Modeling of an ejector in Thermoptim

An ejector or injector (Figure 1) receives as input two fluids normally gaseous but which may also be liquid or two-phase (Chunnanond, 2004):

- the high pressure fluid called primary fluid or motive;
- the low pressure fluid, called secondary fluid or aspirated.

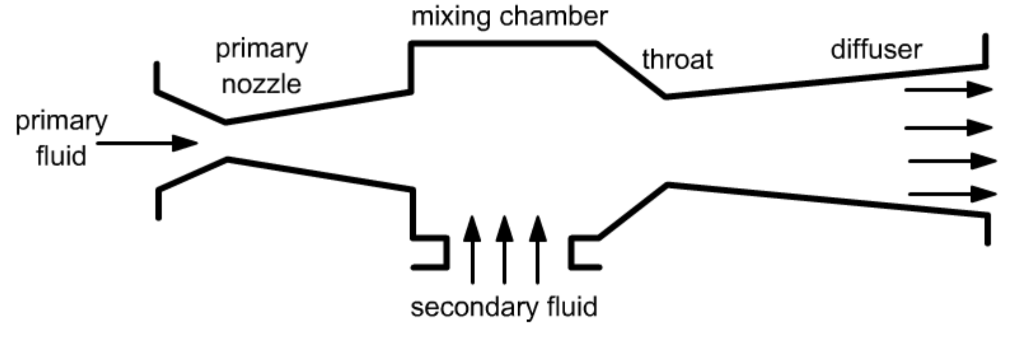


Figure 1: Cross section of an ejector

The primary fluid is accelerated in a converging-diverging nozzle, creating a pressure drop in the mixing chamber, which has the effect of drawing the secondary fluid. The two fluids are then mixed and a shock wave may take place in the following zone (throat in Figure 1). This results in an increase in pressure of the mixture and reduction of its velocity which becomes subsonic. The diffuser then converts the residual velocity into increased pressure.

The ejector thus achieves a compression of the secondary fluid at the expense of a decrease in enthalpy of the primary fluid.

The three most important parameters to characterize the overall efficiency of an ejector are:

- the **entrainment ratio w**, ratio of the secondary to primary mass flow-rates;
- the **pressure lift ratio**, the ratio of the static pressure at the outlet of the diffuser to the static pressure of the secondary fluid;
- a **section ratio** (minimal on maximum or primary flow on entrained flow etc.), which determines its geometry.

Note here that one of the practical problems encountered in the use of an ejector in a cycle is that its efficiency depends on many of its operating conditions: the compression ratio obtained is obviously a function of the entrainment ratio but a variation of the latter induces a change in the optimum geometry of the ejector, which is obviously impossible to achieve.

It follows that an ejector is poorly adapted to operation outside of design conditions.

Ejector model

Ejector modeling is mostly based on the assumption that primary and secondary fluids can be treated as ideal gases in the mixing chamber, which is roughly justified given the low pressure therein. However, there are also cases where one of these fluids is liquid or two-phase, so that the mixture can be biphasic, and several hypotheses can be selected: either carry out calculations with the properties of a real fluid while using the hypothesis of a single-phase flow, or neglect the liquid phase when computing velocities, or consider an equivalent fluid.

The calculation of the ejector (one-dimensional model) is based on the following assumptions:

- the expansion of the primary and secondary fluids in the inlet nozzle is assumed to be adiabatic, taking into account the irreversibilities by an isentropic efficiency;
- the pressure remains constant in the mixing chamber (there are constant mixing section ejectors, but they are

less efficient than others and we do not consider them here);

- when the mixed flow is supersonic, a normal shock can take place in the mixing chamber, which slows down the fluid and creates an overpressure;
- compression in the diffuser is assumed to be adiabatic, taking into account the irreversibilities by an isentropic efficiency;
- fluid properties are uniform in any section.

The model we used is that proposed by Li and Groll (2005), which presents the advantage of being formulated in a manner independent of fluid properties. We have slightly reformulated and expanded it to take account of any shock, these authors implicitly limiting themselves to cases where the mixed flow is subsonic.

In this model, we simplify the calculations assuming *a priori* that we know the pressure drop between the secondary flow suction pressure Pe and that at the inlet of the mixing zone Pb We will consider that it is proportional to Pe. It is clear that in practice, we do not know the value of this factor, which is not directly measurable. If we want to set a value of the output pressure ejector Pd or pressure lift ratio Pd/Pe, we must then iterate on the value of this factor, which is easy to do once the model is set and validated.

The model assumes that the ejector is comprised of 4 main areas:

- the motive flow expansion area;
- the entrained flow expansion area;
- the mixing zone, with possible shock;
- the diffuser.

Setting equations

m_{mi} and m_{si} being the mass-flows of motive and secondary streams, the entrainment ratio is:

$$w = \frac{m_{si}}{m_{mi} + m_{si}}$$

Motive flow expansion area

We will use the index mb to characterize the outlet state of the fluid, which expands adiabatically:

$$s_{mb,is} = s_{mi}$$

$$h_{mb,is} = f(s_{mi}, P_b)$$

$$h_{mb} = h_{mi} - \eta_s (h_{mi} - h_{mb,is})$$

The fluid velocity at the inlet being negligible, its output value is:

$$C_{mb} = \sqrt{2 (h_{mi} - h_{mb})}$$

The specific volume v_{mb} allows you to know the flow area per unit of total mass flow rate:

$$a_{mb} = \frac{v_{mb}}{C_{mb} (1 + w)}$$

Entrained flow expansion area

We will use index sb to characterize the outlet state of the fluid, which expands adiabatically:

$$s_{sb,is} = s_{si}$$

$$h_{sb,is} = f(s_{si}, P_b)$$

$$h_{sb} = h_{si} - \eta_s (h_{si} - h_{sb,is})$$

The fluid velocity at the inlet being negligible, its output value is:

$$C_{sb}=\sqrt{2\ (h_{si} - h_{sb})}$$

The specific volume v_{sb} allows you to know the flow area per unit of total mass flow rate:

$$a_{sb} = \frac{v_{sb}}{C_{sb}} \frac{w}{(1 + w)}$$

Mixing zone

We will use index mix to characterize the mix output state, which can only be determined iteratively by solving simultaneously in pressure the conservation equations of mass flow, momentum and enthalpy.

Solving these equations shows that there may be one or two mixing pressure(s). The physical explanation is as follows: if the mixed flow is supersonic, a recompression with "normal" shock wave takes place because the outlet pressure is greater than that of the flow. The two mathematical solutions correspond to pressures before and after the shock.

The conservation of momentum reads:

$$P_b\ (a_{mb} + a_{sb}) + \frac{C_{mb}}{(1 + w)} + \frac{w\ C_{sb}}{(1 + w)} = P_{mix}\ (a_{mb}+ a_{sb}) + C_{mix}$$

The enthalpy equation writes:

$$h_{mi} + w\ h_{si} = (1 + w)\ (\ h_{mix} + \frac{C_{mix}^2}{2}\ )$$

The conservation of the mass flow writes:

$$v_{mix} = (a_{mb} + a_{sb})\ C_{mix}$$

We will retain as a solution, if there are two, the highest pressure, which is equivalent to taking the lowest, and solve the equations of the shock wave below. This way is more precise, the following equations being strictly speaking valid only for perfect gases.

Schock wave

If there is a shock wave, the equations are as follows, x being the inlet (supersonic), and y the outlet (subsonic):

$$M_y^2 = \frac{M_x^2 + \frac{2}{\gamma - 1}}{\frac{2\ \gamma}{\gamma - 1} M_x^2 - 1}$$

$$\frac{P_y}{P_x} = \frac{1 + \gamma\ M_x^2}{1 + \gamma\ M_y^2}$$

$$\frac{T_y}{T_x} = \left(\frac{P_y}{P_x}\right)^2 \left(\frac{M_y}{M_x}\right)^2$$

References

Li, D., Groll A., Transcritical CO2 refrigeration cycle with ejector-expansion device, International Journal of Refrigeration 28 (2005) 766–773

Modeling of an ejector in Thermoptim

An ejector acts as a mixer receiving two fluids: the primary flow and the secondary flow, and out of which exits the recompressed total flow.

Study of the external class FluidEjector

Consistency tests are performed by the node method checkConsistency() to check if the connected fluids are those expected.

Please refer to Volume 3 of the reference manual for explanations on this point, valid for all external nodes.

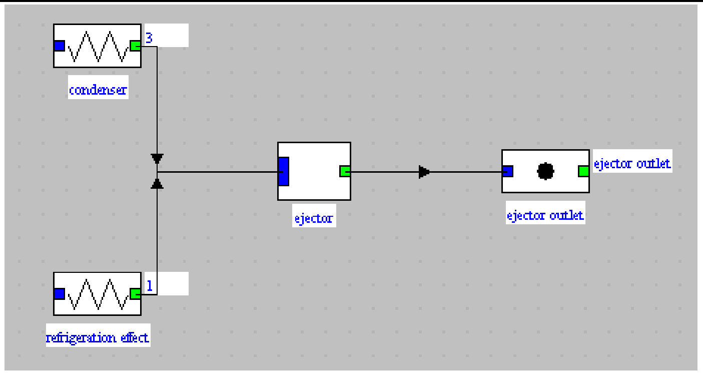


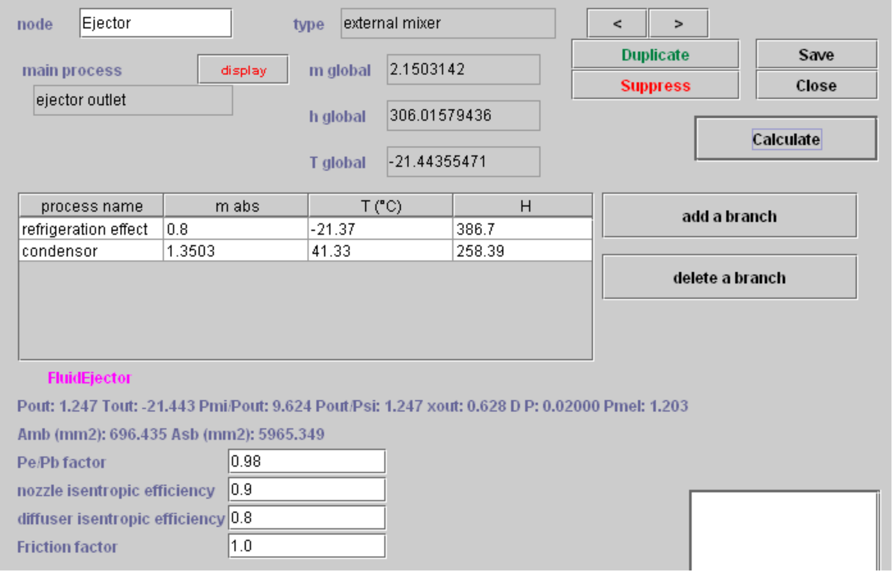
Figure 2: Diagram of the ejector component

The screen of the ejector is shown in Figure 3. It has four parameters:

- The Pe/Pb pressure drop factor at the entrance of the secondary fluid in the ejector, which determines the minimum pressure in the ejector
- The isentropic efficiency of the two nozzles (motive fluid and aspirated fluid)
- The isentropic efficiency of the exit diffuser
- The friction factor to take into account a possible pressure drops in the mixing zone.

The calculation results are displayed above the parameter input fields. The first line indicates:

- Pout, outlet pressure (bar)
- Tout, outlet temperature (°C)
- Pmi/Pout, ratio of the pressure of the motive fluid to the outlet pressure
- Pout/Psi, report of the outlet pressure to the aspirated fluid pressure
- Xout, output quality
- DP, initial pressure drop of the entrained flow
- Pmel, mixture pressure (bar)



node Ejector
type external mixer
main process ejector outlet
m global 2.1503142
h global 306.01579436
T global -21.44355471

process name	m abs	T (°C)	H
refrigeration effect	0.8	-21.37	386.7
condensor	1.3503	41.33	258.39

FluidEjector

Pout: 1.247 Tout: -21.443 Pmi/Pout: 9.624 Pout/Psi: 1.247 xout: 0.628 D P: 0.02000 Pmel: 1.203

Amb (mm2): 696.435 Asb (mm2): 5965.349

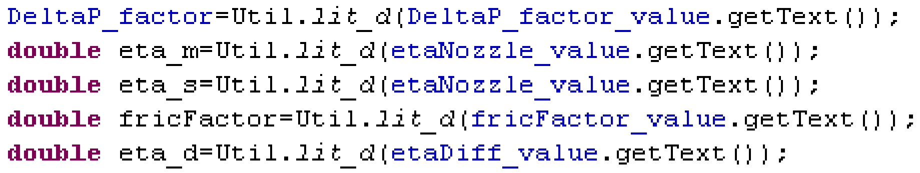
Pe/Pb factor 0.98
nozzle isentropic efficiency 0.9
diffuser isentropic efficiency 0.8
Friction factor 1.0

Figure 2: Screen of the ejector component

The second line provides the sections (mm^2) at the entrance of the mixing zone for the motive fluid (Amb) and the aspirated fluid (Asb).

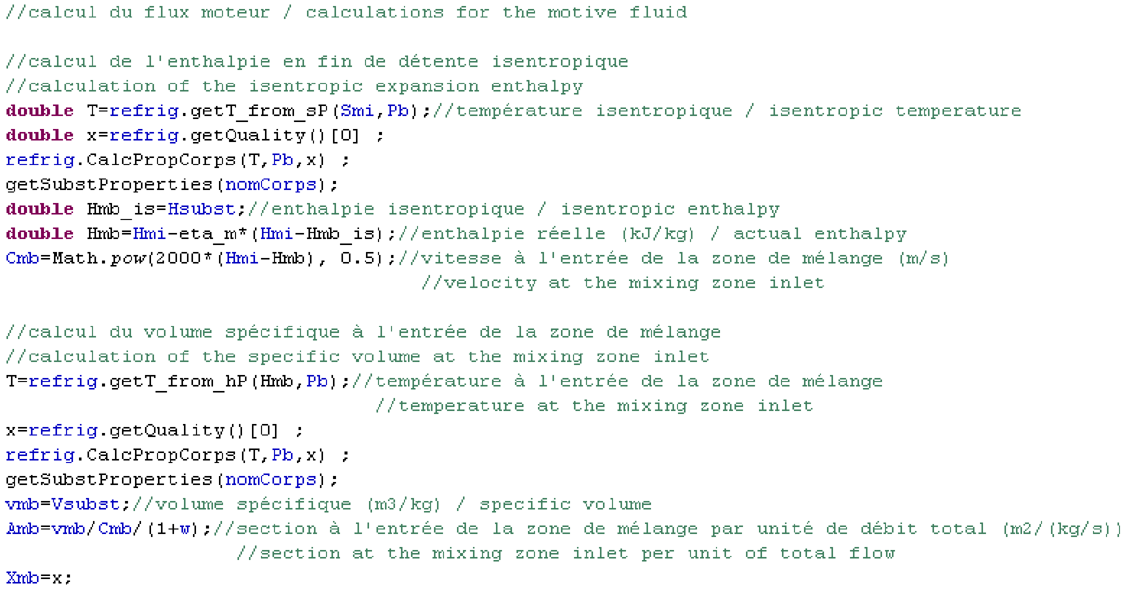
The study of the external class FluidEjector shows how the model has been implemented. As can be seen, six simple steps allow one to perform the calculations:

1) we begin by reading the settings entered on the screen (we added here a friction factor to account for friction losses in the mixing zone):



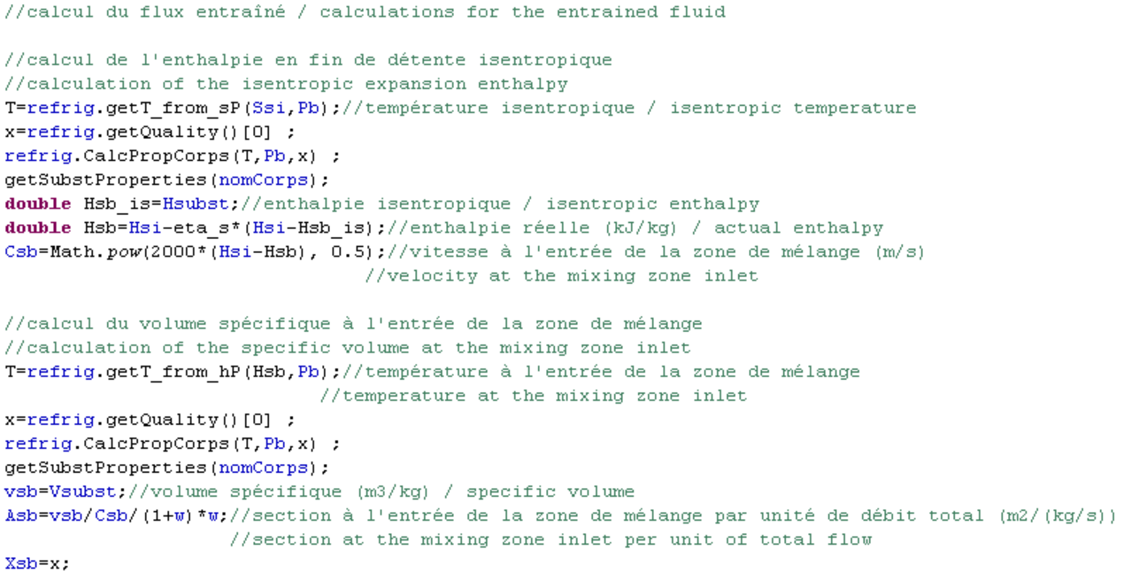
```
DeltaP_factor=Util.lit_d(DeltaP_factor_value.getText());
double eta_m=Util.lit_d(etaNozzle_value.getText());
double eta_s=Util.lit_d(etaNozzle_value.getText());
double fricFactor=Util.lit_d(fricFactor_value.getText());
double eta_d=Util.lit_d(etaDiff_value.getText());
```

2) we then carry out calculations for the motive fluid:



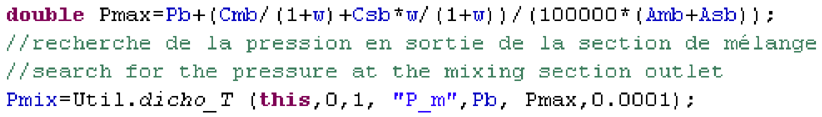
```
//calcul du flux moteur / calculations for the motive fluid

//calcul de l'enthalpie en fin de détente isentropique
//calculation of the isentropic expansion enthalpy
double T=refrig.getT_from_sP(Smi,Pb);//température isentropique / isentropic temperature
double x=refrig.getQuality()[0] ;
refrig.CalcPropCorps(T,Pb,x) ;
getSubstProperties(nomCorps);
double Hmb_is=Hsubst;//enthalpie isentropique / isentropic enthalpy
double Hmb=Hmi-eta_m*(Hmi-Hmb_is);//enthalpie réelle (kJ/kg) / actual enthalpy
Cmb=Math.pow(2000*(Hmi-Hmb), 0.5);//vitesse à l'entrée de la zone de mélange (m/s)
                                    //velocity at the mixing zone inlet

//calcul du volume spécifique à l'entrée de la zone de mélange
//calculation of the specific volume at the mixing zone inlet
T=refrig.getT_from_hP(Hmb,Pb);//température à l'entrée de la zone de mélange
                                //temperature at the mixing zone inlet
x=refrig.getQuality()[0] ;
refrig.CalcPropCorps(T,Pb,x) ;
getSubstProperties(nomCorps);
vmb=Vsubst;//volume spécifique (m3/kg) / specific volume
Amb=vmb/Cmb/(1+w);//section à l'entrée de la zone de mélange par unité de débit total (m2/(kg/s))
                    //section at the mixing zone inlet per unit of total flow
Xmb=x;
```

3) and those for the entrained fluid:

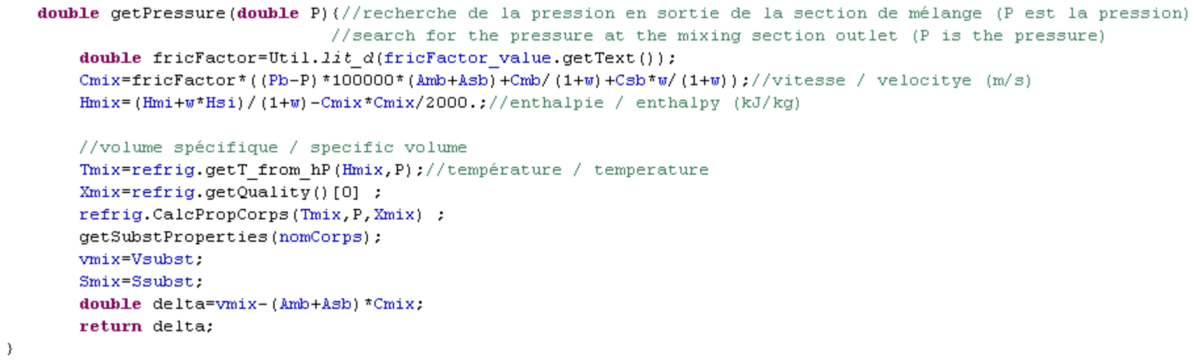


```
//calcul du flux entraîné / calculations for the entrained fluid

//calcul de l'enthalpie en fin de détente isentropique
//calculation of the isentropic expansion enthalpy
T=refrig.getT_from_sP(Ssi,Pb);//température isentropique / isentropic temperature
x=refrig.getQuality()[0] ;
refrig.CalcPropCorps(T,Pb,x) ;
getSubstProperties(nomCorps);
double Hsb_is=Hsubst;//enthalpie isentropique / isentropic enthalpy
double Hsb=Hsi-eta_s*(Hsi-Hsb_is);//enthalpie réelle (kJ/kg) / actual enthalpy
Csb=Math.pow(2000*(Hsi-Hsb), 0.5);//vitesse à l'entrée de la zone de mélange (m/s)
                               //velocity at the mixing zone inlet

//calcul du volume spécifique à l'entrée de la zone de mélange
//calculation of the specific volume at the mixing zone inlet
T=refrig.getT_from_hP(Hsb,Pb);//température à l'entrée de la zone de mélange
                           //temperature at the mixing zone inlet
x=refrig.getQuality()[0] ;
refrig.CalcPropCorps(T,Pb,x) ;
getSubstProperties(nomCorps);
vsb=Vsubst;//volume spécifique (m3/kg) / specific volume
Asb=vsb/Csb/(1+w)*w;//section à l'entrée de la zone de mélange par unité de débit total (m2/(kg/s))
                   //section at the mixing zone inlet per unit of total flow
Xsb=x;
```

4) we then look for the value of the mixture pressure, using for this the dichotomy function Util.dichoT, which seeks the pressure P such as the residue defined by getPressure(P) is zero:

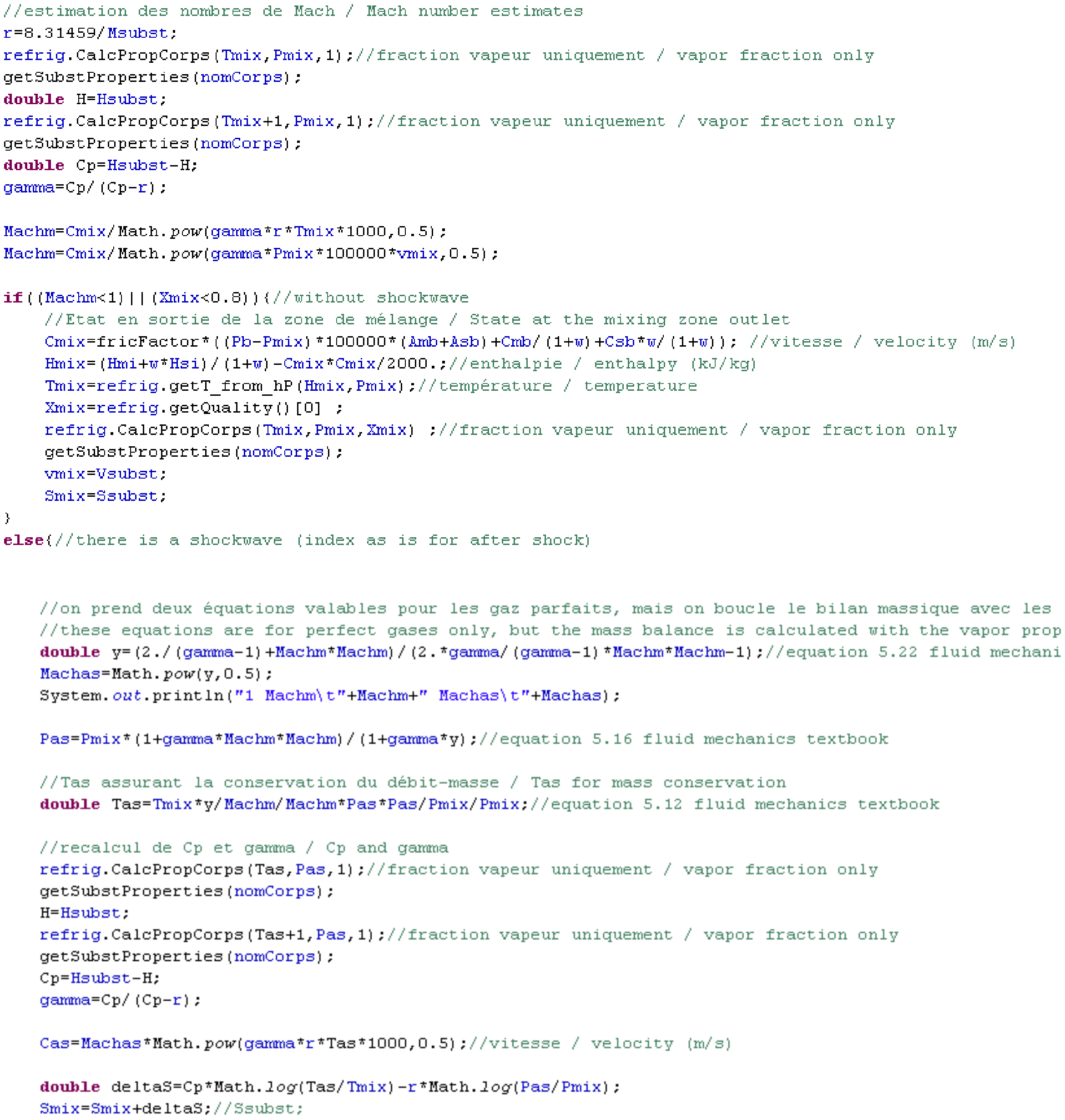


```
double Pmax=Pb+(Cmb/(1+w)+Csb*w/(1+w))/(100000*(Amb+Asb));
//recherche de la pression en sortie de la section de mélange
//search for the pressure at the mixing section outlet
Pmix=Util.dicho_T (this,0,1, "P_m",Pb, Pmax,0.0001);
```



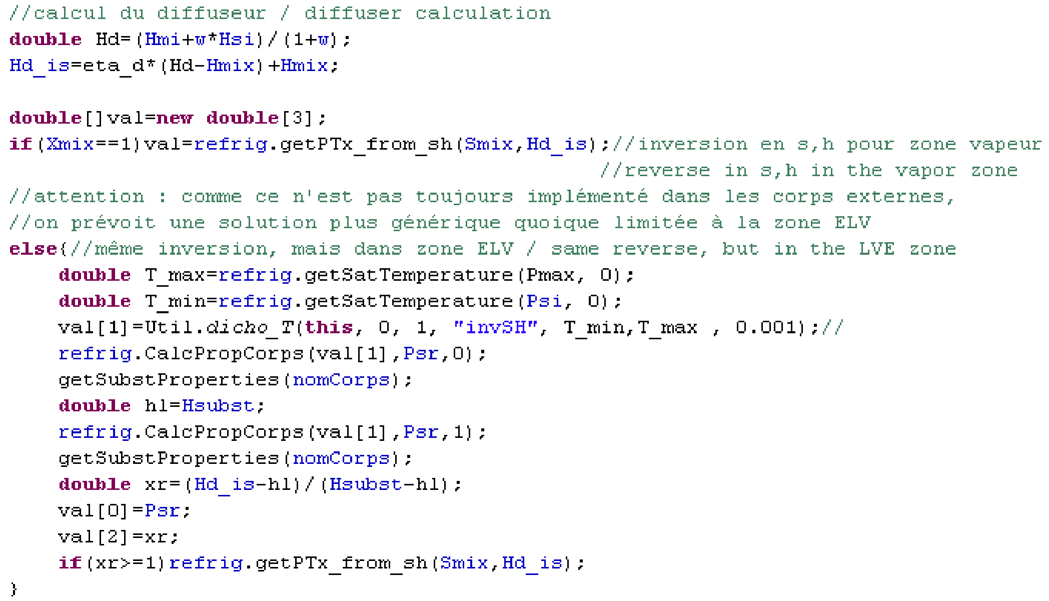
```
  double getPressure(double P){//recherche de la pression en sortie de la section de mélange (P est la pression)
                                //search for the pressure at the mixing section outlet (P is the pressure)
      double fricFactor=Util.lit_d(fricFactor_value.getText());
      Cmix=fricFactor*((Pb-P)*100000*(Amb+Asb)+Cmb/(1+w)+Csb*w/(1+w));//vitesse / velocitye (m/s)
      Hmix=(Hmi+w*Hsi)/(1+w)-Cmix*Cmix/2000.;//enthalpie / enthalpy (kJ/kg)

      //volume spécifique / specific volume
      Tmix=refrig.getT_from_hP(Hmix,P);//température / temperature
      Xmix=refrig.getQuality()[0] ;
      refrig.CalcPropCorps(Tmix,P,Xmix) ;
      getSubstProperties(nomCorps);
      vmix=Vsubst;
      Smix=Ssubst;
      double delta=vmix-(Amb+Asb)*Cmix;
      return delta;
}
```

5) we calculate the Mach number of the mixture to see if the potential shockwave was taken into account. If necessary, we determine the recompression it induces:



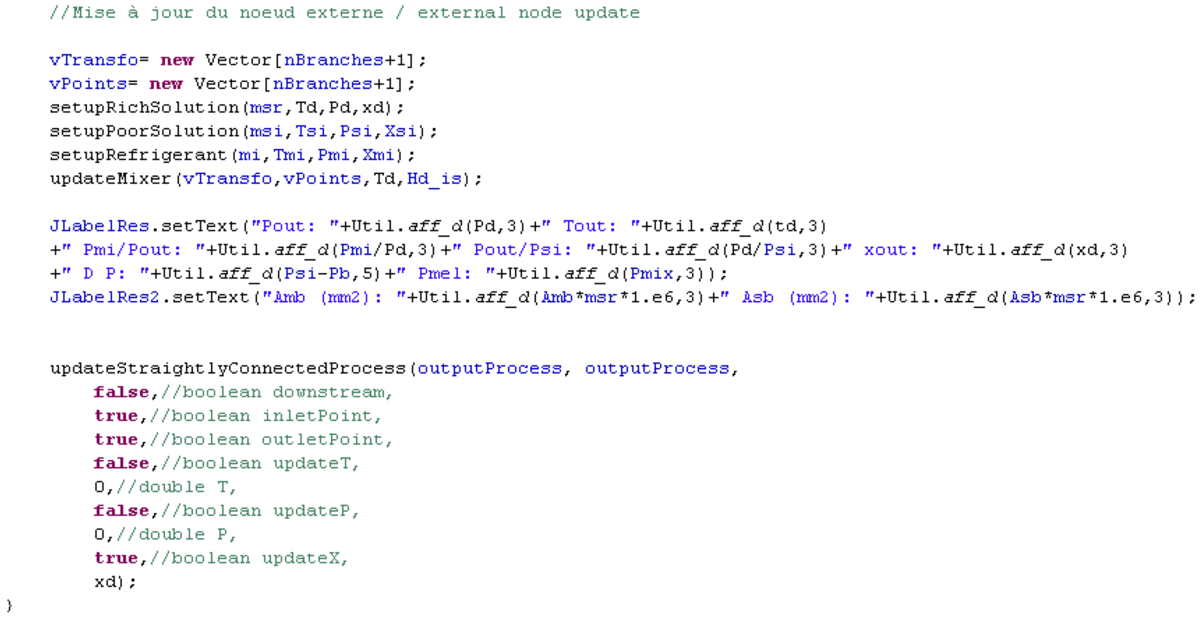
```
//estimation des nombres de Mach / Mach number estimates
r=8.31459/Msubst;
refrig.CalcPropCorps(Tmix,Pmix,1);//fraction vapeur uniquement / vapor fraction only
getSubstProperties(nomCorps);
double H=Hsubst;
refrig.CalcPropCorps(Tmix+1,Pmix,1);//fraction vapeur uniquement / vapor fraction only
getSubstProperties(nomCorps);
double Cp=Hsubst-H;
gamma=Cp/(Cp-r);

Machm=Cmix/Math.pow(gamma*r*Tmix*1000,0.5);
Machm=Cmix/Math.pow(gamma*Pmix*100000*vmix,0.5);

if((Machm<1)||(Xmix<0.8)){//without shockwave
    //Etat en sortie de la zone de mélange / State at the mixing zone outlet
    Cmix=fricFactor*((Pb-Pmix)*100000*(Amb+Asb)+Cmb/(1+w)+Csb*w/(1+w)); //vitesse / velocity (m/s)
    Hmix=(Hmi+w*Hsi)/(1+w)-Cmix*Cmix/2000.;//enthalpie / enthalpy (kJ/kg)
    Tmix=refrig.getT_from_hP(Hmix,Pmix);//température / temperature
    Xmix=refrig.getQuality()[0] ;
    refrig.CalcPropCorps(Tmix,Pmix,Xmix) ;//fraction vapeur uniquement / vapor fraction only
    getSubstProperties(nomCorps);
    vmix=Vsubst;
    Smix=Ssubst;
}
else{//there is a shockwave (index as is for after shock)


   //on prend deux équations valables pour les gaz parfaits, mais on boucle le bilan massique avec les
   //these equations are for perfect gases only, but the mass balance is calculated with the vapor prop
   double y=(2./(gamma-1)+Machm*Machm)/(2.*gamma/(gamma-1)*Machm*Machm-1);//equation 5.22 fluid mechani
   Machas=Math.pow(y,0.5);
   System.out.println("1 Machm\t"+Machm+" Machas\t"+Machas);

   Pas=Pmix*(1+gamma*Machm*Machm)/(1+gamma*y);//equation 5.16 fluid mechanics textbook

   //Tas assurant la conservation du débit-masse / Tas for mass conservation
   double Tas=Tmix*y/Machm/Machm*Pas*Pas/Pmix/Pmix;//equation 5.12 fluid mechanics textbook

   //recalcul de Cp et gamma / Cp and gamma
   refrig.CalcPropCorps(Tas,Pas,1);//fraction vapeur uniquement / vapor fraction only
   getSubstProperties(nomCorps);
   H=Hsubst;
   refrig.CalcPropCorps(Tas+1,Pas,1);//fraction vapeur uniquement / vapor fraction only
   getSubstProperties(nomCorps);
   Cp=Hsubst-H;
   gamma=Cp/(Cp-r);

   Cas=Machas*Math.pow(gamma*r*Tas*1000,0.5);//vitesse / velocity (m/s)

   double deltaS=Cp*Math.log(Tas/Tmix)-r*Math.log(Pas/Pmix);
   Smix=Smix+deltaS;//Ssubst;
```

6) we carry out calculations for the exit diffuser



```
//calcul du diffuseur / diffuser calculation
double Hd=(Hmi+w*Hsi)/(1+w);
Hd_is=eta_d*(Hd-Hmix)+Hmix;

double[]val=new double[3];
if(Xmix==1)val=refrig.getPTx_from_sh(Smix,Hd_is);//inversion en s,h pour zone vapeur
                                                 //reverse in s,h in the vapor zone
//attention : comme ce n'est pas toujours implémenté dans les corps externes,
//on prévoit une solution plus générique quoique limitée à la zone ELV
else{//même inversion, mais dans zone ELV / same reverse, but in the LVE zone
    double T_max=refrig.getSatTemperature(Pmax, 0);
    double T_min=refrig.getSatTemperature(Psi, 0);
    val[1]=Util.dicho_T(this, 0, 1, "invSH", T_min,T_max , 0.001);//
    refrig.CalcPropCorps(val[1],Psr,0);
    getSubstProperties(nomCorps);
    double hl=Hsubst;
    refrig.CalcPropCorps(val[1],Psr,1);
    getSubstProperties(nomCorps);
    double xr=(Hd_is-hl)/(Hsubst-hl);
    val[0]=Psr;
    val[2]=xr;
    if(xr>=1)refrig.getPTx_from_sh(Smix,Hd_is);
}
```

7) the node is updated using the generic methods described in the reference manual



```
    //Mise à jour du noeud externe / external node update

    vTransfo= new Vector[nBranches+1];
    vPoints= new Vector[nBranches+1];
    setupRichSolution(msr,Td,Pd,xd);
    setupPoorSolution(msi,Tsi,Psi,Xsi);
    setupRefrigerant(mi,Tmi,Pmi,Xmi);
    updateMixer(vTransfo,vPoints,Td,Hd_is);

    JLabelRes.setText("Pout: "+Util.aff_d(Pd,3)+" Tout: "+Util.aff_d(td,3)
    +" Pmi/Pout: "+Util.aff_d(Pmi/Pd,3)+" Pout/Psi: "+Util.aff_d(Pd/Psi,3)+" xout: "+Util.aff_d(xd,3)
    +" D P: "+Util.aff_d(Psi-Pb,5)+" Pmel: "+Util.aff_d(Pmix,3));
    JLabelRes2.setText("Amb (mm2): "+Util.aff_d(Amb*msr*1.e6,3)+" Asb (mm2): "+Util.aff_d(Asb*msr*1.e6,3));


    updateStraightlyConnectedProcess(outputProcess, outputProcess,
        false,//boolean downstream,
        true,//boolean inletPoint,
        true,//boolean outletPoint,
        false,//boolean updateT,
        0,//double T,
        false,//boolean updateP,
        0,//double P,
        true,//boolean updateX,
        xd);
}
```