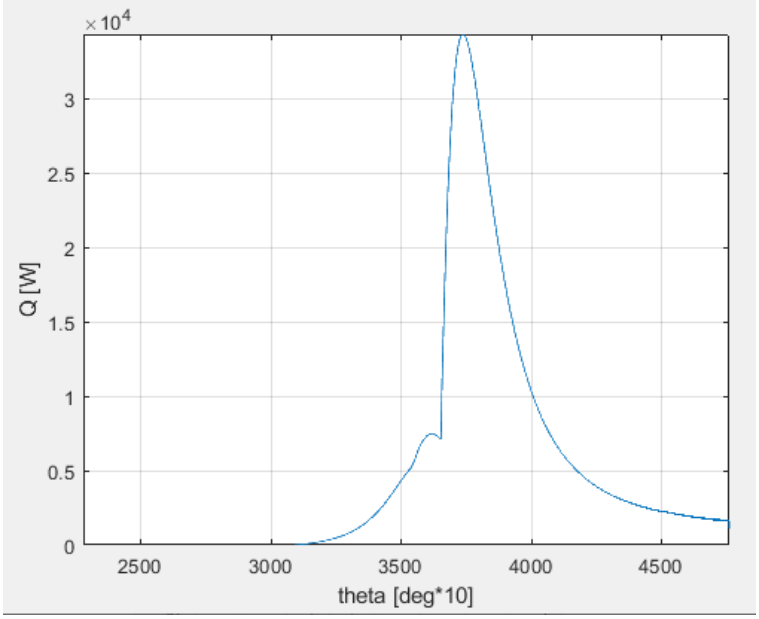
The values of k are computed using the Gatowsky correlation:

$$k_{(i)} = 1.392 - 8.13 \cdot 10^{-5} \cdot T(i)$$

```
T(i)=p_L(i)*V(i)/10/(m_tot*R); %[K]
T(i)=real(T(i));
k(i)=1.392-8.13*10^-5*T(i);
k(i)
w(i) = C_1*U/1000+C_2*(V_u/V(IVC))*T(IVC)/p_L(IVC)*(abs((p_L(i-1)-p_motored(i)))); %[m/s]
w(i)=real(w(i));
h(i) = 3.26*(d*10^-3)^(-0.2)*((p_L(i))*10^2)^(0.8)*w(i)^0.8*T(i)^-0.55; %[W/m^2K]
h(i)=real(h(i));
Q(i) = h(i)*(S_testa*(T_testa-T(i))+S_canna(i)*(T_canna-T(i))+S_stantuffo*(T_stantuffo-T(i)))*10^-6; %[W]
Q(i)=real(Q(i));
p_L(i+1)=p_L(i-1)+1/V(i)*((k(i)-1)*((x_bw(i+1)-x_bw(i-1))*10*(m_b)*Hi+(10*Q(i)/omega)*(theta(i+1)-theta(i-1)))-k(i)*p_L(i)*(V(i+1)-V(i-1)));%[bar]
p_L(i+1)=real(p_L(i+1));
```


1500
1000
500
0
T [K]
2500
3000
3500
4000
4500
theta [deg*10]


2500
2000
1500
1000
500
0
h [W/(m2 K)]
2500
3000
3500
4000
4500
theta [deg*10]


×10^4
3
2.5
2
1.5
1
0.5
0
Q [W]
2500
3000
3500
4000
4500
theta [deg*10]

6.7 Evaluation of HRR and NHRR

It is now possible to compute the net heat release NHRR with the formula:

$$NHRR_i = \frac{k_i}{k_i - 1} \cdot p_i \cdot \frac{(V_{i+1} - V_{i-1})}{(\vartheta_{i+1} - \vartheta_{i-1})} + \frac{1}{k_i - 1} \cdot V_i \cdot \frac{(p_{i+1} - p_{i-1})}{(\vartheta_{i+1} - \vartheta_{i-1})} [J/°]$$

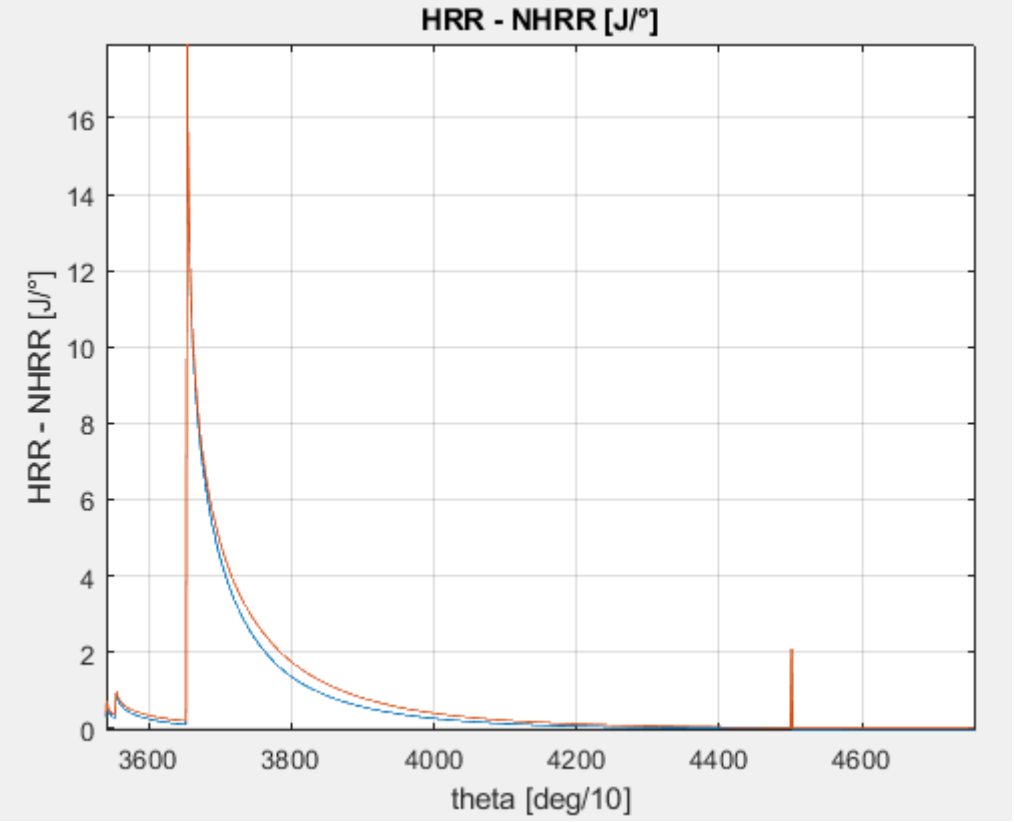
In the same way it is possible to compute the Neat Release Rate subtracting the term of heat exchange with the walls at the already computed NHRR:

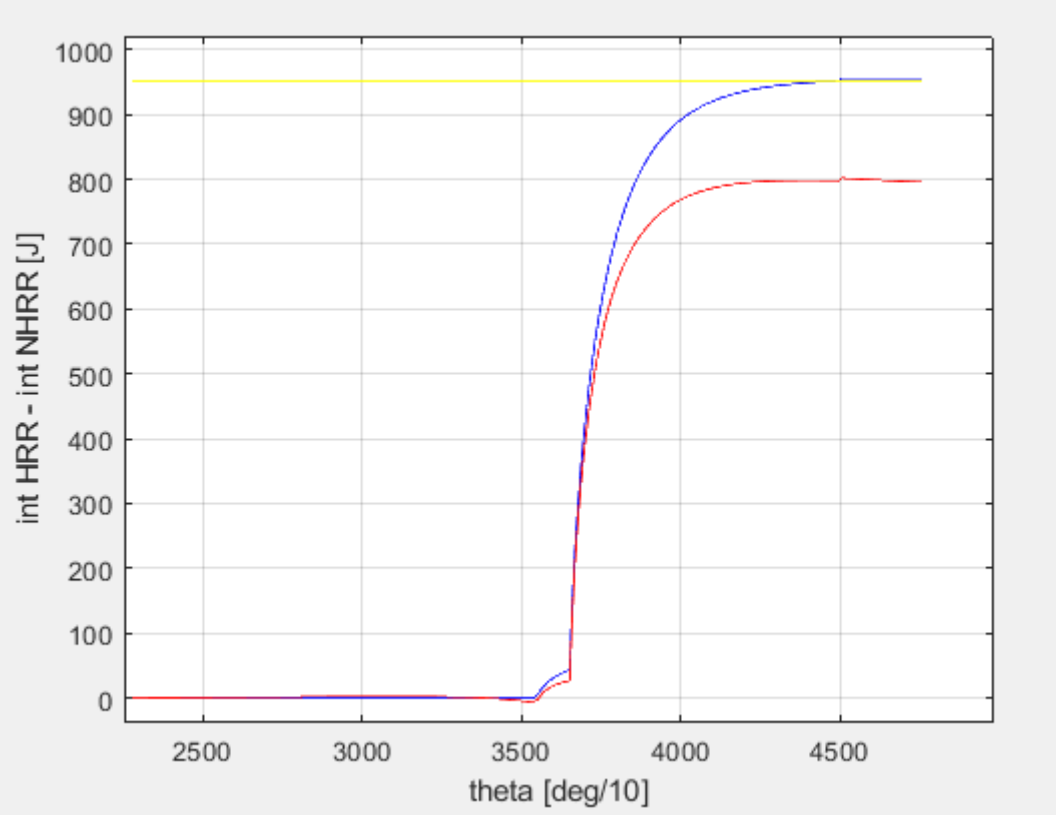
$$HRR_i = \frac{k_i}{k_i - 1} \cdot p_i \cdot \frac{(V_{i+1} - V_{i-1})}{(\vartheta_{i+1} - \vartheta_{i-1})} + \frac{1}{k_i - 1} \cdot V_i \cdot \frac{(p_{i+1} - p_{i-1})}{(\vartheta_{i+1} - \vartheta_{i-1})} - \frac{\dot{Q}_i}{\omega} [J/°]$$

Then starting from the instantaneous values of HRR and NHRR we can compute the integral values of the corresponding heat release laws:

$$\text{int}_{NHRR} = \sum_i NHRR_i [J]$$

$$\text{int}_{HRR} = \sum_i HRR_i [J]$$





6.8 Maximum pressure variation inside the combustion chamber

Referring to the crank angle degree increment chosen, the maximum pressure variation along the cycle is computed:

$$p_{R,\max} = \max\left(\frac{\Delta p_i}{\Delta \vartheta_i}\right)$$

$$where: \Delta p_i = p_{i+1} - p_i$$

The Matlab code uses the functions “max” and “diff” on the chosen interval of the reconstructed pressure cycle.

6.9 Imeph computation

The gross value of the indicated mean effective pressure, is evaluated with respect to the reference point where the intake starts, considering the interval between 180° and 540°.
The formula used for the computation is:

$$imep_{h,calc} = \frac{\sum_{\vartheta=180}^{\vartheta=540} p_i \cdot \Delta V_i}{V_u}$$

Using the “cumtrapz” function implemented in Matlab.

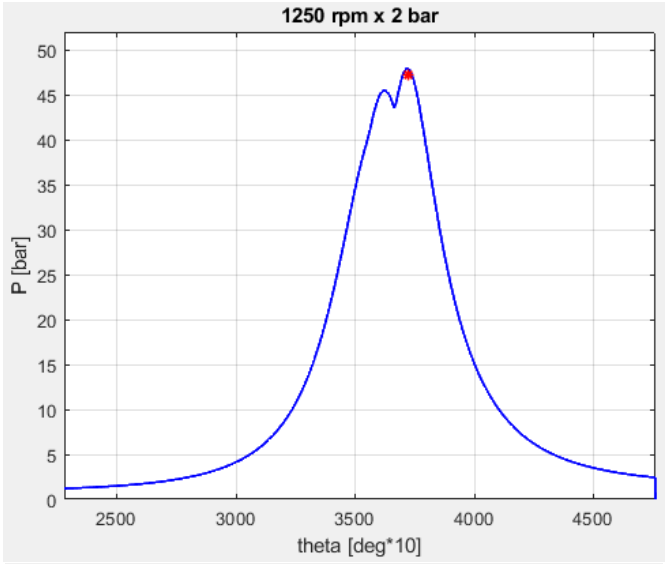
7.Results

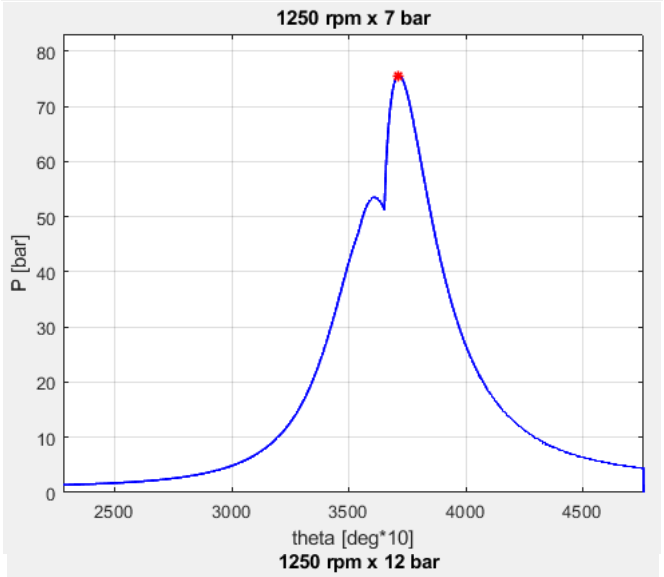
It can be useful now, to analyse some particular points to which the code had been applied, in order to evaluate its reliability. For each fixed engine speed range, three different points had been tested, one at low load, one at a middle load and one at full load. The results are grouped and saved in an Excel sheet on the main Excel file.

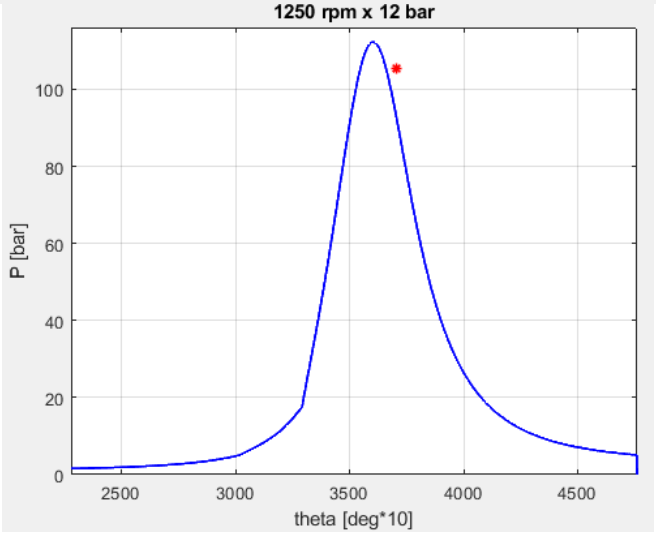
Engine_speed	bmep	imep	p_max	r_c	imep_h_calc	p_max_calc	r_c_eff	Mean_square_error_Multiwiebe	ERR_p_max
[rpm]	[bar]	[bar]	[bar]	[-]	[bar]	[bar]	[-]	[-]	[%]
1250	2	2,85	47,25	16,5	2,1706	45,24	15,607	0,0164	-4,25
1250	7	7,98	75,53	16,5	5,707	72,82	15,607	0,0135	-3,59
1250	11	12,48	105,48	16,5	4,48	109,58	15,607	0,0146	3,89
1500	4	4,84	56,733	16,5	3,45	57,37	15,607	0,0297	1,12
1500	7	7,9	74,2	16,5	5,49	72,14	15,607	0,0168	-2,78
1500	14	15,49	117,98	16,5	11,37	116,83	15,607	0,0182	-0,97
1750	2,01	3,41	44,07	16,5	2,15	43,076	15,607	0,032	-2,26
1750	8	9,28	81,47	16,5	6,18	80,53	15,607	0,0136	-1,15
1750	16	17,71	133,08	16,5	12,18	129,93	15,607	0,0174	-2,37
2000	2	3,17	46,25	16,5	2,03	47,42	15,607	0,0283	2,53
2000	8	9,51	81,49	16,5	5,77	80,19	15,607	0,014	-1,60
2000	16	18,04	136,87	16,5	11,9	143,25	15,607	0,0183	4,66
2250	4	5,55	60,65	16,5	3,5	61,42	15,607	0,026	1,27
2250	10	11,77	95,61	16,5	7,13	95,02	15,607	0,0145	-0,62
2250	16	18,17	137,79	16,5	10,98	133,2	15,607	0,0182	-3,33
2500	2	3,14	53,19	16,5	2,05	53,11	15,607	0,0227	-0,15
2500	8	9,87	82,93	16,5	5,41	80	15,607	0,0149	-3,53
2500	14,99	17,11	134,83	16,5	10,66	141,17	15,607	0,0179	4,70
2750	1	2,46	55,2	16,5	1,89	53,053	15,607	0,0278	-3,89
2750	8	9,77	85,34	16,5	6,306	89,64	15,607	0,015	5,04
2750	15	17,36	138,39	16,5	10,82	138,1	15,607	0,0179	-0,21
3000	2	3,5	63,054	16,5	2,58	63,41	15,607	0,0326	0,56
3000	6	7,7	76,74	16,5	4,61	77,53	15,607	0,017	1,03
3000	15	17,53	143,75	16,5	10,8	142,4	15,607	0,0167	-0,94

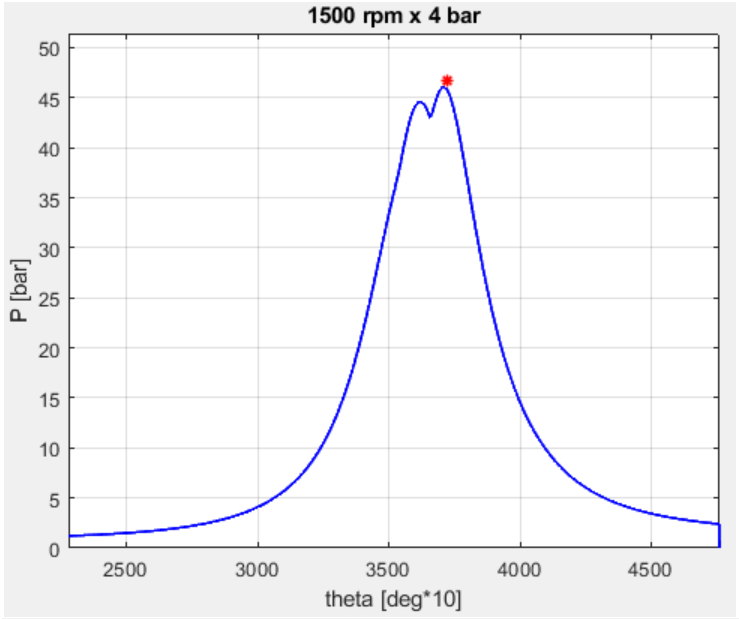
The considered parameters are the maximum pressure inside the combustion chamber, the theoretical and the effective compression ratio, the indicated mean effective pressure (with the given one and the evaluated gross one) and the variation between the experimental curve of mass fraction burned and the modeled Multiwiebe curve (evaluated in terms of the rooted mean square error).
Those parameters have been chosen because they represent in the best possible way the proceeding of the combustion process itself, as well as the main quantities behavior inside the cylinder.
Looking at the table, we can clearly see how the model tends to slightly overestimate the maximum pressure for some points at high loads, while at partial and full ones it is quite accurate.
The accuracy of the model is also noticed in some tests performed to evaluate the overall error between the experimental pressure curve and the modeled ones.
It is also worth to say that the value of the gross indicated mean effective pressure calculated using the model is quite higher than the real one since this quantity is obtained through an integral of the pressure signal, over only a part of the engine cycle.

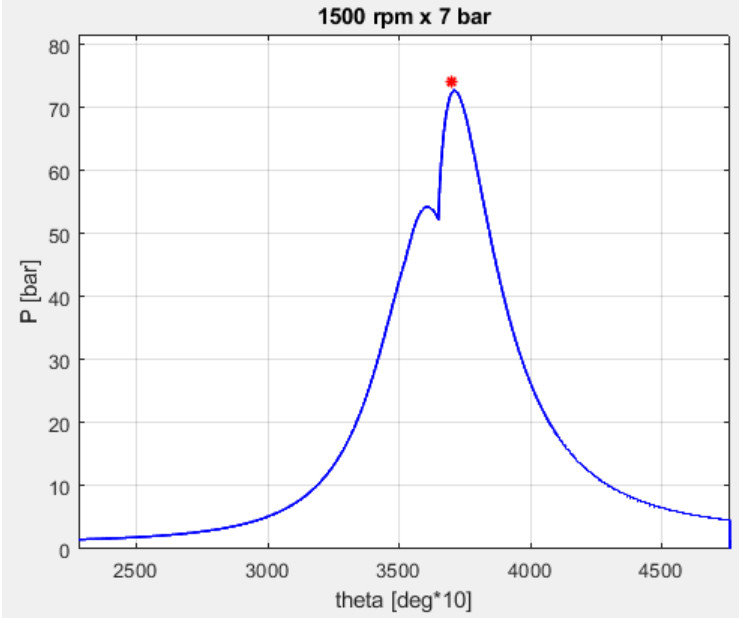
The error on the modeled Multiwiebe processes is observed to be quite low in magnitude, this results in a very good approximation of the combustion process representation since the difference between the modeled process and the real one is always above a value of the 5%.

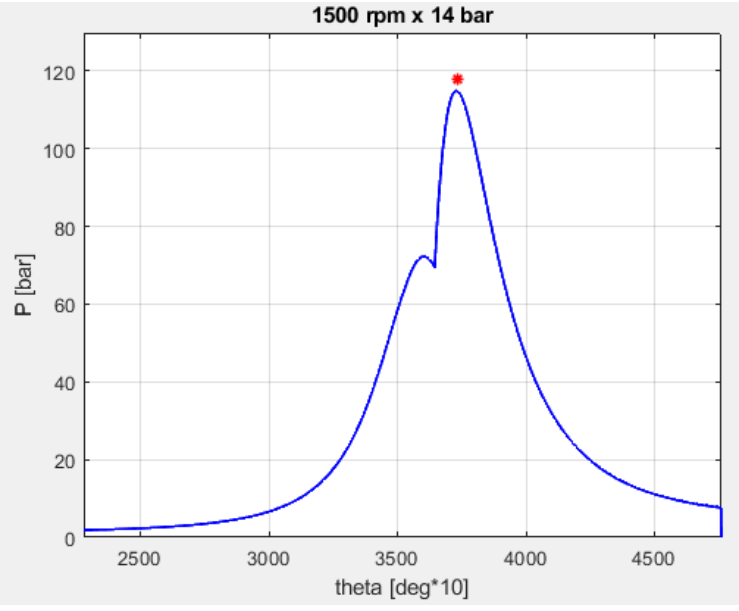


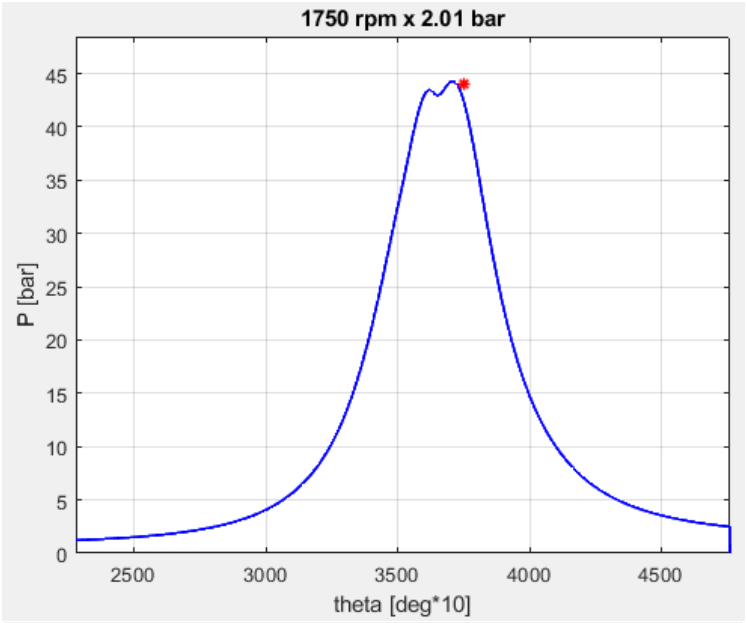


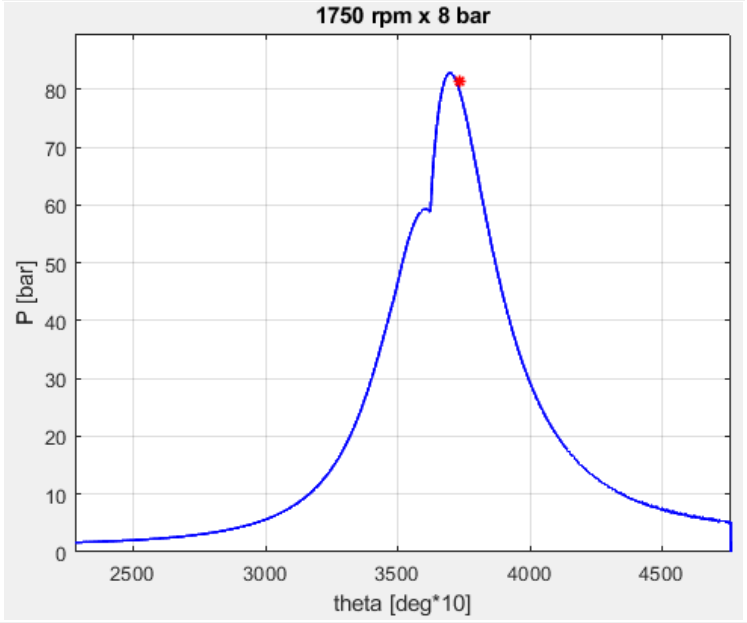


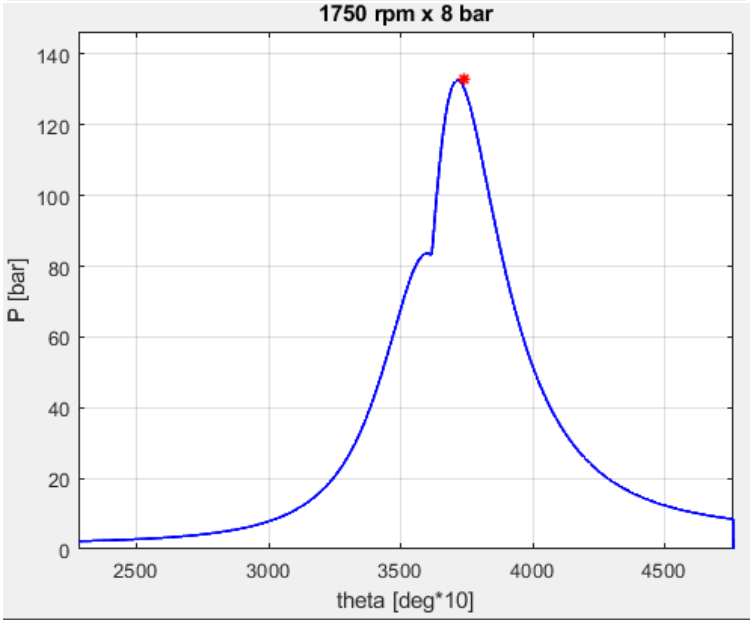

1500 rpm x 4 bar
P [bar]
theta [deg*10]
0
5
10
15
20
25
30
35
40
45
50
2500
3000
3500
4000
4500

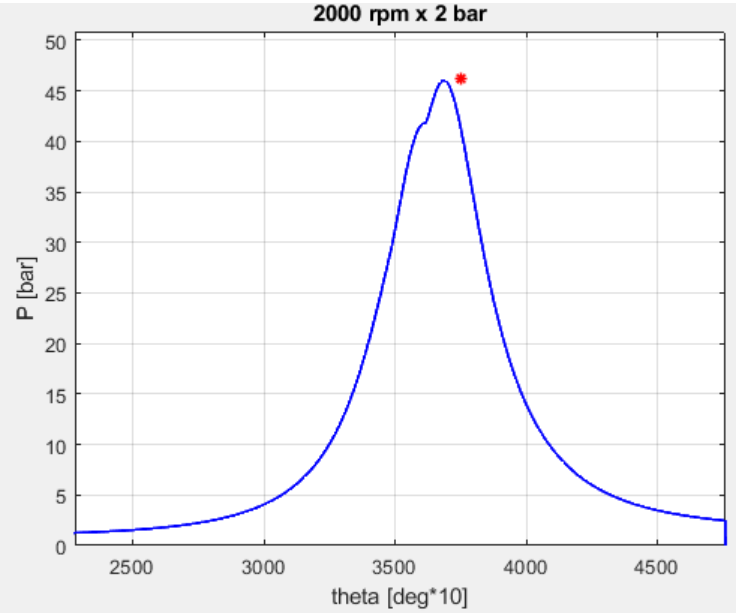

1500 rpm x 7 bar
P [bar]
theta [deg*10]
0
10
20
30
40
50
60
70
80
2500
3000
3500
4000
4500

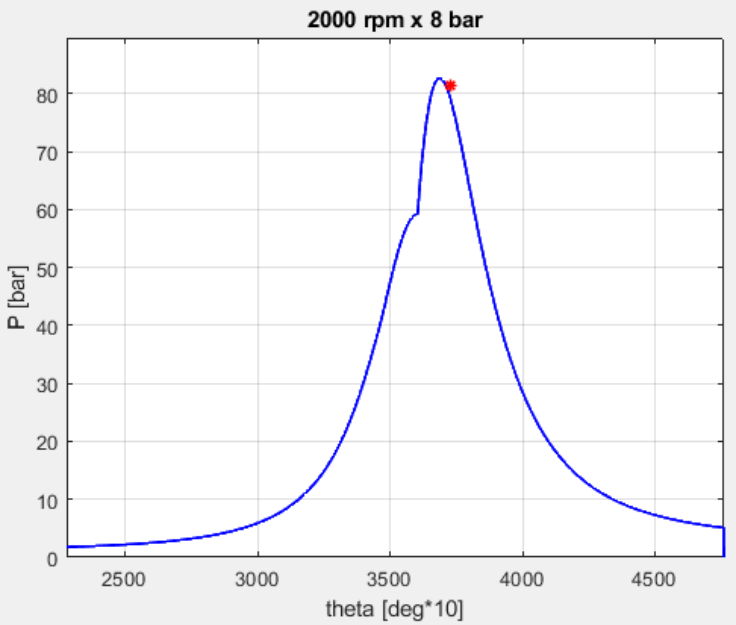

1500 rpm x 14 bar
P [bar]
theta [deg*10]
0
20
40
60
80
100
120
2500
3000
3500
4000
4500

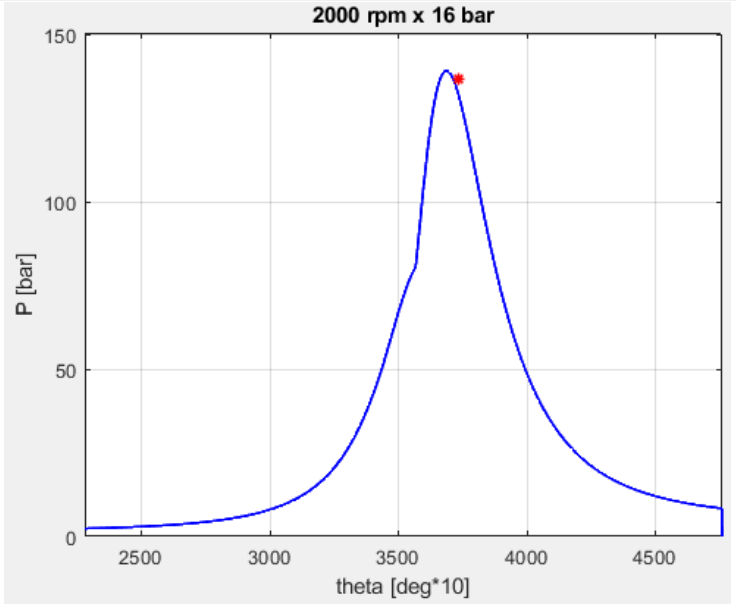

1750 rpm x 2.01 bar
P [bar]
theta [deg*10]
0
5
10
15
20
25
30
35
40
45
2500
3000
3500
4000
4500

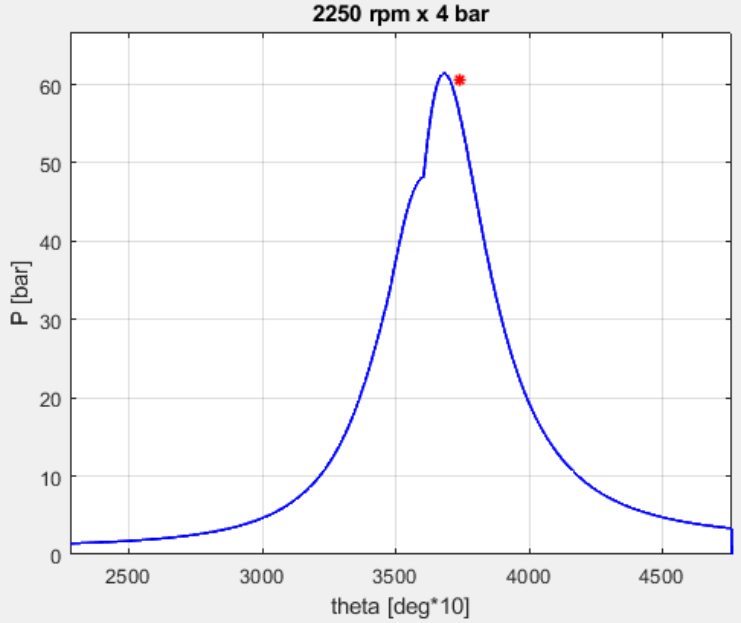

1750 rpm x 8 bar
P [bar]
theta [deg*10]
0
10
20
30
40
50
60
70
80
2500
3000
3500
4000
4500

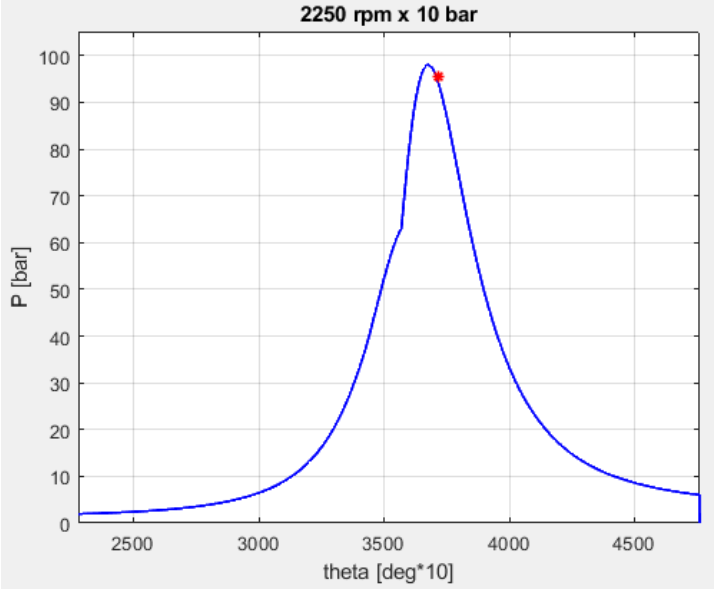

1750 rpm x 8 bar
P [bar]
theta [deg*10]
0
20
40
60
80
100
120
140
2500
3000
3500
4000
4500

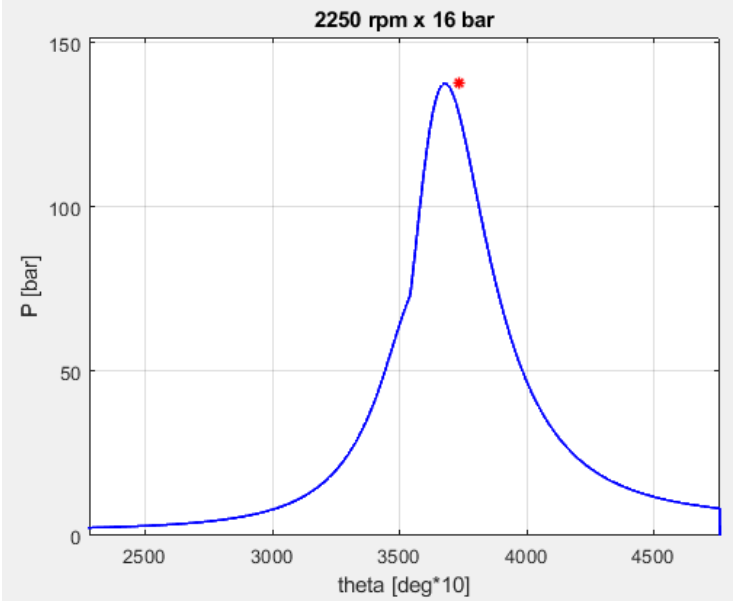

2000 rpm x 2 bar
P [bar]
theta [deg*10]
0
5
10
15
20
25
30
35
40
45
50
2500
3000
3500
4000
4500

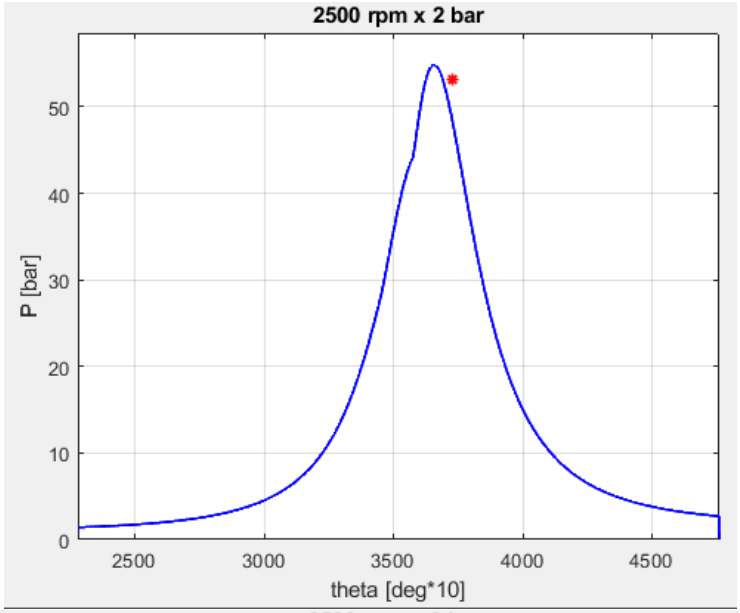

2000 rpm x 8 bar
P [bar]
theta [deg*10]
0
10
20
30
40
50
60
70
80
2500
3000
3500
4000
4500

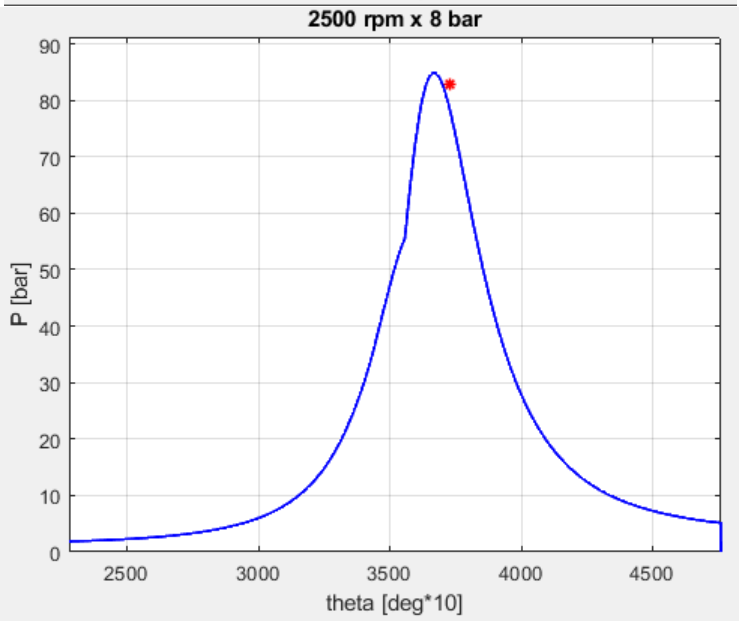

2000 rpm x 16 bar
P [bar]
theta [deg*10]
0
50
100
150
2500
3000
3500
4000
4500

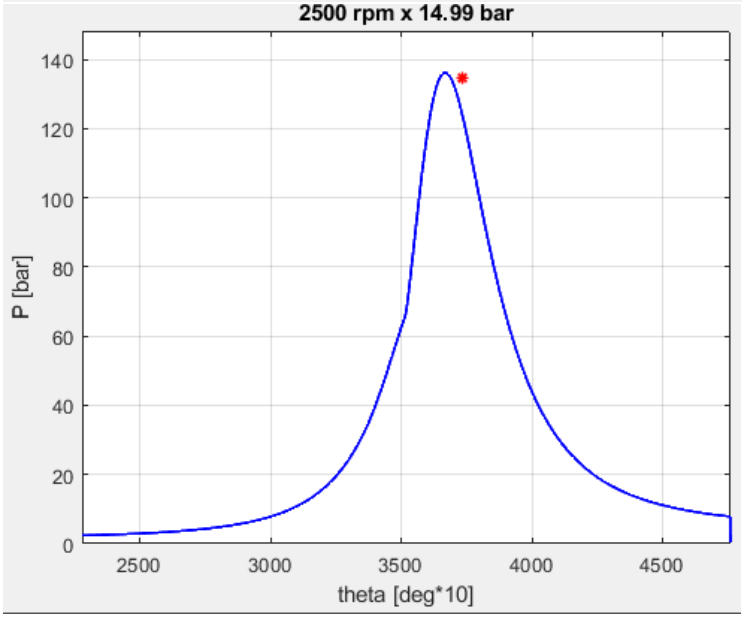

2250 rpm x 4 bar
P [bar]
theta [deg*10]
0
10
20
30
40
50
60
2500
3000
3500
4000
4500

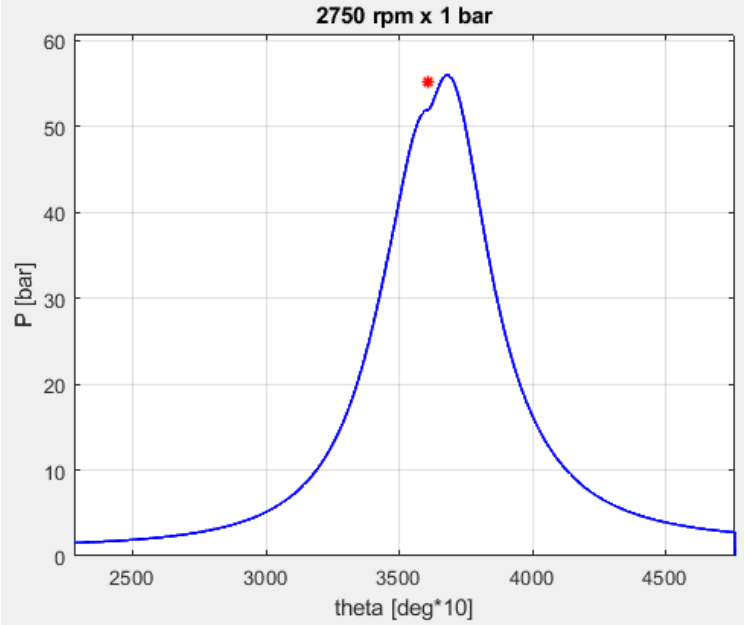

2250 rpm x 10 bar
P [bar]
theta [deg*10]
0
10
20
30
40
50
60
70
80
90
100
2500
3000
3500
4000
4500

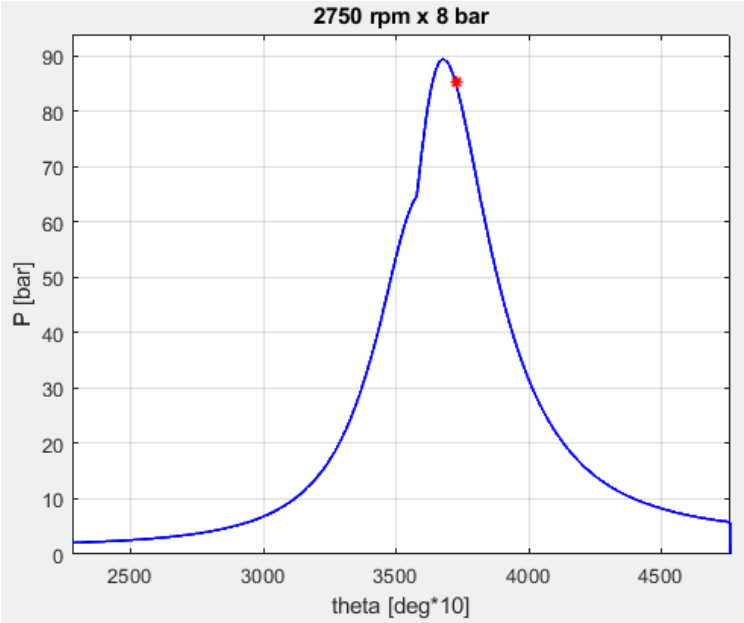

2250 rpm x 16 bar
P [bar]
theta [deg*10]
0
50
100
150
2500
3000
3500
4000
4500

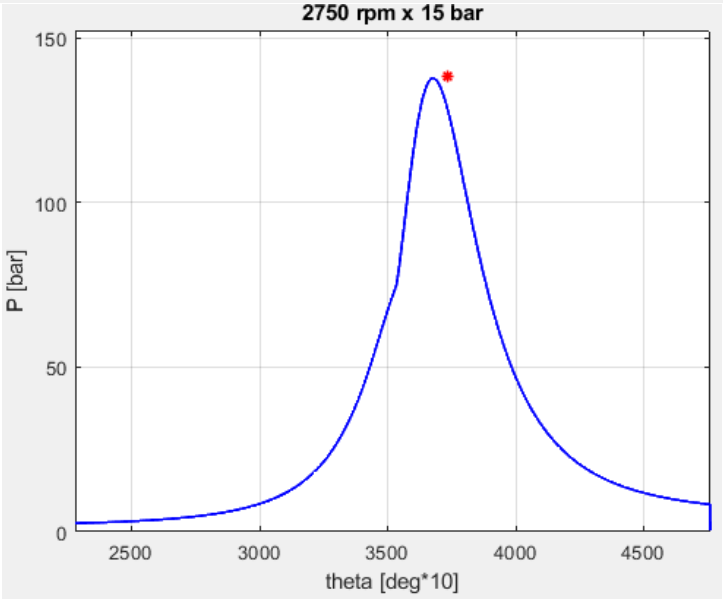

2500 rpm x 2 bar
P [bar]
theta [deg*10]
0
10
20
30
40
50
2500
3000
3500
4000
4500

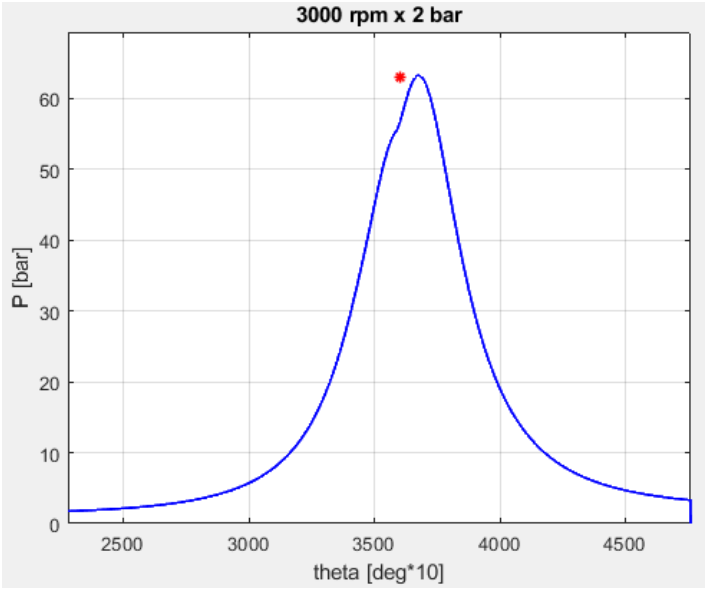

2500 rpm x 8 bar
P [bar]
theta [deg*10]
0
10
20
30
40
50
60
70
80
90
2500
3000
3500
4000
4500

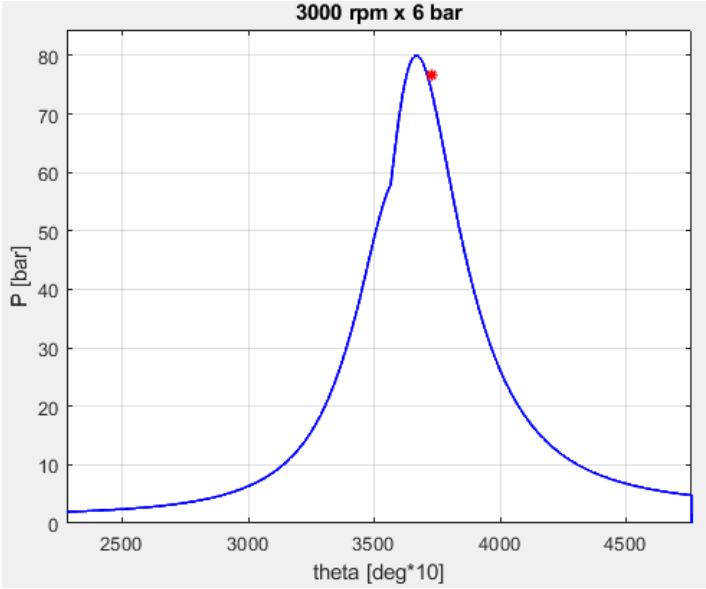

2500 rpm x 14.99 bar
P [bar]
theta [deg*10]
0
20
40
60
80
100
120
140
2500
3000
3500
4000
4500

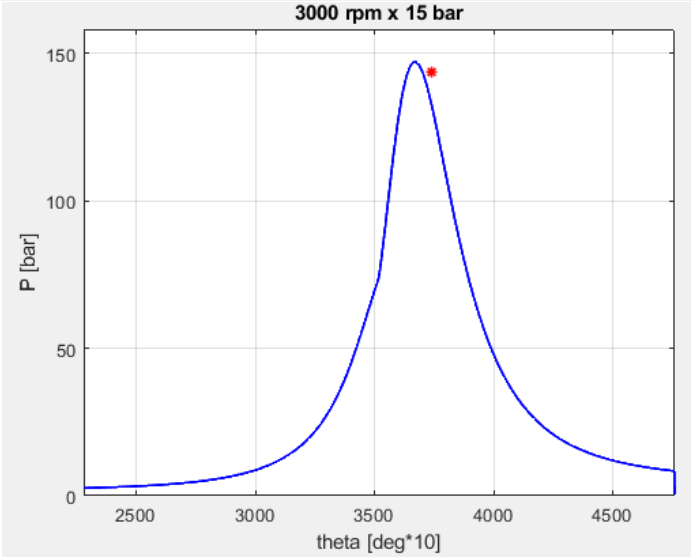

2750 rpm x 1 bar
P [bar]
theta [deg*10]
0
10
20
30
40
50
60
2500
3000
3500
4000
4500


2750 rpm x 8 bar
P [bar]
theta [deg*10]
0
10
20
30
40
50
60
70
80
90
2500
3000
3500
4000
4500


2750 rpm x 15 bar
P [bar]
theta [deg*10]
0
50
100
150
2500
3000
3500
4000
4500


3000 rpm x 2 bar
P [bar]
theta [deg*10]
0
10
20
30
40
50
60
2500
3000
3500
4000
4500


3000 rpm x 6 bar
P [bar]
theta [deg*10]
0
10
20
30
40
50
60
70
80
2500
3000
3500
4000
4500


3000 rpm x 15 bar
P [bar]
theta [deg*10]
0
50
100
150
2500
3000
3500
4000
4500

8.Conclusions

Looking at the results obtained, we can clearly state that the robustness of the code had reached a satisfying level in modelling the expected engine behavior. The development of the Multiwiebe curves for the combustion process, at first was kind of a raw one, but then considering the delay period calculation with the correspondent in-cylinder mass adjustment for every single injection event, it became a very reliable model, showing in a very clear way the combustion evolution.
The model had been carried out to obtain the parameters "*a"* and "*m"* as outputs, to be later considered fixed and used in the computation of the Multiwiebe curve; to further refine the model, a separate calculation of those two exponents for each portion of the Mutliwiebe curve could be carried out; in that way we would obtain a peculiar behavior for each injection event instead of a more common one as it happens in the case of script developed, with not such a great increasing in the code complexity.
As mentioned before, the differences in the evaluation of the two outputs "*a"* and "*m"*, between the case taking as reference the experimental heat release and the one considering just some sampled metrics, is not of a great extent; and of course the tests made had shown that as we increase the number of sampled metrics, the approximation of the calculated Multiwiebe curve is better, and the two coefficients becomes more similar to the ones calculated in the other script.
Another improvement on the robustness of that model could be done by exactly knowing, for each pilot an main injection, the exact value of the Nozzle Opening Delay that had been considered constant for all the injections and equal to 250 ms.
Speaking about the pressure cycle reconstruction, the heat exchange model considered for the computation was the Woschni one, using as coefficients C_1 and C_2 respectively the values found in literature of 2.28 and $3.24*10^{-3}$ but they are approximated given values for the combustion and expansion phase, to be more precise the couple of coefficients can change in values for the different phases of the engine cycle in order to model in a more precise way the heat exchange process; but usually their tuning is made on the basis of experimental data observation and so the choice for those fixed values had been done considering them as sufficiently representative for the whole process.
Another refinement could be done in the calculation of the total mass present inside the cylinder, that had been carried out using the perfect gas law and could be improved implementing the corrected real gas equation to obtain a more precise value.
Also the engine filling process could be modeled in a more representative way if we consider also an exact valve lift profile in order to establish the exact air charge and compression level at each instant (or crank angle degree value) during the intake and exhaust phases, while this model starts its computation of the pressure cycle at Inlet Valve Closing (IVC) time, and ends at the Exhaust Valve Opening (EVO).
The pressure cycles obtained, and the heat release rates had been compared to the experimental values taken from the excel file and the approximation had reached a

satisfying confidence level for almost all the cycles for which the analysis had been carried out.

Software

Data acquisition

```
% close all
% clear all
% clc

%% FIXED DATA
% stroke=90.5; %[mm]
% bore=82; %[mm]
% conrod=145; %[mm]
% CR=16.5;
%% EXCEL DATA ACQUISITION

% [N,T]=xlsread('A20DTH_Beta1plusHW_EngineMapping.xls');

%Matrix of data acquired from excel
% for(i=1:size(N,2)) %definition of matrix columns
%
%    m=1;
%
%    for(j=1:size(N,1)) %definition of matrix rows
%
%       A(m,i)=N(j,i);
%       m=m+1;
%
%    end
%
% end

%A is the matrix corresponding to the excel file

% rpm=A(:,8);
% bmep=A(:,7);
%
% n_i=202; %suddivisioni della meshgrid
%
% px=(max(rpm)-min(rpm))/(n_i-1);
% py=(max(bmep)-min(bmep))/(n_i-1);
%
%
% [Xq,Yq] = meshgrid(min(rpm):px:max(rpm),min(bmep):py:max(bmep));
%
% [Xq,Yq]=meshgrid(sort(rpm),sort(bmep));
%
% xnodes=sort(rpm);
% ynodes=sort(bmep);


%% DATA FITTING

% for(i=1:size(A,2))
%
%    F=scatteredInterpolant(rpm,bmep,A(:,i),'linear','none');
%
```

```
%     vq1=F(Xq,Yq);
%     B(:,:,i)=vq1;
%     ti=strcat(T(1,i),'[',T(2,i),']');
%
% %     figure
% %     surf(Xq,Yq,vq1);
% %     table(ti);
% %     xlabel('n [rpm]');
% %     ylabel('bmep [bar]');
% %     drawnow
% end



%% USEFUL DATA

% P_u=B(:,:,6); %kW
% torque=B(:,:,12); %Nm
% mfb_g=B(:,:,15); %g
% p_max=B(:,:,217); %bar
% QFUEL_MG=B(:,:,205); %mg/c
% imep=B(:,:,199);%bar
% imeph=B(:,:,216); %bar
% bmep=B(:,:,7); %bar
% BSFC_g=B(:,:,203);%g/kWh
% FB_mact=B(:,:,16); %kg/h
% H_i=45500; %kJ/kg
% T_asp=B(:,:,68);
% T_coll=B(:,:,72);
% p_coll=B(:,:,152); %[mbar]
% p_asp=B(:,:,140); %[mbar]
% lambda=B(:,:,158);
% p_amb=B(:,:,151); %[mbar]

%% EFFICIENCIES EVALUATION
% r_c=16.5;
% y_comp=1.3440;
% y_e=1.3063;

% eta_u=10^6*P_u./(4*FB_mact.*rpm/120*H_i);%rendimento utile calcolato a partire da
P_u e m_b
%
% eta_u_obj=10^6*P_u./(4*QFUEL_MG.*rpm/120*H_i);%rendimento utile calcolato a
partire da P_u e m_b_obj
%
% eta_u_q=1./(H_i*BSFC_g/1000/3600);%rendimento utile calcolato a partire dai bsfc
dichiarati
%
%  eta_m=bmep./imeph; %rendimento organico
%
%  eta_lim=1-1./(r_c.^(y_e-1));%rendimento limite calcolato con rapporto di
compressione geometrtico
%                         %e con k dei gas di scarico
% [a,b,l]=find(lambda<1);
%
%     for(i=1:size(a))
%
%         eta_lim(a(i),b(i))=eta_lim(a(i),b(i))*lambda(a(i),b(i));% approssimazione
per miscele ricche
```

```
%
%    end
%
% eta_thetai=eta_u_q./(eta_m.*eta_lim);
% rpm=B(:,:,9);
% figure
% surf(Xq,Yq,eta_m);
% title('eta_M [-]');
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
% drawnow
% figure
% surf(Xq,Yq,eta_u_q);
% title('eta_u_q [-]');
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
% drawnow
% B=[1 2 3 5 8 9 10 12 15 16 18 20 22 25];
% G=[1250 1251 1500 1750 2000 2250 2500 2750 3000 3250 3500 4000 4500 5000];
% y_e=[1.3426 1.3162 1.2876 1.3261 1.3020 1.2785 1.3123 1.3245 1.2673 1.3326 1.3612
1.2786 1.3322 1.2761];
%
% for(i=1:14)
%
%     eta_lim=1-1./(r_c.^(y_e(i)-1));
%     M=[G(i),B(i),eta_lim];
%
% end
% figure
% surf(G,B,M);
% title('eta_lim [-]');
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
% drawnow

%% CALCOLO DI m_b DA bsfc
% m_b_bsfc=1000*(P_u.*BSFC_g./3600)./(4*rpm./120);
% Dm_b=(m_b_bsfc-QFUEL_MG)./QFUEL_MG*100;
% figure
% surf(Xq,Yq,m_b_bsfc);
% title('m_{b_{bsfc}} [-]');
% xlabel('n [rpm]');
% ylabel('pme [bar]');
% drawnow
% figure
% surf(Xq,Yq,Dm_b);
% title('delta_{m_b} [%]');
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
% drawnow

%% PLOTS

%figure
% C=[0:0.1:4];
% [c,h]=contour(Xq,Yq,p_v,C);
% clabel(c,h);
% hold on
% plot(rpm_WOT,BMEP_WOT,'--r');
% title('p_v[bar]');
```

```
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
% drawnow

% figure
% surf(Xq,Yq,p_v);
% title('p_v [bar]');
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
% drawnow

% figure
% surf(Xq,Yq,p_max_ch);
% title('p_{max} [bar]');
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
% drawnow

% figure
% surf(Xq,Yq,p_v_calc);
% title('p_v_{calc} [bar]');
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
% drawnow

%% ENGINE FILLING

% alpha_st=14.6;
% iV=1956;
% R=287;
% % analisi dimensionale
% rho_air_asp=(p_asp*100)/R./(T_asp+273); %[kg/m^3]
% rho_air_coll=((p_coll+p_amb)*100)/R./(T_coll+273);%[kg/m^3]
%
% m_id_asp=(rho_air_asp)*iV/4;%[mg/stk]
% m_id_coll=(rho_air_coll)*iV/4;%[mg/stk]
%
% m_air_eff=alpha_st*lambda.*QFUEL_MG; %[mg]
%
% l_v_asp=m_air_eff./(m_id_asp);
% l_v_coll=m_air_eff./(m_id_coll);

% l_v_asp_EGR=(m_air_eff+m_EGR)./(m_id_asp);
% l_v_coll_EGR=(m_air_eff+m_EGR)./(m_id_coll);

%% PLOTS

% figure
% surf(Xq,Yq,p_asp);
% title('p_{air,asp} [bar]');
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
% drawnow
%
% figure
% surf(Xq,Yq,p_coll);
% title('p_{air,coll} [bar]');
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
```

```
% drawnow
%
% figure
% surf(Xq,Yq,T_asp);
% title('T_{air,asp} [°C]');
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
% drawnow
%
% figure
% surf(Xq,Yq,T_coll);
% title('T_{air,coll} [°C]');
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
% drawnow
%
% figure
% surf(Xq,Yq,rho_air_asp);
% title('rho_{air,asp} [kg/m^{3}]');
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
% drawnow
%
% figure
% surf(Xq,Yq,rho_air_coll);
% title('rho_{air,coll} [kg/m^{3}]');
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
% drawnow
%
% figure
% surf(Xq,Yq,mfb_g);
% title('m_b [mg/stk]');
% xlabel('n [rpm]');
% ylabel('bmep [bar]');
% drawnow
%
% figure
% surf(Xq,Yq,m_air_eff);
% title('m_{air mot}[mg/stk]');
% xlabel('n [rpm]');
% ylabel('pme [bar]');
% drawnow
%
% figure
% surf(Xq,Yq,m_id_asp);
% title('m_{id asp}; [mg/stk]');
% xlabel('n [rpm]');
% ylabel('pme [bar]');
% drawnow
%
% figure
% surf(Xq,Yq,m_id_coll);
% title('m_{id coll} [mg/stk]');
% xlabel('n [rpm]');
% ylabel('pme [bar]');
% drawnow
%
% figure
% surf(Xq,Yq,l_v_coll);
% title('lambda_{v,coll} [-]');
```

```
% xlabel('n [rpm]');
% ylabel('pme [bar]');
% drawnow
%
% figure
% surf(Xq,Yq,l_v_asp);
% title('lambda_{v,asp} [-]');
% xlabel('n [rpm]');
% ylabel('pme [bar]');
% drawnow

%% TORQUE-POWER CURVES

%[rpm,index]=unique(rpm);

% C=max(B(:,1:end,12));
% P=max(B(:,1:end,6));

%
% figure
% title('Torque-Power curves as a function of rpm')
% yyaxis left
% plot(rpm,P,'b')
% ylabel(' P [kW]')
% ylim([0 150])
% hold on
% yyaxis right
% plot(rpm,C,'r')
% ylabel(' T [Nm]')
% ylim([0 800])
% grid on
%
%
% figure
% title('Bsfc_min as a function of rpm')
% plot(rpm,bsfc_min, 'og')
% ylabel(' bsfc [g/kWh]')
% grid on
```

AVL Concerto experimental data acquisition

```
% clear all
% close all
% clc
% %codice filtraggio pressione ULTIMA VERSIONE
%
% %ifile = load_ifile('20081106_TdOff_IMA0_Csam_injmap.003',1);
% file = uigetfile('*.*','PLEASE SELECT THE CORRESPONDING IFILE');
% ifile = load_ifile(file);
% P_raw =ifile.PCYL4.data;
% Pman=ifile.PMAN4.data;
%
% %referenzialità pressione
% CAstep=0.1;
% CA=(0:CAstep:720-CAstep)';
%
%
% Pman_mean= mean(Pman,1); %media per ogni colonna pressione manifold
% [~,pointerleft]=min(abs(CA-175));
% [~,pointerright]=min(abs(CA-185));
%
% P_raw_abs=P_raw-(mean(P_raw(pointerleft:pointerright,:))-Pman_mean); %pressione
grezza assoluta in TAF
%
% %COSTANTS
%    x0 = 0.;
%  xn = 720.;
%    precs=10^-5;
%    itertot= 100;
%    A = inputdlg('please input hintv'); %sampling frequency interval
%    hintv = str2num(cell2mat(A));
%  rh=0.1;  %acquiring step
%    nptmax=7201;
%    nptciclo_p=7200;
%    nintmax=720;
%    nclmax=256;
%    ncicliv=100;
%    nintma2=722;
%    nintmap1=721;
%
%    Thm= zeros(1,nintma2+1);
%
%    dataconv=[1,720000];
%    high_f=[1,720000];
%  whp = hintv / rh;
%  npi =   (whp) + 1;
%  hint = (npi - 1) * rh;
%  whp = (0 - x0) / hint;
%  nintprima = (whp);
%
%  whp = (xn - 0) / hint;
%  ninterv = (whp) + nintprima;
%
%  ni2 = ninterv + 2;
%  xini = 0 - nintprima * hint;
%  xn = xini + hint * ninterv;
%  npt = (npi - 1) * ninterv + 1;
%
```

```
%   for j = 2 :ninterv + 1
%     Thm(j) = xini + 0.5 * hint + (j - 2) * hint;
%     end
%
%       Thm(1) = xini;
%   Thm(ninterv + 2) = xn;
%   whp = (xini - x0) / rh;
%   iprimo =  (whp) + 1;
%   iult = iprimo + npt - 1;
%
%     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     % INIZIALIZATION
%       U_i =zeros(7202,1);
%     Um_i=zeros(nintmax+1,nclmax+1);
%     Zm_i=zeros(nptmax+2,ncicliv+2);
%     ut_i=zeros(nptmax+2,ncicliv+2);
%    % Thi=zeros(1,nptmax);
%     Um_new=zeros(nintma2,1);
%     Um_spl=zeros(nintma2,1);
%      c=zeros(1,2170);
%     p0=zeros(7200,1);
%
%   for iciclo = 1: ncicliv
%       for i = 1: nptmax-1
%
%        dataconv((iciclo-1)*(nptmax-1)+i)=P_raw_abs(i,iciclo);
%
%         end
%     end
%
%
%    %inizio iterazioni
%     for iciclo = 1:ncicliv
%
%        for i = 1: nptmax -1
%             U_i(i)=dataconv((iciclo-1)*(nptmax-1)+i);
%         end
%
%     %valmed
%   for i =1: ninterv
%
%       ii = iprimo + (i - 1)*(npi - 1);
%
%       inf =1+i * (npi - 1);
%
%       Um_i(i+1,iciclo) = 0.5 * (U_i(ii) + U_i(inf));
%
%       for j = 2: npi - 1
%          jj = ii + j - 1;
%          Um_i(i+1,iciclo) = Um_i(i+1,iciclo)+U_i(jj);
%
%         end
%       Um_i(i+1,iciclo) = Um_i(i+1,iciclo)/(npi - 1) ;
%     end
%
%     %%%%splmed 1
%     ic = ni2 - 1;
%   dx = hint * 0.5;
%   for  i = iprimo:iult
%     Thi(i) = xini + (i - iprimo) * rh;
%     end
```

```
%   for(j = 2: ninterv +2)
%
%      Um_new(j)= Um_i(j - 1,iciclo);
%     end
%     %%%%estrap
%      npih = npi * 0.5;
%   we1 = U_i(iprimo) / npih;
%   for(i = iprimo + 1: iprimo + npih-1)
%      we1 = we1+  U_i(i) / npih;
%     end
%
%     we2 = U_i(iult) / npih;
%   for(i = iult -1:-1:  iult - npih + 1)
%      we2 =  we2+ U_i(i) / npih;
%
%     end
%   Um_new(2) = 0.75 * we1 + 0.375 * Um_i(2,iciclo) - 0.125 * Um_i(3,iciclo);
%   Um_new(ninterv +3) = 0.75 * we2 + 0.375 * Um_i(ninterv+1,iciclo) - 0.125 *
Um_i(ninterv ,iciclo);
%   Um_i(1,iciclo) = Um_new(2);
%   Um_i(ninterv +2 ,iciclo) = Um_new(ninterv+3);
%
%      %%% icscu
%
%      [c,cnx]=icscu_new(Thm, Um_new, ni2, ic);
%
%
%      %%% correzione spline per iterazioni
%       kflag=true;
%       niter=0;
%
%        while (kflag)
%
%      kflag=false;
%
%     Um_spl(3) = 0.5 * (Um_new(2) + Um_new(3)) + dx * (0.25 * (c(1) + c(2)) + dx *
(0.1666667 * (c(1 + (ic + 1)) + c(2 + (ic + 1))) + 0.125 * dx * (c(1 + 2 * (ic + 1))
+ c(2 + 2 * (ic + 1)))));
%     err2 = (Um_i(2,iciclo) - Um_spl(3)) / Um_i(2,iciclo);
%
%     if (abs(err2) >= precs)
%       Um_new(3) =Um_new(3)+ 0.75 * err2 * Um_i(2,iciclo);
%       kflag= true;
%      end
%
%
%     for k = 4: ic+1
%       Um_spl(k) = 0.5 * (Um_new(k-1) + Um_new(k)) + dx * (0.75 * c(k-2)+0.25 * c(k-1)
+ dx * (1.1666667 * c(k -2 + (ic + 1)) + 0.1666667 * c(k-1 + (ic + 1)) +dx * (1.875
* c(k - 2 + 2 * (ic + 1)) + 0.125 * c((k-1) + 2 * (ic + 1)))));
%
%       err2 = (Um_i(k - 1,iciclo) - Um_spl(k)) / Um_i(k - 1,iciclo);
%       if abs(err2) >= precs
%          Um_new(k) =Um_new(k)+ 0.75 * err2 * Um_i(k - 1,iciclo);
%             kflag=true;
%        end
%     end
%
%     niter=niter+1;
%
%     if (kflag ==true  && niter <= itertot)
```

```
%     [c]=icscu_new(Thm, Um_new, ni2, ic);
%
%
%     else
%       break
%      end
%
%        end
%
%
%    if(kflag==true)
%       for(i = iprimo: iult)
%
%          Zm_i(i,iciclo) = Zm_i(i,iciclo - 1);
%          ut_i(i,iciclo) = ut_i(i,iciclo - 1);
%         end
%
%    else
%
%         Zm_i(1,iciclo) = Um_new(2);
%       Zm_i(nptmax-1,iciclo) = Um_new(ni2+1);
%
%        k0=1;
%
%  for i = iprimo+1:iult-1
%      k1 = k0 + 1;
%   if Thi(i) >= Thm(k1)
%       k0=k0+1;
%
%   end
%
% D=Thi(i)-Thm(k0);
% Zm_i(i,iciclo) = ((c(k0 + 2 * (ic + 1)) * D + c(k0 + (ic + 1))) * D + c(k0)) * D
+ Um_new(k0+1);
% end
%
%   end
%
%     end
%
%  % valutazione errore nella soluzione
% ZTAF=Zm_i(1:7200,1:100) ; %pressione filtrata da Matlab
% Pe_TAF= mean(ZTAF,2);
%
%   %%%%%%%%% SINGLE ZONE MODEL %%%%%%%%%%%%%%%%%%
%
% V=ifile.V.data;
% V=V(1:2:end);    %solo valori dispari (step 0.1)
% V=V*(10^-9);    %m^3
%
%  %% parameter
% gamma=1.35;
% LHV=42910;
% RHO=0.835e-6;   %kg/mm^3 %KJ/kg;
% ERHO=RHO*LHV;   %KJ/mm3
%
% prompt='Insert test number'
% ans=input(prompt);
% counterz=ans+3;
% for(i=counterz)
%
```

```
%    line_n=strcat('A',int2str(i),':','GD',int2str(i));
%
%    line=xlsread('20070508(1-133)_e_09(1-43)_PQ_EMISSIONI_AF_EI_BS_calcoli.xls',
'20070508_PQ_EMISSIONI_01-133_ca',line_n);
%
%    [N,T]=xlsread('20070508(1-133)_e_09(1-43)_PQ_EMISSIONI_AF_EI_BS_calcoli.xls',
'20070508_PQ_EMISSIONI_01-133_ca',strcat('B',int2str(i)));
%
%    if(length(line)==0)
%
%       i=i+1;
%       k=1;
%       j=j+1;
%       %close all
%
%       continue
%    end
%
% %%VARIABLES FOR PRESSURE CYCLE%%
%
%    IVC_deg_read=xlsread('Z19DTH_ValveLiftProfiles.xls','Profiles','G3');% [°
ATDC]
%    IVC_deg=180+IVC_deg_read;
%
%    n_rec=line(3); %[rpm]
%    imep_rec=line(81);
%    P_u_rec=line(93);
%    bmep_rec=line(4);
%    r_c=16.5;
%    lambda_a_rec=1/line(83);
%
%    m_b_kg=line(71)/line(3)/2/60;%[kg/c]
%
%    p_atm_rec= line(98)/1000;% pressione atmosferica[bar]
%    %p_rel_mot_rec= P(IVC_deg*10);        % (line(45)/1000); %pressione collettore
aspirazione, sovralimentato [bar]
%    p_exh_rec=(line(99)/100);%pressione motore uscita, sovralimentato [bar]
%    theta_p_max_rec=line(9)+360;
%
%    T_out_rec=line(126)+273; %cyl4
%    T_coll_rec=line(116)+273;
%
%    SOIMAIN_rec=360-line(38);
%
%
% %%VARIABLES FOR SUB-FUNCTIONS%%
%
%    qTot_fromECU=line(62)*0.835;%[mm3/c]
%
%    %ID=4; %[°]
%    qPil1=line(32)*0.835; %per averle in mm3/c ho moltiplicato per 0.835 sennò era
in mg/c
%    qPil2=line(34)*0.835;
%    qPil3=line(48)*0.835;
%    q_Main=(line(62)*0.835)-qPil1-qPil2-qPil3;
%    m_b=line(62)*0.835;
%
%    SOI_Pil1=((360-line(38)-((((line(33)*10^-6)))*(line(3)*360/60))));
%    SOI_Pil2=SOI_Pil1-(line(35)*10^-6)*(line(3)*360/60);
%    SOI_Pil3=360-line(49);
%    SOI_MAIN=((360-line(38)));
```

```
%
% %     NOD=250; %[us]
% %     teta0_Pil1=SOI_Pil1+((NOD*10^-6)*(line(3)*360/60))+ID;
% %     teta0_Pil2=SOI_Pil2+((NOD*10^-6)*(line(3)*360/60))+ID;
% %     teta0_Pil3=SOI_Pil3+((NOD*10^-6)*(line(3)*360/60))+ID;
% %     teta0_MAIN=SOI_MAIN+((NOD*10^-6)*(line(3)*360/60))+ID;
% %
%      SOI_vector=[SOI_Pil3 SOI_Pil2 SOI_Pil1 SOI_MAIN];
% %     teta0_vector=[teta0_Pil3 teta0_Pil2 teta0_Pil1 teta0_MAIN];
%      q_vector=[qPil3 qPil2 qPil1 q_Main];
%      logicalq=[qPil3>0 qPil2>0 qPil1>0 q_Main>0];
%      first_non_zeroq=find(logicalq,1);
% %
% %     teta0_first=teta0_vector(first_non_zeroq);
%      SOI_PILfirst=SOI_vector(first_non_zeroq);
% end
%
% table_engine = xlsread('A20DTR_Engine_Data.xlsx');
% %% A20DTR engine data8
%
% stroke =ifile.stroke;                 % in mm
% bore =ifile.bore;                    % in mm
% eps = ifile.compression_ratio;          % compression ratio
% conrod = ifile.conrod_length;           % in mm
% Ncyl = ifile.number_of_cylinders;       % number of cylinders
%
% EVC = table_engine(6);        % exhaust valve closing angle after TDC
% EVO_read = table_engine(7);
% EVO_deg=540-EVO_read;% exhaust valve open after TDC
% Tw = table_engine(8);         % temperature of clyinder body in K
%
% displ = pi * bore^2 /4 * stroke * 10^-9; %volume unitario
%
% SOI=SOI_PILfirst;%single zone approach starting point, should be equal to the
earliest SOI
% EOC=450;   %single zone approach ending point
%
%
% teta_PMI_rad= pi-asin(0/(conrod-(stroke/2)));
% teta_PMS_rad= asin(0/(-conrod-(stroke/2)));
% sfas_deg_PMI= teta_PMI_rad*180/(pi-180);
% sfas_deg_PMS= teta_PMS_rad*180/pi;
% TDC_ASP   =0 +sfas_deg_PMS;
% TDC_ESP   =360 +sfas_deg_PMS;
%
%
% ip_fine_pol=  330/rh+ 1;
% ip_fine_comb= 450/rh+1;
% ip_ch_asp = 198/rh+ 1;
% ip_ap_sca=    (TDC_ESP+EVO_read)/rh+1;
% ip_fine_elab= EOC/rh+1;
%
% Qnetdot=zeros(length(CA),ncicliv);
% %% Qnet by single zone approach
% for j=1:ncicliv
%  P=ZTAF(:,j);
% dP=[0;diff(P)];
% dV=[0;diff(V)];
%
% p_rel_mot_rec= P(IVC_deg*10); %da importare per ricostruzione ciclo pressione, è
la p nel manifold a IVC
```

```
%
% q=[q_vector(first_non_zeroq) qTot_fromECU-q_vector(first_non_zeroq)];   %mm3 fuel
quantity for each injection shot
%
% tmp1(:,j)=(gamma/(gamma-1).*P*1e5.*dV+1/(gamma-1).*V.*dP*1e5)/CAstep;   %J/deg
%
%
% [~,pointerSOI]=min(abs(CA-SOI));
% [~,pointerEOC]=min(abs(CA-EOC));
%
% Qnetdot(pointerSOI:pointerEOC,j)=tmp1(pointerSOI:pointerEOC,j);   %J/deg
%
%
% %calcolo EOC%%%%%%%%%%%
%
% EOC_calc=calc_fine_comb(P,V,Qnetdot(:,j));
%
%   EOC_m(j)=EOC_calc- TDC_ESP;
%
% theta=[1:1:7200];
%
p_r_max_rec=max(diff(P(IVC_deg*10:EVO_deg*10))./diff(theta(IVC_deg*10:EVO_deg*10)
));
%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% %% Qch by shifting and scaling
% Qnet(:,j)=cumtrapz(CA,Qnetdot(:,j));
%
% [~,pointermin]=min(Qnet(:,j));
% [~,pointermax]=max(Qnet(:,j));
% scalefac=sum(q)*ERHO/(Qnet(pointermax,j)-Qnet(pointermin,j));
% Qch=zeros(length(CA),j);
%
Qch(pointermin:pointermax,j)=(Qnet(pointermin:pointermax,j)-Qnet(pointermin,j))*s
calefac;
% Qch(pointermax+1:end,j)=Qch(pointermax,j);
% Xr(:,j)=Qch(:,j)/sum(q)/ERHO;
%
% %MFB50
% [~,pointer]=min(abs(Xr(:,j)-0.5));
% if Xr(pointer,j)>0.5
%    ind=[pointer-1 pointer];
% elseif Xr(pointer,j)<0.5
%    ind=[pointer pointer+1];
% end
% %MFB10
% [~,pointer10]=min(abs(Xr(:,j)-0.1));
% if Xr(pointer10,j)>0.1
%    ind10=[pointer10-1 pointer10];
% elseif Xr(pointer10,j)<0.1
%    ind10=[pointer10 pointer10+1];
% end
% %MFB5
% [~,pointer5]=min(abs(Xr(:,j)-0.05));
% if Xr(pointer5,j)>0.05
%    ind5=[pointer5-1 pointer5];
% elseif Xr(pointer5,j)<0.05
%    ind5=[pointer5 pointer5+1];
% end
% %MFB90
% [~,pointer90]=min(abs(Xr(:,j)-0.9));
```

```
% if Xr(pointer90,j)>0.9
%    ind90=[pointer90-1 pointer90];
% elseif Xr(pointer90,j)<0.9
%    ind90=[pointer90 pointer90+1];
% end
% %MFB1
% [~,pointer1]=min(abs(Xr(:,j)-0.01));
% if Xr(pointer1,j)>0.01
%    ind1=[pointer1-1 pointer1];
% elseif Xr(pointer1,j)<0.01
%    ind1=[pointer1 pointer1+1];
% end
% %MFB65
% [~,pointer65]=min(abs(Xr(:,j)-0.65));
% if Xr(pointer65,j)>0.65
%    ind65=[pointer65-1 pointer65];
% elseif Xr(pointer65,j)<0.65
%    ind65=[pointer65 pointer65+1];
% end
% %MFB75
% [~,pointer75]=min(abs(Xr(:,j)-0.75));
% if Xr(pointer75,j)>0.75
%    ind75=[pointer75-1 pointer75];
% elseif Xr(pointer75,j)<0.75
%    ind75=[pointer75 pointer75+1];
% end
% %MFB95
% [~,pointer95]=min(abs(Xr(:,j)-0.95));
% if Xr(pointer95,j)>0.95
%    ind95=[pointer95-1 pointer95];
% elseif Xr(pointer95,j)<0.95
%    ind95=[pointer95 pointer95+1];
% end
%
% %%%%%%%%%%% valutazione parametri statistici%%%%%%%%%%%%%%%%%%%%%
% MFB50(j)=interp1(Xr(ind,j),CA(ind),0.5)-360;
% MFB5(j)=interp1(Xr(ind,j),CA(ind5),0.5)-360;
% MFB10(j)=interp1(Xr(ind,j),CA(ind10),0.5)-360;
% MFB90(j)=interp1(Xr(ind,j),CA(ind90),0.5)-360;
% MFB65(j)=interp1(Xr(ind,j),CA(ind65),0.5)-360;
% MFB75(j)=interp1(Xr(ind,j),CA(ind75),0.5)-360;
% MFB95(j)=interp1(Xr(ind,j),CA(ind95),0.5)-360;
% MFB1(j)=interp1(Xr(ind,j),CA(ind1),0.5)-360;
%
% MFB5_1=mean(MFB5);
% MFB10_1=mean(MFB10);
% MFB50_1=mean(MFB50);
% MFB90_1=mean(MFB90);
%
% %%%%%%
% Pmax(j) = max(P);   %PRESSIONE MASSIMA
% [~,pointerPmax]=max(P);
% Apmax(j)=CA(pointerPmax)-360; %angolo pressione max
% dp_dteta_max(j) =max(diff(P)/CAstep);
% [~,pointerdp_dteta_max]=max(diff(P)/CAstep);
% Adp_dteta_max(j)=CA(pointerdp_dteta_max)-360;
% p_max_rec=mean(Pmax);%[bar]
%
% %%%%%%%%% IMEP %%%%%
% IMEP(j)=sum(P(1:length(P)).*dV)/displ;
%
```

```
% %%%%%%SOC%%%%%%%
%
% m_comp= log(P(ip_ch_asp)/P(ip_fine_pol))/log(V(ip_fine_pol)/V(ip_ch_asp));
% m_esp= log(P(pointerEOC)/P(ip_ap_sca))/log(V(ip_ap_sca)/V(pointerEOC));
%
%
%
% for i=ip_ch_asp:ip_fine_pol-1
%    p0(i)=P(i);
% end
%  V_mc_prec=V(ip_fine_pol-1);
%
% for i=ip_fine_pol:ip_ap_sca
%
%    V_mc_act=V(i);
%    if i< 3601
%      p0(i)=p0(i-1)*((V_mc_prec/V_mc_act)^m_comp);
%    else
%        p0(i)=p0(i-1)*((V_mc_prec/V_mc_act)^m_esp);
%    end
%
%    if p0(i)>P(i)
%       p0(i)=P(i);
%    end
%       V_mc_prec= V_mc_act;
%
%
% end
%
% ip_comb=ip_fine_elab;
% while ((P(ip_comb)-p0(ip_comb))/p0(ip_comb))> (0.1/100)
%
%    ip_comb=ip_comb-1;
% end
% SOC(j)= (ip_comb-1)*rh+sfas_deg_PMS-360;
%
% %%%%calcolo hrr
%
%  ip_EOC= ((EOC_m(j)+TDC_ESP-TDC_ASP)*10)+1;
%
% hrr(:,j)=Qnetdot(:,j)/Qnet(int64(ip_EOC),j);
% hrr_max(j) = max(hrr(:,j));   %HRR MAX
% [~,pointerHRRmax]=max(hrr(:,j));
% AHRR_MAX(j)=CA(pointerHRRmax)-360; %angolo pressione max
%
% end
%
% %%Calculation of ID for each teta0
%
% NOD=250; %[us]
%
% IVC=round(IVC_deg);
% A=40.44;
% n=1.19;
% CN=45;
% Ea=618.840/(CN+25);
% R=8.13;
% N_ID=n_rec;
%
% p3=P(round(SOI_Pil3)*10)*0.986923;
% m_tot_cyl3=(1.2*p_rel_mot_rec*V(IVC*10)/(287*T_coll_rec)/10);
```

```
% T3=(P(round(SOI_Pil3)*10)*V(round(SOI_Pil3)*10))/(10*287*m_tot_cyl3);
% ID_ms3=A*p3^-n*exp(Ea/(R*T3));
% ID_deg3=ID_ms3*0.006*N_ID;
%
% teta0_Pil3=SOI_Pil3+((NOD*10^-6)*(line(3)*360/60))+ID_deg3;
% teta0_Pil3=double(teta0_Pil3);
%
%
% p2=P(round(SOI_Pil2)*10)*0.986923;
% m_tot_cyl2=m_tot_cyl3+((qPil3/0.835)*10^-9);
% T2=(P(round(SOI_Pil2)*10)*V(round(SOI_Pil2)*10))/(10*287*m_tot_cyl2);
% ID_ms2=A*p2^-n*exp(Ea/(R*T2));
% ID_deg2=ID_ms2*0.006*N_ID;
%
% teta0_Pil2=SOI_Pil2+((NOD*10^-6)*(line(3)*360/60))+ID_deg2;
% teta0_Pil2=double(teta0_Pil2);
%
% p1=P(round(SOI_Pil1)*10)*0.986923;
% m_tot_cyl1=m_tot_cyl2+((qPil2/0.835)*10^-9);
% T1=(P(round(SOI_Pil1)*10)*V(round(SOI_Pil1)*10))/(10*287*m_tot_cyl1);
% ID_ms1=A*p1^-n*exp(Ea/(R*T1));
% ID_deg1=ID_ms1*0.006*N_ID;
%
% teta0_Pil1=SOI_Pil1+((NOD*10^-6)*(line(3)*360/60))+ID_deg1;
% teta0_Pil1=double(teta0_Pil1);
%
% pMAIN=P(round(SOI_MAIN)*10)*0.986923;
% m_tot_cylMAIN=m_tot_cyl1+((qPil1/0.835)*10^-9);
% TMAIN=(P(round(SOI_MAIN)*10)*V(round(SOI_MAIN)*10))/(10*287*m_tot_cylMAIN);
% ID_msMAIN=A*pMAIN^-n*exp(Ea/(R*TMAIN));
% ID_degMAIN=ID_msMAIN*0.006*N_ID;
%
% teta0_MAIN=SOI_MAIN+((NOD*10^-6)*(line(3)*360/60))+ID_degMAIN+3;
% teta0_MAIN=double(teta0_MAIN);
%
% teta0_vector=[teta0_Pil3 teta0_Pil2 teta0_Pil1 teta0_MAIN];
% teta0_first=teta0_vector(first_non_zeroq);
% teta0_first=double(teta0_first)
%
%
%
%
%
%
%
%
%
stat_1=[SOC',EOC_m',IMEP',Pmax',Apmax',dp_dteta_max',Adp_dteta_max',hrr_max',AHRR
_MAX',MFB5',MFB10',MFB50',MFB90'];
%
[stat_2,xb_ens,hrr_ens,Qnet_ens]=stat2(Pe_TAF,V,SOI_PILfirst,qPil1,qTot_fromECU,d
ispl,TDC_ASP,TDC_ESP,sfas_deg_PMS,sfas_deg_PMI,rh);
%
[a,m,Multiwiebe,delta_teta_0,mean_square_error_Multiwiebe,mean_square_error_Wiebe
]=Wiebe02(jj,xb_ens,ip_comb,ip_fine_comb,ip_EOC,counterz,SOI_PILfirst,SOI_Pil1,SO
I_Pil2,SOI_Pil3,SOI_MAIN,qPil1,qPil2,qPil3,q_Main,m_b,teta0_Pil1,teta0_Pil2,teta0
_Pil3,teta0_MAIN,teta0_first);
% %[am,mm,delta_teta_0m,Wiebe_metriche,Multiwiebe_metrics,mean_square_error_Multi
wiebe_metrics,mean_square_error_Wiebe_metrics] =
Wiebe_metriche(counterz,xb_ens,jj,ip_comb,ip_EOC,ip_fine_comb,pointer1,pointer5,p
```

```
ointer10,pointer,pointer65,pointer75,pointer90,pointer95,SOI_PILfirst,qPil1,qPil2
,qPil3,q_Main,m_b,teta0_Pil1,teta0_Pil2,teta0_Pil3,teta0_MAIN,teta0_first);
%
[r_ceff,p_max_clc,imep_clc,bmep_clc,imep_h,pmep,p_r_max]=Untitled29(IVC_deg,EVO_d
eg,n_rec,lambda_a_rec, m_b_kg,p_atm_rec,p_rel_mot_rec,...
%
p_exh_rec,m_comp,m_esp,p_max_rec,theta_p_max_rec,imep_rec,bmep_rec,a,m,delta_teta
_0,T_coll_rec,T_out_rec,...
%
SOI_PILfirst,qPil1,qPil2,qPil3,q_Main,m_b,teta0_Pil1,teta0_Pil2,teta0_Pil3,teta0_
MAIN,teta0_first,ip_fine_comb,Multiwiebe,P);
%
```

Multiwiebe function as a function of x_b

```
% function
[a,m,Multiwiebe,delta_teta_0,mean_square_error_Multiwiebe,mean_square_error_Wiebe
]=
Wiebe02(jj,xb_ens,ip_comb,ip_fine_comb,ip_EOC,counterz,SOI_Pil1,SOI_Pil2,SOI_Pil3
,SOI_MAIN,SOI_PILfirst,qPil1,qPil2,qPil3,q_Main,m_b,teta0_Pil1,teta0_Pil2,teta0_P
il3,teta0_MAIN,teta0_first)
%
% delta_teta_0=ip_EOC-ip_comb;
% t=[round(SOI_PILfirst-30)*10:1:ip_fine_comb]';
% xb_ens_cal=xb_ens(round(SOI_PILfirst-30)*10:ip_fine_comb);
%
% f=@(B,t)
1*(((qPil3/m_b).*(1-exp((-B(1).*((t-(teta0_Pil3*10))./(delta_teta_0)).^(B(2)+1)))
)).*(t>teta0_Pil3*10)+... %pilot injection 1
%
((qPil2/m_b).*(1-exp((-B(1).*((t-(teta0_Pil2*10))./(delta_teta_0)).^(B(2)+1))))).
*(t>teta0_Pil2*10)+...%pilot injection 2
%
((qPil1/m_b).*(1-exp((-B(1).*((t-(teta0_Pil1*10))./(delta_teta_0)).^(B(2)+1))))).
*(t>teta0_Pil1*10)+...%pilot injection 3
%
((q_Main/m_b).*(1-exp((-B(1).*((t-(teta0_MAIN*10))./(delta_teta_0)).^(B(2)+1)))))
.*(t>teta0_MAIN*10)); %Main
%
% B0=[0 0];
% [B]=lsqcurvefit(f,B0,t,xb_ens_cal);
%
% a=real(B(1));
% m=real(B(2));
%
%
%
% CE=1;
%
Multiwiebe=CE.*(((qPil3/m_b).*(1-exp((-a.*((t-((teta0_Pil3*10)))./(delta_teta_0))
.^(m+1))))).*(t>teta0_Pil3*10)+... %pilot injection 1
%
((qPil2/m_b).*(1-exp((-a.*((t-((teta0_Pil2*10)))./(delta_teta_0)).^(m+1))))).*(t>
teta0_Pil2*10)+...%pilot injection 2
%
((qPil1/m_b).*(1-exp((-a.*((t-(teta0_Pil1*10))./(delta_teta_0)).^(m+1))))).*(t>te
ta0_Pil1*10)+...%pilot injection 3
%
((q_Main/m_b).*(1-exp((-a.*((t-((teta0_MAIN*10)))./(delta_teta_0)).^(m+1))))).*(t
>teta0_MAIN*10)); %Main
%
%
multiwiebe_test=1.*(1-exp((-a.*(((t-teta0_first*10))./(delta_teta_0)).^(m+1)))).*
(t>teta0_first*10);
%
%  figure
%  plot(t,Multiwiebe,'r')
%  title('Multiwiebe as a function of x_b')
%  xlabel('theta [deg*10]')
%  hold on
%  plot(t,xb_ens_cal)
%
% mean_square_error_Multiwiebe=real(rmse(Multiwiebe,xb_ens_cal))
% mean_square_error_Wiebe=real(rmse(multiwiebe_test,xb_ens_cal))
% end
```

Multiwiebe function as a function of given metrics

```
% function
[am,mm,delta_teta_0m,Wiebe_metrics,Multiwiebe_metrics,mean_square_error_Multiwieb
e_metrics,mean_square_error_Wiebe_metrics] =
Wiebe_metriche(counterz,xb_ens,jj,ip_comb,ip_EOC,ip_fine_comb,pointer1,pointer5,p
ointer10,pointer,pointer65,pointer75,pointer90,pointer95,SOI_PILfirst,qPil1,qPil2
,qPil3,q_Main,m_b,teta0_Pil1,teta0_Pil2,teta0_Pil3,teta0_MAIN,teta0_first)
%
% mfb=[0 0.01 0.05 0.1 0.5 0.65 0.75 0.9 0.95 1];
%
t=[ip_comb-100,pointer1,pointer5,pointer10,pointer,pointer65,pointer75,pointer90,
pointer95,ip_fine_comb];
% xb_ens_cal=xb_ens(round(SOI_PILfirst-30)*10:ip_fine_comb);
% delta_teta_0m=ip_EOC-ip_comb;
%
% f=@(B,t)
1*(((qPil3/m_b).*(1-exp((-B(1).*((t-(teta0_Pil3*10))./(delta_teta_0m)).^(B(2)+1))
))).*(t>teta0_Pil3*10)+... %pilot injection 1
%
((qPil2/m_b).*(1-exp((-B(1).*((t-(teta0_Pil2*10))./(delta_teta_0m)).^(B(2)+1)))))
.*(t>teta0_Pil2*10)+...%pilot injection 2
%
((qPil1/m_b).*(1-exp((-B(1).*((t-(teta0_Pil1*10))./(delta_teta_0m)).^(B(2)+1)))))
.*(t>teta0_Pil1*10)+...%pilot injection 3
%
((q_Main/m_b).*(1-exp((-B(1).*((t-(teta0_MAIN*10))./(delta_teta_0m)).^(B(2)+1))))
).*(t>teta0_MAIN*10)); %Main
%
%
% B0=[0 0];
% [B]=lsqcurvefit(f,B0,t,mfb);
%
% am=real(B(1));
% mm=real(B(2));
%
% x=[round(SOI_PILfirst-30)*10:1:ip_fine_comb]';
% Wiebe_metrics=
(1-exp((-am.*((x-(teta0_first*10))./delta_teta_0m).^(mm+1)))).*(x>teta0_first*10)
;
%
% CE=1;
%
Multiwiebe_metrics=CE.*(((qPil3/m_b).*(1-exp((-am.*((x-((teta0_Pil3*10))./(delta
_teta_0m)).^(mm+1))))).*(x>teta0_Pil3*10)+... %pilot injection 1
%
((qPil2/m_b).*(1-exp((-am.*((x-((teta0_Pil2*10))./(delta_teta_0m)).^(mm+1))))).*
(x>teta0_Pil2*10)+...%pilot injection 2
%
((qPil1/m_b).*(1-exp((-am.*((x-(teta0_Pil1*10))./(delta_teta_0m)).^(mm+1))))).*(x
>teta0_Pil1*10)+...%pilot injection 3
%
((q_Main/m_b).*(1-exp((-am.*((x-((teta0_MAIN*10))./(delta_teta_0m)).^(mm+1))))).
*(x>teta0_MAIN*10)); %Main
%
%
% figure
% plot(x,Multiwiebe_metrics,'b')
```

```
% title('Wiebe as function of metrics')
% xlabel(' Theta [deg*10]')
% axis([3000 4600 0 1])
% grid on
% hold on
% plot(t,mfb,'r*')
% % hold on
% % plot(x,xb_ens_cal)
% % hold on
% % plot(x,Multiwiebe_metrics,'b')
%
%
%
mean_square_error_Multiwiebe_metrics=real(rmse(Multiwiebe_metrics,xb_ens_cal)) %L
'MSE ed RMSE non sono quantità a-dimensionali, bensì assumono l'unità di misura della
grandezza considerata (RMSE) ed il suo quadrato (MSE)
% mean_square_error_Wiebe_metrics=real(rmse(Wiebe_metrics,xb_ens_cal))
% end
```

Pressure cycle reconstruction

```
% function
[r_ceff,p_max_clc,imep_clc,bmep_clc,imep_h,pmep,p_r_max]=Untitled29(IVC_deg,EVO_d
eg,n_rec,lambda_a_rec, m_b_kg,p_atm_rec,p_rel_mot_rec,...
%
p_exh_rec,m_comp,m_esp,p_max_rec,theta_p_max_rec,imep_rec,bmep_rec,a,m,delta_teta
_0,T_coll_rec,T_out_rec,SOI_PILfirst,qPil1,qPil2,qPil3,q_Main,m_b,teta0_Pil1,teta
0_Pil2,teta0_Pil3,teta0_MAIN,teta0_first,ip_fine_comb,Multiwiebe,P);
% d_theta=10;
%
% theta=[0.1:1/d_theta:720]; %[deg°CA]
%
% p_in_mot = p_rel_mot_rec;%[bar]
% p_out_mot= 1.2*p_exh_rec;%[bar]
%
% d=90.5; %[mm]
% c=82; %[mm]
% Vtot=1956; %[cm^3]
% epsilon=16.5;
% lambda=0.28; %elongation value
% U=2*c*n_rec/60; %[mm/s]
% omega=6*n_rec;%[°/S]
%
% k_t=1;
% k_p=1.2;
% R=287; %[J/kgK]
% Hi=44*10^6; %[J/kg]
% alpha_st=14.6;
% alpha_r=0.7;
% n_cil=4;
% d_theta=10;
% IVC=round(IVC_deg);
% EVO=round(EVO_deg);
%
% alpha=alpha_st*lambda_a_rec;
%
% r=c/2;%[mm] crank radius
% V_u=Vtot/n_cil; %[cm^3] volume
% S=pi*d^2/4;%[mm^2] piston surface
% V_mu=V_u/(epsilon-1);%[cm^3] BDC volume
%
x=r*[(1-cos(theta*(pi)/180))+(1/lambda)*(1-sqrt(1-(lambda*sin(theta*(pi)/180)).^2
))];%[mm] actual instantaneous stroke
% V=V_mu+(S*(x))/1000;%[cm^3] chamber volume
% %% Woschni
%
% C_1= 2.28;
% C_2= 3.24*10^-3;
%
% %% Cycle reconstruction
%
% % wall temperatures extimation
%
% T_testa=500; %[K]
% T_canna=430; %[K]
% T_stantuffo=550; %[K]
%
% % in-cylinder masses
```

```
%
% m_a=alpha*(m_b_kg); % massa d'aria [kg]
% m_r=alpha_r*(m_b_kg); % massa gas residui [kg]
%
% m_tot=p_in_mot*V(IVC*10)/(R*T_coll_rec)/10 %[kg]
%
% % wall surfaces
%
% S_testa=pi/4*d^2*k_t; %[mm^2]
% S_canna=pi*d*(x); %[mm^2]
% S_stantuffo=pi/4*d^2*k_p; %[mm^2]
%
% S_tot=S_testa+S_canna+S_stantuffo;
% %
% % Wiebe functions
%
% theta=[0.1:0.1:720];
%
% multiwiebe=zeros;
%
%
for(i=(round(SOI_PILfirst-30)*10)+100:1:(round(SOI_PILfirst-30)*10)+100+round(siz
e(Multiwiebe,[1]))-1)
%     multiwiebe(i)=Multiwiebe(i-((round(SOI_PILfirst-30)*10)+100-1));
% end
% multiwiebe((round(SOI_PILfirst-30)*10)+100+round(size(Multiwiebe,[1])))=1;
% for(i=(round(SOI_PILfirst-30)*10)+100+round(size(Multiwiebe,[1]))+1:1:7200)
%     multiwiebe(i)=1;
% end
% multiwiebe=multiwiebe';
% % figure
% % plot(multiwiebe)
% % grid on
%
% p_motored=[p_atm_rec*(V(IVC*10)./V(IVC*10:3600)).^m_comp,...
%
p_atm_rec*(V(IVC*10)./V(3600)).^m_comp*(V(3600)./V(3600:EVO*10)).^m_esp]; %[bar]
%
%
p_motored=[p_atm_rec*ones(IVC*10-1,1);p_motored';1.1*p_atm_rec*ones(size(theta,2)
-1-(EVO*10),1)]';
%
% p_o=[p_in_mot*(V(IVC*10)./V(IVC*10:3600)).^m_comp,...
%
p_in_mot*(V(IVC*10)./V(3600)).^m_comp*(V(3600)./V(3600:EVO*10)).^m_esp]; %[bar]
%
%
p_o=[p_in_mot*ones(IVC*10-1,1);p_o';p_in_mot*ones(size(theta,2)-1-(EVO*10),1)]';
%
% % figure
% % plot(V)
% % figure
% % plot(p_motored)
% % title('Motored Pressure and Motored Pressure with boost vs Crank Angle');
% % xlabel('theta [deg*10]');
% % ylabel('P [bar]');
% % grid on
% % hold on
% % plot(p_o)
%
%
```

```
% p_L=p_o;
% T=zeros(1,7200);
% w=zeros(1,7200);
% h=zeros(1,7200);
% Q=zeros(1,7200);
% IVC=IVC_deg*10;
% EVO=EVO_deg*10;
% for(i=(IVC-2):(EVO)) %EVO
%
%    T(i) = p_L(i)*V(i)/10/(m_tot*R); %[K]
%
%    k(i) = 1.25;         %1.392-8.13*10^-5*T(i-1);
%
%    w(i) =
C_1*U/1000+C_2*(V_u/V(IVC))*T(IVC)/p_L(IVC)*(abs((p_L(i-1)-p_motored(i)))); %[m/s
]
%
%    h(i) =
3.26*(d*10^-3)^(-0.2)*((p_L(i))*100)^(0.8)*w(i)^0.8*T(i)^-0.55; %[W/m^2K]
%
%    Q(i) =
h(i)*(S_testa*(T_testa-T(i))+S_canna(i)*(T_canna-T(i))+S_stantuffo*(T_stantuffo-T
(i)))*10^-6; %[W]
%
%
p_L(i+1)=p_L(i-1)+1/V(i)*((k(i)-1)*((multiwiebe(i+1)-multiwiebe(i-1))*10*(m_b_kg)
*Hi+(10*Q(i)/omega)*(theta(i+1)-theta(i-1)))-k(i)*p_L(i)*(V(i+1)-V(i-1)));%[bar]
%
% end
%
% theta=[1:1:7200];
% figure
% plot(theta,p_L)
% title('Pressure: experimental vs calculated')
% grid on
% hold on
% plot(P)
% % figure
% % plot(theta,T,'r')
% % xlabel('theta [deg*10]');
% % ylabel('T [K]');
% % axis([IVC EVO 0 max(T)])
% % grid on
% % figure
% % plot(theta,h)
% % xlabel('theta [deg*10]');
% % ylabel('h [W/(m^2 K)]');
% % axis([IVC EVO 0 max(h)])
% % grid on
% % figure
% % plot(theta,-Q)
% % xlabel('theta [deg*10]');
% % ylabel('Q [W]');
% % axis([IVC EVO 0 max(-Q)])
% % grid on
% for(i=2:EVO-1)
%
%    NHRR(i)=
k(i)/(k(i)-1).*p_L(i)/10*(V(i+1)-V(i-1))./(theta(i+1)-theta(i-1))+...
%       1/(k(i)-1).*V(i)/10*(p_L(i+1)-p_L(i-1))./(theta(i+1)-theta(i-1));
%
```

```
%    HRR(i)=k(i)/(k(i)-1).*p_L(i)/10*(V(i+1)-V(i-1))./(theta(i+1)-theta(i-1))+...
%
1/(k(i)-1).*V(i)/10*(p_L(i+1)-p_L(i-1))./(theta(i+1)-theta(i-1))-(Q(i))/(10*omega
);
%
% end
% p_r_max=max(diff(p_L(IVC:EVO))./diff(theta(IVC:EVO)));%[bar/°]
%
% p_L(EVO:end-1)=p_out_mot;
%
% p_L(end)=p_in_mot;
%
% L_i=cumtrapz(V,p_L)/10;%[J]
%
% imep_clc=(L_i(end)/V_u*10); %[bar]
%
%
imep_h=cumtrapz(V(180*d_theta:540*d_theta),p_L(180*d_theta:540*d_theta))/V_u; %[b
ar]
% imep_h=(imep_h(end)/1.4);
%
% pmep=imep_h-imep_clc; %[bar]
%
% p_max_clc=max(p_L);
%
%
p_v=1.0595-0.0094.*p_max_clc+-0.0062.*(n_rec/1000)+3.4038*10^-5.*(n_rec/1000).^2;
                                                    %1.0595-0.0094.*p_max_
clc+-0.2602.*(n_rec/1000)+0.0601.*(n_rec/1000).^2;
%
% bmep_clc=imep_clc-p_v;
%
% r_ceff=V(IVC)/V(720*d_theta);
%
% Q_tot=-cumtrapz(theta(IVC:EVO),Q(IVC:EVO));
%
% int_HRR=cumtrapz(theta(IVC:EVO-1),(HRR(IVC:EVO-1)));
% int_NHRR=cumtrapz(theta(IVC:EVO-1),(NHRR(IVC:EVO-1)));
%
% max_HRR=m_b_kg*max(Multiwiebe)*Hi*ones(EVO-1,1)';
%
% figure
% plot(theta,p_L,'b',theta_p_max_rec*10,p_max_rec,'r*','linewidth',1.2)
% title('3000 rpm x 15 bar');
% xlabel('theta [deg*10]');
% ylabel('P [bar]');
% axis([IVC EVO 0 (p_max_rec*1.1)])
% grid on
% % hold on
% % plot(theta,p_o,'r')
% % hold on
% % plot(theta,p_motored,'g')
%
% % figure
% %
plot(theta(IVC:EVO-1),int_HRR,'b',theta(IVC:EVO-1),int_NHRR,'r',theta(IVC:EVO-1),
max_HRR(IVC:EVO-1),'y')
% % xlabel('theta [deg/10]');
% % ylabel('int HRR - int NHRR [J]');
% % axis([teta0_first*10 EVO min(NHRR) max(HRR)])
% % grid on
```

```
% for(i=2:EVO-1)
%
%  if(i<360*d_theta)
%
%  k(i+1)=m_comp;
%
%  else
%
%  k(i+1)=m_esp;
%
%  end
% end
%
% mfb_calc=int_HRR/(m_b_kg*Hi);
%
% err05=999999;
% err10=999999;
% err50=999999;
% err90=999999;
%
% for i=1:length(mfb_calc)
%
%  delta05=abs(mfb_calc(i)-0.05);
%  delta10=abs(mfb_calc(i)-0.10);
%  delta50=abs(mfb_calc(i)-0.50);
%  delta90=abs(mfb_calc(i)-0.90);
%
%  if delta05<err05
%  err05=delta05;
%  indice05=i;
%  end
%
%  if delta10<err10
%  err10=delta10;
%  indice10=i;
%  end
%
%  if delta50<err50
%  err50=delta50;
%  indice50=i;
%  end
%  if delta90<err90
%  err90=delta90;
%  indice90=i;
%  end
% end
% mfb_calc_index=(IVC+[indice05,indice10,indice50,indice90])/d_theta-360+1.47;
%
% % figure
% % plot(NHRR)
% % title('HRR - NHRR [J/°]');
% % xlabel('theta [deg/10]');
% % ylabel('HRR - NHRR [J/°]');
% % axis([teta0_first*10 EVO min(NHRR) max(HRR)])
% % grid on
% % hold on
% % plot(HRR)
%
% end
```