HGS user manual

Universitat Politècnica de Catalunya

Manel Soria, Caleb Fuster, Arnau Miró

HGS 2.1 - Manual version 1.0 - November 2021

Contents

1 Introduction

HGS is an open source Thermochemistry calculator for mixtures of ideal gases at high temperature, intended to be used in the context of rocket engines combustion chambers and nozzles. Be aware that liquid rocket engines also have a propellant feed system, where the assumption of ideal gas is not (in general) realistic at all. In this case, NFP code[1] has to be used.

This is a user manual of HGS and only an overview of the Thermodynamics behind the calculations is given. For a more detailed information, please check [1][2][3].

HGS is encoded in Matlab and the reader of this manual is assumed to be familiar with Matlab.

2 Summary of the Thermodynamics expressions used by HGS

2.1 Single species properties

HGS is based on the *Third Millennium Ideal Gas and Condensed Phase Thermochemical Database for Combustion with Updates from Active Thermochemical Tables* [4], usually known as Burcat's coefficients or the NASA 7 coefficients polynomial.

For each of the species, Burcat Database has 7 coefficients that here will be denoted by a_1 to a_7. They allow to obtain c_p, h and s values as a function of the temperature. They are an alternative to tabulated data from JANAF and other sources [5]. Burcat's coefficients are more restricted than the 9 NASA polynomials [4], as their limits are up to 6000K while 9 coefficients are up to 20000K.

2.1.1 Constant Pressure Specific Heat

The Burcat coefficients are defined so that c_p can be calculated as:

$$\frac{c_p}{R} = a_1 + a_2 \cdot T + a_3 \cdot T^2 + a_4 \cdot T^3 + a_5 \cdot T^4 \tag{1}$$

Where R is the universal gas constant, $R = 8.3144621 \times 10^{-3} \frac{\mathrm{kJ}}{\mathrm{mol*K}}$.

2.1.2 Enthalpy

HGS is based on absolute enthalpies, that allow to evaluate reaction enthalpy changes in a very convenient way. It has two terms: formation enthalpy and enthalpy variation due to the temperature:

$$\begin{aligned} dh &= c_p \cdot dT \\ h &= \int c_p \cdot dT + C \end{aligned} \tag{2}$$

$$h = \Delta h_f^0 + \int_{298}^{T} c_p \cdot dT \tag{3}$$

The coefficient a_6 expresses the reference the enthalpy, $a_6 \cdot R = \Delta h_f^0$, so the final expression is:

$$\frac{h}{R} = a_6 + a_1 \cdot T + \frac{a_2 \cdot T^2}{2} + \frac{a_3 \cdot T^3}{3} + \frac{a_4 \cdot T^4}{4} + \frac{a_5 \cdot T^5}{5} \tag{4}$$

2.1.3 Entropy

Entropy is calculated from the integration of the second Gibbs' equation,

$$T \cdot ds = dh - \nu \, dP \tag{5}$$

For the case of an ideal gas, $v = \frac{RT}{P}$ and $dh = c_p dT$, so we have:

[1]Formerly known as INIST

$$
\begin{aligned}
ds &= \frac{dh}{T} - \frac{v}{T} \cdot dP \\
ds &= \frac{c_p}{T} \cdot dT - \frac{R}{P} \cdot dP \\
\Delta s &= \int \frac{c_p}{T} \cdot dT - \int \frac{R}{P} \cdot dP
\end{aligned}
\tag{6}
$$

Two terms are distinguished in the previous equation, the temperature contribution and the pressure contribution. The contribution of the temperature follows the same procedure of the enthalpy, a formation term and the integrated term referenced to 298 K:

$$\Delta s_T = \Delta s_f^0 + \int_{298}^{T} c_p \cdot dT \tag{7}$$

In this case, the seventh coefficient is used as $a_7 \cdot R = \Delta S_f^0$:

$$\frac{s^0}{R} = a_7 + a_1 \cdot ln(T) + a_2 \cdot T + \frac{a_3 \cdot T^2}{2} + \frac{a_4 \cdot T^3}{3} + \frac{a_5 \cdot T^4}{4} \tag{8}$$

The pressure contribution is:

$$\Delta s_p = \int_{P_{ref}}^{p} \frac{R}{P} dP = R \cdot ln(\frac{P}{P_{ref}}) \tag{9}$$

And finally, the gas entropy calculated as:

$$s = \Delta s_T - \Delta s_p \tag{10}$$

2.1.4 Gibbs' Free Energy

Once the enthalpy and the entropy are available, Gibbs' free energy is calculated from its definition:

$$g = h - T \cdot s \tag{11}$$

2.2 Specific Gas Constant

The specific gas constant is obtained as:

$$R_g = \frac{R}{M_m} \tag{12}$$

The database stores the molecular mass of each species.

2.3 Specific Gas Constant

The specific gas constant R_g, different for each species but independent of the temperature is widely used. It is obtained as:

$$R_g = \frac{R}{M_m} \tag{13}$$

2.3.1 Constant Volume Specific Heat and Heat Capacity Ratio

First, c_v is obtained from c_p (at a given temperature) and Mayer relation,

$$R_g = c_p - c_v \tag{14}$$

Then, and heat capacity ratio γ is obtained as:

$$\gamma = \frac{c_p}{c_v} \tag{15}$$

2.4 Sound speed

Once γ and R_g are available, the sound speed is obtained as:

$$a = \sqrt{\gamma R_g T} \tag{16}$$

2.5 Mixtures properties

Now that we have seen how can the properties for a single species be obtained, we turn our attention to gas mixtures. Be aware that `HGS` evaluates the total (and not specific) extensive properties of a mixture, such as H instead of h.

There are different ways to define the composition of a mixture:

- Mass
- Mass fraction
- Molar concentration
- Molar fraction

They are defined as follows:

The mass (m): of a mixture is the summation of each one of the species.

$$m = \sum_{i=1}^{N} m_i \tag{17}$$

The mass fraction (mf): is the ratio of the mass of a species to the mass of the mixture.

$$mf_i = \frac{m_i}{\sum_{i=1}^{N} m_i} \tag{18}$$

The molar concentration (n): of a mixture is the summation of the mols of each species.

$$n = \sum_{i=1}^{N} n_i \tag{19}$$

The molar fraction (y): is the ratio of the mols of a species to the mols of the mixture.

$$y_i = \frac{n_i}{\sum_{i=1}^{N} n_i} \tag{20}$$

2.6 Evaluation of mixture extensive properties

We have to consider two types of properties: **Intensive properties**, like density or speed of sound do not change when the amount of substance varies; and **extensive properties** which vary proportionally when the substance amount changes, such as volume or enthalpy. For the case of a generic extensive property Φ, if we assume an ideal mixture (as we can do in the case of gases), we can simply write,

$$\Phi_T = \frac{\sum_{i=1}^{n} n_i \cdot \phi_i}{\sum_{i=1}^{n} n_i} = \sum_{i=1}^{n} \frac{n_i}{n} \cdot \phi_i = \sum_{i=1}^{n} y_i \cdot \phi_i \tag{21}$$

However, entropy (and thus Gibbs' free energy) require to consider the partial pressure of each species. This is done using Dalton's law that is enunciated as: "the total pressure exerted is equal to the sum of the partial pressures of the individual gases". So the entropy of each species is evaluated at P_i [1]:

$$P_i = P \cdot \frac{n_i}{n_t} \tag{22}$$

Where P is the total pressure and P_i are the partial pressures.

2.6.1 $\mathbf{C_p}$ and $\mathbf{C_v}$ of the mixture

Note that unlike in the case of single species, HGS evaluates the total C_p and C_v, in kJ/K, defined so that for instance $dH = C_p dT$. Note that lowercase c_p is used for the specific properties, $kJ/molK$.

For clarity, lets obtain the expression of C_p

$$H = \Sigma n_i h_i \tag{23}$$

$$dH = \Sigma n_i dh_i = \Sigma n_i c_{p_i} dT \tag{24}$$

$$C_p = \frac{dH}{dT} = \Sigma n_i c_{p_i} \tag{25}$$

The expression for C_v is equivalent.

2.7 Evaluation of mixture intensive properties

Lets turn our attention to the mixture intensive properties.

2.7.1 Average molecular mass of the mixture

The average molecular mass is total mass of the mixture divided by the number of molecules in the mixture (expressed as mol). This is,

$$M_m = \frac{\Sigma_{i=1}^{n} M_{m_i} n_i}{\Sigma_{i=1}^{n} n_i} \tag{26}$$

2.7.2 $\mathbf{R_g}$, γ and sound speed of the mixture

Once the averaged molecular mass and the specific heats of the mixture have been obtained, R_g, γ can be easily obtained from their definitions.

For the case of the sound speed, the expression to be used is equivalent to the expression for a single species.

3 Main HGS functions

3.1 HGSsingle

Purpose. This is the basic function of HGS and it just calculates one property of one species.

```
[Res] = HGSsingle(species,property,T,P)

HGSsingle returns the property of a species. Usually, the name of the
    species is used (as a string) but it is also possible to use its
    code. The later feature reduces the execution time. See HGSid.

Inputs:

species : String or numbers of species

property : Property requested (see below)

T : [K] Temperature

P : [bar] Pressure

Outputs:

Res : Property to be calculated:

Mm [g/mol]

cp [kJ/(mol*K)]

cv [kJ/(mol*K)]

h [kJ/mol]

s [kJ/(mol*K)]

g [kJ/mol]
```

Example 1. Calculate the specific enthalpy of H_2 at 400K, 10bar, and at the same temperature but 100bar, in $kJ/molK$

```
>> HGSsingle('H2','h',400,10)
ans =
     2.9650
>> HGSsingle('H2','h',400,100)
ans =
     2.9650
```

Note that, as expected for an ideal gas, the enthalpy does not change with the pressure.

Example 2. Calculate the specific enthalpy of H_2, O_2 and H_2 at 400K, 10bar in $kJ/molK$

```
>>  HGSsingle('H2','h',400,10)
ans =
    2.9650
>> HGSsingle('O2','h',400,10)
ans =
    3.0270
>> HGSsingle('H2O','h',400,10)
ans =
 -238.3721
```

As was discussed in the introduction and can be seen now, `HGS` returns absolute enthalpies.

Example 3. Evaluate and plot the specific enthalpy of C_4H_{10} and CO_2 in the range from 300K to 4000K.

```
clear
close all
T=linspace(300,4000,200);
for  i=1:numel(T)
    cpC4H10(i)=HGSsingle('C4H10','cp',T(i),1);
    cpCO2(i)=HGSsingle('CO2','cp',T(i),1);

end
plot(T,cpC4H10,'LineWidth',2)
hold on
plot(T,cpCO2,'LineWidth',2)
legend({'cp C4H10','cp CO2'})
set(findall(gcf,'-property','FontSize'),'FontSize',14)
xlabel('T (K)')
ylabel('cp kJ/molK')
```

You can see the resulting plot in figure 1.

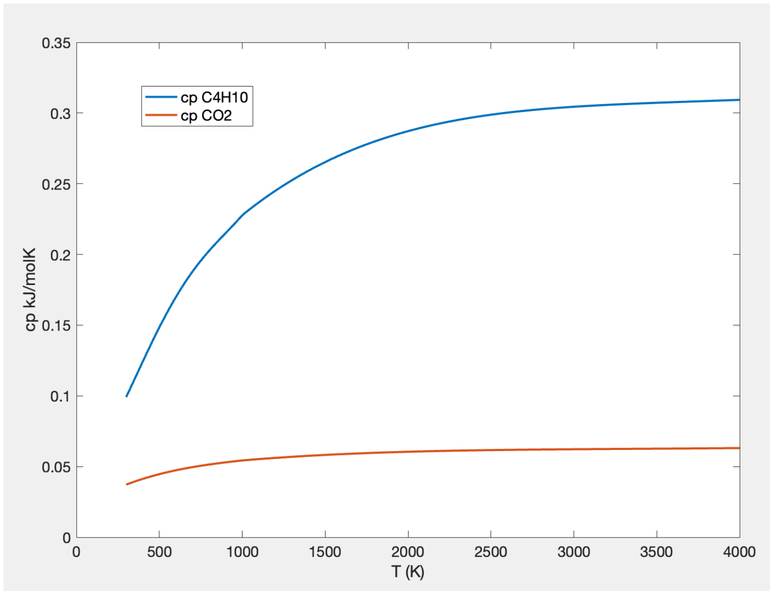


Figure 1: Example: c_p versus temperature plot, with HGSsingle.

Note that `HGSsingle` can not be called in vector form, in its present version a loop is needed.

Exercise 1

Plot c_p heat in $kJ/molK$ for linear hydrocarbons C_nH_{2n+1} with n from 1 to 5, in the range from 300K to 4000K .

3.2 `HGSfind`

HGSfind allows to search the database of species.

```
HGSfind(name,complete)

HGSfind finds the species and the mixtures that contain the string name.

Inputs:
name : String to be found
complete : Prints the complete name if complete = 1

Outputs:
Nothing
```

Example 4. Find the species containing the string 'C4H10'.

```
>> HGSfind('C4H10',0)
Species that contain C4H10
<1152>   C4H10
<1153>   C4H10(II)
<1154>   C4H10FO2P
<1155>   C4H10N2
<1156>   C4H10O
<1157>   2-C4H10O
<1158>   C4H10O(II)
<1159>   T-C4H10O
<1160>   C4H10O(III)
<1161>   n-C4H10O2
<1162>   C4H10O2
<1163>   C4H10O2(II)
<1164>   C4H10O3
<1165>   C4H10O4
<1166>   C4H10S
<1167>   C4H10S(II)
Mixtures that contain C4H10
None
```

Example 5. Find the species containing the string 'C4H10' and display additional information.

```
Species that contain C4H10
<1152>  C4H10[n-butane]
<1153>  C4H10[isobutane]
<1154>  C4H10FO2P[Sarin]
<1155>  C4H10N2[1,4-PIPE]
<1156>  C4H10O[n-butanol]
<1157>  2-C4H10O[2-Butan]
<1158>  C4H10O[Isobutanol]
<1159>  T-C4H10O[T-Butan]
<1160>  C4H10O[DEE Eth]
<1161>  n-C4H10O2[n-]
<1162>  C4H10O2[t-]
<1163>  C4H10O2[1,4 b.diol]
<1164>  C4H10O3[Ethyle]
<1165>  C4H10O4[Erythritol]
<1166>  C4H10S[C4H9SH]
<1167>  C4H10S[(C2H5)2S]
Mixtures that contain C4H10
None
```

The number before each species is the code of the species, that can be used with `HGSsingle` (among other functions) to save some computing time. This feature is designed for advanced users.

As you will see, there are many exotic species; the `HGS` data base contains all the Burcat (ref) list. Do not worry too much if you are not familiar with most of them. In case of doubt, an internet search can be useful. For instance,

`C4H10N2[1,4-PIPE]`

turns out to be Piperazine, a molecule of interest in Biology but probably not so much in Thermochemistry.

3.3 HGSprops

HGSprops is the probably the most frequently used HGS function, as it is called by many other functions. It calculates a group of properties of a given mixture, at certain temperature and pressure. The properties (for instance, entropy) are not specific but total. The mixture is specified by means of a list of strings with the names of the species and a vector with the number of mol of each of them.

```
[varargout] = HGSprop(species,n,T,P,varargin)
      e.g. [Mm,Cp,Cv,H,S,G,Rg,gamma,a] = HGSprop({'H2' 'O2'},[1 2],300,1)
           [H,S] = HGSprop({'H2' 'O2'},[1 2],300,1,'H','S')
           [S,H] = HGSprop({'H2' 'O2'},[1 2],300,1,'S','H')

  HGSprop returns the properties of a mixture of gasses

  Inputs:

  species : String or code of species
  n : [mols] Number of mols per species
  T : [K] Temperature
  P : [bar] Pressure
  varargin : Expected return: 'Mm' 'Cp' 'Cv' 'H' 'S' 'G' 'Rg' 'gamma' 'a'
  If it is empty, all the properties will be returned.

  Outputs:

  varargout :  Property result
     Mm [g/mol]
     Cp [kJ/K]
     Cv [kJ/K]
     H [kJ]
     S [kJ/K]
     G [kJ]
     Rg [kJ/(kg*K)]
     gamma
     a [m/s]
```

Example 6. Find the properties of a mixture of 1 mol of H_2 and 1 mol of O_2 at 300K and 1 bar.

```
> [Mm,Cp,Cv,H,S,G,Rg,gamma,a] = HGSprop({'H2' 'O2'},[1 2],300,1)
Mm =
    22.0040
Cp =
     0.0876
Cv =
     0.0627
H =
     0.1621
S =
     0.5574
G =
  -167.0566
Rg =
     0.3779
gamma =
     1.3979
a =
   398.0787
```

This is the basic usage of the function, where all the properties are evaluated.

Example 7. It is also possible to calculate only some of the properties:

```
>> [H,G,S] = HGSprop({'H2' 'O2'},[1 2],300,1,'H','G','S')
H =
    0.1621
G =
  -167.0566
S =
    0.5574
```

Example 8. Note that, for extensive properties such as H, G or S, `HGSprop` returns the **total** property and not the specific property. Try this and compare with the previous call:

```
>> [H,G,S] = HGSprop({'H2' 'O2'},2*[1 2],300,1,'H','G','S')
H =
    0.3242
G =
  -334.1132
S =
    1.1148
```

The same holds for C_p and C_v, but not for the molecular mass, R_g, γ and the sound speed.

Exercise 2

Plot γ for a gas mixture 80-20 of N_2-O_2 in the range 300K to 4000K

3.4 `HGSrebuild`

`HGS` provides support for mixtures of gases that are two be considered as a single component, such as it is sometimes done with air. There are two built in mixtures:

- RP1 surrogate [3], a mixture of 61.25% $C_{10}H_{22}$ and 38.75% $C_{10}H_{18}$ (in number of mol).
- Air7921, a mixture of 70% of N_2 and 21% of O_2 (in number of mol).

HGSrebuild breaks down a mixture into a its components.

```
[species,n0] = HGSrebuild(species,n0)

HGSrebuild transform the mixtures in to their species. It also transforms
the mols of the mixtures with the proportions assigned.

Inputs:

species : String or numbers of species
n0 : [mols] Mols

Outputs:

species : String of species
n0 : [mols] Mols
```

Example 9. Find out the number of mol the individual species present in 10 mol of RP1 surrogate.

```
>> [sp,n]=HGSrebuild('RP1',10)
sp =
  1x2 cell array
    {'C10H22'}    {'C10H18'}
n =
    6.1250    3.8750
```

Example 10. Obtain the composition in mol of each species and the enthalpy in kJ of two mol of RP1 surrogate at 300K and 10 bar.

```
>> [sp,n]=HGSrebuild('RP1',2)
sp =
  1x2 cell array
    {'C10H22'}    {'C10H18'}
n =
    1.2250    0.7750
>> HGSprop({'RP1'}, [2],300,10,'H')
ans =
  -397.9173
```

Note that mixtures can not be used by `HGSsingle`.

3.5 HGSeq

Up to now, the `HGS` functions discussed have been useful but perhaps not as much as to justify the existence of `HGS`. `HGSeq` is the first function that performs a complex calculation: the equilibrium state of a mixture of species at a given pressure and temperature. It is carried out by direct minimization of Gibbs energy.

`HGSeq` is the workhorse of many algorithms of interest for analysis of rocket engines, that use `HGSeq` as a function.

```
[species,n,deltaG] = HGSeq(species,n,T,P,options)

\texttt{HGSeq} calculates the species quantities in equilibrium
   conditions at a certain temperature and pressure

Inputs:

species : String or code of species
n0 : [mol] Initial mixture
T : [K] Temperature.
P : [bar] Pressure
options : Options for the fmincon (optional)

Outputs:

species : Species
n : [mol] Final mixture
Gmin : [kJ] Minimum Gibbs free energy
```

Example 11. Two mol of H_2 and 1 mol of O_2 are in a vessel at the constant conditions of 2000K and 1 bar. Find their equilibrium state and the total Gibbs energy in equilibrium, knowing that the species involved are H_2, O_2, H_2O, H, O and OH.

```
>> [sp,n,G]=HGSeq({'H2','O2','H2O','H','O','OH'},[2;1;0;0;0;0],2000,1)
sp =
  1x6 cell array
    {'H2'}    {'O2'}    {'H2O'}    {'H'}    {'O'}    {'OH'}
n =
    0.0118
    0.0047
    1.9857
    0.0003
    0.0001
    0.0048
G =
  -1.3976e+03
```

Example 12. Note that it is immediate to obtain the rest of the properties of the mixture calling `HGSprop` at the equilibrium pressure and temperature:

```
>> [sp,n,G]=HGSeq({'H2','O2','H2O','H','O','OH'},[2;1;0;0;0;0],2000,1);
>> [Mm,Cp,Cv,H,S,G,Rg,gamma,a] = HGSprop(sp,n,2000,1)
Mm =
   17.9494
Cp =
    0.1034
Cv =
    0.0867
H =
 -333.7075
S =
    0.5320
G =
  -1.3976e+03
Rg =
    0.4632
gamma =
    1.1925
a =
   1.0511e+03
```

Example 13. The initial quantities are arbitrary (this is, they should not be the stoichiometric reaction quantities). Actually, they can even be what you would consider products (as in the second function call in the next example box). Note also that if some function outputs are not wanted, you can ask Matlab to omit them.

```
>> [~,n,~]=HGSeq({'H2','O2','H2O','H','O','OH'},[10;1;0;0;0;0],2000,1)
n =
    7.9929
    0.0000
    1.9996
    0.0145
    0.0000
    0.0004
>> [~,n,~]=HGSeq({'H2','O2','H2O','H','O','OH'},[0;0;10;0;0;1],4000,20)
n =
    2.7226
    0.8871
    5.2609
    2.2870
    1.2189
    2.7460
```

Example 14. Note how the composition and thus the properties of the mixture change with the temperature. At 400K, water is water but at 4000K it has been decomposed and the mixture of species has significantly different properties.

```
>> clear all
>> P=10;
>> sp={'H2','O2','H2O','H','O','OH'};
>> [~,n,~]=HGSeq(sp,[0;0;10;0;0;0],400,P);
>> [Mm,~,~,~,~,~,Rg,gamma,~] = HGSprop(sp,n,400,P)
Mm =
   18.0150
Rg =
    0.4615
gamma =
    1.3204
>> [~,n,~]=HGSeq(sp,[0;0;10;0;0;0],4000,P);
>> [Mm,~,~,~,~,~,Rg,gamma,~] = HGSprop(sp,n,4000,P)
Mm =
   11.5280
Rg =
    0.7212
gamma =
    1.2781
```

Exercise 3

Find the equilibrium state of a stoichiometric mixture of H_2 and O_2 from 400K to 4000K and 1 bar. Plot the molar fraction of each species (see previous examples) and in a separate plot γ. In the γ plot, draw three lines at P=1, P=10 and P=100 bar.

Exercise 4

Find the equilibrium state of a stoichiometric mixture of one mol of CH_4 and air (consider 7.52 mol of N_2 per mol of O_2), from 400K to 4000K and 1 bar. Consider the following species: CH_4, H_2O, CO_2, CO, O_2, H_2, OH, H, O, NO, N_2. Plot the total enthalpy of the mixture versus the temperature.

3.6 `HGStp`

`HGStp` (standing for "temperature of the products") finds the combustion temperature, with dissociation, assuming thermodynamic equilibrium state.

As we have seen, with `HGSeq` we can find the equilibrium state of a gas mixture at constant T and P, as a function of T and P. In the case of a rocket combustion chamber in steady state, we can assume that the temperature and pressure are kept constant and that the equilibrium is reached instantaneously. The pressure is usually a design parameter that is known. The rocket feed system has to be able to provide the needed flow rate of propellants at a higher pressure, to be able to maintain the chamber pressure, considering the pressure losses at the injectors.

However, the temperature is a consequence of the reaction itself and therefore it is now known. To find it, a heat balance in the combustion chamber has to be carried out. Assume for a moment that it is adiabatic (later we will see how to proceed in case it is not adiabatic). In this case, according to the first principle in open systems, the sum of the enthalpies of the reactants (that is known) has to be equal to the sum of the enthalpies of the products (that can be computed as a function of their temperature). The products are considered to be in equilibrium state.

So the algorithm would be: First, the enthalpy of the reactants is evaluated. Then, for a given temperature, we can find the outlet composition (`HGSeq`) and then its enthalpy (`HGSprop`). Only the outlet temperature that yields the correct enthalpy will be the combustion chamber temperature.

```
[Tp,n,species,flag] = HGStp(species,n0,type,V0,P,options)

HGStp calculates the reaction temperature considering
dissociation, and the products composition in equilibrium

Inputs:

species : String or code of species
n0 : [mols] Number of mols of each species.
      Usually, the number of mol of the products will be zero)
type : Entry type that defines the state of the input.
        It can be 'T' or 'H'
V0 : Value of type : 'T' V0=T [K] input temperature
                   : 'H' V0=H [kJ] input enthalpy
P : [bar] Pressure
options : (OPTIONAL) Structure with the options for the secant method.
                .xmin [K] Temperature minimum for the solver;
                .xmax [K] Temperature maximum for the solver;
                .maxiter Max iterations for the solver;
                .epsx Diferential T where the solver reaches the solution;
                .epsy Diferential S where the solver reaches the solution;
                .fchange T difference where secant method is
                          changed by bisection method;
                .type Select between: 'Frozen' for frozen flow
                                      'Shifting' for shifting flow
                .info Detailed info == 1; No info == 0.
                .dTp Improve the velocity with the approximation of
                parabola. +- dTp
         For instance, by default:
         options=struct('xmin',300,'xmax',6000,'maxiter',50,...
              'epsx',0.1,'epsy',0.5,'fchange',5,...
              'type','Shifting','info',0,'dTp',100)

Outputs:

Tp : [K] Exit temperature
n : [mols] Species resultant mols
species : String or code of species
flag : Solver error detection:
                1  Solver has reached the solution
               -1  Solver failed. Maximum iterations
               -2  Solver failed. Initial sign change not found
```

Example 15. 1 mol/s of CH_4 at 298K reacts with air in stoichiometric quantity (one mol/s of O_2 and 7.52 mol/s of N_2) in an adiabatic chamber. Find the temperature of the reaction products, considering dissociation. The species to be considered are: CH_4, H_2O, CO_2, CO, O_2, H_2, OH, H, O, NO, N_2.

```
>> sp={'CH4','H2O','CO2','CO','O2','H2','OH','H','O','NO','N2'};
>> ni=[1;0;0;0;2;0;0;0;0;0;7.52];
>> [Tp,np]=HGStp(sp,ni,'T',298,1);
>> Tp
Tp =
    2.2238e+03
```

Example 16. 1 mol/s of CH_4 at 298K reacts with air in stoichiometric quantity (one mol/s of O_2 and 7.52 mol/s of N_2) in an adiabatic chamber. Find the temperature of the reaction products, considering dissociation. The species to be considered are: CH_4, H_2O, CO_2, CO, O_2, H_2, OH, H, O, NO, N_2.

```
>> sp={'CH4','H2O','CO2','CO','O2','H2','OH','H','O','NO','N2'};
>> ni=[1;0;0;0;2;0;0;0;0;0;7.52];
>> [Tp,np]=HGStp(sp,ni,'T',298,1);
>> Tp
Tp =
    2.2238e+03
```

Example 17. 2 mol of H_2 and 1 mol of O_2 react in an adiabatic combustion chamber. They enter as saturated liquids at the pressure of 10 bar, and it is known that in this state their total enthalpy (in the reference of `HGS`) is -26.3367 kJ (we will see how can this value be obtained with NFP). Find their reaction temperature, considering dissociation, and compare it with the reaction temperature that would be obtained if both were in gas state at 300K.

```
>> species={'H2','O2' , 'H2O','H','O','OH'};
>> nr=[2;1;0;0;0;0];
>> P=10;
>> HinLIQ=-26.3367;
>> HGStp(species,nr,'H',HinLIQ,P)
ans =
    3.3349e+03
>> HGStp(species,nr,'T',300,P)
ans =
    3.3875e+03
```

Exercise 5

H_2 and O_2 enter a reaction chamber at 300K and 80bar. Considering dissociation, plot the temperature of the reaction products as a function of the OF ratio, defined as the ratio between the mass of O_2 and the mass of H_2.

Exercise 6

Repeat the previous example with CH_4 and O_2.

Exercise 7

If the heat lost by the combustion chamber (kW) is known, or it is a function of the combustion temperature, can `HGStp` be used to find the outlet temperature? How?

3.7 HGSisentropic

```
[Tp, species ,n,V2, flag ] = HGSisentropic ( species ,n0 ,T0,P0, Fro_Shift ,
                                              typeexit ,V1, options1 , options2 )

HGSisentropic calculates the outlet variables for an isentropic expansion
beginning with a very low velocity .

Inputs:
species : String or code of species
n0 : [mols] Number of mols of each species
T0 : [K] Initial temperature
P0 : [bar] Inlet pressure
Fro_Shift : Select between: 'Frozen' for frozen flow
                            'Shifting' for shifting flow
typeexit : Entry type that defines the state of the input.
           It can be 'P' or 'M'
V1 : Value for type: 'P'   V1=P [bar] output pressure
                     'M'   V1=M [] output Mach. Has to be >=1
options1 : (optional) Structure with the options to be passed to
    HGSeqcond
              that solves for the temperature with HGSsecant.
              .xmin [K] Temperature minimum for the solver;
              .xmax [K] Temperature maximum for the solver;
              .maxiter Max iterations for the solver;
              .epsx Diferential T where the solver reachs the solution;
              .epsy Diferential S where the solver reachs the solution;
              .fchange T difference where secant method is
                       changed by bisection method;
              .maxrange Max range to fit in a parabola
              .info Detailed info == 1; No info == 0.
              .dTp Improve the velocity with the approximation of
              parabola. +- dTp
        struct('xmin',300,'xmax',4000,'maxiter',50,'epsx',0.1,'epsy',0.5,
                'fchange',5,'maxrange',1500,'info',0,'dTp',100)
options2 : (optional) Structure with the options for HGSsecant to be
               called here to solve for the Pressure
        struct('xmin',0.01,'xmax',<P0,'maxiter',50,'epsx',0.01,'epsy',0.01
                'fchange',1,'info',0)

Outputs:
Tp : [K] Exit temperature
species : String or code of species
n : [mols] Species c mols
V2 : If typeexit 'P' ->[Mach] Mach of the mixture at a P
     If typeexit 'M' ->[bar] Pressure at the Mach
flag : Solver error detection:
              1  Solver has reached the solution
             -1  Solver failed. Maximum iterations in T loop
             -2  Solver failed. Initial sign change not found in T loop
             Only for typeexit=='M'
             -3  Solver failed. Maximum iterations in P loop
             -4  Solver failed. Initial sign change not found in P loop
```

`HGSisentropic` evaluates the isentropic expansion of a gas mixture in a nozzle, beginning from a state of very low (neglectable) velocity. During the expansion, the flow can be considered ”shifting” or ”frozen”.

- Shifting means that the equilibrium is assumed to be reached instantaneously at each nozzle point. Thus, for each point the composition of the mixture changes.
- Frozen means that the composition is considered constant along the nozzle. However, the properties of the mixture change with the temperature (so this is not to be confused with a perfect gas model, where the properties remain constant during the expansion).

There are two options:

- The expansion proceeds until the desired pressure is reached.
- The expansion proceeds until the desired Mach number is reached.

In both cases, a secant method is used to solve the non-linear equation. This is to avoid using derivatives, that cause numerical problems when the function being derived (`HGSeq`) is evaluated with fsolve.

Example 18. A mixture of 5 mol/s of H_2 and 1 mol/s of O_2 enter a combustion chamber at 300K. The combustion chamber pressure is 80bar.

1. Find the temperature and composition of the reaction products (assuming that the chamber is adiabatic).

```
clear
close all
species = {'H2', 'O2', 'H2O', 'OH', 'O', 'H'};
n=zeros(length(species),1);
n(1)=5; % mol/s
n(2)=1; % mol/s
Tin=300; % K
Pchamber=80; % bar

% Combustion chamber:
[Tcomb,~, ncomb] = HGStp(species,n,'T', Tin, Pchamber);
```

2. Isentropic expansion case 1. Assuming shifting flow, find the Mach number at the exit of the nozzle if the exit pressure is 0.5 bar.

```
% Assuming Shifting flow:
% Expansion up to 0.5 bar
[Texit1,~,nexit1,Mexit1,flag1] =
    HGSisentropic(species,ncomb,Tcomb,Pchamber,'Shifting','P',0.5)
Texit1 =
  910.8030
nexit1 =
    3.0000
    0.0000
    2.0000
    0.0000
    0.0000
    0.0000
Mexit1 =
    3.9181
flag1 =
     1
```

3. Isentropic expansion case 2. Assuming shifting flow, find the pressure and temperature at the exit of the nozzle if the Mach exit is 4.

```
% Expansion up to Mach 4
[Texit2,~,nexit2,Pexit2,flag2] =
    HGSisentropic(species,ncomb,Tcomb,Pchamber,'Shifting','M',4)
Texit2 =
  883.7606
nexit2 =
    3.0000
    0.0000
    2.0000
    0.0000
    0.0000
    0.0000
Pexit2 =
    0.4234
flag2 =
     1
```

Note how the exit pressure and temperature are lower than in the previous case.

4. Isentropic expansion case 3. Assuming frozen flow, find the Mach number at the exit of the nozzle if the exit pressure is 0.5 bar.

```
% Assuming Frozen flow:
% Expansion up to 0.5 bar
[Texit3,~,nexit3,Mexit3,flag3] =
    HGSisentropic(species,ncomb,Tcomb,Pchamber,'Frozen','P',0.5)

Texit3 =
   1.9254e+03
nexit3 =
    2.9866
    0.0000
    1.9931
    0.0068
    0.0000
    0.0337
Mexit3 =
    1.9080
flag3 =
     1
```

5. Isentropic expansion case 4. Assuming frozen flow, find the pressure at the exit of the nozzle if the Mach exit is 4.

```
% Expansion up to Mach 4
[Texit4,~,nexit4,Pexit4,flag4] =
    HGSisentropic(species,ncomb,Tcomb,Pchamber,'Frozen','M',4,[])
Texit4 =
     []
nexit4 =
     []
Pexit4 =
     []
flag4 =
    -4
```

In this case, the solver has failed to produce a result. Remember to check the exit flag before accepting a result as correct!. The solution will be to use a starting pressure closer to the actual value:

```
% We use a better intial guess for the pressure P=xmax=0.5
% and also turn information on (info=1) to see the iterative process
opt2=struct('xmin',0.01,'xmax',0.5,'maxiter',50,...
         'epsx',0.01,'epsy',0.01,'fchange',1,'info',1);
[Texit4,~,nexit4,Pexit4,flag4] =
    HGSisentropic(species,ncomb,Tcomb,Pchamber,'Frozen','M',4,[],opt2)

ii=1 x1=1.000000e-02 y1=-2.814260e+00 xc=3.102752e-01
    yc=1.527887e+00 x2=5.000000e-01 y2=1.778152e+00
ii=2 x1=1.000000e-02 y1=-2.814260e+00 xc=1.601376e-01
    yc=1.151035e+00 x2=3.102752e-01 y2=1.527887e+00
ii=3 x1=1.000000e-02 y1=-2.814260e+00 xc=8.506880e-02
    yc=7.391283e-01 x2=1.601376e-01 y2=1.151035e+00
ii=4 x1=1.000000e-02 y1=-2.814260e+00 xc=4.753440e-02
    yc=2.871752e-01 x2=8.506880e-02 y2=7.391283e-01
ii=5 x1=1.000000e-02 y1=-2.814260e+00 xc=2.876720e-02
    yc=-1.976270e-01 x2=4.753440e-02 y2=2.871752e-01
ii=6 x1=2.876720e-02 y1=-1.976270e-01 xc=3.815080e-02
    yc=8.792537e-02 x2=4.753440e-02 y2=2.871752e-01
Texit4 =
  892.9014
nexit4 =
    2.9866
    0.0000
    1.9931
    0.0068
    0.0000
    0.0337
Pexit4 =
    0.0382
flag4 =
     1
```

3.8 HGSnozzle

`HGSnozzle` returns the state of the flow along the nozzle, assuming an isentropic expansion. It is based on `HGSisentropic`, and it is intended to be used to study and plot different flow parameters as a function of the local pressure.

```
  [species,n,T,v,M,A,F,Isp] = HGSnozzle(species,n0,T0,P0,Pe,Pa,Fro_Shift,
                                        options1,options2)

HGSnozzle evaluates different flow properties as a function of the
pressure, during a isentropic expansion beginning with a very low
velocity

Inputs:
species : String or code of inlet species
n0 : [mols] Number of mols/s of each inlet species
T0 : [K] Inlet temperature
P0 : [bar] Inlet pressure
P :  [bar] Pressure vector (with all the pressures that have to be
       evaluated)
Pa : [bar] Atmospheric pressure
Fro_Shift : Select between: 'Frozen' for frozen flow
                            'Shifting' for shifting flow

Outputs:
species : String or code of species
n : [mols] Matrix of pecies mols, sorted as: n(species, pressure)
T : [K] Exit temperature
M : [] Exit Mch
A : [m^2] Exit area
F : [N] Thrust
Isp : [s^]Specific impulse, g0 = 9.807 m/s^2
```

Example 19. From the known conditions at the nozzle inlet of a Vinci engine [REF], operating with H_2 and O_2, and a range of pressures from 10 bar below the chamber pressure to 0.01 bar, calculate as a function of the pressure:

- The mol/s of each of the species.
- The temperaure.
- The velocity.
- The Mach number.
- The nozzle area.
- the Thrust (if the nozzle ended at the local pressure).
- the Specif impulse (if the nozzle ended at the local pressure).

Assume that the engine operates in the vacuum.

```
% The inlet conditions to combustion chamber of the Vinci engine are:
species={'H2','O2','H2O','OH','O','H'};
n0 = [2875.496,1052.566,0,0,0,0]; % mol/s (obtained from kg/s)
P0=62;
Pa=0; % Vacuum
Hin=-19.0609; % kJ (liquid inlet, obtained with NFP)

[Tp,~,np,~] = HGStp(species,n0,'H',Hin,P0);

% Nozzle expansion
% We generate a vector with the pressure points to be obtained
% From 10 bar below the chamber pressure to 0.01 in 2 diferent steps
% 10 bar is arbitrary, we just want to make sure that we begin before the
% throat but far from the inlet, where the velocity would be 0 and the
    area
% infinite.
P = [P0-10:-2:1, linspace(1,0.01,20)];
[species,n,T,v,M,A,F,Isp] = HGSnozzle(species,np,Tp,P0,P,Pa,'Shifting');
```

In the example **Vinci Rocket nozzle study** (Section 5.5) you will find more information about how to post-process the results obtained here.

4 Advanced and auxiliary `HGS` functions

4.1 Low level Thermodynamic properties evaluation functions

The functions `HGSh`, `HGSs`, `HGScp` and `HGScv` are called by `HGSsingle` and `HGSprops` to perform the calculation of enthalpy, entropy, constant pressure specific heat and constant volume specific heat. In all cases, the inputs are the species and its temperature (and pressure in the case of the entropy).

These are not functions that you will normally call, but please check the source code if you want to know the details.

4.2 `HGSr`

This function just defines R, the universal gas constant, as a global variable.

4.3 `HGSid`

You may want to call this function to accelerate the code, in case you have to call many times `HGSprops` with the same species.

```
[id] = HGSid(species)

HGSid finds the id of the species to improve the velocity of the code

Inputs:
species : String or code of species

Outputs:
id : Species code
```

Example 20. For an arbitrary mixture of gases, compare the execution times using HGSprop with the species names and with the codes. Check that the results are the same.

```
clear
close all
sp={'C4H10','CH4','N2','CO2','O2'};
n=[1,1,1,1,1];

id=HGSid(sp);

tic
for i=1:10000
    [H,G,S]=HGSprop(sp,n,400,10,'H','G','S');
end
H
G
S
toc

tic
for i=1:10000
    [H,G,S]=HGSprop(id,n,400,10,'H','G','S');
end
H
G
S
toc
```

(The previous code is saved as testHGSspeed).

```
>> testHGSspeed
H =
  -568.6965
G =
   -1.0288e+03
S =
     1.1504
Elapsed time is 3.408228 seconds.
H =
  -568.6965
G =
   -1.0288e+03
S =
     1.1504
Elapsed time is 1.406984 seconds.
```

Thus, a significant amount of time is spent looking for the codes.

4.4 HGSprintInfo

Sometimes you require to know the complete data from a determinate species. This functions allows to know the name, composition, state, molar mass (only for species) and the id.

Example 21. Printing a Mixture (Air7921), a species with symbolic composition name (Mg2SiO4(l)) and a species with a non-symbolic composition name (RDX).

```
>> HGSprintInfo('Air7921')
Combination = <Air7921>   <code = 2968>
   Composition— N2: 79.00%.  O2: 21.00%.
       ——————————————————
>> HGSprintInfo('Mg2SiO4(l)')
Species = Mg2SiO4(l),    state = C,   <code = 2381>
   Composition: Mg—2  Si—1  O—4
  Mm = 140.6910
       ——————————————————
>> HGSprintInfo('RDX')
Species = RDX,    state = G,   <code = 947>
   Composition: C—3  H—6  N—6  O—6
  Mm = 222.1170
       ——————————————————
```

4.5 HGSelements

`HGSelements` searches the `HGS` database and returns the elements that contains a species and their number.

```
HGSelements(name)

HGSelements returns the elements present in the species name and their
number

Inputs:
name: Name or code of the species

Outputs:
eln: List with the name of the elements present in the species
elq: Vector with their quantities
————————————————————————————————————————————————————————————————————————
Examples:
[a,b]=HGSelements('H2O');
[a,b]=HGSelements(2247);
```

4.6 HGSparamFmincon

```
[A,b,Aeq,beq,lb,ub] = HGSparamFmincon(id,n0)

HGSparamFmincon returns the parameters to be used by HGSeq for fmincon
    solver

Inputs:

id : Id of species
n0 : [mol] Species mols

Outputs:

A : fmincon A inequality matrix A*x < b
b : fmincon b inequality vector A*x < b
A : fmincon Aeq equality matrix Aeq*x = beq
b : fmincon beq equality vector Aeq*x = beq
lb : fmincon lb lower boundary limit
ub : fmincon ub upper boundary limit
```

`HGSeq` function is based on direct minimization of Gibbs energy. `HGSparamFmincon` is called by `HGSeq` to create the constrain parameters for the Matlab function `fmincon`. `fmincon` allows to minimize a non-linear function subject to linear and non-linear constraints. To implement the stoichiometric constraints, `HGS` only uses two of the `fmincon` restrictions: lower boundary and linear equality constraint.

The returns of `HGSparamFmincon` are:

- A from $Ax <= b$, in this case empty.
- b from $Ax <= b$, in this case empty.
- Aeq from $A_{eq}x = b_{eq}$, from mass conservation restriction.
- beq from $A_{eq}x = b_{eq}$, from mass conservation restriction.
- lb lower boundary, in this case 0 for all the species.
- ub upper boundary, in this case Inf for all the species.

The equality constrain is used to set the mass conservation of each of the elements. To do so, a matrix A_{eq} is created from the list of species. Each column is the species and each row, the vector of elements it contains. For instance, in a mixture of H_2, O_2, H_2O, H, O, OH, the matrix (with a column at the right hand side to express the element associated to each row) is:

$$A_{eq} = \left[\begin{array}{cccccc} 2 & 0 & 2 & 1 & 0 & 1 \\ 0 & 2 & 1 & 0 & 1 & 1 \end{array}\right] \left|\begin{array}{c} H \\ O \end{array}\right. \tag{27}$$

The `HGS` database has 2 fields to store the elements (.ena) and the number of it (.nat) for each species. This information, that can be easily retrieved with the function `HGSelements`, is used to assemble A_{eq}.

Constinuing with the previous mixture, if 3 mol of H_2 and 5 mol of O_2 are initially present in the mixture, the stoichiometric restriction will be:

$$A_{eq}n = b_{eq} \tag{28}$$

Where n is a vector column with the number of mol of each species. In our case,

$$n = \begin{bmatrix} 3 \\ 5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{29}$$

and b_{eq} is obtained as:

$$b_{eq} = A_{eq} n = \begin{bmatrix} 6 \\ 10 \end{bmatrix} \left| \begin{matrix} H \\ O \end{matrix} \right.$$

Exercise 8

Using `HGSelements`, write a code to prepare the stoichiometric restriction (A_{eq} and b_{eq}) for a mixture of CH_4, H_2O, CO_2, CO, O_2, H_2, OH, H, O, NO, N_2, containing one mol of CH_4, two mol of O_2 and three mol of N_2.

4.7 HGSsecant

`HGSsecant` is an solver used to find zeros of functions of one variable, using a method based in the secant algorithm. This is needed to avoid computing numerical derivatives of functions calculated with Newton or similar algorithms (as used in `fsolve`), that can have numerical noise. `HGSsecant` is not a general purpose algorithm, as it expects that the function being solved is `HGSeq`.

```
[Tp,n,flag] = HGSsecant(f,n0,options)
HGSsecant solves a function using a combination of the secant method and
bisection method

Inputs:

f : Function
n0 : [mol] Initial mixture
options : Structure with the options for the secant method.
              .xmin [K] Temperature minimum for the solver;
              .xmax [K] Temperature maximum for the solver;
              .maxiter Max iterations for the solver;
              .epsx Diferential T where the solver reaches the solution;
              .epsy Diferential S where the solver reaches the solution;
              .fchange T difference where secant method is
                       changed by bisection method;
              .maxrange Max range to fit in a parabola
              .info Detailed info == 1; No info == 0.
              .dTp Improve the velocity with the approximation of
              parabola. +- dTp
        struct('xmin',300,'xmax',6000,'maxiter',200,'epsx',0.1,'epsy',
          1,'fchange',5,'info',0,'dTp',100,'maxrange',1500)

Outputs:

Tp : [K] Final temperature
n : [mol] Final mixture
flag : Solver error detection:
              1  Solver has reached the solution
             -1  Solver failed. Maximum iterations
             -2  Solver failed. Initial sign change not found
```

4.8 HGSmendeliev

```
[Mendeleiev] = HGSmendeliev
HGSmendeliev returns an array with the molar mass of each element
Periodic table data: https://www.ptable.com/?lang=es
Inputs:
Outputs:
Mendeleiev : [g/mol] Molar mass of each substance of the periodic table
```

4.9 HGSresetDataBase

This function downloads the `HGS` data base, calling `HGSdataDownload`, and then adds the mixtures. Currently, two mixtures are supported: RP1 and Air7921 (see section3.4).

4.10 HGSdataDownload

This rather long function downloads the original Burcat database and parses it to form the `HGS` database. The location of the database is hardcoded as http://garfield.chem.elte.hu/Burcat/THERM.DAT.

4.11 HGSaddMixture and HGSsubtMixture

These functions -not designed to be called by the end user- allow to create or remove mixtures from the database. By default, the database contains mixtures an air and RP-1 surrogate but these functions allow the users to manage their own mixtures.

Example 22. Create another air surrogate with more species than N_2 and O_2 and delete it.

```
HGSfind('Another')
Species that contain Another
None
Mixtures that contain Another
None
>> HGSaddMixture('AnotherAirSurrogate',{'N2','O2','Ar','CO2'},...
                    [78.054,20.6,0.934,0.412])
>> HGSfind('Another')
Species that contain Another
None
Mixtures that contain Another
<2969>  AnotherAirSurrogate
>> HGSsubtMixture('AnotherAirSurrogate')
>> HGSfind('Another')
Species that contain Another
None
Mixtures that contain Another
None
```

5 HGS additional examples

The purpose of this section is to present application examples that combine different functions. Some of them are based on the examples included in the HGS source code but not necessarily in the same order.

5.1 Combustion without dissociation

Example 23. A mixture of 0.1 kg/s de CH_4 and double of the air needed for a stoichiometric reaction enter a combustion chamber at Tr=400K. Assuming that there is no dissociation of the products and that the combustion chamber is adiabatic, determine the outlet temperature Tp. Use a 79% N_2 - 21% O_2 composition for air.

The stoichiometric reaction with oxygen would be:

$$CH_4 + 2O_2 \longrightarrow CO_2 + 2H_2O \tag{30}$$

If we double the quantity of oxygen, the reaction can be expressed as:

$$CH_4 + 4O_2 \longrightarrow CO_2 + 2H_2O + 2O_2 \tag{31}$$

However, if we use air, for each molecule of O2 we carry 79/21 molecules of N2: If we double the quantity of oxygen, the reaction can be expressed as:

$$CH_4 + 4\left(O_2 + \frac{79}{21}\right) \longrightarrow CO_2 + 2H_2O + 2O_2 + 4\frac{79}{21}N_2 \tag{32}$$

It is useful to express the reaction with the stoichiometric indices representing the flow rates of the different substances in mol/s. To do so, we convert the 0.1 kg/s of CH4 to 6.25 mol/s and multiply the previous reaction by 6.25:

$$6.25CH_4 + 25\left(O_2 + \frac{79}{21}\right) \longrightarrow 6.25CO_2 + 12.5H_2O + 12.5O_2 + 25\frac{79}{21}N_2 \tag{33}$$

Next, we treat each of the species as a separate inlet or outlet to the chamber. For the inlets, the temperature is known, while the outlet temperature is obtained from an energy balance:

$$\Sigma \dot{n}_i h_i = \Sigma \dot{n}_o h_o \tag{34}$$

First, we define a set of helper function that return the enthalpy of the species involved:

```
clear;

Tr=400;  % K
p=1;     % not relevant

% Specific enthalpy (h) of each species as a function of T:

hCH4=@(T) HGSsingle('CH4','h',T,p);

hO2=@(T)  HGSsingle('O2','h',T,p);

hN2=@(T)  HGSsingle('N2','h',T,p);

hH2O=@(T) HGSsingle('H2O','h',T,p);

hCO2=@(T) HGSsingle('CO2','h',T,p);
```

Then, we express our energy balance equation 34 as an equation and we solve it with 1000K as our first guess:

```
% Combustion equation to solve: sum(hreactives) - sum(hproducts) = 0


eq=@ (Tp) 6.25*hCH4(Tr) ...
          + 25*hO2(Tr)...
          + 25*(79/21)*hN2(Tr)...
          - 12.5*hH2O(Tp)...
          - 12.5*hO2(Tp)...
          - 25*(79/21)*hN2(Tp)...
          - 6.25*hCO2(Tp);


[Tp,fval,exitflag]=fsolve(eq,1000);
fprintf('Adiabatic flame temperature: %.3f K \n',Tp);
```

```
Adiabatic flame temperature: 1562.086 K
```

Exercise 9

Find the reaction temperature in the previous example, but considering dissociation.

Exercise 10

Continuing with the same example (without dissociation), find the heat that has to be rejected so that the fumes are at 450K.

Exercise 11

Write a function that accepts as inputs a string with the name of an arbitrary species (containing only H and C) and a inlet temperature. Using `HGSelements`, the function will have to generate a energy balance equation and find the reaction temperature, assuming that there is no dissociation.

5.2 H_2O_2 as gas generator

Example 24. H_2O_2 at high concentration has been used a gas generator in different applications such as the turbopump of the V2 (the first liquid engine rocket) and also is a candidate to replace hydrazine in space propulsion. With a catalyzer, it decomposes exothermically into O_2 and H_2O:

$$H_2O_2 \longrightarrow H_2O + \frac{1}{2}O_2 \tag{35}$$

If H_2O_2 enters at 300K in gaseous state in a catalytic reactor (adiabatic), determine the temperature of the products, assuming that there is no dissociation.

This problem is similar to the previous:

```
clear

Tr=300;           % K
p=1;              % not relevant

% Enthalpy of each species
hH2O2=@(T) HGSsingle('H2O2','h',T,p);
hO2=@(T)   HGSsingle('O2','h',T,p);
hH2O=@(T)  HGSsingle('H2O','h',T,p);

% Equation to be solved
eq=@(Tp) hH2O2(Tr) -...
          (hH2O(Tp)+0.5*hO2(Tp));

[Tp,fval,exitflag]=fzero(eq,3000);

fprintf('Temperature of the products: %.2f K \n',Tp);
```

```
Temperature of the products: 2047.82 K
```

Exercise 12

Solve the previous example considering the dissociation of H_2O and O_2 and assuming that the species are in equilibrium.

5.3 Understanding HGStp

Example 25. Consider a mixture of H_2, O_2, H_2O, H, O, and OH at 10 bar and 350K that initially contains two mol of H_2 and 1 mol of O_2 (and a neglectable quantity of the others species). Plot its enthalpy in equilibrium as a function of its temperature, together with the enthalpy of the initial state. Then, without using `HGStp`, find the reaction temperature if the initial mixture is in an adiabatic reaction chamber at constant pressure.

First, we plot the enthalpy:

```
species={'H2','O2','H2O','H','O','OH'};
Tr=350;                 % K
P=10;                   % bar
nr=[2;1;0;0;0;0];       % mol

Hin = HGSprop(species,nr,Tr,P,'H')

% Plot products enthalpy vs. T
T=linspace(300,4000,20);
Hout = zeros(length(T),1);
for i=1:length(T)
    [~,comp,~] = HGSeq(species,nr,T(i),P);
    Hout(i) =HGSprop(species,comp,T(i),P,'H');
end
plot(T,Hout,'-or',T,Hin*ones(length(T),1),'-b','LineWidth',2);
legend('Hout','Hin');
xlabel('Temperature (K)'); ylabel('Enthalpy (kJ/molK)');
set(gca,'FontSize',18)
```

You can see the resulting plot in figure 2.

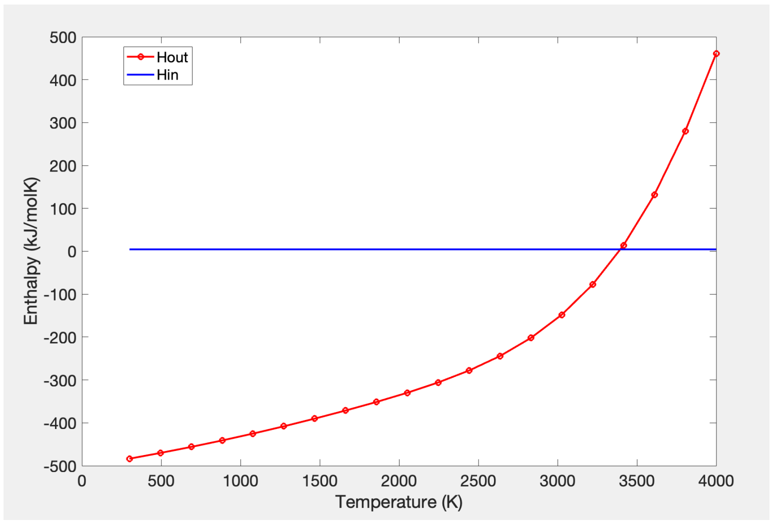


Figure 2: Example: Initial enthalpy and mixture enthalpy.

Next, we have to find the intersection point of both lines

```

% Function to be solved to find T so that deltaH=0
    function DH=deltaH(Tp)
        [~,neq,~] = HGSeq(species,nr,Tp,P); % find equilibrium
            composition
        Ho = HGSprop(species,neq,Tp,P,'H');
        DH=Ho-Hin;
    end

% examples
fprintf('deltaH @ 400 K = %.2f kJ/molK \n',deltaH(400));
fprintf('deltaH @ 4000 K = %.2f kJ/molK \n',deltaH(4000));

% solving the equation
Tflame=fzero(@deltaH,3000,optimset('Display','iter'));

fprintf('Adiabatic flame temperature Tp = %.2f K \n',Tflame);
```

Note that, before calling the solver (fzero in this case), we test the function for a couple of values, to check that the results obtained are coherent. This is a very good practice. The result obtained is:

```
Adiabatic flame temperature Tp = 3397.73 K
```

5.4 Comparison with RPA

Example 26. To ensure the correct running of `HGStp`, we compared the results of a combustion of a LH2-LOX cycle with the ones of RPA code [6]. RPA is a well-know code for analyse the combustion and expansion of species in a rocket Combustion chamber and Nozzle.

For both codes, we use the following conditions:

- LH2-LOX combustion
- o_f ratio: 4.1
- Tin: 90 K
- Pin: 45 bar
- mass of H_2: 1 kg

RPA usually shows multiple species that are negligible, to speed-up our code we reduce the species to 6 (H_2, O_2, H_2O, OH, O and H). RPA results, under this conditions, are:

Molar concentration:
$H_2 = 0.4730081$, $O_2 = 0.0000519$, $H_2O = 0.5040833$, $OH = 0.0063140$, $O = 0.0001233$, $H = 0.0164185$

To run our `HGStp`, we require the inlet enthalpy of our species, requiring the `NFP` difference. This is due to enthalpy reference of both codes are not the same.

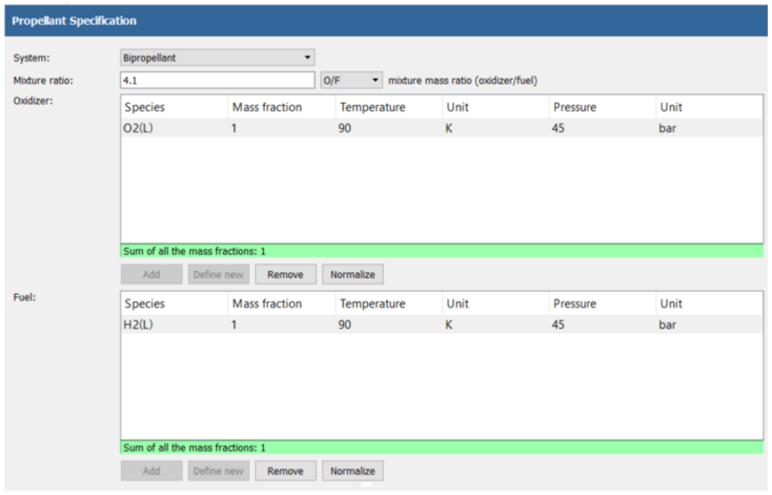
```
mH2 = 1; % [kg]
mO2 = mH2 * of_ratio; % [kg]

MmH2 = HGSsingle('H2','Mm');
MmO2 = HGSsingle('O2','Mm');

n0 = zeros(length(species),1);
n0(1) = mH2 / (MmH2/1000);
n0(2) = mO2 / (MmO2/1000);

% First of all, we need to calculate de inlet enthalpy using NFP
% (only the values) from a reference T of 300 K.
% deltaH_H2 = (INIST('H2','h_pt',45,Tref) - INIST('H2','h_pt',45,Tin)) *
    mH2
% deltaH_O2 = (INIST('O2','h_pt',45,Tref) - INIST('O2','h_pt',45,Tin)) *
    mO2

deltaH_H2 = 2.8577525044e+03 * mH2 ;
deltaH_O2 = 393.3914211 * mO2 ;

H_H2_HGS = HGSsingle('H2','h',Tref,Pin)*n(1);
H_O2_HGS = HGSsingle('O2','h',Tref,Pin)*n(2);

% Inlet enthalpy
Hin = H_H2_HGS - deltaH_H2 + H_O2_HGS - deltaH_O2;
```

With the previous calculations, we have the inputs for the functions: `Hin`, `n0` and `Pin`.

```
[Tcomb,~, ncomb] = HGStp(species,n,'H', Hin, Pin);
```

The comparative results between RPA and `HGStp` is:

```
After the combustion
     RPA: 3004.078 K              HGS: 3022.940 K
Molar fractions
H2         RPA:4.724e-01          HGS: 4.730e-01
O2         RPA:5.907e-05          HGS: 5.190e-05
H2O        RPA:5.033e-01          HGS: 5.041e-01
OH         RPA:6.776e-03          HGS: 6.314e-03
O          RPA:1.403e-04          HGS: 1.233e-04
H          RPA:1.735e-02          HGS: 1.642e-02
```



Propellant Specification

System: Bipropellant

Mixture ratio: 4.1 O/F mixture mass ratio (oxidizer/fuel)

Oxidizer:

Species	Mass fraction	Temperature	Unit	Pressure	Unit
O2(L)	1	90	K	45	bar

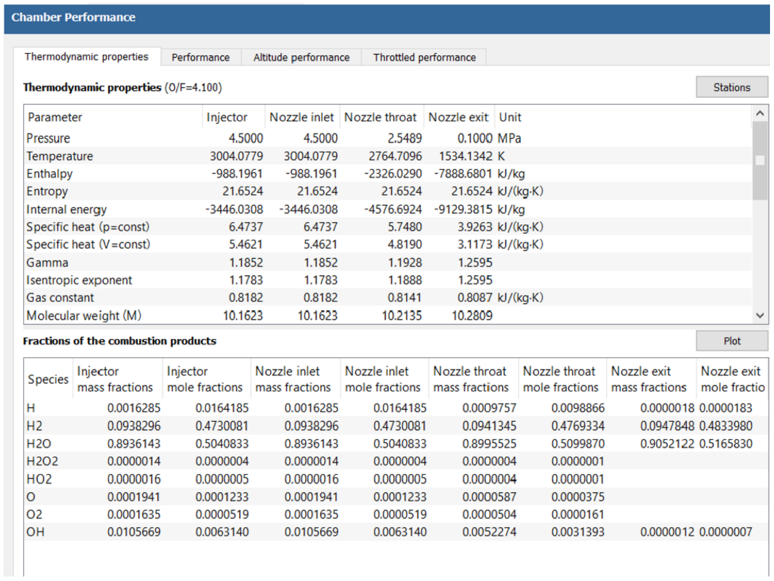
Sum of all the mass fractions: 1

Fuel:

Species	Mass fraction	Temperature	Unit	Pressure	Unit
H2(L)	1	90	K	45	bar

Sum of all the mass fractions: 1

(a) Inlet of RPA code



Chamber Performance

Thermodynamic properties (O/F=4.100)

Parameter	Injector	Nozzle inlet	Nozzle throat	Nozzle exit	Unit
Pressure	4.5000	4.5000	2.5489	0.1000	MPa
Temperature	3004.0779	3004.0779	2764.7096	1534.1342	K
Enthalpy	-988.1961	-988.1961	-2326.0290	-7888.6801	kJ/kg
Entropy	21.6524	21.6524	21.6524	21.6524	kJ/(kg·K)
Internal energy	-3446.0308	-3446.0308	-4576.6924	-9129.3815	kJ/kg
Specific heat (p=const)	6.4737	6.4737	5.7480	3.9263	kJ/(kg·K)
Specific heat (V=const)	5.4621	5.4621	4.8190	3.1173	kJ/(kg·K)
Gamma	1.1852	1.1852	1.1928	1.2595	
Isentropic exponent	1.1783	1.1783	1.1888	1.2595	
Gas constant	0.8182	0.8182	0.8141	0.8087	kJ/(kg·K)
Molecular weight (M)	10.1623	10.1623	10.2135	10.2809	

Fractions of the combustion products

Species	Injector mass fractions	Injector mole fractions	Nozzle inlet mass fractions	Nozzle inlet mole fractions	Nozzle throat mass fractions	Nozzle throat mole fractions	Nozzle exit mass fractions	Nozzle exit mole fractio
H	0.0016285	0.0164185	0.0016285	0.0164185	0.0009757	0.0098866	0.0000018	0.0000183
H2	0.0938296	0.4730081	0.0938296	0.4730081	0.0941345	0.4769334	0.0947848	0.4833980
H2O	0.8936143	0.5040833	0.8936143	0.5040833	0.8995525	0.5099870	0.9052122	0.5165830
H2O2	0.0000014	0.0000004	0.0000014	0.0000004	0.0000004	0.0000001		
HO2	0.0000016	0.0000005	0.0000016	0.0000005	0.0000004	0.0000001		
O	0.0001941	0.0001233	0.0001941	0.0001233	0.0000587	0.0000375		
O2	0.0001635	0.0000519	0.0001635	0.0000519	0.0000504	0.0000161		
OH	0.0105669	0.0063140	0.0105669	0.0063140	0.0052274	0.0031393	0.0000012	0.0000007

(b) RPA results

Figure 3: RPA calculations

5.5 Vinci Rocket nozzle study.

Example 27. The Vinci Cycle is the upper stage of the Ariane 6 [7] will be used as an example. From its known conditions at the inlet of the combustion chamber (see the first part of the code), plot the following nozzle expansion parameters, as a function of the pressure, knowing that the exit pressure is 0.01 bar:

- Temperature
- H_2O fraction.
- Velocity.
- Mach number.
- Thrust in the vacuum (if the nozzle ends at the current pressure).
- Nozzle diameter.

Calcule also:

- The throat diameter.
- The nozzle exit diameter.
- The expansion ratio.
- The thrust.

Use the shifting flow approximation.

First we obtain the nozzle inlet conditions. Note that the mol/s values have been obtained from kg/s data and that the total inlet enthalpy (in the HGS reference) has been obtained with `NFP` from the known inlet conditions (liquid). All these calculations are omited to focus our attention in the nozzle.

```
close all
clear
% The inlet conditions to combustion chamber of the Vinci engine are:
species={'H2','O2','H2O','OH','O','H'};
n0 = [2875.496,1052.566,0,0,0,0]; % mol/s (obtained from kg/s)
P0=62;
Pa=0; % Vacuum
Hin=-19.0609; % kJ (liquid inlet, obtained with NFP)

[Tp,~,np,~] = HGStp(species,n0,'H',Hin,P0);
```

Next we call `HGSnozzle`. The vector with the pressures has been obtained after some trial and error.

```
% We generate a vector with the pressure points to be obtained
% From 10 bar below the chamber pressure to 0.01 in 2 different steps
% 10 bar is arbitrary, we just want to make sure that we begin before the
% throat but far from the inlet, where the velocity would be 0 and the
   area
% infinite.
P = [P0-10:-2:1, linspace(1,0.01,20)];
[species,n,T,v,M,A,F,Isp] = HGSnozzle(species,np,Tp,P0,P,Pa,'Shifting');
```

Then we do the plots and calculations we have been asked for, beginning with the temperature:

```
% Temperature plot
figure
plot(P,T, '-b','LineWidth',2)
set ( gca, 'xdir', 'reverse')
title('T versus P')
xlabel('P [bar]')
ylabel('T [K]')
set(gca,'FontSize',18)
```

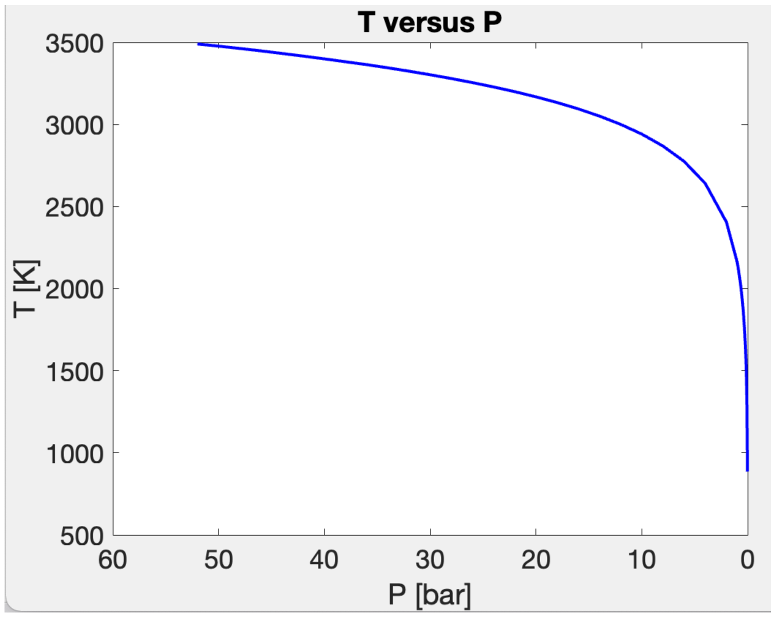


Figure 4: Vinci Nozzle: Temperature versus pressure.

Next, the H_2O molar fraction. Note that `sum(n(:,3)` is the total number of mol of each species, that depends on the pressure as the composition is shifting.

```
%H2O fraction plot
figure
hold on
plot(P,n(3,:)./sum(n(:,3)),'LineWidth',2)
set ( gca, 'xdir', 'reverse' )
title('H2O fraction versus P')
xlabel('P [bar]')
ylabel('H2O Molar fraction []')
set(gca,'FontSize',18)
```

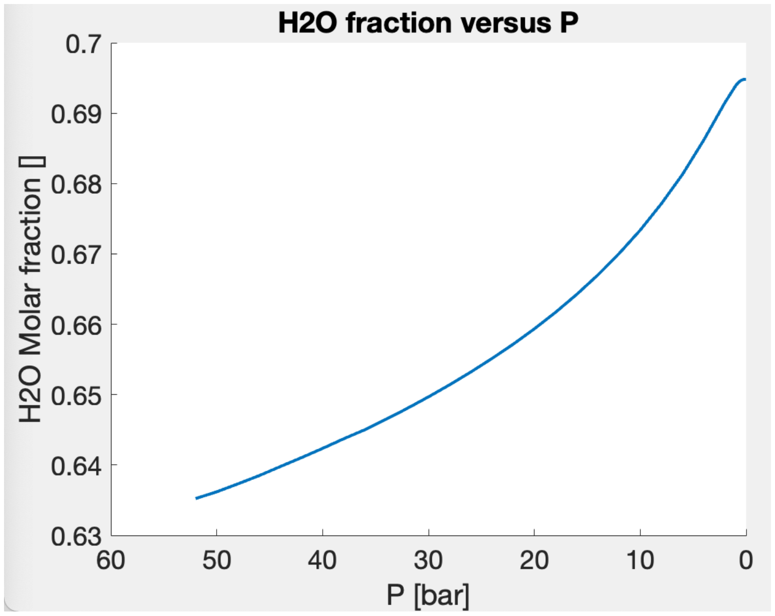


Figure 5: Vinci Nozzle: H_2O molar fraction versus pressure.

Next, the velocity:

```
%velocity plot
figure
plot(P,v,'LineWidth',2)
title('v vs P')
set ( gca, 'xdir', 'reverse' )
xlabel('P [bar]')
ylabel('v [m/s]')
set(gca,'FontSize',18)
```

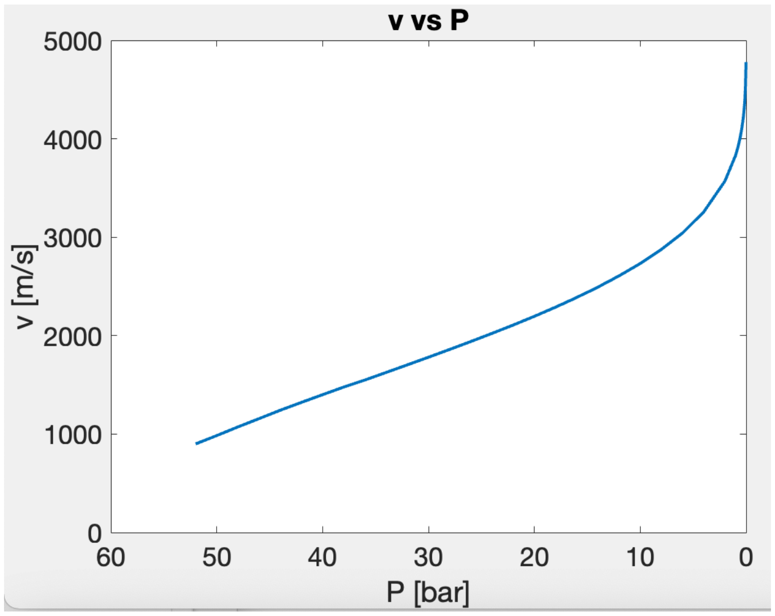


Figure 6: Vinci Nozzle: Flow velocity versus pressure.

Next, we will plot Mach and calculate the exact pressure of the throat with `HGSisentropic`:

```
% Mach plot
figure
plot(P,M,'LineWidth',2)
hold on
plot(P,ones(size(P)),'r') % Horizontal line at M=1
title('Mach vs P')
set ( gca, 'xdir', 'reverse' )
xlabel('P [bar]')
ylabel('Mach []')
set(gca,'FontSize',18)

%%
% We can find the exact throat pressure with:
[~,~,~,Pt,flag] = HGSisentropic(species,np,Tp,P0,'Shifting','M',1);
plot([Pt Pt],[0 6],'r') % vertical line at the throat
```

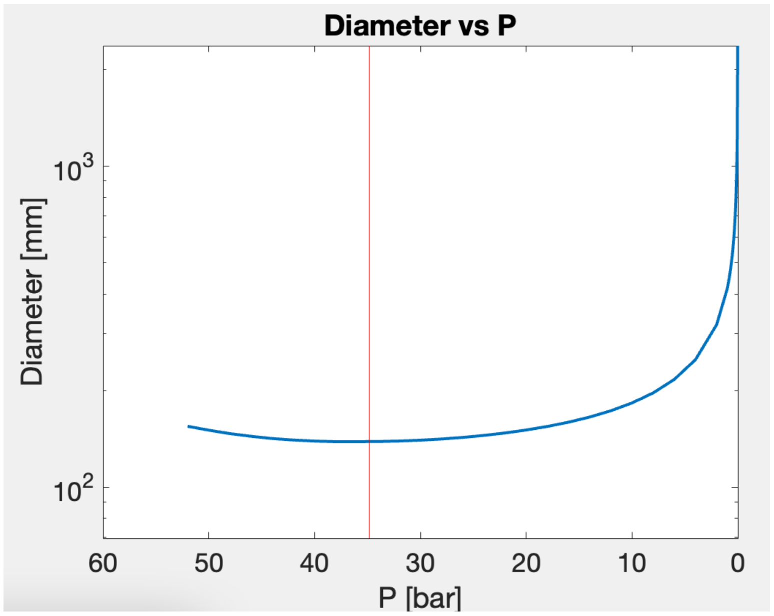


Figure 7: Vinci Nozzle: Nozzle diameter versus pressure.

Next, the thrust:

```
figure
plot(P,F,'LineWidth',2)
title('Thrust vs P')
set ( gca, 'xdir', 'reverse' )
xlabel('P [bar]')
ylabel('F [N]')
set(gca,'FontSize',18)
```

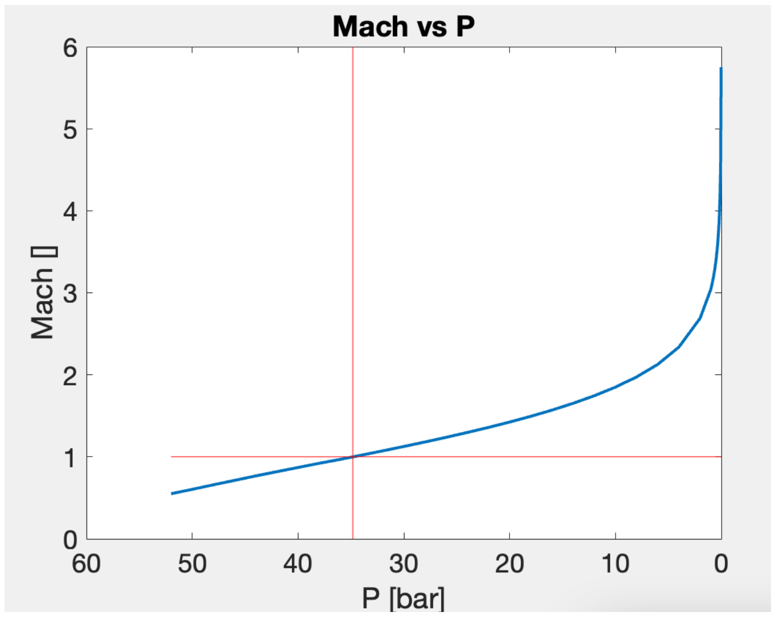


Figure 8: Vinci Nozzle: Thrust (if the nozzle ends at a given pressure) versus the pressure.

Finally, the nozzle diameter, the expansion ratio and the thrust:

```
figure
D=1000*2*sqrt(A/pi);
semilogy(P,D,'LineWidth',2)
hold on
semilogy([Pt Pt],[min(D)/2 max(D)],'r')
set ( gca, 'xdir', 'reverse' )
title('Diameter vs P')
xlabel('P [bar]')
ylabel('Diameter [mm]')
set(gca,'FontSize',18)

Dt=interp1(P0-P,D,P0-Pt); % P0-P to have x values in ascending order
De=D(end);

fprintf('Throat diameter %.2f mm\nExit diameter %.2f mm \n',Dt,De);
fprintf('Nozzle expansion ratio = %.2f \n',(De/Dt)^2);
fprintf('Thrust = %.2f kN\n',F(end)/1000);
```

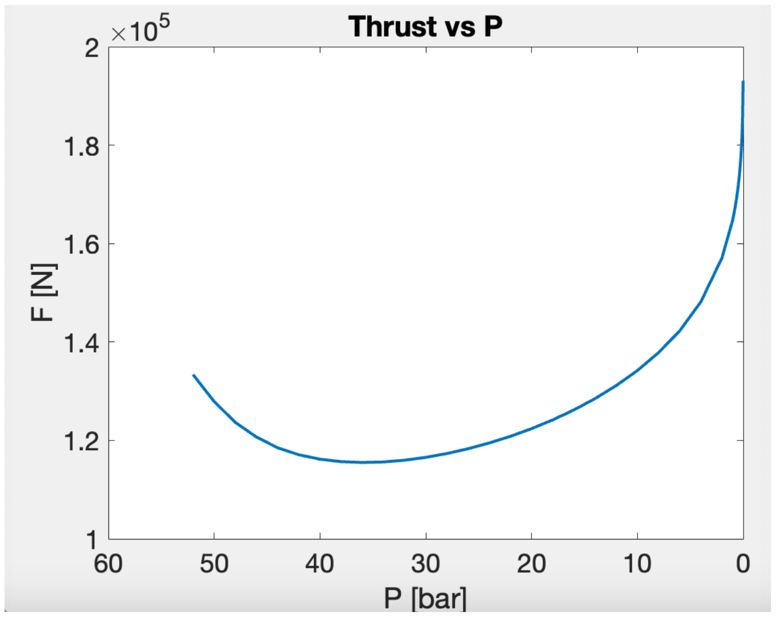


Figure 9: Vinci Nozzle: Diameter versus pressure.

And the numeric results are:

```
P = 52.000000,  1 /46
P = 50.000000,  2 /46
...
P = 0.062105,  45 /46
P = 0.010000,  46 /46
Throat diameter 138.74 mm
Exit diameter 2374.95 mm
Nozzle expansion ratio = 293.03
Thrust = 193.09 kN
```

Note that `HGSnozzle` prints the progress, as it can be quite slow.

Exercise 13

In the previous example, find the local density and check mass conservation equation.

References

[1] Yunus A. Çengel et al. *Thermodynamics: an engineering approach.* New York [etc.] : McGraw-Hill, 2006. ISBN: 9701056116.

[2] Manel Soria. *Extension of Rocket Engines class notes.* Class Notes. 2021.

[3] Caleb Fuster. *Study: Algorithms for Rocket Engine Thermochemistry.* Master Thesis. 2021. URL: `https://upcommons.upc.edu/handle/2117/353102`.

[4] A Burcat and R Branko. "Third millennium ideal gas and condensed phase thermochemical database for combustion with updates from active thermochemical tables". In: *Technical Report* ANL-05/20.September (2005), ANL–05/20 TAE 960. URL: `http://garfield.chem.elte.hu/Burcat/Introduction.pdf`.

[5] NIST. *NIST Chemistry WebBook - SRD 69.* 2017. DOI: `https://doi.org/10.18434/T4D303`. URL: `https://webbook.nist.gov/chemistry/` (visited on 11/14/2021).

[6] *RPA - Rocket Propulsion Analysis.* `https://www.rocket-propulsion.com/index.htm`. Accessed: 2021-11-11.

[7] Armin Herbertz. "Component Modeling for Rocket Engine Cycle Analysis". In: *Transactions of the Japan Society for Aeronautical and Space Sciences, Aerospace Technology Japan* 14.ists30 (2016), pp. 119–127. ISSN: 1884-0485. DOI: `10.2322/tastj.14.pa_119`.