UNIVERSITAT POLITÈCNICA DE CATALUNYA

Flujo Compresible. Estudio de una tobera

Métodos numéricos. Dinámica de gases y transferencia de calor y masa

Boyan Naydenov

27/06/2016

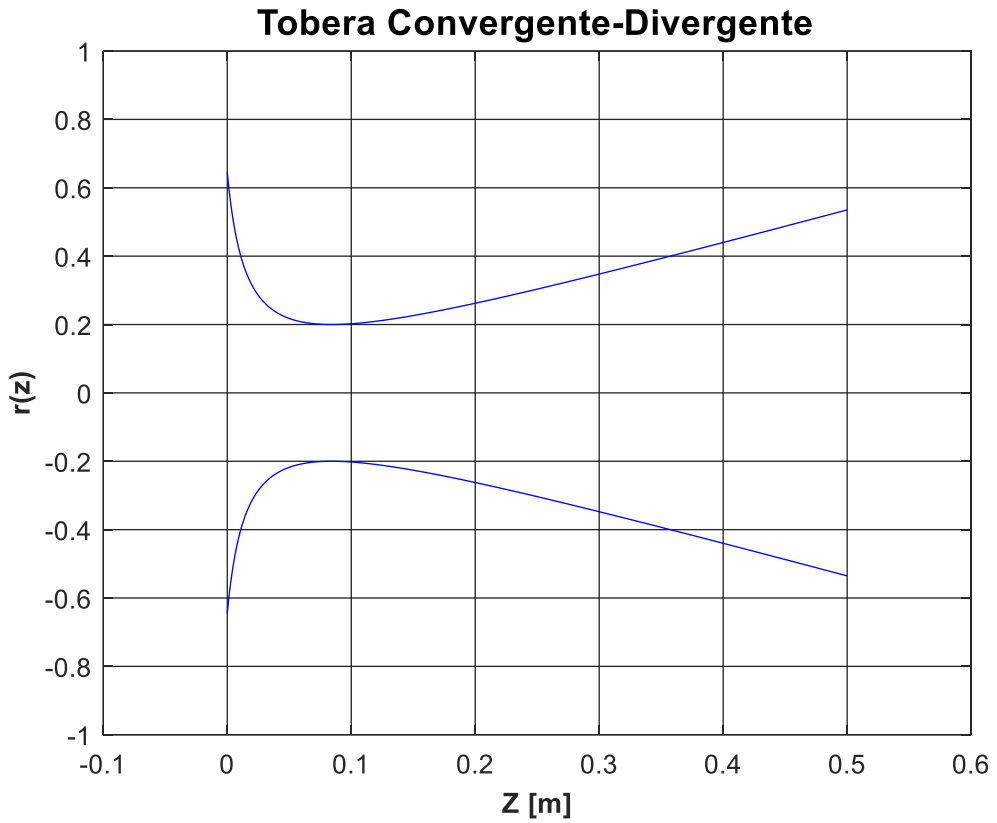
UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeries
Industrial i Aeronàutica de Terrassa

Índice

1. Descripción del caso

En el presente estudio se busca evaluar la distribución media y por lo tanto, unidimensional, de la velocidad, presión, temperatura, número de Mach, entropías y otros, de una tobera convergente-divergente, similar a la utilizada por los cohetes espaciales.

Se muestra a continuación una representación de la geometría del problema. Nótese que dicha geometría se verá sometida a variaciones posteriormente.

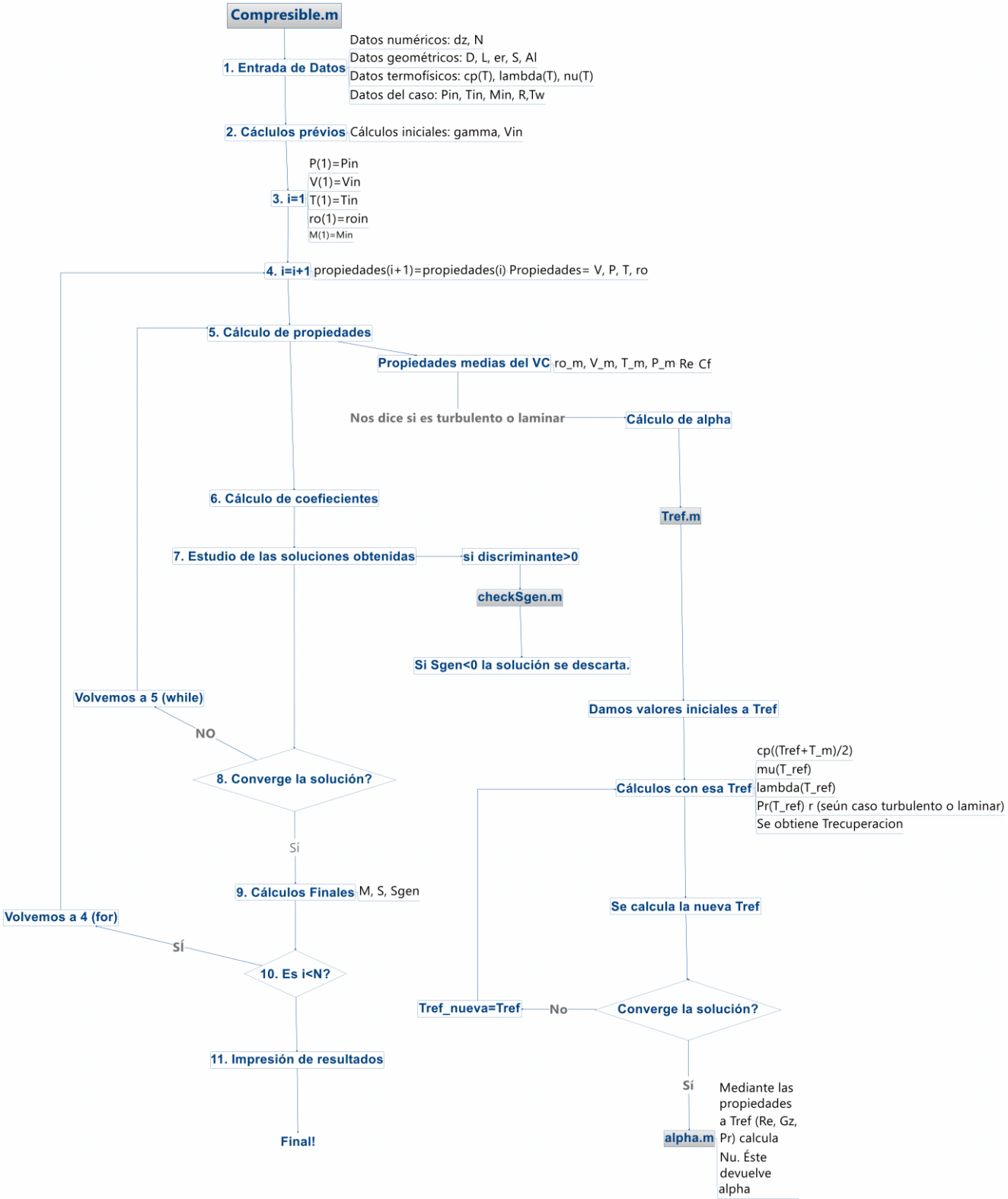


Para la resolución del problema se ha seguido un esquema numérico, discretizando la tobera, que presenta una simetría de revolución, en N volúmenes de control.

En la resolución se tendrá en cuenta el calor de convección así como la existencia de flujo compresible debido a las altas velocidades (M>1).

El algoritmo se detalla en el siguiente diagrama[1]:

[1] Nótese que se utilizan los mismos nombres de las variables y de los archivos que los del código adjunto para una mejor comprensión de este.


Compresible.m
1. Entrada de Datos
Datos numéricos: dz, N
Datos geométricos: D, L, er, S, Al
Datos termofísicos: cp(T), lambda(T), nu(T)
Datos del caso: Pin, Tin, Min, R,Tw
2. Cáclulos prévios
Cálculos iniciales: gamma, Vin
3. i=1
P(1)=Pin
V(1)=Vin
T(1)=Tin
ro(1)=roin
M(1)=Min
4. i=i+1
propiedades(i+1)=propiedades(i) Propiedades= V, P, T, ro
5. Cálculo de propiedades
Propiedades medias del VC
ro_m, V_m, T_m, P_m Re Cf
Nos dice si es turbulento o laminar
Cálculo de alpha
6. Cálculo de coefiecientes
7. Estudio de las soluciones obtenidas
si discriminante>0
checkSgen.m
Si Sgen<0 la solución se descarta.
Volvemos a 5 (while)
NO
8. Converge la solución?
Sí
9. Cálculos Finales
M, S, Sgen
Volvemos a 4 (for)
SÍ
10. Es i<N?
11. Impresión de resultados
Final!
Tref.m
Damos valores iniciales a Tref
cp((Tref+T_m)/2)
mu(T_ref)
lambda(T_ref)
Pr(T_ref) r (seún caso turbulento o laminar)
Se obtiene Trecuperacion
Cálculos con esa Tref
Se calcula la nueva Tref
Converge la solución?
No
Tref_nueva=Tref
Sí
alpha.m
Mediante las propiedades a Tref (Re, Gz, Pr) calcula Nu. Éste devuelve alpha

2. Estudios de verificación

Para verificar y asegurar el correcto funcionamiento del código se ha seguido la siguiente metodología:

1. Realización del algoritmo de resolución de un caso de sección constante.
2. Verificación de balances globales y análisis de las soluciones obtenidas.
3. Comparación de resultados del caso del tubo de sección constante con los presentados en la asignatura "*Dinámica de Gases*".
4. Modificación del algoritmo para adaptarlo al caso de tubo de sección variable (tobera).
5. Verificación de balances globales y análisis de las soluciones obtenidas.
6. Comparación de resultados del caso de tobera con los presentados en la asignatura "*Dinámica de Gases*".

3. Resultados del estudio numérico

3.1. Estudio de convergencia

3.2. Factores de relajación

4. Resultados del estudio físico

4.1. Análisis de los resultados obtenidos

4.2. Influencia de la geometría

4.3. Influencia del material

5. Anexo

5.1 Código utilizado

Complet.m

```
clc;
clear all;
tic
```

Entrada de datos.

```
ra=0;
d=10^-6;
%%Gas : aire
cp=@(T) 1022-0.1626*T+3.5025*10^-4*T^2;
lambda=@(T) 3.807*10^-3+7.4*10^-5*T;
nu=@(T) (2.53928*10^-5*sqrt(T/273.15))/(1+(122/T));

%%Caso de tobera convergente-divergente
er=0.0001;
L=0.5;
N=555;
dz=L/N;
conv_div=1;
for i=1:(N+1)
    Rs(i)=1/(100*i*dz+1.5)+i*dz+0.015;
    D(i)=0.01;
    S(i)=pi*Rs(i)^2;
end
[MIN , Nmin]=min(Rs(:));
for i=1:(N)
    theta(i)=atan((Rs(i+1)-Rs(i))/dz);
    Al(i)=2*pi*Rs(i)*dz/cos(theta(i));
end
    theta(N+1)=atan((Rs(N+1)-Rs(N))/dz);
    Al(N+1)=2*pi*Rs(N+1)*dz/cos(theta(N+1));

Pin=5*100000;
Tin=600;
R=287;
gamma=(cp(Tin))/(cp(Tin)-R);
Tw=300; %in K and cte
```

Obtención de Datos a diferentes M de entrada

```
for Min=[0.01 0.03 0.04 0.05571];
Vin=Min*sqrt((gamma*R*Tin)); %m/s
ra=ra+1;
% Caso de tobera no-crítica (obtenido por pruebas)
                % Onda de choque se ve a producir despues del tubo
                % (no hay onda de choque)
                choq=N+5;
                %%Se llama al programa que realiza todos los calculos
                compresible;
```

```
                % Se guardan los resultados
                Pnc(ra,:)=P; Tnc(ra,:)=T; Vnc(ra,:)=V; Mnc(ra,:)=M;
                entrnc(ra,:)=entr; Sgennc(ra,:)=Sgen;
% Caso de tobera crítica
          if Min==0.05571
                %puntos donde se va a calcular la onda de choque
                %(Nmin=garganta)
                rw=1;
                w=round(linspace(Nmin+100,N,4));
                for choq=w
                %%Se llama al codigo que realiza todos los calculos para cada onda de
                %%choque
                compresible;
                Pc(rw,:)=P; Tc(rw,:)=T; Vc(rw,:)=V; Mc(rw,:)=M;
                entrc(rw,:)=entr; Sgenc(rw,:)=Sgen;
                rw=rw+1;
                end
          end
end
DataFINAL =
struct('Pnc',Pnc,'Pc',Pc,'Tnc',Tnc,'Tc',Tc,'Vnc',Vnc,'Vc',Vc,'Mnc',Mnc,'Mc',Mc,'entrnc',
entrnc,'entrc',entrc,'Sgennc',Sgennc,'Sgenc',Sgenc);
```

Tratamiento de Datos

```
graficas(DataFINAL,L,N);
toc
```

```
Elapsed time is 2.807843 seconds.
```

Compresible.m

Matrices de datos (caso 1D)

```
P=zeros(1,N+1); T=zeros(1,N+1); V=zeros(1,N+1); ro=zeros(1,N+1);
M=zeros(1,N+1); Sgen=zeros(1,N+1); entr=zeros(1,N+1);
%Variable que se activa una vez se produce el salto de M
salt=false;
%Variable que se activa una vez llegamos al punto donde aparecería onda de
%choque
xoc=false;
```

Caso inicial

```
P(1)=Pin; T(1)=Tin; V(1)=Vin;
ro(1)=P(1)/(R*T(1));
mins=ro(1)*V(1)*S(1);
M(1)=Min; Sgen(1)=0; entr(1)=0;
```

Comienzan las iteraciones

```
for i=1:(N);
    if i>=choq
        xoc=true;
    end
    % Alteración de los valores para que nunca estemos en M=1.
```

```
                if M(i)>=0.97 && salt==false
                    Vch=V(i);
                    Pch=P(i);
                    Tch=T(i);
                    roch=ro(i);
                    Mch=M(i);
                    V(i)=V(i)*1.05;
                    P(i)=P(i)*0.99999;
                    T(i)=T(i)*0.99999;
                    ro(i)=P(i)/(R*T(i));
                    mins=ro(i)*V(i)*S(i);
                    M(i)=V(i)/(sqrt((gamma*R*T(i))));
                    salt=true;
                    entra=true;
                end
    %Valores a la salida supuestos
    P(i+1)=P(i);
    T(i+1)=T(i);
    V(i+1)=V(i);
    ro(i+1)=ro(i);
    % Variable que se activa cuando el VC converge
    bien=false;
while  bien==false
%Propiedades medias supuestas
        ro_m=(ro(i+1)+ro(i))/2;
        V_m=(V(i+1)+V(i))/2;
        T_m=(T(i+1)+T(i))/2;
        P_m=ro_m*R*T_m;
        Re=ro_m*V_m*D(i)*2/nu(T_m);
```

T*

```
        %PARA GASES
        if Re>2000
            Data=Tref(1,d,T_m,P_m,V_m,Tw,D(i),L,R,S(i));
        else
            Data=Tref(2,d,T_m,P_m,V_m,Tw,D(i),L,R,S(i));
        end
        %Propiedades  a T_ref
        Cpi=Data.cp; lambdai=Data.lambda; nui=Data.nu; alphai=Data.a;
        r=Data.r; T_r=Data.T_r;
        %Se calcula "f" a partir del Re calculado
                if (Data.Re<2000)
                    f=16/Data.Re;
                elseif (Data.Re>5*10^3 && Data.Re<3*10^4 && er<=0.0001)
                    f=0.079*Data.Re^-0.25;
                elseif (Data.Re>5*10^3 && Data.Re<3*10^4 && er>0.003 && er<0.005)
                    f=0.096*Data.Re^-0.25;
                elseif (Data.Re>3*10^4 && Data.Re<3*10^6 && er<=0.0001)
                    f=0.046*Data.Re^-0.2;
                elseif (Data.Re>3*10^4 && Data.Re<3*10^6 && er>0.003&& er<0.005)
                    f=0.078*Data.Re^-0.2;
                end
                        %Churchill General Expression
        %         AS=(2.457*log(1/((7/Re)^0.9+0.27*er)))^16;
        %         BS=(37530/Re)^16;
        %         f=2*((8/Re)^12+1/(AS+BS)^(3/2))^(1/12);
```

Cálculo de coeficientes que resuelven ecuaciones N-S

```
        Al_m=(Al(i+1)+Al(i))/2;
        Av=mins+f*ro_m*abs(V_m)/4*Al_m*cos(theta(i));
        Bv=(S(i)+Al_m/2*sin(theta(i)));
        Cv=(S(i)+Al_m/2*sin(theta(i)))*P(i)+(mins-
f*ro_m*abs(V_m)/4*Al_m*cos(theta(i)))*V(i);

        At=mins*cp(T_m)+alphai*Al(i)/2;
        Bt=mins/2+r*alphai*Al(i)/(4*Data.cptref);
        Ct=(mins*cp(T_m)-alphai*Al(i)/2)*T(i)+(mins/2-
r*alphai*Al(i)/(4*Data.cptref))*V(i)^2+alphai*Tw*Al(i);

        A=Av*At*S(i)-Bv*Bt*mins*R;
        B=Cv*At*S(i);
        C=Bv*Ct*mins*R;

        dis=B^2-4*A*C;
```

Únicamente se prosigue si discriminante es positivo

```
        if dis>0
            %Se obtienen 2 soluciones.
            %Revisión de las soluciones mediante entropia generada

            positiva=1;% para raíz positiva
            DataFinal_a =
checkSgen(dis,A,B,C,Av,Bv,Cv,At,Bt,Ct,positiva,T(i),P(i),mins,Data,dz,Al(i),S(i),Tw,V(i)
,f,Data.ro);
            Sgena=DataFinal_a.sgen;

            positiva=0;% para raíz negativa
            DataFinal_b =
checkSgen(dis,A,B,C,Av,Bv,Cv,At,Bt,Ct,positiva,T(i),P(i),mins,Data,dz,Al(i),S(i),Tw,V(i)
,f,Data.ro);
            Sgenb=DataFinal_b.sgen;

            if  Sgena>=0 && Sgenb<0
                V_i=DataFinal_a.v;
                P_i=DataFinal_a.p;
                T_i=DataFinal_a.T;
                ro_i=DataFinal_a.ro;
                % Se revisa convergenica
                    if abs(P_i-P(i+1))<d && abs(T_i-T(i+1))<d && abs(V_i-V(i+1))<d
                        bien=true;
                    end
                 % Calculo de los valores a la salida del VC
                V(i+1)=V_i;
                P(i+1)=P_i;
                T(i+1)=T_i;
                ro(i+1)=ro_i;
                Sgen(i+1)=Sgena;
                entr(i+1)=entr(i)+DataFinal_a.S;
                gamma=(cp(T(i+1))/R)/(cp(T(i+1))/R-1);
                M(i+1)=V(i+1)/(sqrt((gamma*R*T(i+1))));

            elseif Sgena<0 && Sgenb>=0
```

```
                V_i=DataFinal_b.v;
                P_i=DataFinal_b.p;
                T_i=DataFinal_b.T;
                ro_i=DataFinal_b.ro;
                    if abs(P_i-P(i+1))<d && abs(T_i-T(i+1))<d && abs(V_i-V(i+1))<d
                        bien=true;
                    end
                V(i+1)=V_i;
                P(i+1)=P_i;
                T(i+1)=T_i;
                ro(i+1)=ro_i;
                Sgen(i+1)=Sgenb;
                entr(i+1)=entr(i)+DataFinal_b.S;
                gamma=(cp(T(i+1))/R)/(cp(T(i+1))/R-1);
                M(i+1)=V(i+1)/(sqrt((gamma*R*T(i+1))));
            else
                % En caso de flujo crítico, se obtienen dos solcuiones
                % positivas

                a=abs(DataFinal_a.v-V(i));% Salto
                b=abs(DataFinal_b.v-V(i));
                if a<b
                       if i ~=choq
                           % Se escoge el caso sin salto
                            V_i=DataFinal_a.v;
                            P_i=DataFinal_a.p;
                            T_i=DataFinal_a.T;
                            ro_i=DataFinal_a.ro;
                       if abs(P_i-P(i+1))<d && abs(T_i-T(i+1))<d && abs(V_i-V(i+1))<d
                                bien=true;
                            end
                            V(i+1)=V_i;
                            P(i+1)=P_i;
                            T(i+1)=T_i;
                            ro(i+1)=ro_i;
                            Sgen(i+1)=Sgena;
                            entr(i+1)=entr(i)+DataFinal_a.S;
                            gamma=(cp(T(i+1))/R)/(cp(T(i+1))/R-1);
                            M(i+1)=V(i+1)/(sqrt((gamma*R*T(i+1))));
                       else
                            % Se escoge el caso que genera salto (onda de
                            % choque)
                            V_i=DataFinal_b.v;
                            P_i=DataFinal_b.p;
                            T_i=DataFinal_b.T;
                            ro_i=DataFinal_b.ro;
                        if abs(P_i-P(i+1))<d && abs(T_i-T(i+1))<d && abs(V_i-V(i+1))<d
                                bien=true;
                            end
                            V(i+1)=V_i;
                            P(i+1)=P_i;
                            T(i+1)=T_i;
                            ro(i+1)=ro_i;
                            Sgen(i+1)=Sgenb;
                            entr(i+1)=entr(i)+DataFinal_b.S;
                            gamma=(cp(T(i+1))/R)/(cp(T(i+1))/R-1);
                            M(i+1)=V(i+1)/(sqrt((gamma*R*T(i+1))));
                        end
```

```
                    else
                          if i ~=choq
                             V_i=DataFinal_b.v;
                             P_i=DataFinal_b.p;
                             T_i=DataFinal_b.T;
                             ro_i=DataFinal_b.ro;
                        if abs(P_i-P(i+1))<d && abs(T_i-T(i+1))<d && abs(V_i-V(i+1))<d
                                                  bien=true;
                                             end
                             V(i+1)=V_i;
                             P(i+1)=P_i;
                             T(i+1)=T_i;
                             ro(i+1)=ro_i;
                             Sgen(i+1)=Sgenb;
                             entr(i+1)=entr(i)+DataFinal_b.S;
                             gamma=(cp(T(i+1))/R)/(cp(T(i+1))/R-1);
                             M(i+1)=V(i+1)/(sqrt((gamma*R*T(i+1))));
                          else
                             V_i=DataFinal_a.v;
                             P_i=DataFinal_a.p;
                             T_i=DataFinal_a.T;
                             ro_i=DataFinal_a.ro;
                          if abs(P_i-P(i+1))<d && abs(T_i-T(i+1))<d && abs(V_i-V(i+1))<d
                                  bien=true;
                             end
                             V(i+1)=V_i;
                             P(i+1)=P_i;
                             T(i+1)=T_i;
                             ro(i+1)=ro_i;
                             Sgen(i+1)=Sgena;
                             entr(i+1)=entr(i)+DataFinal_a.S;
                             gamma=(cp(T(i+1))/R)/(cp(T(i+1))/R-1);
                             M(i+1)=V(i+1)/(sqrt((gamma*R*T(i+1))));
                          end
                    end
                end
            else
                display('Dis negatiu. Error');
               break;
            end
end
end
```

Tref.m

Proceso iterativo para el cálculo de Tref

```
function Datos = Tref(caso,d,T_m,P_m,V_m,Tw,D,L,R,S)
T_reff=T_m;
T_rn=T_m+1;
listo=false;
while listo==false;
        cp=(1022-0.1626*T_reff+(3.5025*10^-4)*T_reff^2);
        nu=(2.53928*10^-5*sqrt(T_reff/273.15))/(1+(122/T_reff));
        lambda=(3.807*10^-3+7.4*10^-5*T_reff);
        Pr=(cp*nu/lambda);
        %Según turubulento(1) o laminar (2)
        if caso==1
```

```
            r=Pr^(1/3);
        else
            r=Pr^(1/2);
        end
        cptref=1/(T_rn-T_m)*(1022*(T_rn-T_m)-0.1626/2*(T_rn^2-T_m^2)+((3.5025*10^-
4)/3)*(T_rn^3-T_m^3));
        T_rn=T_m+r*(V_m)^2/(2*cptref);
        T_ref=T_m+0.5*(Tw-T_m)+0.22*(T_rn-T_m);

        % Revisamos convergencia
        if abs(T_ref-T_reff)<d
            listo=true;
            T_reff=T_ref;
            T_r=T_rn;
        else
            T_reff=T_ref;
            T_rn=T_rn;
        end
end
ro_Tref=P_m/(R*T_reff);
Ref=ro_Tref*V_m*D/nu;
Gz=pi*D/(4*L)*Ref*Pr;
a=alpha(Ref,Gz,Pr,lambda,D);
% Tupla que devuelve la funcion
Datos = struct('cp',cp,'nu',nu,'lambda',lambda, 'Pr', Pr, 'r', r, 'T_ref',T_reff, 'T_r',
T_r, 'Re', Ref, 'Gz', Gz, 'a',a, 'ro', ro_Tref,'cptref',cptref);
end
```

checkSgen.m

```
function DataFinal = checkSgen(
dis,A,B,C,Av,Bv,Cv,At,Bt,Ct,positiva,Tc,pc,min,Data,dz,Al,S,Tw,V,f,rop)
R=287;
cp=@(T) 1022-0.1626*T+3.5025*10^-4*T^2;
    if positiva
        v=(B+sqrt(dis))/(2*A);
    else
        v=(B-sqrt(dis))/(2*A);
    end
p=(Cv-Av*v)/Bv;
T=(Ct-Bt*v*v)/At;
ro=p/(R*T);

%Check with conservation equations
% Ec de la masa
ecm=min-ro*v*S;
% Ec del momentum
eccm=min*(v-V)-pc*S+p*S+f*(ro+rop)/2*abs((v+V)/2)*((v+V)/2)/2*Al;
% Ec de l'energia
ece=min*cp((T+Tc)/2)*(T-Tc)+min*(v^2/2-V^2/2)-Data.a*(Tw-
((T+Tc)/2+Data.r*(((v+V)/2)^2)/(2*Data.cptref)))*Al;

%Calculamos Entropia generada
Ds=1022*log(T/Tc)-0.1626*(T-Tc)+((3.5025*10^-4)/2)*(T^2-Tc^2)-R*log(p/pc);
Sgen=min*Ds/(S*dz)-Data.a*Al*(Tw-Data.T_r)/(S*dz*Tw);
DataFinal = struct('sgen',Sgen,'Ds',Ds,'v',v, 'p', p, 'T', T, 'ro',ro, 'S', Ds);
end
```

alpha.m

```
function alphai = alpha( Re_Tref,Gz_Tref,Pr_Tref,lambdai,D)
            if Re_Tref<2000 && Gz_Tref>10
                        C=1.86;
                        m=1/3;
                        n=1/3;
                        K=(D/L)^(1/3)*(nui/nu(Tw))^(0.14);
            elseif Re_Tref<2000 && Gz_Tref<10
                        C=3.66;
                        m=0;
                        n=0;
                        K=1;
            elseif Re_Tref>2000
                        C=0.023;
                        m=0.8;
                        n=0.4;
                        K=1;
            end
%Nusselt
Nu=C*Re_Tref^m*Pr_Tref^n*K;
%Coeficeinte de calor por convección
alphai=Nu*lambdai/(D);
end
```

graficas.m

```
function  graficas( DataFINAL,L,N)
figure1 = figure('Color',[1 1 1]);
% legendInfo{ra} = ['Min = ' num2str(Min)];
x=linspace(0,L,N+1);
                subplot(3,2,1)
                plot(x,DataFINAL.Pnc,x,DataFINAL.Pc);
                grid on;
                title('Pressió','FontWeight','bold','FontSize',14);
                xlabel('X [m]','FontWeight','bold'); ylabel('P
[Pa]','FontWeight','bold');

                subplot(3,2,2)
                plot(x,DataFINAL.Tnc,x,DataFINAL.Tc);
                hold on;
                grid on;
                title('Temperatura','FontWeight','bold','FontSize',14);
                xlabel('X [m]','FontWeight','bold'); ylabel('T
[K]','FontWeight','bold');

                subplot(3,2,3)
                plot(x,DataFINAL.Vnc,x,DataFINAL.Vc);
                hold on;
                grid on;
                title('Velocitat','FontWeight','bold','FontSize',14);
                xlabel('X [m]','FontWeight','bold'); ylabel('V
[m/s]','FontWeight','bold');

                subplot(3,2,4)
                plot(x,DataFINAL.Mnc,x,DataFINAL.Mc);
                hold on;
```

```
                grid on;
                title('Mach','FontWeight','bold','FontSize',14);
                xlabel('X [m]','FontWeight','bold'); ylabel('Mach','FontWeight','bold');

                subplot(3,2,5)
                semilogy(x,abs(DataFINAL.entrnc),x,abs(DataFINAL.entrc));
                hold on;
                grid on;
                title('Entropia Específica','FontWeight','bold','FontSize',14);
                xlabel('X [m]','FontWeight','bold'); ylabel('s
[J/KgK]','FontWeight','bold');

                subplot(3,2,6)
                plot(x,DataFINAL.Sgennc,x,DataFINAL.Sgenc);
                hold on;
                grid on;
                title('Entropía generada per unitat de
Volum','FontWeight','bold','FontSize',14);
                xlabel('X [m]','FontWeight','bold'); ylabel('W
[W/m3K]','FontWeight','bold');
end
```

5.2 Comportamiento Analítico Esperado y Criterio de Summerfield