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

Curso: Grado en ingeniería en tecnologías aeroespaciales

Estudiante: Andrés Adam Alberdi

Título del proyecto:

Estudio del desarrollo de una nueva aplicación Android para flujo compresible

"App4Comp"

Director: Pedro Javier Gamez Montero

Tutor: Daniel Garcia-Almiñana

Fecha de entrega: 12 de junio de 2015

Contenido: Documento 2 - Anexos

Contenido

Contenido i

Lista de Figuras ii

Lista de tablas iv

A. Funcionamiento de la App 1

A.1. Atmósfera Estándar Internacional ISA 1

A.1.1. Presentación *1*

A.1.2. Bloques *1*

A.2. Flujo isentrópico 4

A.2.1. Lista de gases *4*

A.2.2. Bloques *6*

A.3. Onda de choque normal 10

A.3.1. Bloques *10*

A.4. Tobera convergente-divergente 15

A.4.1. Modo tobera adaptada *15*

Presentación *15*

Bloques *16*

A.4.2. Modo genérico *20*

A.5. Flujo de Fanno 24

A.5.1. Propiedades de gases y materiales incluidos en la App *24*

A.5.2. Bloques *25*

B. Códigos de verificación de resultados 30

B.1. Atmósfera ISA 30

B.2. Flujo isentrópico 31

B.3. Onda de choque 32

B.4. Tobera convergente-divergente 34

B.4.1. Modo tobera adaptada 34

B.4.2. Modo genérico 37

B.5. Flujo de Fanno 43

Lista de Figuras

Ilustración A.1. Función ISA en cálculo normal, ante input incorrecto y elección de unidades de altitud 1
Ilustración A.2. Declaración de variables en ISA 2
Ilustración A.3. Bloque principal en ISA 3
Ilustración A.4. Bloques extras en ISA 3
Ilustración A.5. Elección del factor gamma 5
Ilustración A.6. Bloque principal de la función flujo isentrópico 7
Ilustración A.7. Función de T/T0 en flujo isentrópico 7
Ilustración A.8. Manipulación de visibilidades en la función flujo isentrópico 8
Ilustración A.9. Cambio de etiquetas para los casos directo e inverso 9
Ilustración A.10. Bloque principal de la función onda de choque 12
Ilustración A.11. Procedimiento Ma2 13
Ilustración A.12. Procedimiento P_relation 13
Ilustración A.13. Procedimiento T_relation 13
Ilustración A.14. Procedimiento rho_relation 14
Ilustración A.15. Procedimiento P_stag_relation 14
Ilustración A.16. Procedimiento area_relation 14
Ilustración A.17. Selección del input en modo tobera adaptada 15
Ilustración A.18. Resultados para el modo tobera adaptada con relación de áreas como input 16
Ilustración A.19. Bloque principal del modo tobera adaptada 18
Ilustración A.20. Función M_from_P 18
Ilustración A.21. Función M_from_T 19
Ilustración A.22. Función Arel_from_M 19
Ilustración A.23. Proceso iterativo solver_Ma_from_Arel 19
Ilustración A.24. Derivada numérica derivative_for_Arel 20
Ilustración A.25. Bloque principal del modo genérico 22
Ilustración A.26. Proceso iterativo Solver_shock_area 23

Ilustración A.27. Proceso para encontrar la presión en la salida P_exit_for_area 23

Ilustración A.28. Bloque principal de la función flujo de Fanno 27

Ilustración A.29. Función fld_formula..... 28

Ilustración A.30. Función p_stag_rel_chocked 28

Ilustración A.31. Función T_rel_chocked..... 28

Ilustración A.32. Función rho_rel_chocked..... 28

Ilustración A.33. Función P_rel_chocked..... 28

Lista de tablas

Tabla A.1. Lista de gases predeterminada 4

Tabla A.2. Propiedades de gases comunes a 1 atm y 20ºC 24

Tabla A.3. Valores típicos de rugosidad para conductos comerciales 25

A. Funcionamiento de la App

A.1. Atmósfera Estándar Internacional ISA

A.1.1. Presentación

Se muestran diferentes aspectos de la función ISA:

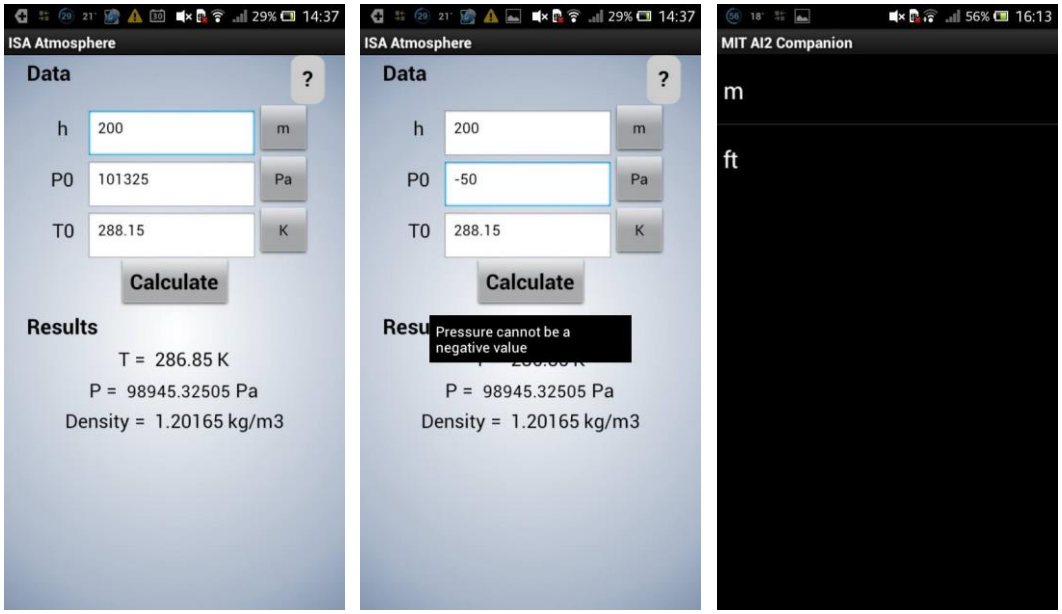


Ilustración A.1. Función ISA en cálculo normal, ante input incorrecto y elección de unidades de altitud

A.1.2. Bloques

Los bloques principales de la función ISA son:

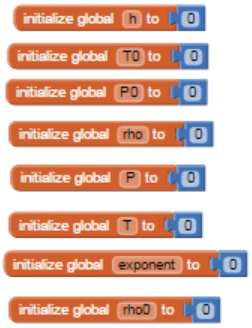
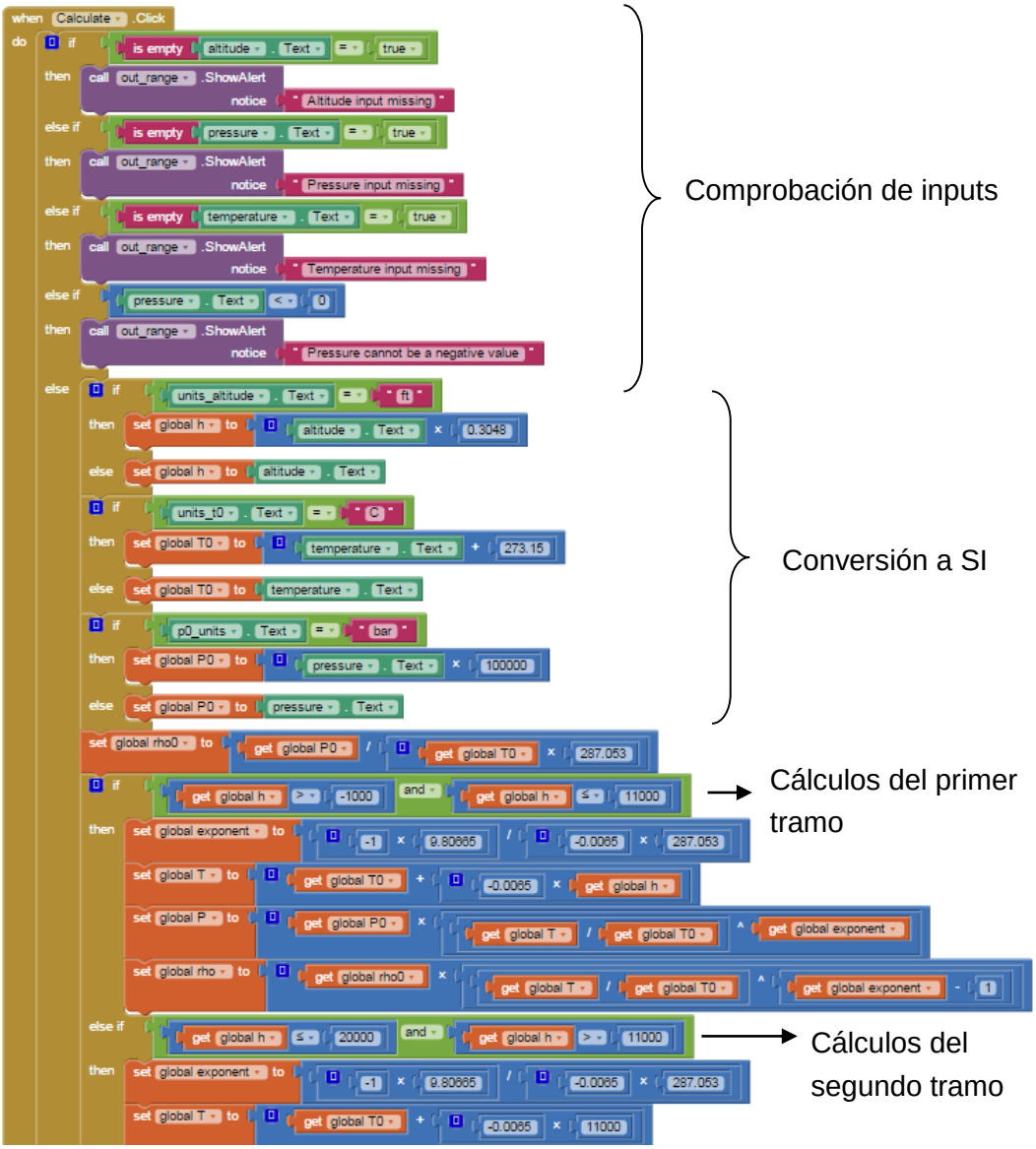


Ilustración A.2. Declaración de variables en ISA



(Continua en la página siguiente)

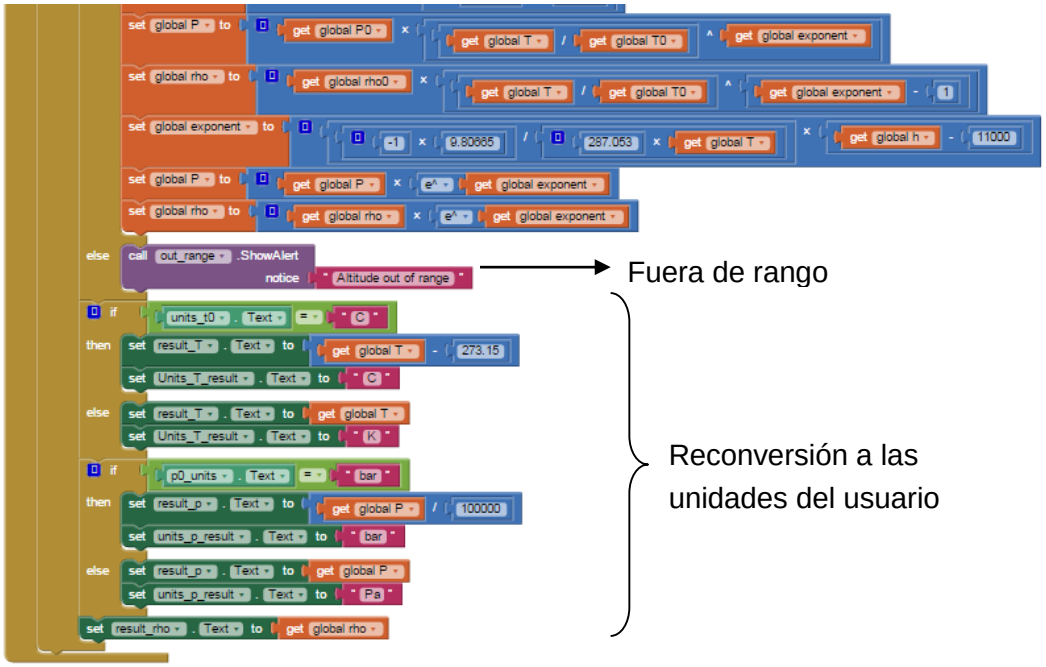


Ilustración A.3. Bloque principal en ISA

En cuanto a las demás funciones de la parte de bloques de la aplicación, están los que cambian el texto de los "listpcker" de las unidades a la selección del usuario, y la información extra para el botón "Más información":

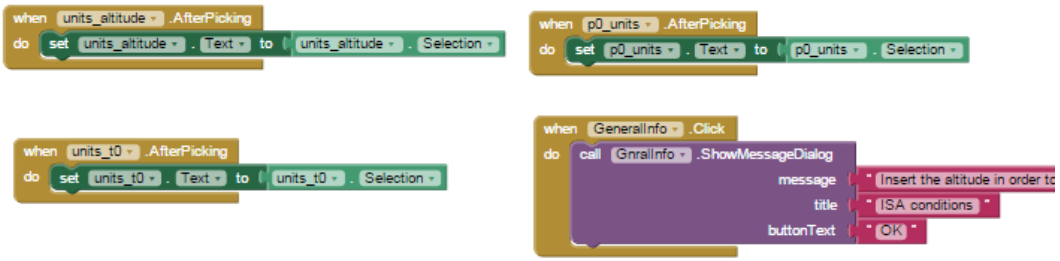


Ilustración A.4. Bloques extras en ISA

Los bloques relacionados con la selección de unidades son siempre de la misma forma en todas las funciones.

A.2. Flujo isentrópico

A.2.1. Lista de gases

Los gases disponibles para escoger son, en el mismo orden que se disponen en la aplicación:

Gas	γ
Aire seco	1.4
Amoníaco	1.31
Argón	1.67
Dióxido de carbono	1.3
Monóxido de carbono	1.4
Gas cloro	1.34
Etano	1.22
Helio	1.66
Hidrógeno	1.41
Kriptón	1.68
Neón	1.64
Óxido de nitrógeno	1.4
Dióxido de nitrógeno	1.31
Nitrógeno gas	1.404
Oxígeno	1.4
Propano	1.13
Dióxido de azufre	1.29
Vapor de agua	1.33
Xenón	1.66

Tabla A.1. Lista de gases predeterminada

Están dispuestos en orden alfabético en inglés, con la excepción del aire seco, que al ser la opción más común se ha puesto en primer lugar. Los datos fueron tomados de “Mecánica de fluidos” [4] y “Lange’s Handbook of Chemistry” [15] a las temperaturas más próximas a 20 ºC posibles. Es fácil ver que muchos de estos gases normalmente no cumplen la condición del problema de tener un comportamiento parecido al de un gas ideal y caloríficamente perfecto, pero la decisión de usarlos y aceptar la semejanza con el comportamiento real depende en última instancia del usuario.

A continuación se muestra cómo se ve la elección en la aplicación:

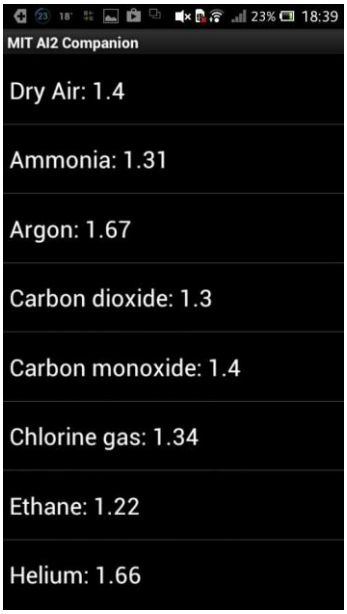
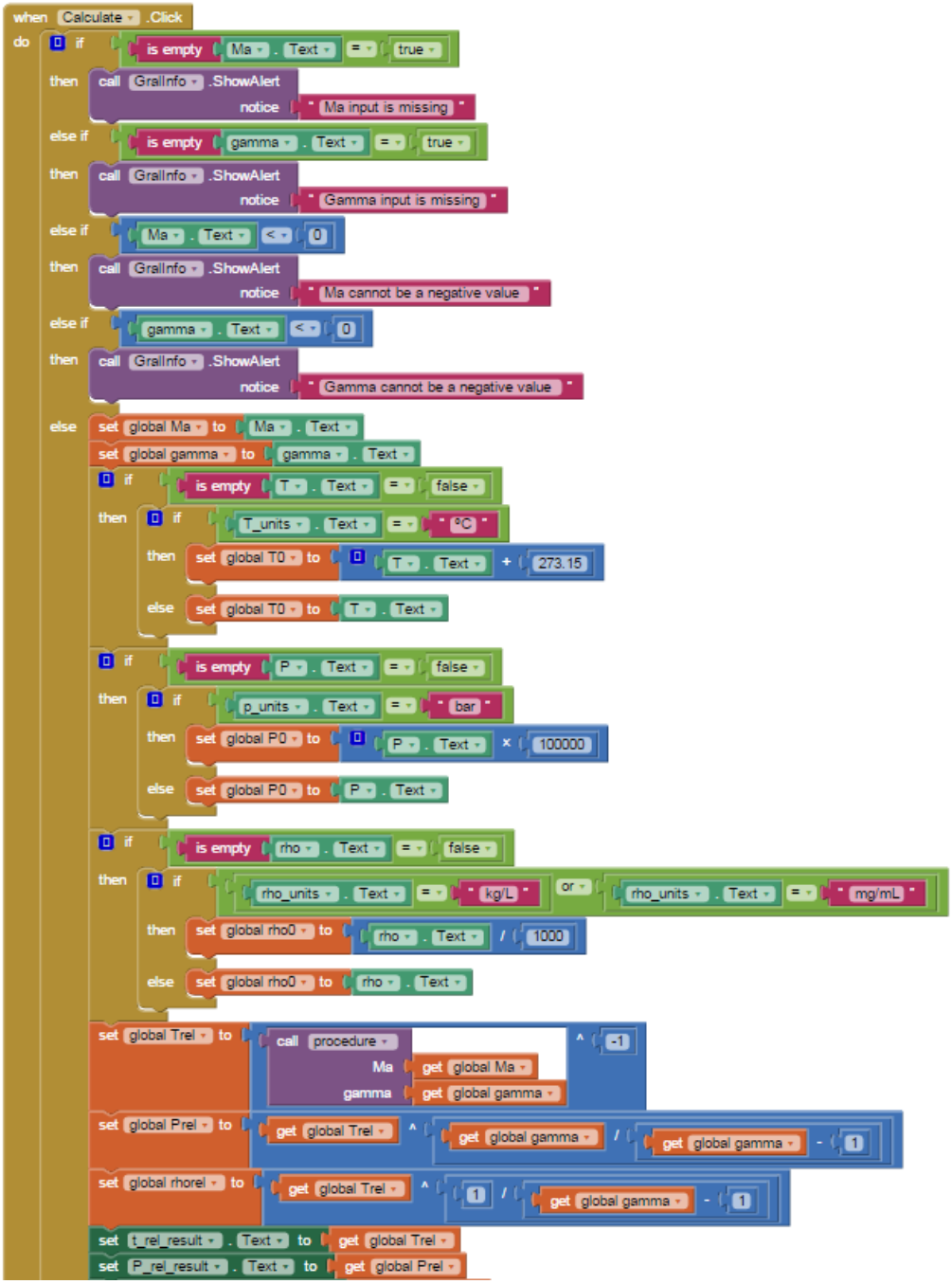


Ilustración A.5. Elección del factor gamma

A.2.2. Bloques

En bloque principal de la aplicación es el siguiente:



(Continúa en la página siguiente)

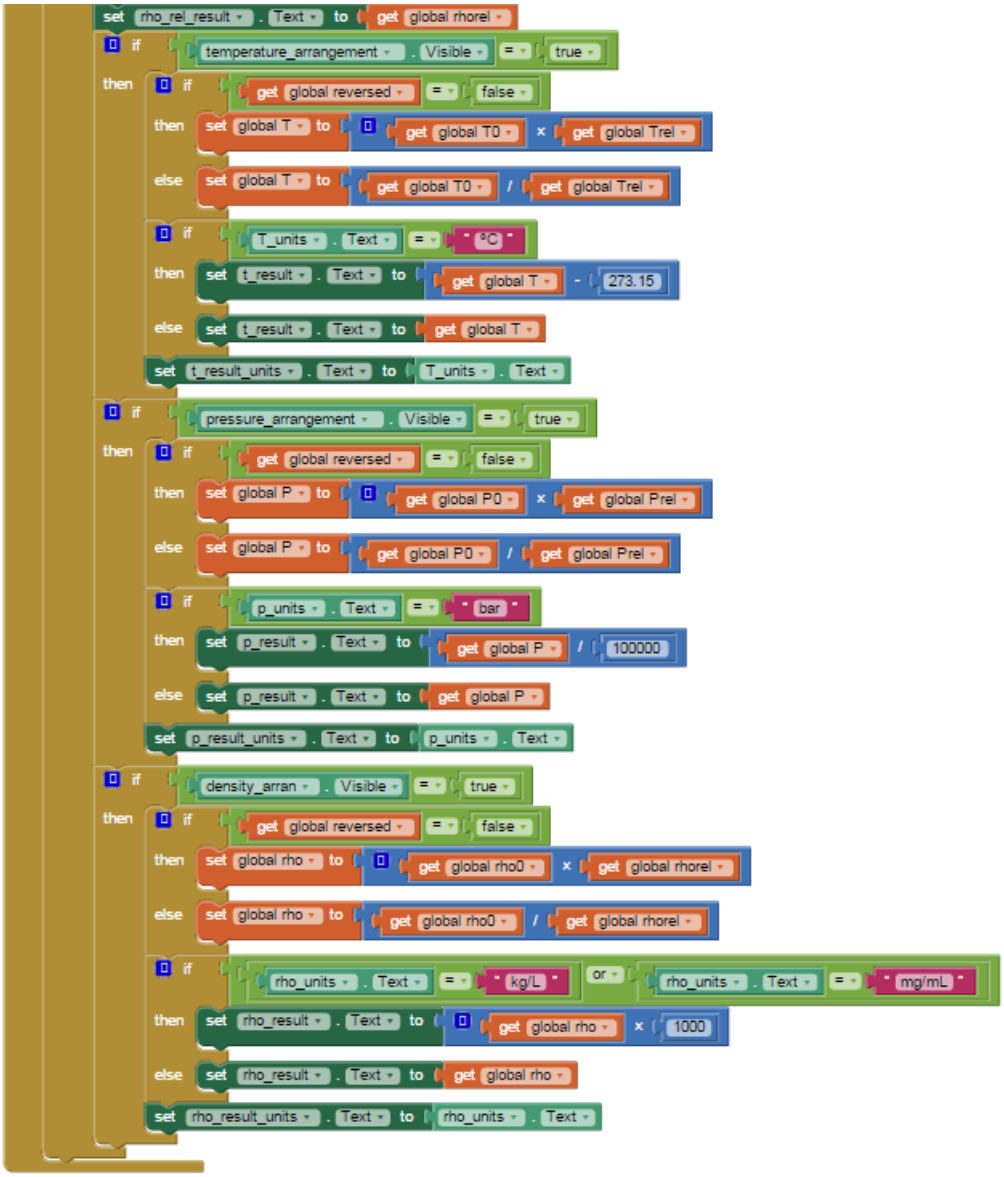


Ilustración A.6. Bloque principal de la función flujo isentrópico

Durante este bloque llama a la función "procedure", la única función creada por el programador, para calcular la relación T/T_0. Este bloque es la representación en App Inventor de la fórmula que relaciona la relación de temperaturas con el número de Mach.

Ilustración A.7. Función de T/T0 en flujo isentrópico

La novedad de esta función es la capacidad de hacer cálculos dimensionales opcionales e invertir el problema. La parte de elegir qué propiedad hacer dimensional se controla por el botón “Dimensional” y con el bloque siguiente:

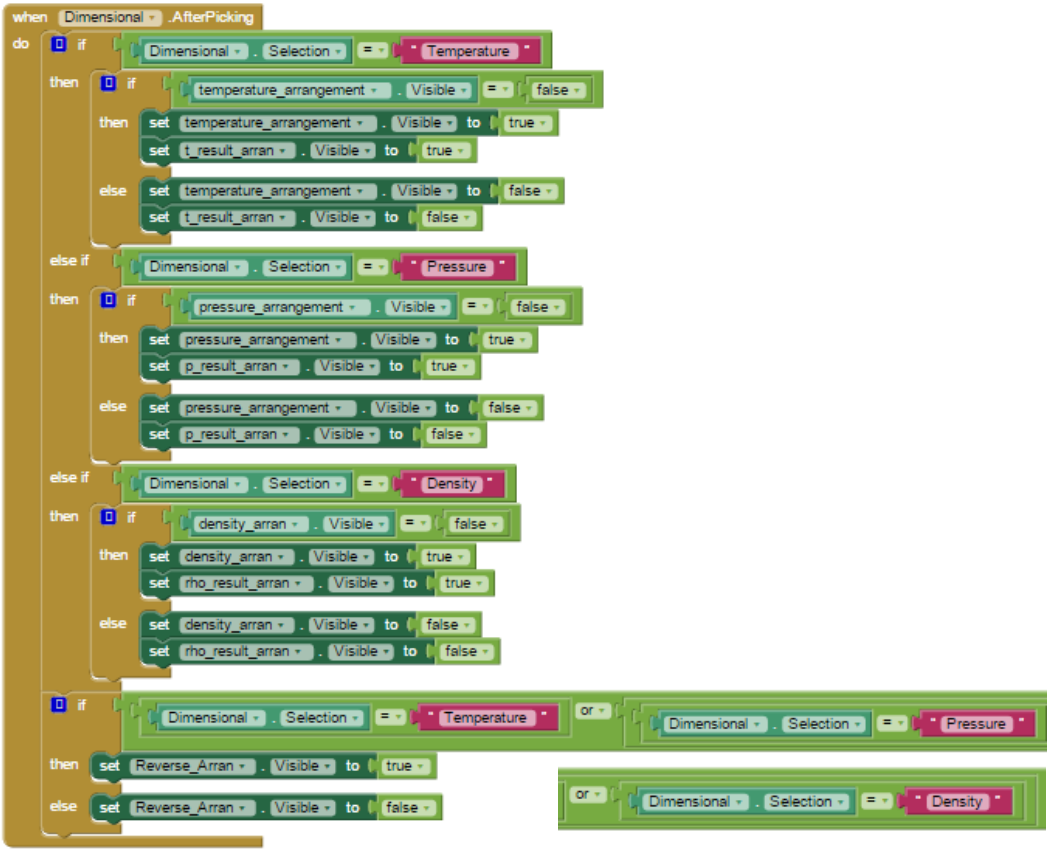


Ilustración A.8. Manipulación de visibilidades en la función flujo isentrópico

En este bloque, se manipula la visibilidad de los “arrangement” que contienen la etiqueta que señala la propiedad, el cuadro de texto para introducirla y el botón de las unidades. Cuando alguno de ellos es visible, el problema se puede invertir (último bloque “if”).

Por lo que respecta a poder invertir el problema con el botón “Reverse”, el bloque que controla su comportamiento es el siguiente:

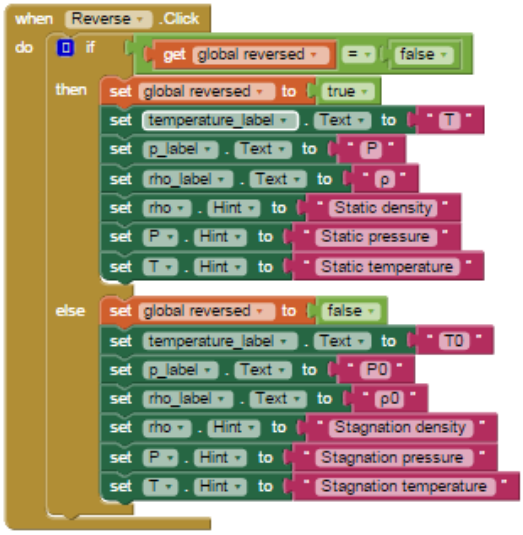
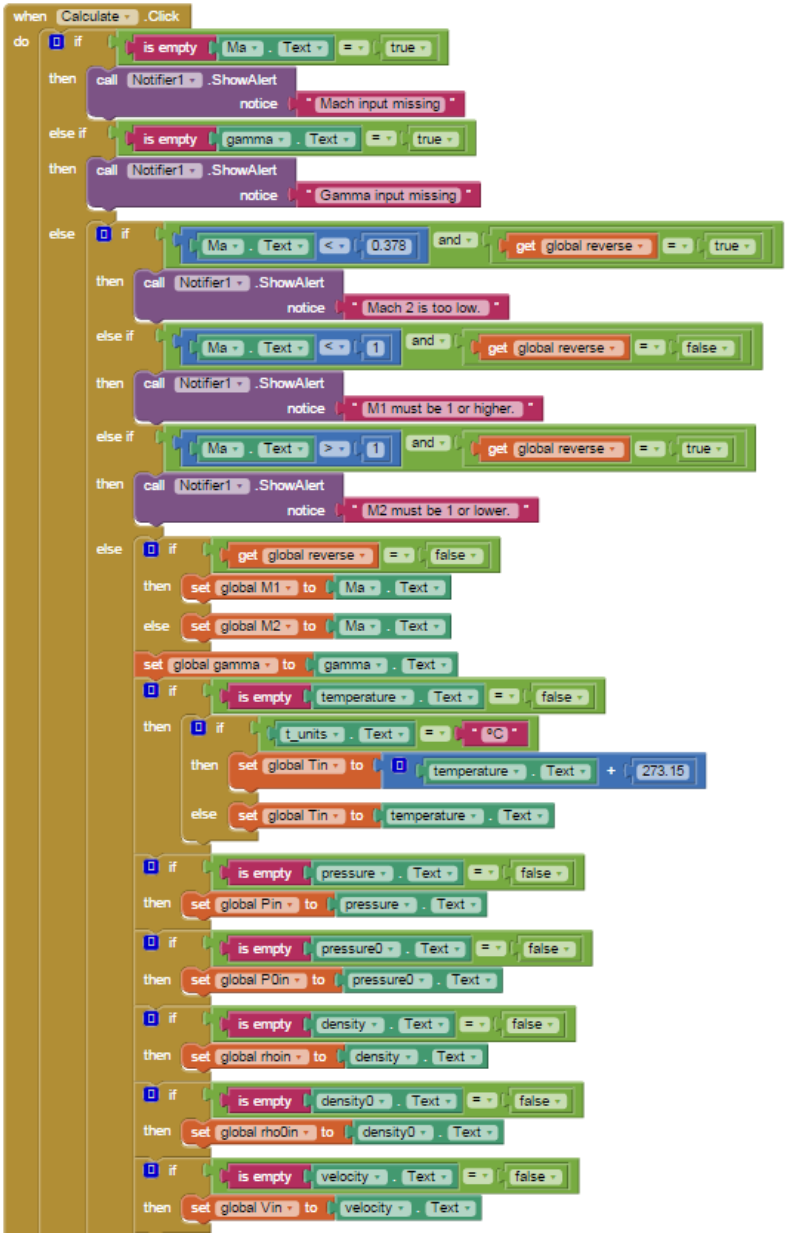


Ilustración A.9. Cambio de etiquetas para los casos directo e inverso

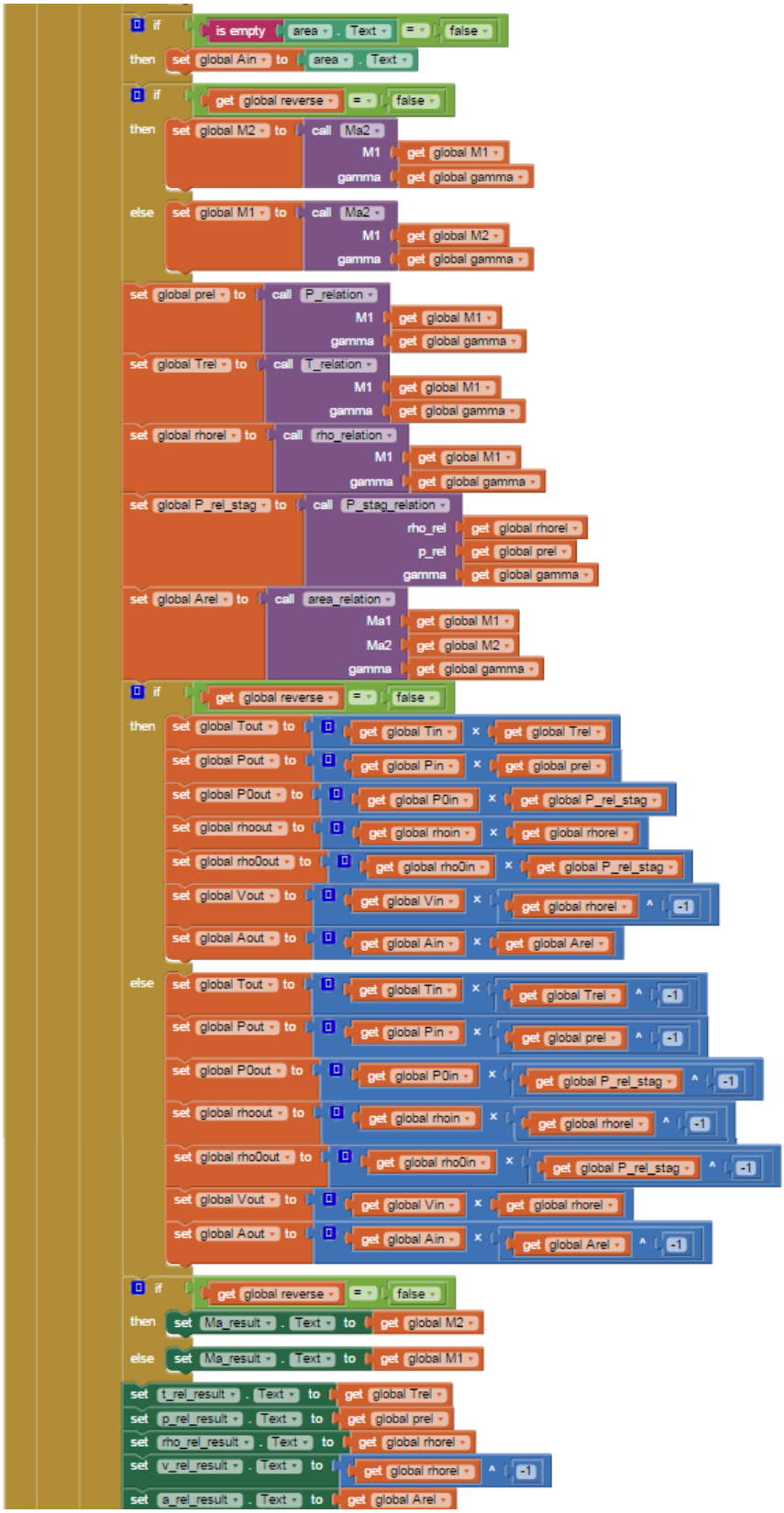
En él se cambian las etiquetas de los inputs según corresponde, para señalizar correctamente qué valor se está introduciendo (estático o estancamiento), y se opera el booleano "reversed" para indicar qué problema se realiza.

A.3. Onda de choque normal

A.3.1. Bloques



(Continúa en la página siguiente)



(Continúa en la página siguiente)

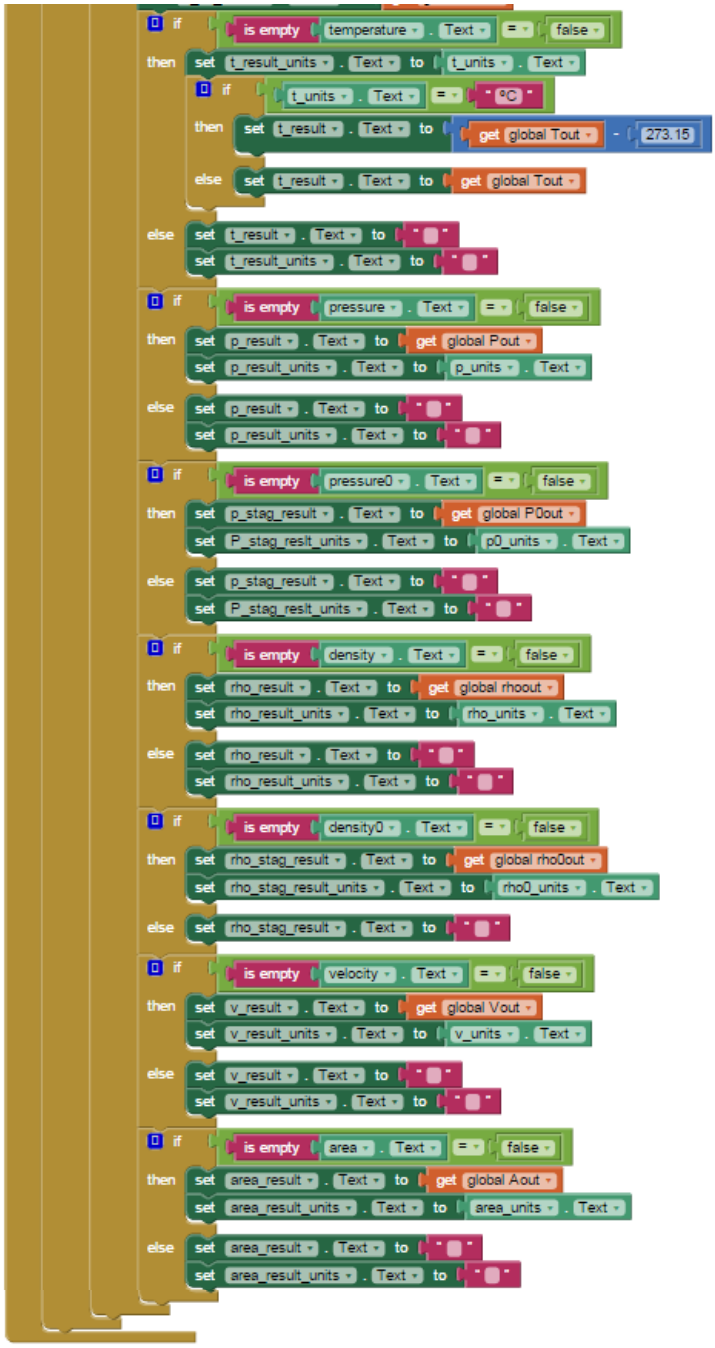


Ilustración A.10. Bloque principal de la función onda de choque

Las funciones utilizadas (bloques de color morado, marcados con “call”) son las operaciones matemáticas utilizadas, que se encuentran en la siguiente página.

- Procedimiento "Ma2". Calcula el Ma_1, antes de la onda de choque. Es la ecuación (3.19) de la memoria. Tiene la particularidad de que aislada para obtener Ma_1 a partir de Ma_2, queda de la misma forma que antes con los números de Mach intercambiados. Eso significa que se puede usar la misma función para ambos problemas, directo e inverso.

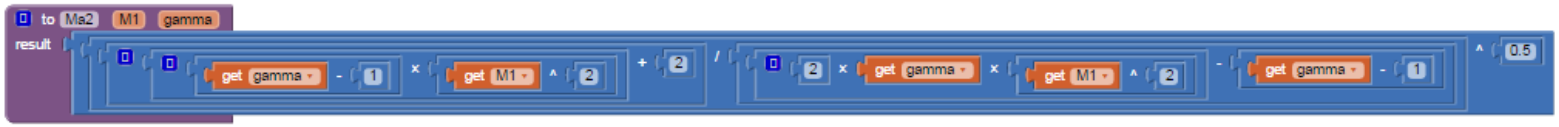


Ilustración A.11. Procedimiento Ma2

- Procedimiento "P_relation". Relaciona las presiones estáticas antes y después de la onda de choque, de la forma P_2/P_1. Ecuación (3.20).

Ilustración A.12. Procedimiento P_relation

- Procedimiento "T_relation". Relaciona las temperaturas estáticas de antes y después de la onda de choque, como T_2/T_1. Ecuación (3.21).

Ilustración A.13. Procedimiento T_relation

- Procedimiento “rho_relation”. Como los anteriores, relaciona densidades estáticas. Ecuación(3.22).

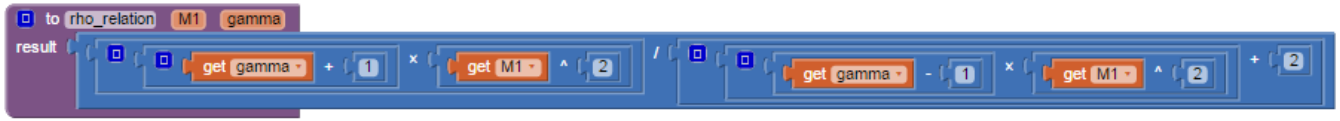


Ilustración A.14. Procedimiento rho_relation

- Procedimiento “P_stag_relation”. Relaciona presiones y densidades de estancamiento de la misma forma que las anteriores. Ecuación(3.23).

Ilustración A.15. Procedimiento P_stag_relation

- Procedimiento “area_relation”. Relaciona áreas de garganta. Ecuación(3.31).

Ilustración A.16. Procedimiento area_relation

A.4. Tobera convergente-divergente

A.4.1. Modo tobera adaptada

Presentación

A continuación se muestran algunos aspectos de la presentación del modo tobera adaptada en la función tobera convergente-divergente. Como por ejemplo, la selección del input a introducir:

Ilustración A.17. Selección del input en modo tobera adaptada

De esos inputs, el de la relación de áreas resulta el más particular. Si se recuerda la teoría, la ecuación que relaciona esas áreas con el número de Mach del fluido (ecuación (3.30)), para una relación de áreas dada, tiene dos soluciones: una supersónica y una supersónica. Para cada una hay una relación de presión, temperatura y densidad, así que se tiene que mostrar ambos casos:

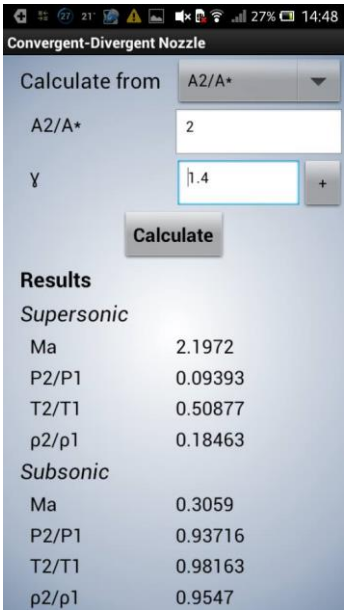
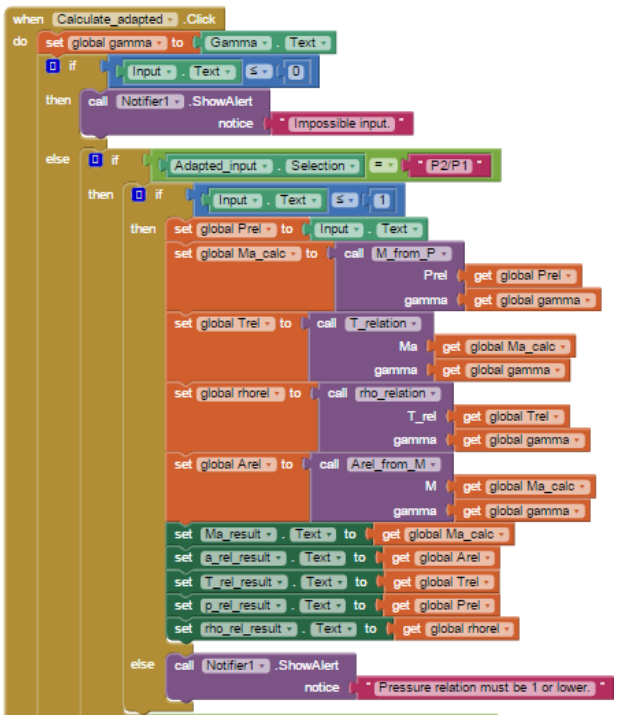


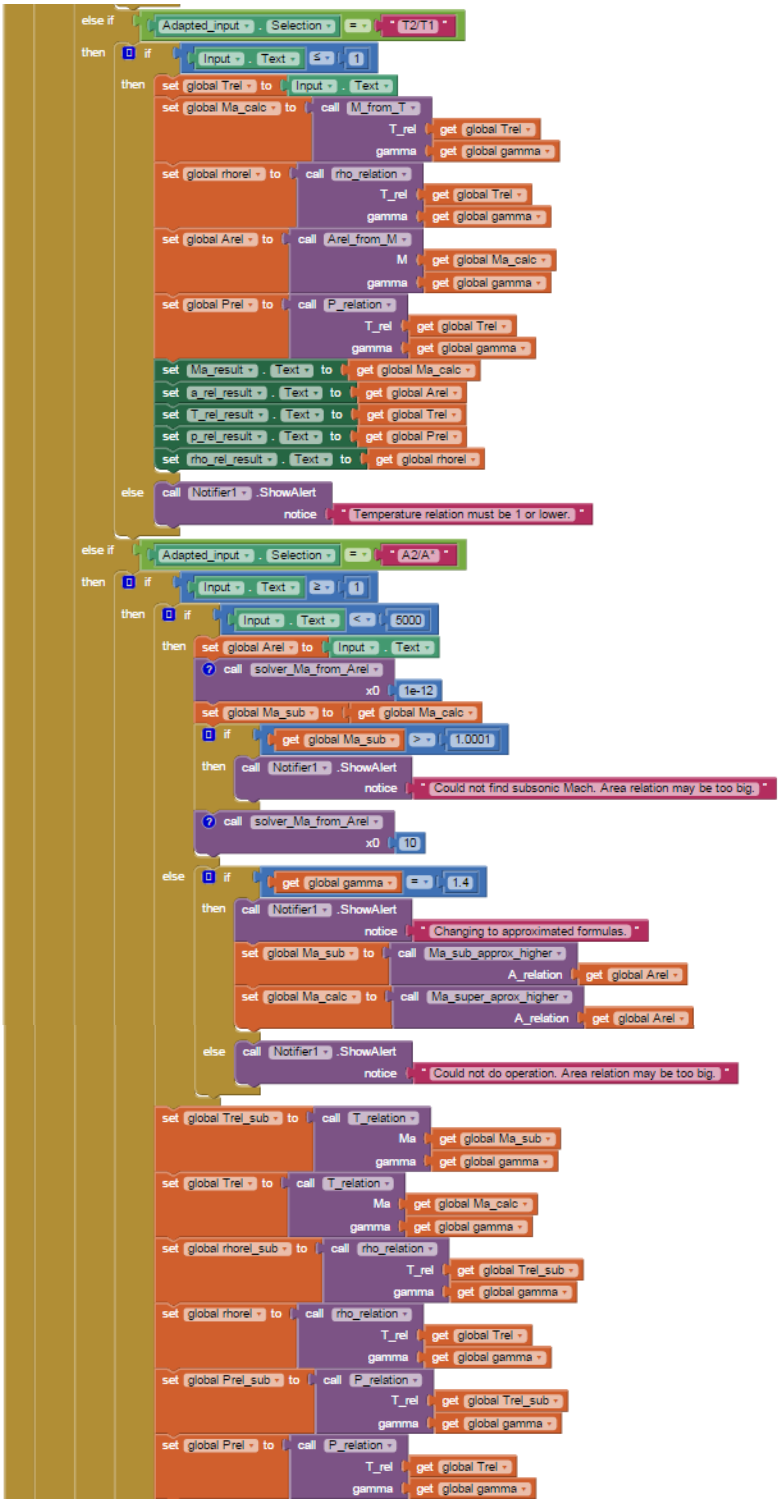
Ilustración A.18. Resultados para el modo tobera adaptada con relación de áreas como input

Bloques

A continuación se muestra el bloque principal del modo tobera adaptada de la función:



(Continúa en la página siguiente)



(Continúa en la página siguiente)

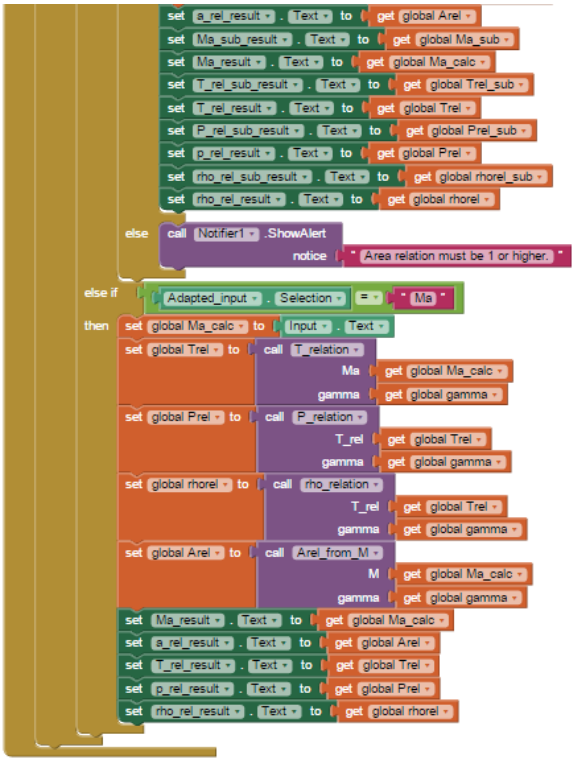


Ilustración A.19. Bloque principal del modo tobera adaptada

Se puede observar que el programa separa los diferentes casos según el input y opera de forma diferente en cada uno. Esto se hace para evitar recalcular el dato introducido, aunque hace que resulte un bloque más grande. Como se usan varias veces en el mismo bloque, las relaciones de presión, temperatura y densidad en función del número de Mach se han hecho funciones, pero siguen teniendo los mismos cálculos que en la función de flujo isentrópico.

Las funciones nuevas en esta función son:

- Función “M_from_P”, que es la ecuación (3.20), aislada para obtener el Mach a partir de la relación de presiones:

Ilustración A.20. Función M_from_P

- Función “M_from_T”, que es la ecuación (3.21) aislada para obtener el Mach a partir de la relación de temperaturas:

Ilustración A.21. Función M_from_T

- Función “Arel_from_M”, que es la ecuación (3.30) utilizada para obtener la relación de áreas a partir del Mach. Como no se puede aislar el Mach de ella, se usa también en la iteración.

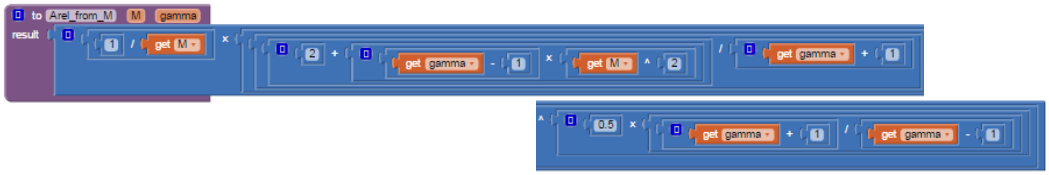


Ilustración A.22. Función Arel_from_M

La iteración de Newton-Raphson se hace en forma de función, pero a diferencia de las utilizadas hasta ahora, esta función no devuelve un resultado, sino que manipula las variables globales del problema:

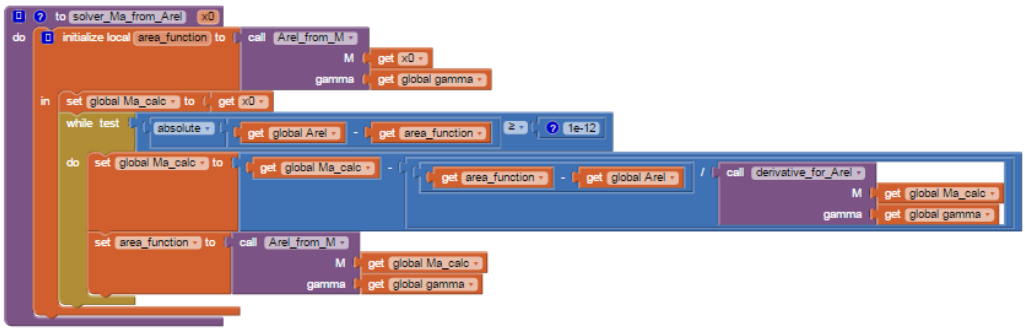


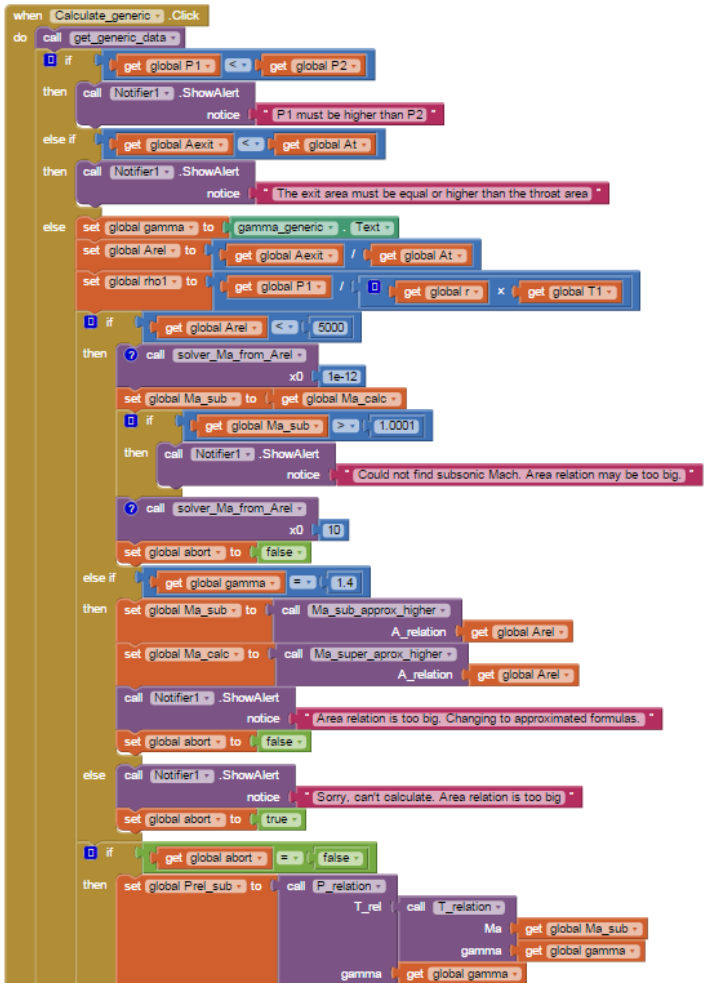
Ilustración A.23. Proceso iterativo solver_Ma_from_Arel

Donde la derivada se implementa de la forma

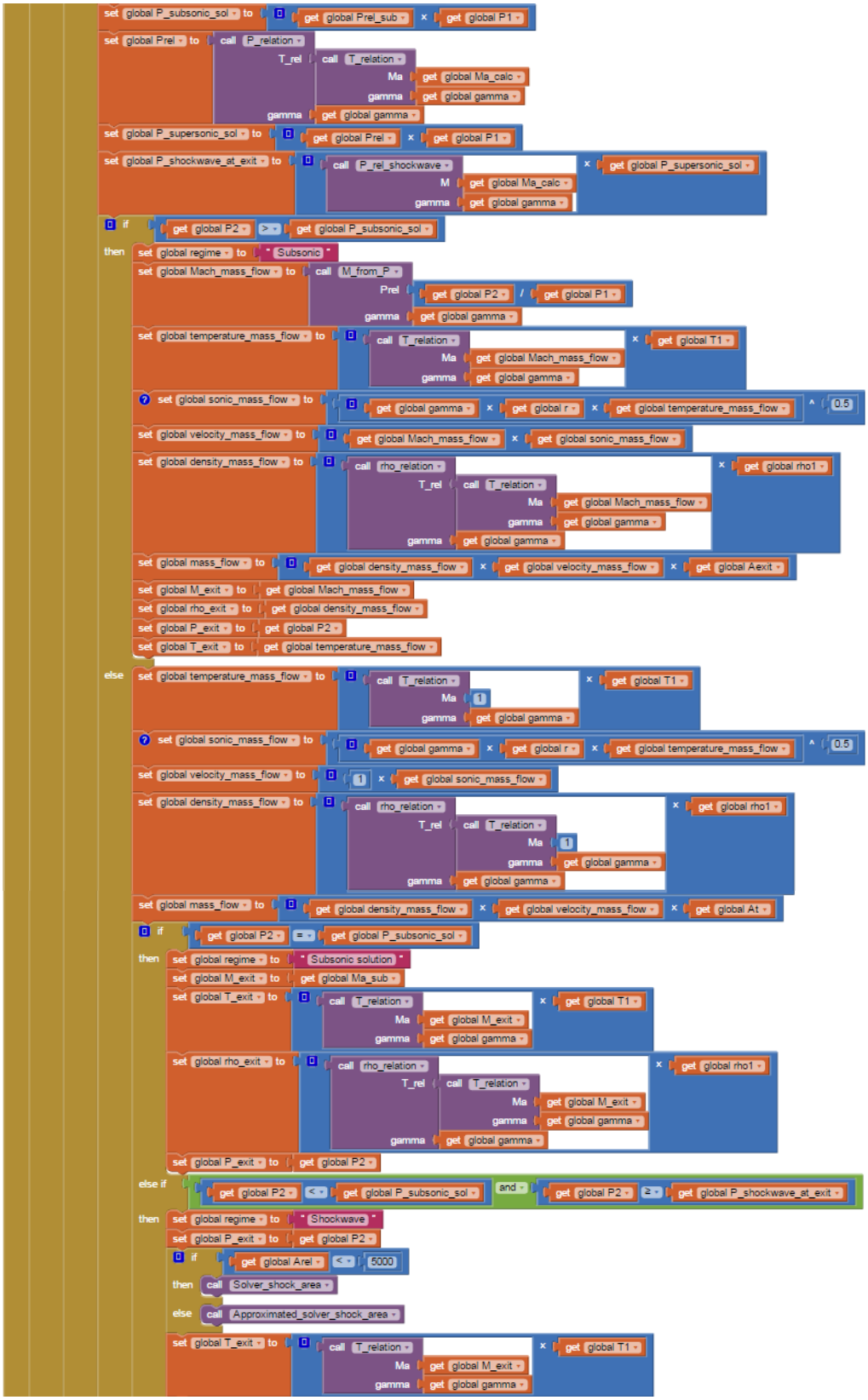
Ilustración A.24. Derivada numérica derivative_for_Arel

A.4.2. Modo genérico

A continuación se muestran los bloques principales de este modo:



(Continúa en la siguiente página)



(Coninúa en la siguiente página)

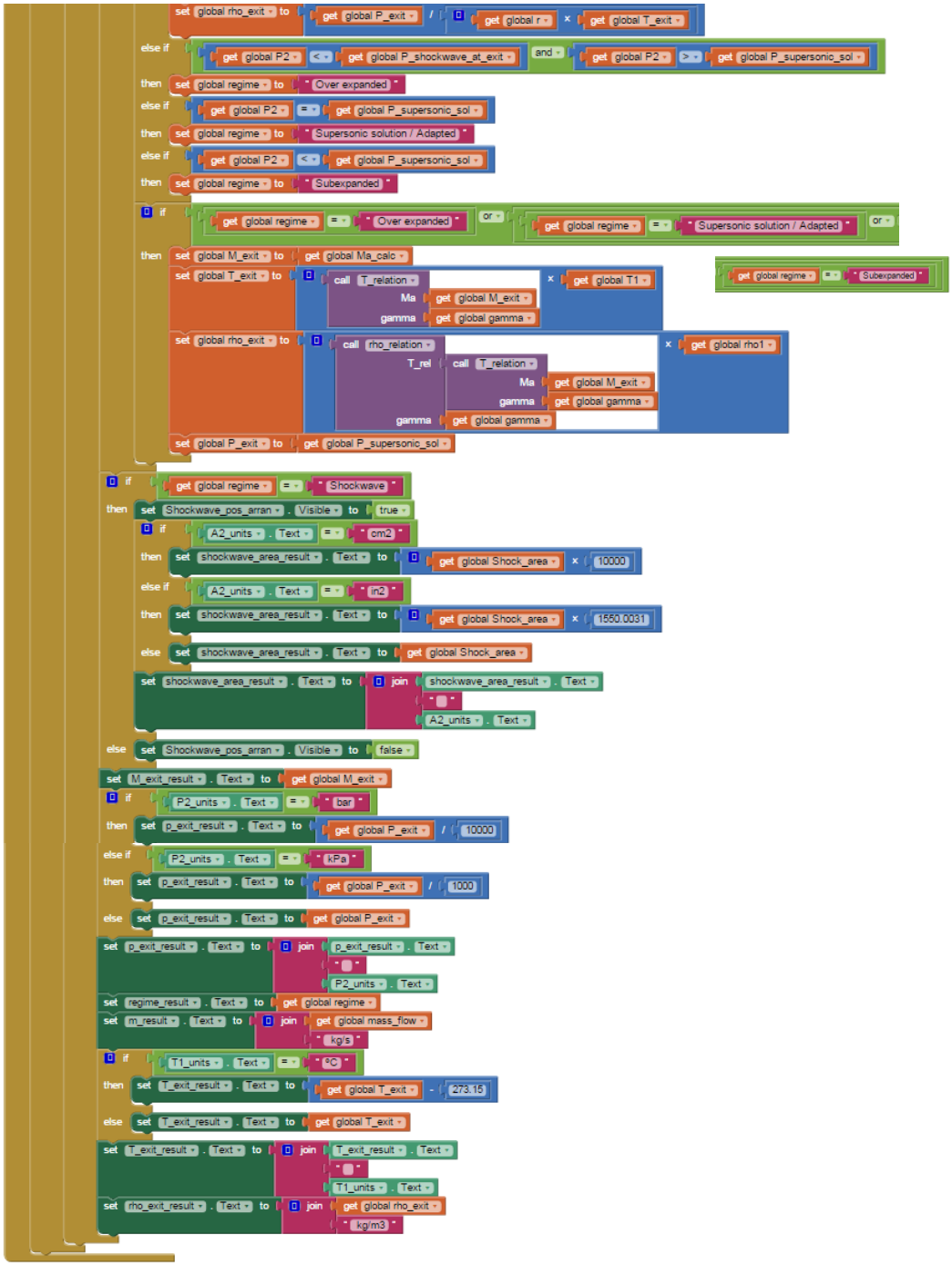


Ilustración A.25. Bloque principal del modo genérico

Donde el algoritmo para encontrar la onda de choque, llamado “Solver_shock_area” es

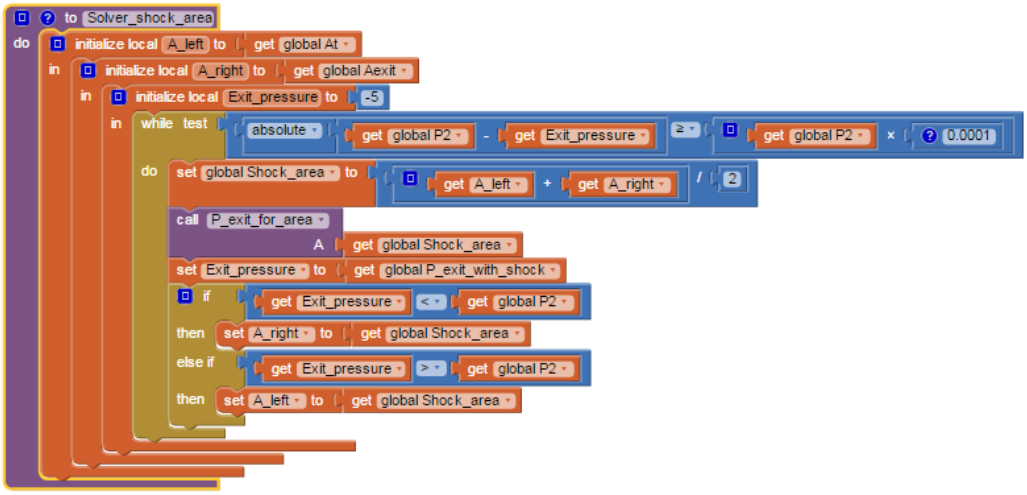


Ilustración A.26. Proceso iterativo Solver_shock_area

Se puede ver que la tolerancia para dicha iteración es del 0.01% de P_2. Esto asegura una buena exactitud de resultados cuando se opera con presiones bajas, mientras que mejora la velocidad de convergencia cuando se opera con presiones altas.

El bloque para encontrar la presión de salida dada un área de onda de choque es el siguiente:

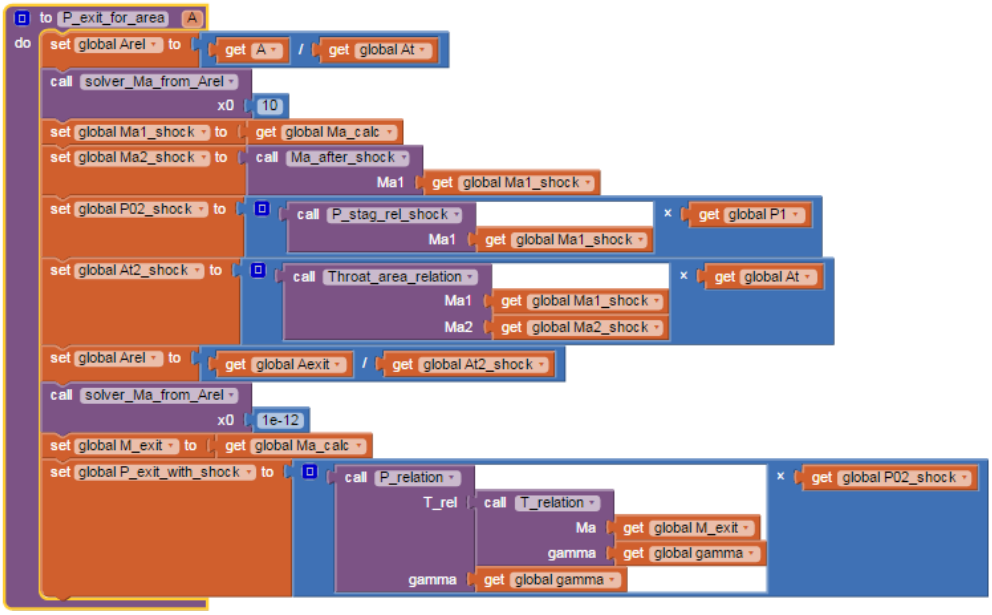


Ilustración A.27. Proceso para encontrar la presión en la salida P_exit_for_area

Igual que el modo tobera adaptada, la iteración para encontrar el Mach a partir de la relación de áreas no funciona para valores de relaciones mayores de 5000, por lo que se usan las fórmulas aproximadas para el caso del aire y se cancela para otros valores de gamma.

A.5. Flujo de Fanno

A.5.1. Propiedades de gases y materiales incluidos en la App

En la función de flujo de Fanno se incluyen, por comodidad del usuario, valores predefinidos para la constante adiabática, constante específica y viscosidad absoluta de diferentes gases. Dichos datos se toman del libro *Mecánica de Fluidos* de Frank M. White, a 20ºC y 1 atmósfera. La lista de gases se ha reducido ligeramente respecto de otras funciones, y el dato nuevo es el de la viscosidad absoluta, que se muestra a continuación:

Gas	Viscosidad absoluta μ [N*s/m^2] (*10^{-5})
H_2	0.905
He	1.97
H_20	1.02
Ar	2.24
Aire seco	1.8
CO_2	1.48
CO	1.82
N_2	1.76
O_2	2.00
NO	1.90
N_2O	1.45
Cl_2	1.03
CH_4	1.34

Tabla A.2. Viscosidad de gases comunes a 1 atm y 20ºC

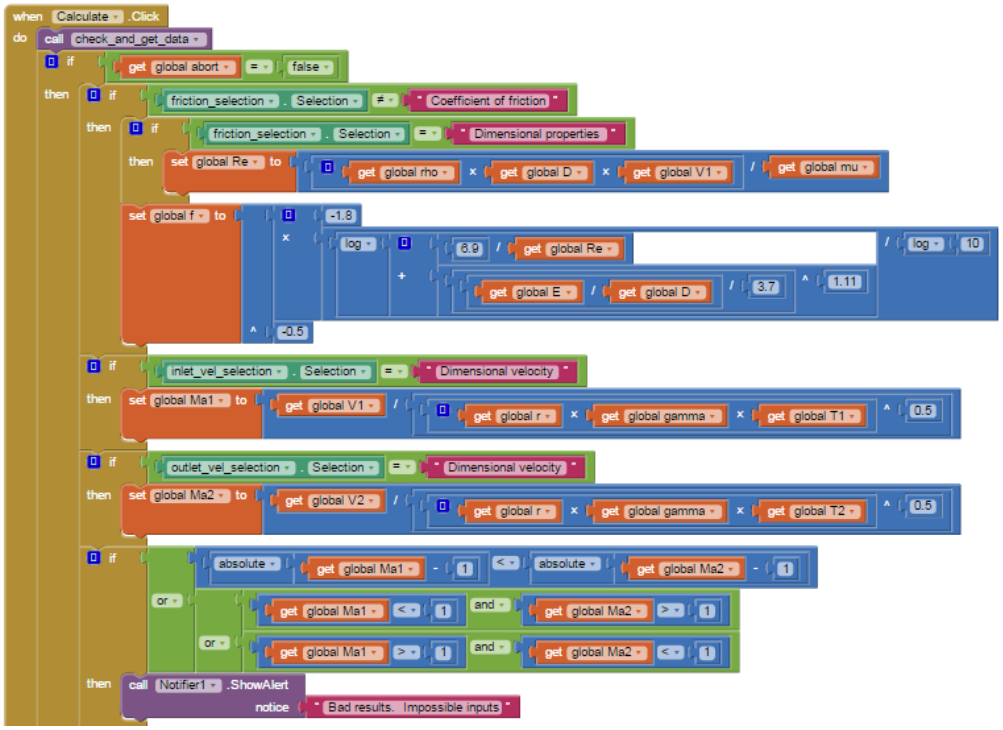
Por otra parte, también se incluyeron las rugosidades típicas de algunos materiales, para servir de referencia al usuario. Dichos datos fueron tomados de la misma fuente, y son los siguientes:

Material	Condición	ε [mm]
Acero	Inoxidable	0.02
	Comercial, nuevo	0.046
	Oxidado	2.0
Hierro	Fundido	0.26
	Forjado	0.046
Latón	Laminado	0.002
Plástico	Tubo laminado	0.0015
Vidrio	-	0
Hormigón	Liso	0.04
	Rugoso	2.0
Caucho	Liso	0.01

Tabla A.3. Valores típicos de rugosidad para conductos comerciales

A.5.2. Bloques

El bloque principal de la función es el siguiente:



(Continua en la página siguiente)

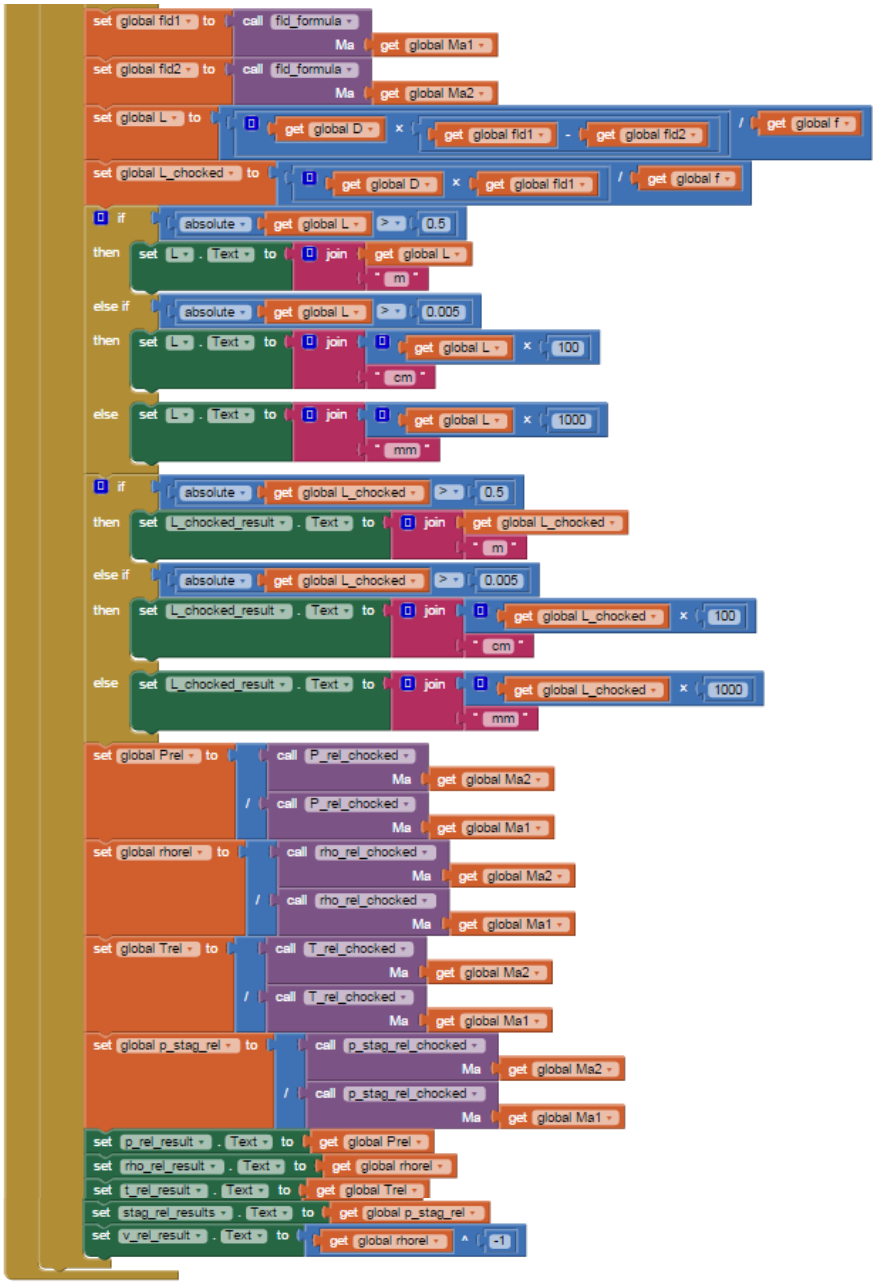


Ilustración A.28. Bloque principal de la función flujo de Fanno

Se puede ver que el programa al principio llama a la función "check_and_get_data", cuya función es detectar qué inputs debe introducir el usuario, comprobar que son válidos y leerlos. Eso hace que el programa principal sea mucho más limpio y aplicado al problema físico. Sin embargo, es un bloque grande y no aporta nueva información teórica, por lo que se omite en este anexo.

A continuación se muestran las demás funciones que utiliza el programa, que son las ecuaciones y fórmulas de la teoría:

Ilustración A.29. Función fld_formula

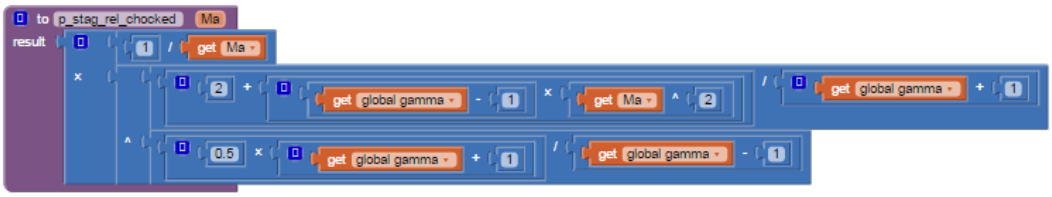


Ilustración A.30. Función p_stag_rel_chocked

Ilustración A.31. Función T_rel_chocked

Ilustración A.32. Función rho_rel_chocked

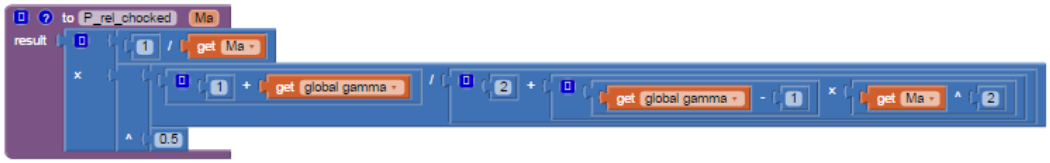


Ilustración A.33. Función P_rel_chocked

En esta función no se lleva a cabo ninguna iteración, por lo que no hay valores numéricos que elegir. Se puede comentar que las unidades del resultado de la

longitud dependen del valor de dicha magnitud. En función de la longitud del conducto, se expresa en metros, centímetros o milímetros.

B. Códigos de verificación de resultados

B.1. Atmósfera ISA

El siguiente es el código para obtener los resultados de la función ISA y crear las gráficas comparativas:

```
clear
close all
T0=288.15; %K
P0=101325; %Pa
%h=20000; %m
g=9.80665;
r=287.053;
rho0=P0/(r*T0);
a=-6.5*10^-3;
h=linspace(0,20000,20000);
for i=1:length(h)
if ((h(i)>=-1000)&&(h(i)<=11000))
    T(i)=T0+a*h(i);
    P(i)=P0*(T(i)/T0)^(-g/(a*r));
    rho(i)=rho0*(T(i)/T0)^((-g/(a*r))-1);

elseif (h(i)>11000)&&(h(i)<=20000)
    T(i)=T0+a*11000;
    P(i)=P0*(T(i)/T0)^(-g/(a*r));
    rho(i)=rho0*(T(i)/T0)^((-g/(a*r))-1);
    P(i)=P(i)*exp(-g/(r*T(i))*(h(i)-11000));
    rho(i)=rho(i)*exp(-g/(r*T(i))*(h(i)-11000));
end
end
T_app=[288.15 287.5 284.9 281.65 262.15 249.15 236.15 216.65 216.65
216.65 216.65 216.65 216.65];
h_app=[0 100 500 1000 4000 6000 8000 11000 14000 16000 17500 19000
20000];
% plot(h,T);
% hold on
% plot(h_app,T_app,'x');
% xlabel('Altitud [m]');
% ylabel('Temperatura [K]');
% grid;
% %fprintf('%f %f %f %f\r\n',h,T,P,rho);
% legend('Verificación por Matlab','Resultados de la App');

% P_app=[101325 100128.81274 95457.84253 89868.89 61624.10469 47162.04
35580.2088 22614.20669 14087.16322 10275.07513 8109.7245 6400.69592
5466.48104];
% plot(h,P);
% hold on
% plot(h_app,P_app,'x');
% xlabel('Altitud [m]');
% ylabel('Presión [Pa]');
% grid;
% legend('Verificación por Matlab','Resultados de la App');

rho_app=[1.22523 1.2135 1.16745 1.11178 0.81907 0.65955 0.52498 0.3637
0.22656 0.16525 0.13043 0.10294 0.08792];
```

```
plot(h,rho);
hold on
plot (h_app,rho_app,'x');
xlabel('Altitud [m]');
ylabel('Densidad [kg/m3]');
grid;
legend('Verificación por Matlab','Resultados de la App');
```

B.2. Flujo isentrópico

El siguiente es código de Matlab utilizado para comprobar los resultados de la función flujo isentrópico y crear las gráficas comparativas:

```
clear
close all
T0=288.15; %K
P0=101325; %Pa
rho0=1.225; %kg/m3
gamma=1.4;
Ma=linspace(0,15);
for i=1:length(Ma)
    P(i)=P0*(1+(gamma-1)/2*Ma(i)^2)^(-gamma/(gamma-1));
    T(i)=T0*(1+(gamma-1)/2*Ma(i)^2)^-1;
    rho(i)=rho0*(1+(gamma-1)/2*Ma(i)^2)^(-1/(gamma-1));
end
M_app=[0 0.5 1 3 5 10 15];
T_app=[288.15 274.42857 240.125 102.91071 48.025 13.72143 6.26413];
P_app=[101325 85418.91 53528.15214 2758.43975 191.50814 2.38753 0.15348];
rho_app=[1.225 1.08433 0.77657 0.09338 0.01389 0.00061 0.00009];
% plot(Ma,T);
% hold on;
% plot(M_app,T_app,'x');
% grid;
% xlabel('Número de Mach');
% ylabel('Temperatura [K]');
% legend('Verificación por Matlab','Resultados de la App');

% plot(Ma,P);
% hold on;
% plot(M_app,P_app,'x');
% grid;
% xlabel('Número de Mach');
% ylabel('Presión [Pa]');
% legend('Verificación por Matlab','Resultados de la App');

% plot(Ma,rho);
% hold on;
% plot(M_app,rho_app,'x');
% grid;
% xlabel('Número de Mach');
% ylabel('Densidad [kg/m3]');
% legend('Verificación por Matlab','Resultados de la App');

P=10000;
T=300;
rho=1;
for i=1:length(Ma)
    P0(i)=P*(1+(gamma-1)/2*Ma(i)^2)^(gamma/(gamma-1));
```

```
    T0(i)=T*(1+(gamma-1)/2*Ma(i)^2)^1;
    rho0(i)=rho*(1+(gamma-1)/2*Ma(i)^2)^(1/(gamma-1));
end
M_app=[0 0.5 1 5 15];
T0_app=[300 315 360 1800 13800];
P0_app=[10000 11862.1264 18929.29159 5.2909e6 6.6016e9];
rho0_app=[1 1.12973 1.57744 88.18163 14351.41024];

% plot(Ma,T0);
% hold on;
% plot(M_app,T0_app,'x');
% grid;
% xlabel('Número de Mach');
% ylabel('Temperatura de estancamiento [K]');
% legend('Verificación por Matlab','Resultados de la App');

% plot(Ma,P0);
% hold on;
% plot(M_app,P0_app,'x');
% grid;
% xlabel('Número de Mach');
% ylabel('Presión de estancamiento [Pa]');
% legend('Verificación por Matlab','Resultados de la App');

plot(Ma,rho0);
hold on;
plot(M_app,rho0_app,'x');
grid;
xlabel('Número de Mach');
ylabel('Densidad de estancamiento [kg/m3]');
legend('Verificación por Matlab','Resultados de la App');
```

B.3. Onda de choque

El siguiente es el código para obtener los resultados de la función onda de choque y crear las gráficas comparativas:

```
clear
close all
gamma=1.4;
P1=101325;
T1=300;
P0_1=101325;
rho1=1.225;
rho0_1=1.225;
v1=100;
A1=1;
Ma1=linspace(1,10);
for i=1:length(Ma1);
P2(i)=P1*(((2*gamma*Ma1(i)^2)-(gamma-1))/(gamma+1));
Ma2(i)=sqrt((((gamma-1)*Ma1(i)^2)+2)/((2*gamma*Ma1(i)^2)-(gamma-1)));
Astar2(i)=A1*(Ma2(i)/Ma1(i))*((2+((gamma-1)*Ma1(i)^2))/(2+((gamma-
1)*Ma2(i)^2)))^(0.5*(gamma+1)/(gamma-1));
rho2(i)=rho1*((gamma+1)*Ma1(i)^2)/((gamma-1)*Ma1(i)^2+2);
v2(i)=v1*(((gamma+1)*Ma1(i)^2)/((gamma-1)*Ma1(i)^2+2))^-1;
```

```
rho0_2(i)=rho0_1*(rho2(i)/rho1)^(gamma/(gamma-1))*(P1/P2(i))^(1/(gamma-
1));
P0_2(i)=P0_1*(rho2(i)/rho1)^(gamma/(gamma-1))*(P1/P2(i))^(1/(gamma-1));
T2(i)=T1*((2+(gamma-1)*Ma1(i)^2)*(2*gamma*Ma1(i)^2-(gamma-
1))/((gamma+1)^2*Ma1(i)^2));
end
Ma1_app=[1 2 4 7 10];
Ma2_app=[1 0.57735 0.43496 0.39736 0.38758];
T2_app=[300 506.25 1214.0625 3140.82 6116.25];
P2_app=[101325 455962.5 1.8745e6 5.776e6 1.18e7];
P0_2_app=[101325 73042.544 14059.5 1555.5 308.51];
rho2_app=[1.225 3.26607 5.6 6.67 7];
rho0_2_app=[1.225 0.88307 0.16998 0.01881 0.00373];
v2_app=[100 37.5 21.875 18.37 17.5];
A2_app=[1 1.3872 7.207 65.14 328.434];

% plot(Ma1,Ma2);
% hold on;
% plot(Ma1_app,Ma2_app,'x');
% xlabel('Ma 1 (antes)');
% ylabel('Ma 2 (después)');
% legend('Verificación con Matlab','Resultados de la App');
% grid;

plot(Ma1,P2);
hold on;
plot(Ma1_app,P2_app,'x');
xlabel('Ma 1 (antes)');
ylabel('Presión estática después [Pa]');
legend('Verificación con Matlab','Resultados de la App');
grid;

% plot(Ma1,P0_2);
% hold on;
% plot(Ma1_app,P0_2_app,'x');
% xlabel('Ma 1 (antes)');
% ylabel('Presión de estancamiento después [K]');
% legend('Verificación con Matlab','Resultados de la App');
% grid;

% plot(Ma1,T2);
% hold on;
% plot(Ma1_app,T2_app,'x');
% xlabel('Ma 1 (antes)');
% ylabel('Temperatura estática después [K]');
% legend('Verificación con Matlab','Resultados de la App');
% grid;

% plot(Ma1,rho2);
% hold on;
% plot(Ma1_app,rho2_app,'x');
% xlabel('Ma 1 (antes)');
% ylabel('Densidad estática después [kg/m3]');
% legend('Verificación con Matlab','Resultados de la App');
% grid;

% plot(Ma1,rho0_2);
% hold on;
% plot(Ma1_app,rho0_2_app,'x');
% xlabel('Ma 1 (antes)');
% ylabel('Densidad de estancamiento después [kg/m3]');
% legend('Verificación con Matlab','Resultados de la App');
% grid;

% plot(Ma1,v2);
```

```
% hold on;
% plot(Ma1_app,v2_app,'x');
% xlabel('Ma 1 (antes)');
% ylabel('Velocidad después [m/s]');
% legend('Verificación con Matlab','Resultados de la App');
% grid;

% plot(Ma1,Astar2);
% hold on;
% plot(Ma1_app,A2_app,'x');
% xlabel('Ma 1 (antes)');
% ylabel('Área de garganta después');
% legend('Verificación con Matlab','Resultados de la App');
% grid;
```

B.4. Tobera convergente-divergente

B.4.1. Modo tobera adaptada

Relación de áreas como input

En este código, llamado Arel_input.m, se calculan las propiedades para el modo tobera adaptada de la función tobera convergente-divergente, para el caso de tener la relación de áreas como input.

```
clear
close all
%Function that gives the pressure and Mach given an area ratio
Area/throat
%area (A2a)
P1=100000; %Pa
T1=300; %K
rho1=1.225;
gamma=1.4;
A2a=linspace(1,5);
for j=1:length(A2a)

    %Area relation (A2/A*) as a function of Ma at the exit. Ma cannot be
    %isolated, so we find it by iteration
    Area_ratio=@(Ma) (1/Ma*((2+(gamma-
1)*Ma^2)/(gamma+1))^((1/2)*(gamma+1)/(gamma-1)))-A2a(j);
    %(This function has to be 0)
    options= optimset('Display','none','TolFun',1e-12,'TolX',1e-12);%We
define the tolerance
    %The Mach equation for a given area relation has two solutions, one
    %subsonic and other supersonic.

    e=0.001; %tolerance for comparing double solution M=1

    %We find the subsonic Mach trying to approach from low starting
points
    for i=0.01:0.01:1
        Ma2_sub(j)=fsolve (Area_ratio, i, options);
        if Ma2_sub(j)<=1
            break
        end
    end
```

```
    %We find the supersonic Mach approaching from higher starting points
    for i=2*A2a:1000*A2a
        Ma2_super(j)=fsolve (Area_ratio, i, options);
        if Ma2_super(j)>=1
            break
        end
    end

    if ((Ma2_sub(j)==1||Ma2_super(j)==1)&&(abs(Ma2_sub(j)-
Ma2_super(j))>=e))||(Ma2_sub(j)>1)||(Ma2_super(j)<1);
        %If both machs are equal, they must be 1.
        %Ma subsonic must be <1 and Ma supersonic must be >1
        error('Error in finding the Ma at the exit given area ratio');
    end
    %By obtaining the Mach we can now compute the propierties at the
outlet
    Pexit(j)=property_mach(P1, Ma2_super(j), gamma,'P');
    Pexit_sub(j)=property_mach(P1,Ma2_sub(j),gamma,'P');
    Texit(j)=property_mach(T1,Ma2_super(j),gamma,'T');
    Texit_sub(j)=property_mach(T1,Ma2_sub(j),gamma,'T');
    rhoexit(j)=property_mach(rho1,Ma2_super(j),gamma,'rho');
    rhoexit_sub(j)=property_mach(rho1,Ma2_sub(j),gamma,'rho');
    P_rel_super(j)=Pexit(j)/P1;
    P_rel_sub(j)=Pexit_sub(j)/P1;
    T_rel(j)=Texit(j)/T1;
    rho_rel(j)=rhoexit(j)/rho1;
end
A2a_app=[1 1.5 2 4 5];
Ma_super_app=[1 1.85412 2.1972 2.94018 3.17478];
Ma_sub_app=[1 0.43026 0.3059 0.14655 0.11669];
Prel_app=[0.52828 0.16018 0.09393 0.02979 0.02099];
T_rel_app=[0.8333 0.59257 0.50877 0.36644 0.33158];
rho_rel_app=[0.63394 0.27031 0.18463 0.08129 0.06331];

% plot(A2a,Ma2_super);
% hold on
% plot(A2a_app,Ma_super_app,'x');
% grid;
% xlabel('Relación de áreas A2/A*');
% ylabel('Mach supersónico');
% legend('Verificación por Matlab','Resultados de la App');

% plot(A2a,Ma2_sub);
% hold on
% plot(A2a_app,Ma_sub_app,'x');
% grid;
% xlabel('Relación de áreas A2/A*');
% ylabel('Mach subsónico');
% legend('Verificación por Matlab','Resultados de la App');

% plot(A2a,P_rel_super);
% hold on
% plot(A2a_app,Prel_app,'x');
% grid;
% xlabel('Relación de áreas A2/A*');
% ylabel('Relación de presiones P2/P1');
% legend('Verificación por Matlab','Resultados de la App');

% plot(A2a,T_rel);
% hold on
% plot(A2a_app,T_rel_app,'x');
% grid;
% xlabel('Relación de áreas A2/A*');
% ylabel('Relación de temperaturas T2/T1');
```

```
% legend('Verificación por Matlab','Resultados de la App');

plot(A2a,rho_rel);
hold on
plot(A2a_app,rho_rel_app,'x');
grid;
xlabel('Relación de áreas A2/A*');
ylabel('Relación de densidades');
legend('Verificación por Matlab','Resultados de la App');
```

Se puede observar que el código llama a la función property_mach, cuyo código es el siguiente:

```
function [ fi ] = property_mach( fi0, Ma,gamma,property)
%   Determines the value of a given propierty as a function of the Mach
%   number and the stagnation value of such propierty
%   property can be pressure 'P', temperature 'T' or density 'rho'
if property=='P'
    %fi0=fi1*(1+(gamma-1)/2*Ma^2)^(gamma/(gamma-1))
    fi=fi0*(1+(gamma-1)/2*Ma^2)^(-gamma/(gamma-1));
elseif property=='T'
    fi=fi0*(1+(gamma-1)/2*Ma^2)^-1;
elseif strcmp(property,'rho')
    fi=fi0*(1+(gamma-1)/2*Ma^2)^(-1/(gamma-1));
end
end
```

Relación de temperaturas como input

En el siguiente código, llamado Trel_input.m, se calcula el número de Mach a partir de la relación de temperaturas y se crea un gráfico comparativo:

```
clear
close all
gamma=1.4;
T1=300;
Trel=linspace(0.001,0.9);
options= optimset('Display','none','TolFun',1e-12,'TolX',1e-12);
for i=1:length(Trel)
    T2=T1*Trel(i);
    fun= @(Ma) property_mach(T1,Ma,gamma,'T')-T2;
    Ma(i)=fsolve(fun,1,options);
    Trel(i)=T2/T1;
end
Ma_app=[6.7082 2.23607 1.46385 0.74536];
Prel_app=[0.1 0.5 0.7 0.9];

plot(Trel, Ma);
hold on
plot(Prel_app,Ma_app,'x');
grid;
xlabel('Relación de temperaturas T2/T1');
ylabel('Mach de salida');
legend('Verificación por Matlab','Resultados de la App');
```

Relación de presiones como input

En éste se calcula el Mach a partir de la relación de presiones. Se llama Prel_input.m.

```
clear
close all
gamma=1.4;
P1=100000;
Prel=linspace(0.001,0.9);
options= optimset('Display','none','TolFun',1e-12,'TolX',1e-12);
for i=1:length(Prel)
    P2=P1*Prel(i);
    fun= @(Ma) property_mach(P1,Ma,gamma,'P')-P2;
    Ma(i)=fsolve(fun,1,options);
    Prel(i)=P2/P1;
    %Area_rel(i)=1/Ma*((2+(gamma-
1)*Ma^2)/(gamma+1))^((1/2)*(gamma+1)/(gamma-1));
end
Ma_app=[3.69296 2.15719 1.04646 0.7324 0.3909];
Prel_app=[0.01 0.1 0.5 0.7 0.9];

plot(Prel, Ma);
hold on
plot(Prel_app,Ma_app,'x');
grid;
xlabel('Relación de presiones P2/P1');
ylabel('Mach de salida');
legend('Verificación por Matlab','Resultados de la App');
```

B.4.2. Modo genérico

Para la comprobación del modo genérico se usa, en primer lugar, el código llamado mass_flow_calculation, para comprobar el caudal másico:

```
clear
close all
P1=100000;
T1=300;
A2=2;
A2a=2;
gamma=1.4;
r=287.058;
P2=linspace(100000,50000,1000);
for i=1:length(P2)
regime='subsonic';

    %Properties of the throat are calculaed
    Pstar=property_mach( P1, 1,gamma,'P');
    rho1=P1/(r*T1);
    rhostar=property_mach(rho1,1,gamma,'rho');
    Tstar=property_mach(T1,1,gamma,'T');
    cstar=sqrt(gamma*r*Tstar);

    %The pressure limit, below it the nozzle is chocked needs to be
    %calculated:
```

```
    [P_super,M_super,P_sub,~]=P_ad( P1, A2a, gamma );

    Astar=A2/A2a; %We could send this, but we can calculate it again.

    if (P2(i)>P_sub)
        choked=false;
        %P1/P2=[1+(gamma-1)/2*Ma2^2]^[gamma/(gamma-1)] - Find Ma2
        Ma2=sqrt(2/(gamma-1)*((P1/P2(i))^((gamma-1)/gamma)-1)); %Might be
faster than solve for function property_mach
        %Complex formula for mass flow:
        %m_f=P1*sqrt(gamma/(r*T1))*Ma2*A2*(1+(gamma-
1)/2*Ma2^2)^((gamma+1)/(2*(1-gamma)));

        %May be easier doing:
        rho2=property_mach(rho1,Ma2,gamma,'rho');
        T2=property_mach(T1,Ma2,gamma,'T');
        u2=Ma2*sqrt(gamma*r*T2);
        m_f(i)=rho2*u2*A2;

    else
        choked=true;
        %Complex formula for mass flow (also correct)
        %m_f=P1*sqrt(gamma/(r*T1))*Astar*((gamma+1)/2)^((gamma+1)/(2*(1-
gamma)));
        %Easier formula for mass flow:
        m_f(i)=Astar*cstar*rhostar;

        %P_chocked is a limiting value for nozzles with shockwaves
        [~,P_forchocked,~,~] =
Shockwave(M_super,P_super,100,Astar,gamma); %The velocity is not
important, so a random value (100) is given.

        if P_sub==P2(i)
            regime='sub_solution';

        elseif (P2(i)<P_sub && P2(i) > P_forchocked )
            regime='shockwave';

        elseif P2(i)==P_forchocked %A shockwave just at the exit
            regime='shockwave_exit';

        elseif P2(i)<P_forchocked && P2(i)>P_super
            regime='overexpanded';

        elseif P2(i)==P_super
            regime='adapted';

        elseif P2(i)<P_super
            regime='subexpanded';
        end

    end
    P_choked=Pstar;
end

plot(P2,m_f);
P2_app=[100 98 96 94 90 80 50]*10^3;
massflow_app=[0 134.837 188.58977 228.37447 233.33228 233.33228
233.33228];
hold on;
plot(P2_app,massflow_app,'x');
grid;
xlabel('Presión del ambiente de descarga P2 [Pa]');
ylabel('Flujo másico [kg/s]');
legend('Verificación por Matlab','Resultados de la App');
```

El código llama a las funciones “P_ad” y “Shockwave”, cuyos códigos se muestran a continuación:

P_ad:

```
function [Pexit,Ma2_super,Pexit_sub,Ma2_sub] = P_ad( P1, A2a, gamma )
%Function that gives the pressure and Mach given an area ratio
Area/throat
%area (A2a)

    %Area relation (A2/A*) as a function of Ma at the exit. Ma cannot be
    %isolated, so we find it by iteration
    Area_ratio=@(Ma) (1/Ma*((2+(gamma-
1)*Ma^2)/(gamma+1))^((1/2)*(gamma+1)/(gamma-1)))-A2a;
    %(This function has to be 0)
    options= optimset('Display','none','TolFun',1e-12,'TolX',1e-12);%We
define the tolerance
    %The Mach equation for a given area relation has two solutions, one
    %subsonic and other supersonic.

    e=0.001; %tolerance for comparing double solution M=1

    %We find the subsonic Mach trying to approach from low starting
points
    for i=0.01:0.01:1
        Ma2_sub=fsolve (Area_ratio, i, options);
        if Ma2_sub<=1
            break
        end
    end

    %We find the supersonic Mach approaching from higher starting points
    for i=2*A2a:1000*A2a
        Ma2_super=fsolve (Area_ratio, i, options);
        if Ma2_super>=1
            break
        end
    end

    if ((Ma2_sub==1||Ma2_super==1)&&(abs(Ma2_sub-
Ma2_super)>=e))||(Ma2_sub>1)||(Ma2_super<1);
        %If both machs are equal, they must be 1.
        %Ma subsonic must be <1 and Ma supersonic must be >1
        error('Error in finding the Ma at the exit given area ratio');
    end
    %By obtaining the Mach we can now compute the propierties at the
outlet
    Pexit=property_mach(P1, Ma2_super, gamma,'P');
    Pexit_sub=property_mach(P1,Ma2_sub,gamma,'P');
end
```

Shockwave:

```
function [ Ma2,P2,v2,Astar2] = Shockwave( Ma1,P1,v1,Astar1,gamma )
%This function calculaes the conditions after a normal shockwave
```

```
%They depend only on Ma1, is a magnitude is not wanted (v2, for example)
a
%random v1 can be sent.
P2=P1*(((2*gamma*Ma1^2)-(gamma-1))/(gamma+1));
Ma2=sqrt((((gamma-1)*Ma1^2)+2)/((2*gamma*Ma1^2)-(gamma-1)));
Astar2=Astar1*(Ma2/Ma1)*((2+((gamma-1)*Ma1^2))/(2+((gamma-
1)*Ma2^2)))^(0.5*(gamma+1)/(gamma-1));
v2=(v1*(2+((gamma-1)*Ma2^2)))/((gamma+1)*Ma1^2);
end
```

Siguiendo con la verificación de resultados, se utiliza el código llamado “generic_limits.m” para obtener los límites de presiones P2 correspondientes a una presión adaptada y a la presión de salida con una onda de choque en ella, que no es más que una modificación del código “P_ad”.

```
clear
close all
P1=100000;
T1=300;
A2=2;
A2a=linspace(1,3);
gamma=1.4;
r=287.058;
P2=0;
for i=1:length(A2a)
regime='subsonic';

    %Properties of the throat are calculaed
    Pstar=property_mach( P1, 1,gamma,'P');
    rho1=P1/(r*T1);
    rhostar=property_mach(rho1,1,gamma,'rho');
    Tstar=property_mach(T1,1,gamma,'T');
    cstar=sqrt(gamma*r*Tstar);

    %The pressure limit, below it the nozzle is chocked needs to be
    %calculated:
    [P_super(i),M_super,P_sub,~]=P_ad( P1, A2a(i), gamma );

    Astar=A2/A2a(i); %We could send this, but we can calculate it again.

    if (P2>P_sub)
        choked=false;
        %P1/P2=[1+(gamma-1)/2*Ma2^2]^[gamma/(gamma-1)] - Find Ma2
        Ma2=sqrt(2/(gamma-1)*((P1/P2)^((gamma-1)/gamma)-1)); %Might be
faster than solve for function property_mach
        %Complex formula for mass flow:
        %m_f=P1*sqrt(gamma/(r*T1))*Ma2*A2*(1+(gamma-
1)/2*Ma2^2)^((gamma+1)/(2*(1-gamma)));

        %May be easier doing:
        rho2=property_mach(rho1,Ma2,gamma,'rho');
        T2=property_mach(T1,Ma2,gamma,'T');
        u2=Ma2*sqrt(gamma*r*T2);
        m_f(i)=rho2*u2*A2;

    else
        choked=true;
        %Complex formula for mass flow (also correct)
```

```
        %m_f=P1*sqrt(gamma/(r*T1))*Astar*((gamma+1)/2)^((gamma+1)/(2*(1-
gamma)));
        %Easier formula for mass flow:
        m_f(i)=Astar*cstar*rhostar;

        %P_chocked is a limiting value for nozzles with shockwaves
        [~,P_forchocked(i),~,~] =
Shockwave(M_super,P_super(i),100,Astar,gamma); %The velocity is not
important, so a random value (100) is given.

        if P_sub==P2
            regime='sub_solution';

        elseif (P2<P_sub && P2> P_forchocked(i) )
            regime='shockwave';

        elseif P2==P_forchocked(i) %A shockwave just at the exit
            regime='shockwave_exit';

        elseif P2<P_forchocked(i) && P2>P_super(i)
            regime='overexpanded';

        elseif P2==P_super(i)
            regime='adapted';

        elseif P2<P_super(i)
            regime='subexpanded';
        end

    end
    P_choked=Pstar;
end

% plot(A2a,P_forchocked);
Pforchocked_app=[52828 61572 51340 43477.04 37595.95 66325.49];
A2a_app=[1 1.5 2 2.5 3 1.1];
P_super_app=[52828 16017 9393.26 6398.43 4729.87 32685.65];

% hold on;
% plot(A2a_app,Pforchocked_app,'x');
% grid;
% xlabel('Relación de áreas A2/A*');
% ylabel('Presión con onda de choque en la salida [Pa]');
% legend('Verificación por Matlab','Resultados de la App');

plot(A2a,P_super);
hold on;
plot(A2a_app,P_super_app,'x');
grid;
xlabel('Relación de áreas A2/A*');
ylabel('Presión de tobera adaptada [Pa]');
legend('Verificación por Matlab','Resultados de la App');
```

Por último, se utiliza el código llamado “shock_area_calc.m” para calcular el área de la onda de choque:

```
clear
close all
%Finds the area where the shockwave is.
%Inputs: Stagnation pressure, exit pressure (ambient), Throat area, exit
```

```
%area and gamma
%It only works well if the inputs are for a nozzle in the shockwave
regime
P1=100000;
Astar=1;
A2=2;
gamma=1.4;
P2=linspace(93716,51341);
for i=1:length(P2);
[ shock_area(i),M2_exit(i) ] = find_shockwave_area(
P1,P2(i),Astar,A2,gamma );
end

P2_app=[93716 80000 70000 60000 51341];
area_app=[1 1.29712 1.51 1.756 2];
plot(P2,shock_area);
hold on
plot(P2_app,area_app,'x');
grid;
legend('Verificación por Matlab','Resultados de la App');
xlabel('Presión ambiente P2 [Pa]');
ylabel('Área de onda de choque [m2]');
```

Donde se utiliza la función “find_shockwave_area”, cuyo código es el siguiente:

```
function [ shock_area,M2_exit ] = find_shockwave_area(
P1,P2,Astar,A2,gamma )
%Finds the area where the shockwave is.
%Inputs: Stagnation pressure, exit pressure (ambient), Throat area, exit
%area and gamma
%It only works well if the inputs are for a nozzle in the shockwave
regime

Amean=(A2+Astar)/2; %Value to start the fsolve iteration, the mean value
between A2 and Astar
options= optimset('Display','none','TolFun',1e-12,'TolX',1e-12);%We
define the tolerance
fun=@(A) pressure(A)-P2; %Function that has to be 0, using the nested
pressure function
shock_area=fsolve(fun,Amean,options); %shock area is calculated.
%M2_exit is calculated in the last iteration of the pressure function,
and
%as it is nested, the value is correctly stored.


function P2_exit=pressure(A)
        a_ratio=A/Astar;
    [P1_shock,M1_shock,~,~]=P_ad(P1, a_ratio, gamma);
    [M2_shock,P2_shock,~,~] =
Shockwave(M1_shock,P1_shock,100,Astar,gamma); %A random velocity (100) is
given for simplicity. It does not matter.
    P20_shock=P2_shock*(1+(gamma-1)/2*M2_shock^2)^(gamma/(gamma-1));
%Stagnation pressure after shockwave
    M2_exit=calculate_Mexit(A,A2,M2_shock); %Mach at the exit
    P2_exit=property_mach(P20_shock,M2_exit,gamma,'P');
end

function M2_exit=calculate_Mexit(A,A2,M2) %Calculates the Mach at the
exit given the area relation shock area/exit area
   ratio=A/A2;
   area_relation=@(M) M/M2*((1+(gamma-1)*(M2^2)/2)/(1+(gamma-
1)*M^2/2))^((gamma+1)/(2*(gamma-1)))-ratio;
   for j=0.001:0.001:1 %As it has two solutions, the subsonic solution is
the one we want. j is the initial value for the iteration.
```

```
        M2_exit=fsolve(area_relation,j,options);
        if M2_exit<=1
            break
        elseif j==1&&M2_exit>1
            error('Could not find subsonic expansion after shockwave');
        end
    end
 end
end
```

B.5. Flujo de Fanno

Para la verificación del flujo de Fanno se utiliza un solo código en el que se calculan todos los resultados a comprobar. El código, llamado "Fanno", es el siguiente:

```
clear
close all
Ma2=0.9;
Ma1=linspace(0.1,0.9);
gamma=1.4;
f=0.024;
D=0.02;
fld=@(M) ((1-
M^2)/(gamma*M^2))+((gamma+1)/(2*gamma))*(log(((gamma+1)*M^2)/(2+(gamma-
1)*M^2))/log(exp(1)));
factor(2)=fld(Ma2);
p_rel_chocked=@(M) 1/M*((gamma+1)/(2+(gamma-1)*M^2))^0.5;
rho_rel_chocked=@(M) 1/M*((gamma+1)/(2+(gamma-1)*M^2))^-0.5;
T_rel_chocked=@(M) ((gamma+1)/(2+(gamma-1)*M^2));
p_rel_stag_chocked=@(M) 1/M*((gamma+1)/(2+(gamma-1)*M^2))^(-
0.5*(gamma+1)/(gamma-1));
for i=1:length(Ma1)
    factor(1)=fld(Ma1(i));
    L(i)=(1/f)*D*(factor(1)-factor(2));
    L_chocked(i)=D/f*factor(1);
    p_rel(i)=p_rel_chocked(Ma2)/p_rel_chocked(Ma1(i));
    rho_rel(i)=rho_rel_chocked(Ma2)/rho_rel_chocked(Ma1(i));
    T_rel(i)=T_rel_chocked(Ma2)/T_rel_chocked(Ma1(i));
    p_rel_stag(i)=p_rel_stag_chocked(Ma2)/p_rel_stag_chocked(Ma1(i));
end

Ma1_app=[0.1 0.2 0.4 0.5 0.9];
L_app=[55.756 12.099 1.912 0.87879 0];
p_rel_app=[0.10318 0.20697 0.41885 0.5281 1];
rho_rel_app=[0.11965 0.23859 0.47161 0.58443 1];
T_rel_app=[0.86231 0.86747 0.88812 0.90361 1];
p_rel_stag_app=[0.17329 0.34043 0.63445 0.75297 1];

% plot(Ma1,L);
% hold on
% plot(Ma1_app,L_app,'x');
% grid;
% xlabel('Ma at the inlet');
% ylabel('Length of the duct');
% legend('Matlab verification','App results');

% plot(Ma1,p_rel);
```

```
% hold on
% plot(Ma1_app,p_rel_app,'x');
% grid;
% xlabel('Ma at the inlet');
% ylabel('P2/P1');
% legend('Matlab verification','App results');

% plot(Ma1,rho_rel);
% hold on
% plot(Ma1_app,rho_rel_app,'x');
% grid;
% xlabel('Ma at the inlet');
% ylabel('?2/?1');
% legend('Matlab verification','App results');

% plot(Ma1,T_rel);
% hold on
% plot(Ma1_app,T_rel_app,'x');
% grid;
% xlabel('Ma at the inlet');
% ylabel('T2/T1');
% legend('Matlab verification','App results');

plot(Ma1,p_rel_stag);
hold on
plot(Ma1_app,p_rel_stag_app,'x');
grid;
xlabel('Ma at the inlet');
ylabel('P0(2)/P0(1)');
legend('Matlab verification','App results');
```