Sumario

Anexo A: Datos técnicos del Sikorski UH 60

A.1: Masa e inercias

Abreviatura	**Magnitud**	**Unidades SI**	**Unidades anglosajonas**
m	Masa de la aeronave	$7493\ kg$	$16400.0\ lb$
I_{xx}	Momento de inercia de alabeo	$7631\ kg \cdot m^2$	$5629\ slug \cdot ft^2.$
I_{yy}	Momento de inercia de cabeceo	$54232\ kg \cdot m^2$	$40000\ slug \cdot ft^2.$
I_{zz}	Momento de inercia de guiñada	$50436\ kg \cdot m^2$	$37200\ slug \cdot ft^2.$
I_{xz}	Producto de inercia	$2264\ kg \cdot m^2$	$1670\ slug \cdot ft^2.$
STA_{CG}	Stationline del centro de gravedad	$9.15\ m$	$360.4\ in.$
WL_{CG}	Waterline del centro de gravedad	$6.28\ m$	$247.2\ in.$
BL_{CG}	Buttline del centro de gravedad	$0.0\ m$	$0.0\ in.$

Tabla A.1. *Masa, inercias y centro de gravedad del Sikorski UH-60*

A.2: Rotor principal

Abreviatura	**Magnitud**	**Unidades SI**	**Unidades anglosajonas**
R_{MR}	Radio del rotor principal	$1.68\ m$	$26.83\ ft$
c_{MR}	Cuerda de la pala del rotor principal	$0.2469\ m$	$1.73\ ft$
Ω_{MR}	Velocidad de rotación del rotor principal	$27\ rad/s$	
$n_{b_{MR}}$	Número de palas del rotor principal	4	
γ_{MR}	Número de inercia de Lock del rotor principal	8.1936	
ϵ	Separación de las articulaciones de batimiento del rotor principal (tanto por uno)	0.04659	
K_β	Constante elástica de batimiento del rotor principal	$0.0\ lb \cdot ft/rad$	
$K_{1_{MR}}$	Factor de acoplamiento entre paso y batimiento del rotor principal ($\tan \delta_3$)	0	
$\beta_{0_{MR}}$	Conicidad preestablecida	$0.0\ rad$	
$\theta_{t_{MR}}$	Torsión constructiva de la pala del rotor principal	$-0.3142\ rad$	
σ_{MR}	Solidez del rotor principal	0.08210	
a_{MR}	Pendiente de la curva de sustentación de la pala del rotor principal	5.73	
$C_{T_{MAX}}$	Máxima tracción del rotor principal	$0.1846\ m$	

i_s	Inclinación longitudinal del eje del rotor principal (positiva hacia delante)	$0.05236\ rad$	
STA_{MR}	Stationline del rotor principal	$8.67\ m$	$341.2\ in.$
WL_{MR}	Waterline del rotor principal	$8.00\ m$	$315.0\ in.$
BL_{MR}	Buttline del rotor principal	$0.0\ m$	$00\ in.$

Tabla A.2. *Características mecánicas del rotor principal del Sikorski UH-60*

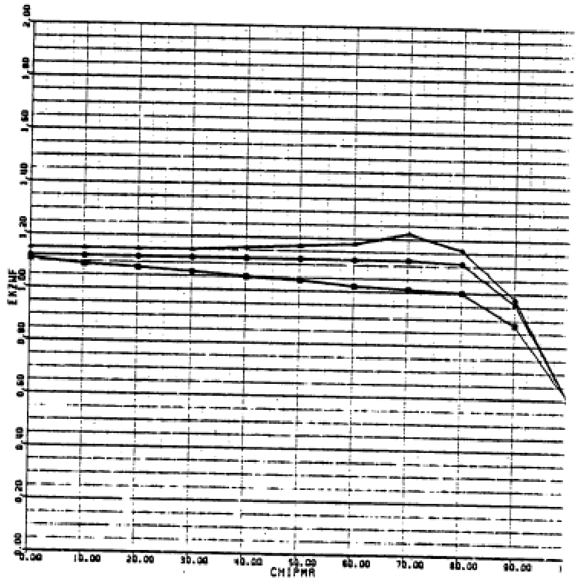


Figura A.1. *Efecto de la estela del rotor principal sobre el fuselaje*

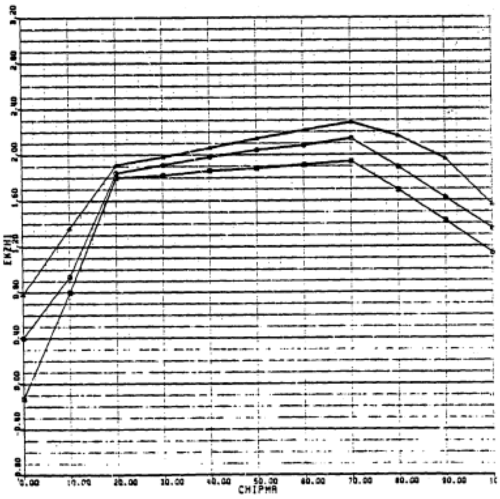


Figura A.2.. *Efecto de la estela del rotor principal sobre los elementos de cola*

A.3: Rotor de cola

Abreviatura	**Magnitud**	**Unidades SI**	**Unidades anglosajonas**
R_{TR}	Radio del rotor de cola	$1.68\,m$	$5.3\,ft$
c_{TR}	Cuerda de la pala del rotor de cola	$0.2469\,m$	$9.7205\,in$
Ω_{TR}	Velocidad de rotación del rotor de cola	$124.62\,rad/s$	
$n_{b_{TR}}$	Número de palas del rotor de cola	4	
γ_{TR}	Número de inercia de Lock del rotor de cola	3.3783	
$K_{1_{TR}}$	Factor de acoplamiento entre paso y batimiento del rotor de cola ($\tan\delta_3$)	0.7002	
$\beta_{0_{TR}}$	Conicidad preestablecida	$0.01309\,rad$	
$\theta_{t_{TR}}$	Torsión constructiva de la pala del rotor de cola	$-0.3142\,rad$	
σ_{TR}	Solidez del rotor de cola	0.1875	
a_{TR}	Pendiente de la curva de sustentación de la pala del rotor de cola	5.73	
STA_{TR}	Stationline del rotor de cola	$18.59\,m$	$732.0\,in.$
WL_{TR}	Waterline del rotor de cola	$8.25\,m$	$324.7\,in.$
BL_{TR}	Buttline del rotor de cola	$-0.36\,m$	$-14.0\,in.$

Tabla A.3. *Características mecánicas del rotor de cola del Sikorski UH-60*

A.4: Estabilizador horizontal

Abreviatura	**Magnitud**	**Unidades SI**	**Unidades anglosajonas**
i_{HT}	Ángulo de incidencia	variable	
S_{HT}	Área	$4.18\ m^2$	$45.0\ ft^2$
AR_{HT}	Relación de aspecto	4.6	
$C_{Lmax_{HT}}$	Pendiente máxima de la curva de sustentación	1.03	
η_{HT}	Ratio de presión dinámica	0.4	
AR_{HT}	Relación de aspecto	4.6	
$k_{v_{MR}}$	Efecto de la velocidad inducida por la estela del rotor principal	1.8	
STA_{HT}	STA del estabilizador horizontal	$6.19\ m$	$700.4\ in.$
WL_{HT}	WL del estabilizador horizontal	$6.19\ m$	$244.0\ in.$
BL_{HT}	BL del estabilizador horizontal	$0.0\ m$	$0.0\ in.$

Tabla A.4. *Características mecánicas del estabilizador horizontal del Sikorski UH-60*

A.5: Estabilizador vertical

Abreviatura	**Magnitud**	**Unidades SI**	**Unidades anglosajonas**
i_{VT}	Ángulo de incidencia	$0\ rad$	
S_{VT}	Área	$3.0\ m^2$	$32.3\ ft^2$
AR_{VT}	Relación de aspecto	1.92	
Λ_{VT}	Ángulo de flecha	$0.7156\ rad$	
$C_{Lmax_{VT}}$	Pendiente máxima de la curva de sustentación	0.89	
η_{VT}	Ratio de presión dinámica	0.651	
AR_{VT}	Relación de aspecto	4.6	
Λ_{VT}	Ángulo de flecha	$0.7156\ rad$	
$k_{v_{TR}}$	Efecto de la velocidad inducida por la estela del rotor de cola	1.0	
STA_{VT}	STA del estabilizador vertical	$17.65\ m$	$695.0\ in.$
WL_{VT}	WL del estabilizador vertical	$6.93m$	$273.0\ in.$
BL_{VT}	BL del estabilizador vertical	$0.0\ m$	$0.0\ in.$

Tabla A.5. *Características mecánicas del estabilizador vertical del Sikorski UH-60*

A.6: Fuselage

Abreviatura	Magnitud	Unidades SI	Unidades anglosajonas
STA_{FU}	STA del punto de referencia aerodinámica del fuselaje	$8.78\ m$	$345.5\ in.$
WL_{FU}	WL del punto de referencia aerodinámica del fuselaje	$5.94\ m$	$234.0\ in.$

Tabla A.6. *Características mecánicas del fuselaje del Sikorski UH-60*

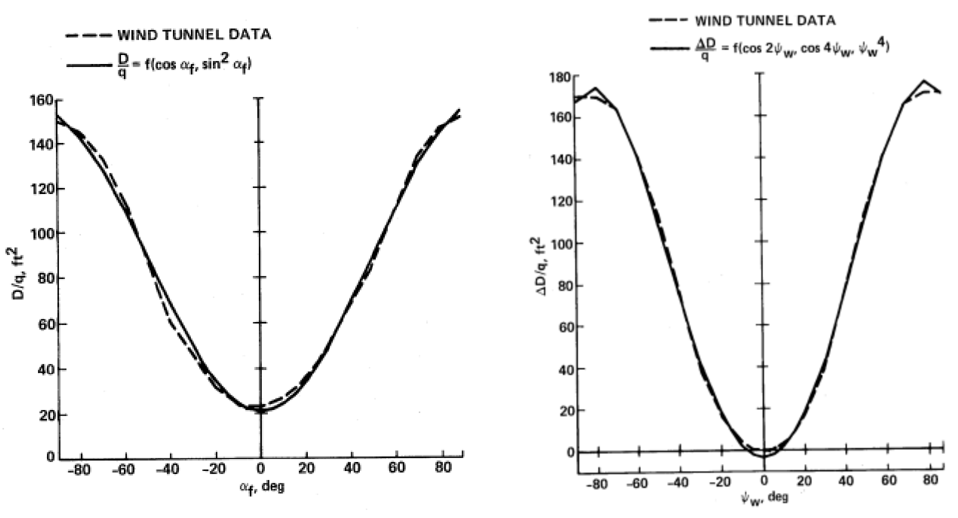


Figure 1.- Fuselage drag vs angle of attack.

Figure 4.- Incremental fuselage drag vs sideslip.

$$\frac{D}{q} = 90.0555\sin^2\alpha_{FU} - 41.5604\cos\alpha_{FU} + 2.94684\cos 4\psi_w - 103.141\cos 2\psi_w - 0.535350\cdot 10^{-6}\psi_w^4 + 160.2049$$

Figura A.3. *Resistencia al avance del fuselaje del Sikorski UH-60*

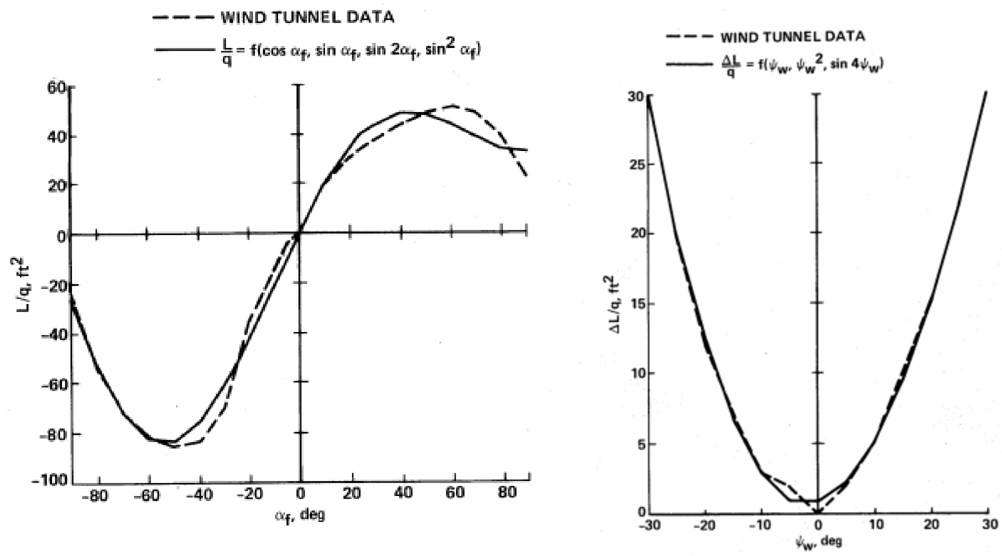


Figure 2.- Fuselage lift vs angle of attack.

Figure 5.- Incremental fuselage lift vs sideslip.

$$\frac{L}{q} = 29.3616\sin\alpha_{FU} + 43.4680\sin 2\alpha_{FU} - 81.8924\sin^2\alpha_{FU} - 84.1469\cos\alpha_{FU} - 0.821406\cdot 10^{-1}\psi_w + 3.00102 sin4\psi_w + 0.0323477\psi_w^2 + 85.3496$$

Figura A.4. *Sustentación del fuselaje del Sikorski UH-60*

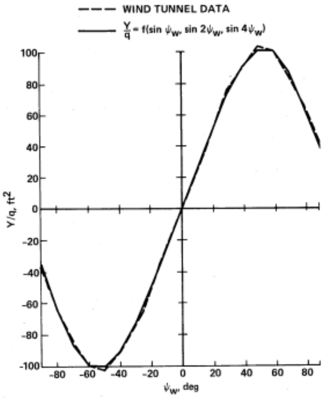


Figure 7.- Fuselage side force vs sideslip.

$$\frac{Y}{q} = 35.3999\sin\psi_w + 71.8019\psi_w - 8.04823\sin 4\psi_w - 0.980257\cdot 10^{-12}$$

Figura A.5. *Fuerza lateral del fuselaje del Sikorski UH-60*

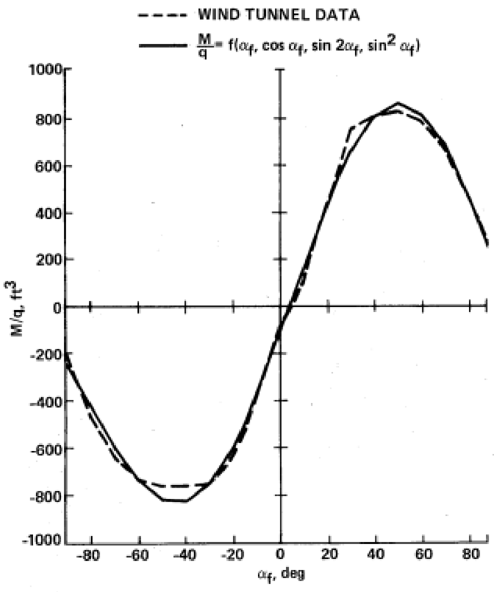
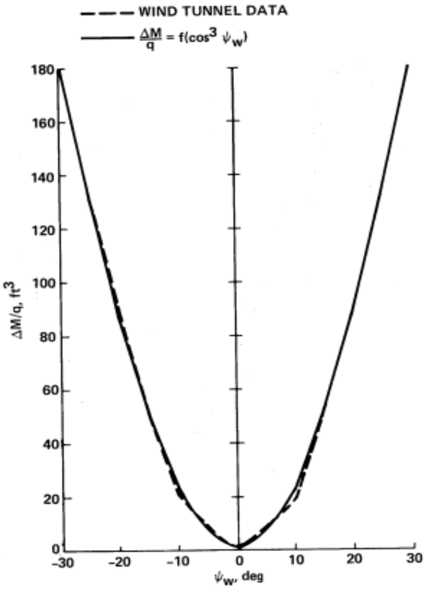


Figure 3.- Fuselage pitching moment vs angle of attack.



$$\frac{M}{q} = 2.37925\alpha_{FU} + 728.026\sin 2\alpha_{FU} + 426.760\sin^2 \alpha_{FU} + 348.072\cos \alpha_{FU} - 510.581\cos^3 \psi_w + 56.111$$

Figura A.6. *Momento de cabeceo del fuselaje del Sikorski UH-60*

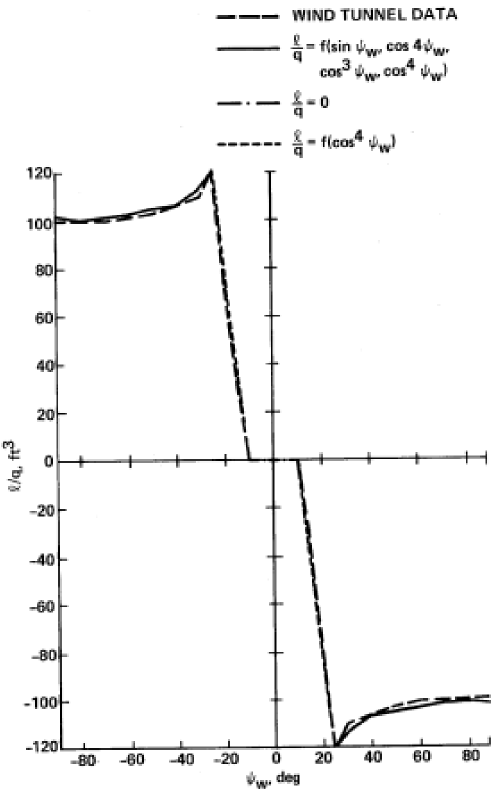


Figure 8.- Fuselage rolling moment vs sideslip.

$$\frac{L}{q} = 614.797\sin\psi_w + \frac{\psi_w}{|\psi_w|}\left(-47.7213\cos 4\psi_w - 290.504\cos^3\psi_w + 735.507\cos^4\psi_w - 669.266\right), \qquad 25^\circ < |\psi_w| \leq 90^\circ$$

$$\frac{L}{q} = \frac{\psi_w}{|\psi_w|}\left(455.707\cos^4\psi_w - 428.639\right), \qquad 10^\circ < \psi_w \leq 25^\circ$$

$$\frac{L}{q} = 0.0, \qquad -10^\circ < \psi_w \leq 10^\circ$$

Figura A.7. *Momento de alabeo del fuselaje del Sikorski UH-60*

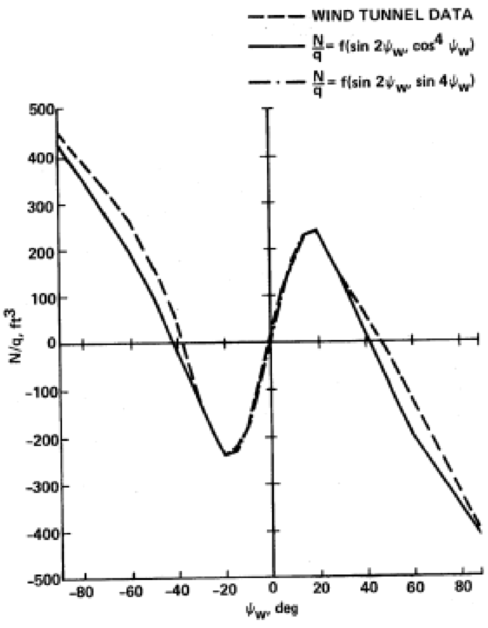


Figure 9.- Fuselage yawing moment vs sideslip.

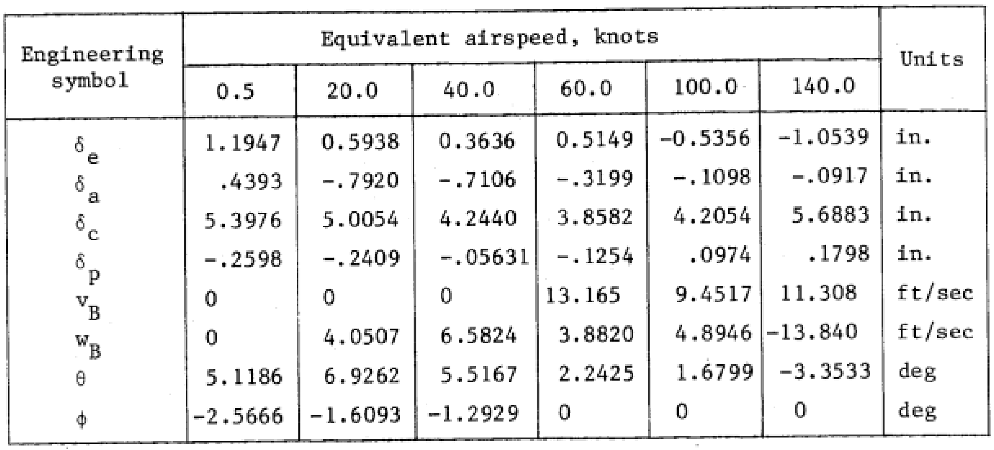
$\frac{N}{q} = 220.0 \sin 2\psi_w + \frac{\psi_w}{|\psi_w|}(671.0 cos^4\psi_w - 429.0)$ $20° < |\psi_w| \leq$

$90° \frac{N}{q} = -278.133 \sin 2\psi_w + 422.644 sin4\psi_w - 1.83172,$ $-20º \leq$

$\psi_w \leq 20°$

Figura A.8. *Momento de guiñada del fuselaje del Sikorski UH-60*

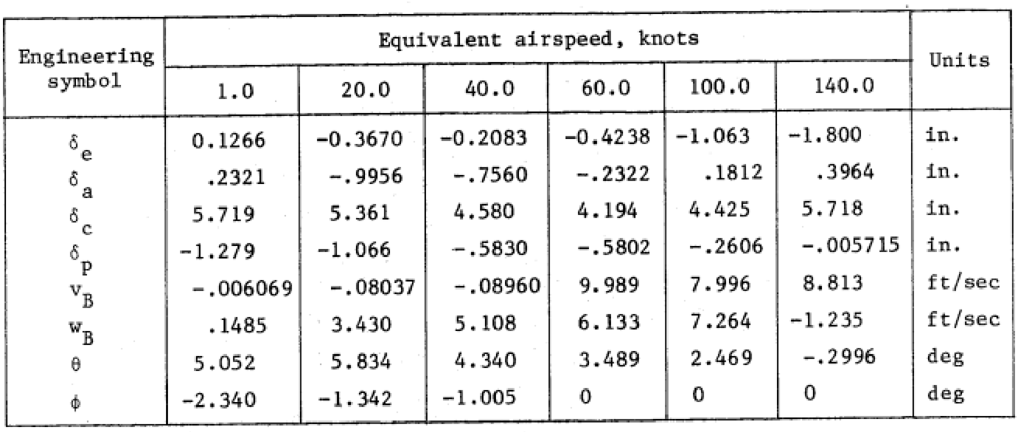
A.8: Parámetros y resultados de vuelo compensado con el modelo de Howlett



Engineering symbol	Equivalent airspeed, knots						Units
	0.5	20.0	40.0	60.0	100.0	140.0	
δ_e	1.1947	0.5938	0.3636	0.5149	-0.5356	-1.0539	in.
δ_a	.4393	-.7920	-.7106	-.3199	-.1098	-.0917	in.
δ_c	5.3976	5.0054	4.2440	3.8582	4.2054	5.6883	in.
δ_p	-.2598	-.2409	-.05631	-.1254	.0974	.1798	in.
v_B	0	0	0	13.165	9.4517	11.308	ft/sec
w_B	0	4.0507	6.5824	3.8820	4.8946	-13.840	ft/sec
θ	5.1186	6.9262	5.5167	2.2425	1.6799	-3.3533	deg
ϕ	-2.5666	-1.6093	-1.2929	0	0	0	deg

Tabla A.7. *Parámetros de control y características de vuelo compensado con el modelo de Howlett*

A.9: Parámetros y resultados de vuelo estabilizado con el modelo de Hilbert



Engineering symbol	Equivalent airspeed, knots						Units
	1.0	20.0	40.0	60.0	100.0	140.0	
δ_e	0.1266	-0.3670	-0.2083	-0.4238	-1.063	-1.800	in.
δ_a	.2321	-.9956	-.7560	-.2322	.1812	.3964	in.
δ_c	5.719	5.361	4.580	4.194	4.425	5.718	in.
δ_p	-1.279	-1.066	-.5830	-.5802	-.2606	-.005715	in.
v_B	-.006069	-.08037	-.08960	9.989	7.996	8.813	ft/sec
w_B	.1485	3.430	5.108	6.133	7.264	-1.235	ft/sec
θ	5.052	5.834	4.340	3.489	2.469	-.2996	deg
ϕ	-2.340	-1.342	-1.005	0	0	0	deg

Tabla A.8. *Parámetros de control y características de vuelo compensado con el modelo de Hilbert*

Referencias bibliográficas

[1] HILBERT, K. *A Mathematical Model of the UH-60 Helicopter, NASA Technical Memorandum 85890,* California, 1984

[2] HOWLETT, J.*, UH–60A BLACK HAWK Engineering Simulation Program: Volume I Mathematical Model,* NASA CR–166309, Dec. 1981

ANEXO B: Datos técnicos del UH60 Modificado

B.1: Masa e inercias

Abreviatura	Magnitud	Unidades SI	Unidades anglosajonas
m	Masa de la aeronave	$8877\ kg$	$20000.0\ lb$
I_{xx}	Momento de inercia de alabeo	$9686\ kg \cdot m^2$	$6176\ slug \cdot ft^2.$
I_{yy}	Momento de inercia de cabeceo	$48443\ kg \cdot m^2$	$34244.0\ slug \cdot ft^2.$
I_{zz}	Momento de inercia de guiñada	$43422\ kg \cdot m^2$	$31596.0\ slug \cdot ft^2.$
I_{xz}	Producto de inercia	$2203\ kg \cdot m^2$	$1839.0\ slug \cdot ft^2.$
STA_{CG}	Stationline del centro de gravedad	$8.95\ m$	$352.36\ in.$
WL_{CG}	Waterline del centro de gravedad	$6.39\ m$	$251.4\ in.$
BL_{CG}	Buttline del centro de gravedad	$0.0\ m$	$0.0\ in.$

Tabla B.1 *Masa e inercias del UH-60 modificado*

B.2: Desplazamiento del estabilizador horizontal

Abreviatura	Magnitud	Unidades SI	Unidades anglosajonas
STA_{HT}	Stationline del estabilizador horizontal	$17.53\ m$	$690.0\ in.$
WL_{HT}	Waterline del estabilizador horizontal	$6.20\ m$	$244.0\ in.$
BL_{HT}	Buttline del estabilizador horizontal	$0.0\ m$	$0.0\ in.$

Tabla B.2 *Posición del estabilizador horizontal del UH-60 modificado*

B.3: Características del ala

Abreviatura	Magnitud	Unidades SI	Unidades anglosajonas
	Envergadura	$13.72\ m$	$45.0\ ft$
S_W	Área efectiva	$21.00\ m^2$	$226.0\ ft^2$
	Cuerda en el encastre	$1.68\ m$	$5.5\ ft$
	Cuerda en la punta	$1.37\ m$	$4.5\ ft$
	Estrechamiento	0.820	
	Cuerda aerodinámica media	$1.53\ m$	$5.02\ ft$
AR_W	Relación de aspecto	9.0	
i_{RW}, i_{LW}	Ángulo de incidencia	$2\ deg$	
	Perfil aerodinámico	$NACA\ 63-412$	
STA_{LW}	Stationline de la semiala izquierda	$8.81\ m$	$347.0\ in.$
WL_{LW}	Waterline de la semiala izquierda	$6.45\ m$	$254.0\ in.$
BL_{LW}	Buttline de la semiala izquierda	$-2.46\ m$	$-97.0\ in.$
STA_{RW}	Stationline de la semiala derecha	$8.81\ m$	$347.0\ in.$
WL_{RW}	Waterline de la semiala derecha	$6.45\ m$	$254.0\ in.$
BL_{RW}	Buttline de la semiala derecha	$2.46\ m$	$97.0\ in.$

Tabla B.3 *Características mecánicas del ala del UH-60 modificado*

$\alpha(deg)$	C_L	C_D	C_m
-180	0	0,088	0,007
-170	0,547	0,314	0,058
-160	1,028	0,66	0,128
-150	1,386	1,084	0,21
-140	1,576	1,536	0,295
-130	1,576	1,96	0,371
-120	1,386	2,306	0,428
-110	1,028	2,532	0,457
-100	0,547	2,61	0,454
-90	0	2,532	0,42
-80	-0,547	2,306	0,359
-70	-1,028	1,96	0,28
-60	-1,386	1,536	0,195
-50	-1,576	1,084	0,114
-40	-1,576	0,66	0,047
-35	-1,504	0,474	0,016
-30	-1,386	0,314	-0,0001
-25	-1,226	0,184	-0,013
-20	-1,028	0,088	-0,028
-18	-0,94	0,06	-0,031
-16	-0,848	0,038	-0,033
-14	-0,751	0,023	-0,036
-12	-0,651	0,013	-0,037
-10	-0,547	0,01	-0,074
-8	-0,615	0,008	-0,075
-6	-0,369	0,006	-0,077
-4	-0,124	0,006	-0,079
-2	0,121	0,005	-0,081
0	0,355	0,006	-0,083
2	0,585	0,007	-0,083
4	0,819	0,007	-0,08
6	1,039	0,009	-0,078
8	1,262	0,01	-0,067
10	1,445	0,012	-0,036
12	1,509	0,016	-0,013
14	1,551	0,024	-0,005
16	1,535	0,041	-0,006
18	1,445	0,075	-0,04
20	1,301	0,133	-0,052
25	1,226	0,208	-0,067
30	1,386	0,344	-0,08
35	1,504	0,51	-0,096
40	1,576	0,7	-0,127
50	1,576	1,129	-0,194
60	1,386	1,58	-0,275
70	1,028	1,999	-0,36
80	0,547	2,334	-0,439
90	0	2,546	-0,5
100	-0,547	2,609	-0,534
110	-1,028	2,515	-0,537
120	-1,386	2,276	-0,508
130	-1,576	1,92	-0,451
140	-1,576	1,491	-0,375
150	-1,386	1,04	-0,29
160	-1,028	0,621	-0,208
170	-0,547	0,286	-0,138
180	0	0,074	-0,087

Tabla B.4 *Coeficientes de sustentación, resistencia al avance y momento de cabeceo del ala*

B.4: Características del propulsor

Abreviatura	Magnitud	Unidades SI	Unidades anglosajonas
R_{PR}	Radio del propulsor	$1.22\,m$	$4\,ft$
$n_{b,PR}$	Número de palas del propulsor	7	
σ_{MR}	Solidez del propulsor	0.34	
Ω_{PR}	Velocidad de giro	$226.20\,rad/s$	
	Torsión en la punta	$34.6\,deg$	
	Torsión en el encastre	$83.8\,deg$	
c_{PR}	Cuerda (0.75R)	$0.27\,m$	$0.88\,ft$
	Torsión (0.75R)	$42.4\,deg$	
	Perfil aerodinámico	$Clark-Y$	
BL_{PR}	Stationline del propulsor	$19.56\,m$	$770.0\,in.$
STA_{PR}	Waterline del propulsor	$5.99\,m$	$236.0\,in.$
WL_{PR}	Buttline del propulsor	$0.0\,m$	$0.0\,in.$

Tabla B.5 *Características mecánicas del propulsor*

r/R [-]	c/R [-]	paso[deg]	t/c [-]
0,1	0,019	83,8	0,122
0,15	0,04	79,3	0,121
0,2	0,066	75	0,122
0,25	0,094	70,9	0,122
0,3	0,123	67	0,122
0,35	0,149	63,4	0,122
0,4	0,172	60	0,121
0,45	0,191	56,8	0,122
0,5	0,207	53,9	0,121
0,55	0,218	51,2	0,122
0,6	0,224	48,7	0,122
0,65	0,227	46,4	0,122
0,7	0,226	44,3	0,121
0,75	0,219	42,4	0,121
0,8	0,208	40,6	0,122
0,85	0,19	38,9	0,121
0,9	0,164	37,4	0,122
0,95	0,121	35,9	0,122
1	0,006	34,6	0,122

Tabla B.6 *Especificación del propulsor*

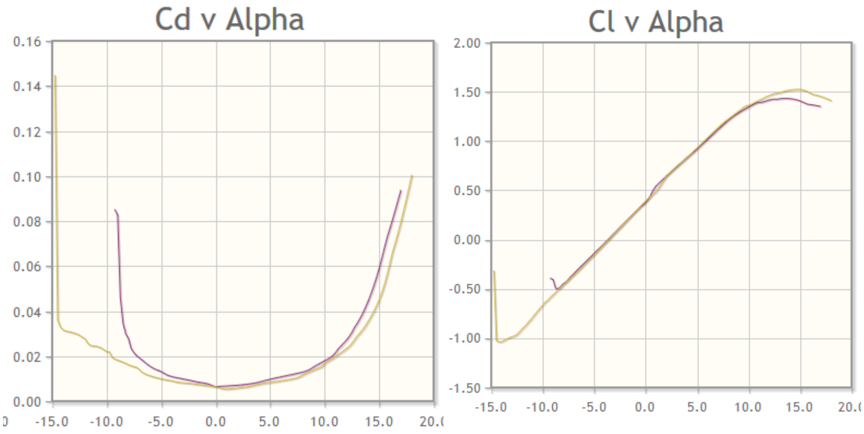


Figura B.1. *Coeficientes de sustentación y resistencia al avance para el perfil Clark-Y de las palas del propulsor*

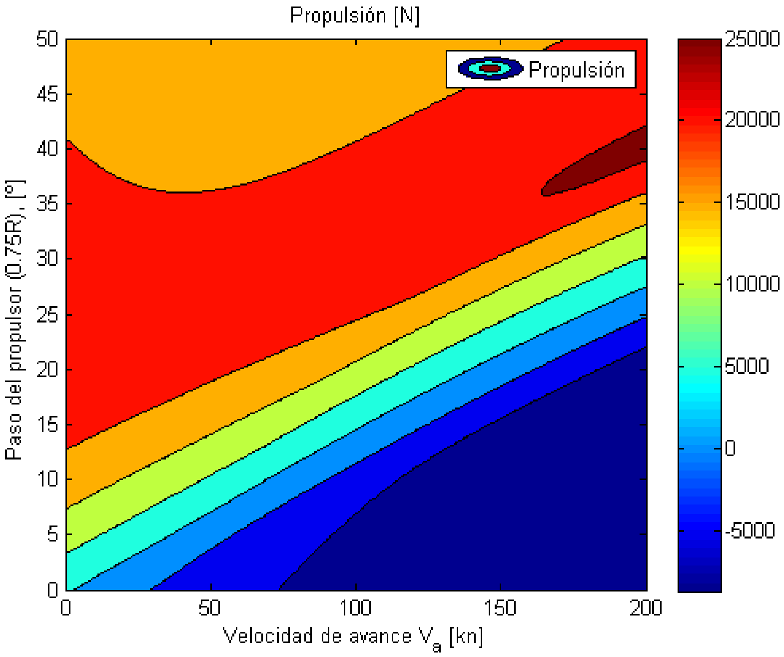


Figura B.2. *Fuerza de empuje del propulsor*

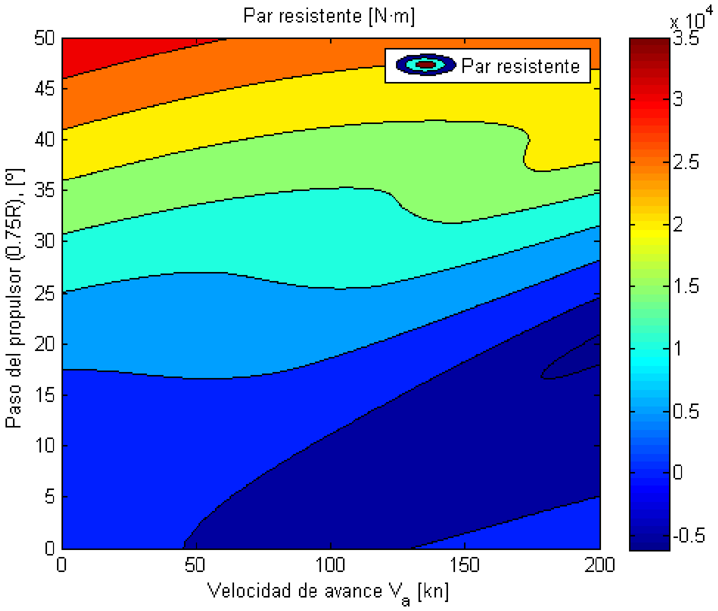


Figura B.3. *Par resistente del propulsor*

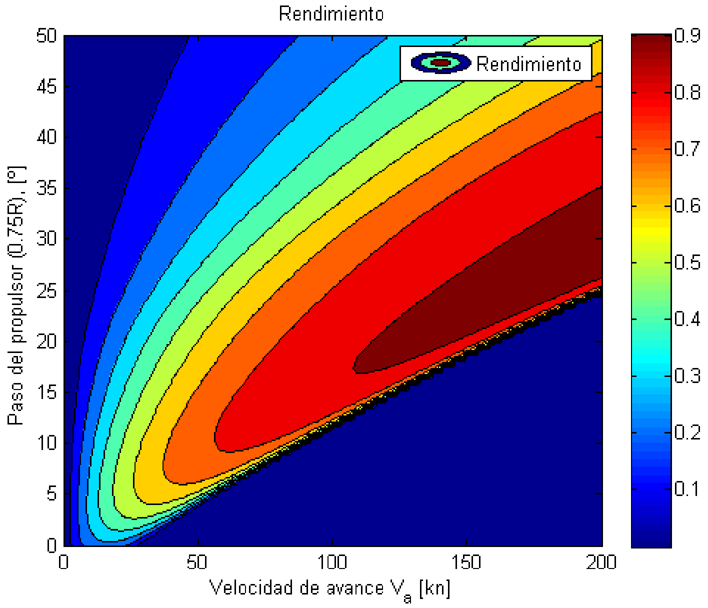


Figura B.4. *Rendimiento del propulsor*

Referencias bibliográficas

[1] OZDEMIR, G. T., *In-Flight Performance Optimization for Rotorcraft with Redundant Controls*, Pennsylvania State University, 2013

[2] THORSEN, A. T.*, Assessment of Control Allocation Optimization on Performance and Dynamic Response Enhancement of a Compound Rotorcraft,* Pennsylvania State University, 2014

Anexo C: Modelos dinámicos

C.1: Parámetros UH60

```
% masa e inercias

P.mass          =       16400.0*P.m_esc1;       % kg
P.Ixx           =       5629.0*P.i_esc;         % kg*m2
P.Iyy           =       40000.0*P.i_esc;            % kg*m2
P.Izz           =       37200.0*P.i_esc;            % kg*m2
P.Ixz           =       1670.0*P.i_esc;         % kg*m2

P.STA_CG        =       360.4*P.l_esc2;         % m

P.WL_CG         =       247.2*P.l_esc2;         % m
P.BL_CG         =       0.0*P.l_esc2;                   % m



% rotor principal

P.R_MR          =       26.83*P.l_esc1;         % m
P.c_MR          =       1.73*P.l_esc1;          % m
P.Omega_MR      =       27.0;                           % rad/sec
P.n_b_MR        =       4.0;                            %   -
P.gamma_MR      =       8.1936;                     %   -
P.eps_MR        =       0.04659;                    %   tanto por 1
P.K_beta_MR     =       0.0;                            %   N.m/rad
P.K_1_MR        =       0.0;                            % -
P.theta_t_MR    =       -0.3142;    %0.0;%              % rad
P.beta_00_MR    =       0.0;                            % rad
P.sol_MR        =       0.08210;                    % -
P.a_MR          =       5.73;                           % 1/rad
P.CTM_MR        =       0.1846;                     % - (maximum thrust)
P.gamma_s_MR    =       0.05236; %                  % rad       (rotor
forward tilt)
P.STA_H         =       341.2*P.l_esc2;         % m
P.WL_H          =       315.0*P.l_esc2;         % m
P.BL_H          =       0.0*P.l_esc2;               % m

P.I_beta_MR     =       1512.6*P.i_esc;           % kg*m2  de Howlett
p5.1-47
P.M_beta_MR     =       86.7*P.m_esc2*P.l_esc1;      % kg*m   de Howlett
p5.1-47
P.P2_MR         =
1+P.K_beta_MR/(P.I_beta_MR*P.Omega_MR^2)+P.eps_MR*P.R_MR*P.M_beta_MR/(P.g
*P.I_beta_MR)+P.gamma_MR*P.K_1_MR/8*(1-4/3*P.eps_MR);

global pr_mi_MR
pr_mi_MR      =         0.1;
```

```
% rotor de cola

P.R_TR          =       5.5*P.l_esc1;               % m
P.Omega_TR      =       124.62;                     % rad/sec
P.gamma_TR      =       3.3783;                     %   -
P.sol_TR        =       0.1875;                     % -
P.K_1_TR        =       0.7002;                     % -
P.beta_00_TR    =       0.01309;                    %   rad
P.theta_t_TR    =       -0.3142;                    %   rad
P.a_TR          =       5.73;                           % 1/rad
P.STA_TR        =       732.0*P.l_esc2;         % m
P.WL_TR         =       324.7*P.l_esc2;         % m
P.BL_TR         =       14.0*P.l_esc2;              % m

P.n_b_TR        =       4.0;                            %   -
P.c_TR          =       P.sol_TR*pi*P.R_TR/P.n_b_TR;            % m
P.I_beta_TR     =       P.rho*P.c_TR*P.a_TR*P.R_TR^4/P.gamma_TR;        %
kg/m2

P.K_ang         =       20*pi/180;                   %rad (20 deg)

global pr_mi_TR
pr_mi_TR      =       0.1;



% fuselaje

P.STA_ACF       =       345.5*P.l_esc2;         % m
P.WL_ACF        =       234.0*P.l_esc2;         % m

P.BL_ACF        =       0.0*P.l_esc2;             % m


% estabilizador horizontal

P.STA_HT        =       700.4*P.l_esc2;         % m
P.WL_HT         =       244.0*P.l_esc2;         % m
%P.alpha_i_HT   =       variable                %rad
P.S_HT          =       45.0*P.s_esc;               % m2
P.AR_HT         =       4.6;                            % -     (HT
aspect ratio)
P.CLM_HT        =       1.03;                           % -     (HT
maximum lift curve slope)
P.nu_HT         =       0.4; %                  % -     (dynamic pressure
ratio)
P.kv_HT         =       1.8;                            % -     (Main
rotor induced velocity effect at HT)

P.BL_HT         =       0.0*P.l_esc2;           % m
```

```
% estabilizador vertical

P.STA_VT        =       695.0*P.l_esc2;         % m
P.WL_VT         =       273.0*P.l_esc2;         % m
P.alpha_i_VT    =       0.0;                            %   rad
P.S_VT          =       32.3*P.s_esc;               % m2
P.AR_VT         =       1.92;                           % -
(aspect ratio)
P.LAMBDA_VT     =       0.7156;                     %   rad     (sweep
angle)
P.CLM_VT        =       0.89;                           % -         (VT
maximum lift curve slope)
P.nu_VT         =       0.651;                      % -         (dynamic
pressure ratio)
P.kv_VT         =       1.0;                            % -         (Tail
rotor induced velocity effect at VT)

P.BL_VT         =       0.0*P.l_esc2;           % m


% sistema de control

P.CA1S          =       0.0;                            % rad
(Swashplate lateral cyclic pitch for zero lateral cyclic stick)
P.CB1S          =       0.0;                            % rad
(Swashplate longitudinal cyclic pitch for zero longitudinal cyclic stick)
P.CK1           =       0.04939;                    % rad/in   (Longitudinal
cyclic control sensitivity
P.CK2           =       0.02792;                    % rad/in   (Lateral
cyclic control sensitivity
P.C5            =       0.2286;                         % rad       (Main
rotor root collective pitch for zero collective stick
P.C6            =       0.02792;                    % rad/in   (Main rotor
collective control sensitivity
P.C7            =       0.1743;                         % rad       (Tail
rotor root collective pitch for zero pedal position
P.C8            =       -0.07734;                   % rad/in   (Pedal
sensitivity

P.PP            =       P.P2_MR-1;
P.C1            =       (1-8/3*P.eps_MR+2*P.eps_MR^2)/(1-
4/3*P.eps_MR)*P.CK2;
P.C2            =       8*P.PP*P.CK2/(P.gamma_MR*(1-4/3*P.eps_MR));
P.C3            =       P.C2*P.CK1/P.CK2;
P.C4            =       -P.C1*P.CK1/P.CK2;


% CONTROL SYSTEM CHARACTERISTICS

% Feedforward gains
P.SK_1          =       1.0;                            % - (in/in, m/m)
(Longitudinal stick to longitudinal cyclic)
```

```
P.SK_5          =       1.0;                          % - (in/in, m/m)
(Lateral stick to lateral cyclic)
P.SK_9          =       1.0;                          % - (in/in, m/m)
(Collective stick to collective control)
P.SK_10         =       1.0;                          % - (in/in, m/m)
(Pedals to directional control)

% Crossfeed gains
P.SK_4          =       -0.1640;                  % - (in/in, m/m)
(Collective stick to longitudinal cyclic)
P.SKM_2         =       -0.5746;                  % - (in/in, m/m)
(Pedals to longitudinal cyclic)
P.SK_8          =       -0.16   ;                     % - (in/in, m/m)
(Collective stick to lateral cyclic)
P.SK_11         =       -0.2889;                  % - (in/in, m/m)
(Collective stick to directional control)

% Feedback gains
P.SKV_3_2       =       1.3;                  % in / (rad/s)  (Pitch rate
to lateral cyclic)
P.SKV_6_1       =       -0.88;                % in / (rad/s)  (Roll rate to
longitudinal cyclic)



% ENGINE CHARACTERISTICS

%   &&&&&&&&&&&&&&&&&&&
P.eng_K         =       1.75;           % HP/LB_fuel  (Engine gain)
%   &&&&&&&&&&&&&&&&&&&

P.eng_T         =       1.25;                     % s
(Engine time constant)
P.thr_T         =       1.25;                         % s
(Throttle time constant)
P.thr_P         =       100.0;                    %   tanto por 100
(Throttle position)
P.Omega_lim     =       9.0;                          % rad/s
(MR rpm lower limit)
P.G_ratio       =       4.62;                         %   -
(Gear ratio)


%   &&&&&&&&&&&&&&&&&&&
P.G_KG1         =       2000.0;                   %   LB_fuel/(rad/s)
(Proportional governor feedback gain)
P.G_KG2         =       2500.0;                   %   LB_fuel/(rad/s)
(Integral governor feedback gain)
P.G_KG3         =       500.0;                    %   LB_fuel/(rad/s)
(Rate governor feedback gain)
%   &&&&&&&&&&&&&&&&&&&
```

```
P.J             =       P.n_b_MR*P.I_beta_MR + P.n_b_TR*P.I_beta_TR;



% ACTUATOR LIMITS


% theta_0, theta_1s, theta_1c, i_HT

P.theta_max = (pi/180)*[25.9; 16.3; 8.0; 36.5; 39];
P.theta_min = (pi/180)*[9.9; -12.5; -8.0; 4.5; -8.0];
```

C.2: Parámetros UH60 modificado

```
% MASS AND INERTIA PROPERTIES

P.mass          =       (16400.0+3170.0)*P.m_esc1;      % kg
P.Ixx           =       (5629.0+1515)*P.i_esc;          % kg*m2
P.Iyy           =       (40000.0-4270)*P.i_esc;         % kg*m2
P.Izz           =       (37200.0-5173)*P.i_esc;         % kg*m2
P.Ixz           =       (1670.0-45)*P.i_esc;            % kg*m2
%P.STA_CG       después de P.STA_H
P.WL_CG         =       (247.2+4.2)*P.l_esc2;           % m
P.BL_CG         =       0.0*P.l_esc2;                   % m


% MAIN ROTOR PARAMETRES

P.R_MR          =       26.83*P.l_esc1;         % m
P.c_MR          =       1.73*P.l_esc1;          % m
P.Omega_MR      =       27.0;                           % rad/sec
P.n_b_MR        =       4.0;                            %   -
P.gamma_MR      =       8.1936;                     %   -
P.eps_MR        =       0.04659;                    %   tanto por 1
P.K_beta_MR     =       0.0;                            %   N.m/rad
P.K_1_MR        =       0.0;                            % -
P.theta_t_MR    =       -0.3142;    %0.0;%              % rad
P.beta_00_MR    =       0.0;                            % rad
P.sol_MR        =       0.08210;                    % -
P.a_MR          =       5.73;                           % 1/rad
P.CTM_MR        =       0.1846;                     % - (maximum thrust)
P.gamma_s_MR    =       0.05236; %       0.0;%          % rad
(rotor forward tilt)

P.STA_H         =       341.2*P.l_esc2;         % m
P.STA_CG        =       P.STA_H+11*P.l_esc2;            % m

P.WL_H          =       315.0*P.l_esc2;         % m
P.BL_H          =       0.0*P.l_esc2;               % m

P.I_beta_MR     =       1512.6*P.i_esc;           % kg*m2  de Howlett
p5.1-47
P.M_beta_MR     =       86.7*P.m_esc2*P.l_esc1;      % kg*m   de Howlett
p5.1-47
P.P2_MR         =
1+P.K_beta_MR/(P.I_beta_MR*P.Omega_MR^2)+P.eps_MR*P.R_MR*P.M_beta_MR/(P.g
*P.I_beta_MR)+P.gamma_MR*P.K_1_MR/8*(1-4/3*P.eps_MR);

global pr_mi_MR
pr_mi_MR      =         0.1;


% TAIL ROTOR PARAMETRES
```

```
P.R_TR          =       5.5*P.l_esc1;               % m
P.Omega_TR      =       124.62;                     % rad/sec
P.gamma_TR      =       3.3783;                     %   -
P.sol_TR        =       0.1875;                     % -
P.K_1_TR        =       0.7002;                     % -
P.beta_00_TR    =       0.01309;                    %   rad
P.theta_t_TR    =       -0.3142;                    %   rad
P.a_TR          =       5.73;                           % 1/rad
P.STA_TR        =       732.0*P.l_esc2;         % m
P.WL_TR         =       324.7*P.l_esc2;         % m
P.BL_TR         =       14.0*P.l_esc2;                  % m

P.n_b_TR        =       4.0;                            %   -
P.c_TR          =       P.sol_TR*pi*P.R_TR/P.n_b_TR;            % m
P.I_beta_TR     =       P.rho*P.c_TR*P.a_TR*P.R_TR^4/P.gamma_TR;        %
kg/m2

P.K_ang         =       20*pi/180;                   %rad (20 deg)

global pr_mi_TR
pr_mi_TR      =       0.1;


% FUSELAGE PARAMETRES

P.STA_ACF       =       345.5*P.l_esc2;         % m
P.WL_ACF        =       234.0*P.l_esc2;         % m

P.BL_ACF        =       0.0*P.l_esc2;             % m


% HORIZONTAL TAIL PARAMETRES

P.STA_HT        =       P.STA_H+348.8*P.l_esc2;         % m
P.WL_HT         =       P.WL_H-71.0*P.l_esc2;   % m
P.BL_HT         =       P.BL_H+0.0*P.l_esc2;        % m


%P.alpha_i_HT   =       variable                %rad
P.S_HT          =       45.0*P.s_esc;               % m2
P.AR_HT         =       4.6;                            % -     (HT
aspect ratio)
P.CLM_HT        =       1.03;                           % -     (HT
maximum lift curve slope)
P.nu_HT         =       0.4;                            % -     (dynamic
pressure ratio)
P.kv_HT         =       1.8;                            % -     (Main
rotor induced velocity effect at HT)



% VERTICAL TAIL PARAMETRES
```

```
P.STA_VT        =       695.0*P.l_esc2;         % m
P.WL_VT         =       273.0*P.l_esc2;         % m
P.alpha_i_VT    =       0.0;                            %   rad
P.S_VT          =       32.3*P.s_esc;               % m2
P.AR_VT         =       1.92;                           % -
(aspect ratio)
P.LAMBDA_VT     =       0.7156;                     %   rad     (sweep
angle)
P.CLM_VT        =       0.89;                           % -         (VT
maximum lift curve slope)
P.nu_VT         =       0.651;                      % -         (dynamic
pressure ratio)
P.kv_VT         =       1.0;                            % -         (Tail
rotor induced velocity effect at VT)

P.BL_VT         =       0.0*P.l_esc2;           % m




% parámetros del ala izquierda


P.S_LW          =       226.0/2*P.s_esc;                % m2
P.AR_LW         =       9.0/2;                          % -       (HT
aspect ratio)
P.MAC_LW        =       5.02*P.l_esc1;
P.alpha_i_LW    =       2.0*(pi/180);       %rad

P.STA_LW        =       P.STA_H+5.8*P.l_esc2;           % m
P.WL_LW         =       P.WL_H-61.0*P.l_esc2;           % m
P.BL_LW         =       P.BL_H-97.0*P.l_esc2;           % m

% parámetros del ala derecha

P.S_RW          =       226.0/2*P.s_esc;                % m2
P.AR_RW         =       9.0/2;                          % -       (WN
aspect ratio)   +
P.MAC_RW        =       5.02*P.l_esc1;
P.alpha_i_RW    =       2.0*(pi/180);       %rad

P.STA_RW        =       P.STA_H+5.8*P.l_esc2;           % m
P.WL_RW         =       P.WL_H-61.0*P.l_esc2;           % m
P.BL_RW         =       P.BL_H+97.0*P.l_esc2;           % m

P.LT_wn         = load ('data_wing.txt');


% parámetros del propulsor

P.STA_PR        =       P.STA_H+428.8*P.l_esc2;         % m
```

```
P.WL_PR         =       P.WL_H-79.0*P.l_esc2;           % m
P.BL_PR         =       P.BL_H-0.0*P.l_esc2;            % m



% ACTUATOR LIMITS
% theta_0, theta_1s, theta_1c, i_HT, beta_prop

P.theta_max = (pi/180)*[25.9; 16.3; 8.0; 36.5; 39; 32.2];
P.theta_min = (pi/180)*[9.9; -12.5; -8.0; 4.5; -8.0; 14.1];



% % programa lineal propulsor 3 de (119 kn, 15.0°) a (200kn, 31.0°) para
dar unos 15kn a
% % 200kn
P.Va_start = 119.0*0.5144;      % m/s
P.beta_start = 14.1*(pi/180); % rad
P.m_beta = (16.9/81)/0.5144*(pi/180); % rad/(m/s)


P.LT_PR2        = load ('data_propeller.mat');
P.LT_PR3        = load ('data_propeller_3D.mat');


%P.J            =       P.n_b_MR*P.I_beta_MR + P.n_b_TR*P.I_beta_TR;
```

C.3: Rotor principal (helicóptero)

```
function out = cN_fm_main_rotor(commands, lv_B, ar_B, d_ar_B,
loc_pr_mi_MR, P)

    global pr_mi_MR;

    % parameters adaptation

    gTOL         =    P.gtol;
    rho          =    P.rho;
    g            =    P.g;

    omega_ref_MR=    P.Omega_MR;
    Omega_MR     =    P.Omega_MR;
    R_MR         =    P.R_MR;

    a_MR         =    P.a_MR;
    c_MR         =    P.c_MR;
    s_MR         =    P.sol_MR;
    n_b_MR       =    P.n_b_MR;
    gamma_MR     =    P.gamma_MR;
    theta_t_MR   =    P.theta_t_MR;

    K_beta_MR    =    P.K_beta_MR;
    I_beta_MR    =    P.I_beta_MR;
    M_beta_MR    =    P.M_beta_MR;
    eps_MR       =    P.eps_MR;
    K_1_MR       =    P.K_1_MR;
    gamma_s      =    P.gamma_s_MR;
    P2_MR        =    P.P2_MR;

    r_H          =    [-(P.STA_H-P.STA_CG); P.BL_H-P.BL_CG ; -(P.WL_H-
P.WL_CG)];

    % input adaptation
    theta_0_MR   =        commands(1);
    theta_1s_MR  =        commands(2);
    theta_1c_MR  =        commands(3);

    % hub velocity and rates in bH
    lv_H         =    rh_j(-gamma_s)*(lv_B + cross(ar_B,r_H));
    ar_H         =    rh_j(-gamma_s)*ar_B;
    d_ar_H       =    rh_j(-gamma_s)*d_ar_B;
    % hub-wind angle, velocity and rate
    if  (abs(lv_H(1))<gTOL)&&(abs(lv_H(2))<gTOL)
        beta_w   =  0.0;
    else
        beta_w   =    atan2(lv_H(2),lv_H(1));
    end
    lv_W         =    rh_k(beta_w)*lv_H;
    ar_W         =    rh_k(beta_w)*ar_H;
```

```
    d_ar_W      =   rh_k(beta_w)*d_ar_H;

    % hub-wind adimensional angular rate and acceleration
    p_hw_MR     =   ar_W(1)/omega_ref_MR;
    q_hw_MR     =   ar_W(2)/omega_ref_MR;
    d_p_hw_MR   =   d_ar_W(1)/omega_ref_MR^2;
    d_q_hw_MR   =   d_ar_W(2)/omega_ref_MR^2;

    % advance and vertical velocity ratio
    mu_MR       =   lv_W(1)/(omega_ref_MR*R_MR);
    mu_z_MR         =   lv_W(3)/(omega_ref_MR*R_MR);

    % pitch angles in bW (pHFD, p199)
   theta_1sw_MR   =   theta_1c_MR*sin(beta_w)+theta_1s_MR*cos(beta_w);
   theta_1cw_MR   =   theta_1c_MR*cos(beta_w)-theta_1s_MR*sin(beta_w);


    % iteration for lambda_0 and C_T

    k_11    =   P2_MR + (gamma_MR*K_1_MR*mu_MR^2/4) * (1/2 - eps_MR +
eps_MR^2/2) ;
    k_12    =   -gamma_MR*mu_MR/4 * (eps_MR/2 - eps_MR^2) ;
    k_13    =   -gamma_MR*K_1_MR*mu_MR/4 *(2/3 - eps_MR) ;
    k_21    =   -gamma_MR*mu_MR/2*(1/3 - eps_MR/2) ;
    k_22    =   P2_MR - 1 + (gamma_MR*K_1_MR*mu_MR^2/8)*(1/2 - eps_MR +
eps_MR^2/2) ;
    k_23    =   gamma_MR/2*(1/4 -2*eps_MR/3 + eps_MR^2/2) +
gamma_MR*mu_MR^2/8*(1/2 - eps_MR + eps_MR^2/2) ;
    k_31    =   -gamma_MR*K_1_MR*mu_MR/2*(2/3 - eps_MR) ;
    k_32    =   -gamma_MR/2*(1/4 -2*eps_MR/3 + eps_MR^2/2) +
gamma_MR*mu_MR^2/8*(1/2 - eps_MR + eps_MR^2/2) ;
    k_33    =   P2_MR - 1 + (3*gamma_MR*K_1_MR*mu_MR^2/8)*(1/2 - eps_MR +
eps_MR^2/2) ;
    K       =   Omega_MR^2*[k_11, k_12, k_13; k_21, k_22, k_23; k_31,
k_32, k_33];

    lambda_0_MR =   loc_pr_mi_MR;
    %lambda_0_MR_init =  lambda_0_MR

    F1      =   -
M_beta_MR/(I_beta_MR*Omega_MR^2)+(1/2)*theta_0_MR*gamma_MR*(1/4-
(1/3)*eps_MR+(1/2)*mu_MR^2*(1/2-
eps_MR+(1/2)*eps_MR^2))+(1/2)*theta_t_MR*gamma_MR*(1/5-
(1/4)*eps_MR+(1/2)*mu_MR^2*(1/3-
(1/2)*eps_MR))+(1/2)*theta_1sw_MR*gamma_MR*mu_MR*(1/3-
(1/2)*eps_MR)+(1/2)*(mu_z_MR-lambda_0_MR)*gamma_MR*(1/3-
(1/2)*eps_MR)+(1/8)*p_hw_MR*gamma_MR*mu_MR*(2/3-eps_MR);
    F2      =   -M_beta_MR/(I_beta_MR*Omega_MR^2)-
(1/2)*theta_1cw_MR*gamma_MR*(1/4-(1/3)*eps_MR+(1/4)*mu_MR^2*(1/2-
eps_MR+(1/2)*eps_MR^2))-d_q_hw_MR-
2*p_hw_MR*(1+eps_MR*R_MR*M_beta_MR/(g*I_beta_MR))-
(1/2)*q_hw_MR*gamma_MR*(1/4-(1/3)*eps_MR);
    F3      =   -(1/2)*theta_0_MR*gamma_MR*mu_MR*(2/3-eps_MR)-
(1/2)*theta_t_MR*gamma_MR*mu_MR*(1/2-(2/3)*eps_MR)-
(1/2)*theta_1sw_MR*gamma_MR*(1/4-(1/3)*eps_MR+(3/4)*mu_MR^2*(1/2-
eps_MR+(1/2)*eps_MR^2))-(1/2)*(mu_z_MR-lambda_0_MR)*gamma_MR*mu_MR*(1/2-
```

```
eps_MR+(1/2)*eps_MR^2)-d_p_hw_MR-(1/2)*p_hw_MR*gamma_MR*(1/4-
(1/3)*eps_MR)+2*q_hw_MR*(1+eps_MR*R_MR*M_beta_MR/(g*I_beta_MR));
    f       =   Omega_MR^2*[F1;F2;F3];
    Beta_m          =   K\f;
    beta_0_MR       =   Beta_m(1);
    beta_1cw_MR     =   -Beta_m(2);
    beta_1sw_MR     =   -Beta_m(3);

    ct_MR       =   a_MR*s_MR/2*((1/2)*(mu_z_MR-lambda_0_MR)*(-
eps_MR^2+1)+theta_0_MR*(1/3+(1/2)*mu_MR^2*(1-
eps_MR))+theta_t_MR*(1/4+(1/4)*mu_MR^2*(-eps_MR^2+1))-
(1/2)*(K_1_MR*beta_1sw_MR-theta_1sw_MR)*mu_MR*(-eps_MR^2+1)-
beta_0_MR*K_1_MR*(1/3+(1/2)*mu_MR^2*(1-eps_MR))-
(1/2)*beta_1cw_MR*mu_MR*eps_MR*(1-eps_MR)+(1/4)*p_hw_MR*mu_MR*(1-
eps_MR)^2);
    Lambda      =   mu_MR^2+(lambda_0_MR-mu_z_MR)^2;
    h_i         =   -(2*lambda_0_MR*sqrt(Lambda)-
ct_MR)*Lambda/(2*Lambda^(3/2)+a_MR*s_MR/4*Lambda-ct_MR*(mu_z_MR-
lambda_0_MR));
    i_iter = 1;
    while abs(h_i/lambda_0_MR)>gTOL && i_iter<50
        lambda_0_MR =   lambda_0_MR+0.6*h_i;
        F1          =   -
M_beta_MR/(I_beta_MR*Omega_MR^2)+(1/2)*theta_0_MR*gamma_MR*(1/4-
(1/3)*eps_MR+(1/2)*mu_MR^2*(1/2-
eps_MR+(1/2)*eps_MR^2))+(1/2)*theta_t_MR*gamma_MR*(1/5-
(1/4)*eps_MR+(1/2)*mu_MR^2*(1/3-
(1/2)*eps_MR))+(1/2)*theta_1sw_MR*gamma_MR*mu_MR*(1/3-
(1/2)*eps_MR)+(1/2)*(mu_z_MR-lambda_0_MR)*gamma_MR*(1/3-
(1/2)*eps_MR)+(1/8)*p_hw_MR*gamma_MR*mu_MR*(2/3-eps_MR);
        F2          =   -M_beta_MR/(I_beta_MR*Omega_MR^2)-
(1/2)*theta_1cw_MR*gamma_MR*(1/4-(1/3)*eps_MR+(1/4)*mu_MR^2*(1/2-
eps_MR+(1/2)*eps_MR^2))-d_q_hw_MR-
2*p_hw_MR*(1+eps_MR*R_MR*M_beta_MR/(g*I_beta_MR))-
(1/2)*q_hw_MR*gamma_MR*(1/4-(1/3)*eps_MR);
        F3          =   -(1/2)*theta_0_MR*gamma_MR*mu_MR*(2/3-eps_MR)-
(1/2)*theta_t_MR*gamma_MR*mu_MR*(1/2-(2/3)*eps_MR)-
(1/2)*theta_1sw_MR*gamma_MR*(1/4-(1/3)*eps_MR+(3/4)*mu_MR^2*(1/2-
eps_MR+(1/2)*eps_MR^2))-(1/2)*(mu_z_MR-lambda_0_MR)*gamma_MR*mu_MR*(1/2-
eps_MR+(1/2)*eps_MR^2)-d_p_hw_MR-(1/2)*p_hw_MR*gamma_MR*(1/4-
(1/3)*eps_MR)+2*q_hw_MR*(1+eps_MR*R_MR*M_beta_MR/(g*I_beta_MR));
        f           =   Omega_MR^2*[F1;F2;F3];
        Beta_m      =   K\f;
        beta_0_MR   =   Beta_m(1);
        beta_1cw_MR =   -Beta_m(2);
        beta_1sw_MR =   -Beta_m(3);
        ct_MR       =   a_MR*s_MR/2*((1/2)*(mu_z_MR-lambda_0_MR)*(-
eps_MR^2+1)+theta_0_MR*(1/3+(1/2)*mu_MR^2*(1-
eps_MR))+theta_t_MR*(1/4+(1/4)*mu_MR^2*(-eps_MR^2+1))-
(1/2)*(K_1_MR*beta_1sw_MR-theta_1sw_MR)*mu_MR*(-eps_MR^2+1)-
beta_0_MR*K_1_MR*(1/3+(1/2)*mu_MR^2*(1-eps_MR))-
(1/2)*beta_1cw_MR*mu_MR*eps_MR*(1-eps_MR)+(1/4)*p_hw_MR*mu_MR*(1-
eps_MR)^2);
        Lambda      =   mu_MR^2+(lambda_0_MR-mu_z_MR)^2;
```

```
         h_i         =    -(2*lambda_0_MR*sqrt(Lambda)-
ct_MR)*Lambda/(2*Lambda^(3/2)+a_MR*s_MR/4*Lambda-ct_MR*(mu_z_MR-
lambda_0_MR));
         i_iter=i_iter+1;
    end
    if i_iter>=50
        lambda_0_MR =    lambda_0_MR+0.3*h_i;
        'peta_MR'
    end
    pr_mi_MR       =    lambda_0_MR;
    %lambda_0_MR_fin =  pr_mi_MR

% ¿es necesario comprobar coincidencia valores de CT y T?

   % wake angle and linear inflow in bW (pHFD, p121)
    chi_MR          =    atan2 (mu_MR,lambda_0_MR-mu_z_MR);

    delta        = 0.009+0.3*(6*ct_MR/(a_MR/s_MR))^2;
    T_MR    =
(1/2)*n_b_MR*rho*a_MR*c_MR*R_MR^3*Omega_MR^2*((1/2)*(mu_z_MR-
lambda_0_MR)*(-eps_MR^2+1)+theta_0_MR*(1/3+(1/2)*mu_MR^2*(1-
eps_MR))+theta_t_MR*(1/4+(1/4)*mu_MR^2*(-eps_MR^2+1))-
(1/2)*(K_1_MR*beta_1sw_MR-theta_1sw_MR)*mu_MR*(-eps_MR^2+1)-
beta_0_MR*K_1_MR*(1/3+(1/2)*mu_MR^2*(1-eps_MR))-
(1/2)*beta_1cw_MR*mu_MR*eps_MR*(1-eps_MR)+(1/4)*p_hw_MR*mu_MR*(1-
eps_MR)^2);
    H_W_MR  =
(1/2)*n_b_MR*rho*a_MR*c_MR*R_MR^3*Omega_MR^2*((1/2)*delta*mu_MR*(-
eps_MR^2+1)/a_MR-(-(1/4)*beta_0_MR*K_1_MR+(1/4)*theta_0_MR)*(2*(mu_z_MR-
lambda_0_MR)*mu_MR*(1-eps_MR)-(eps_MR-
2/3)*beta_1cw_MR+(2/3)*beta_1cw_MR+(2/3)*p_hw_MR)-
(1/4)*theta_t_MR*((mu_z_MR-lambda_0_MR)*mu_MR*(-eps_MR^2+1)-
((2/3)*eps_MR-
1/2)*beta_1cw_MR+(1/2)*beta_1cw_MR+(1/2)*p_hw_MR)+((1/4)*K_1_MR*beta_1cw_
MR-(1/4)*theta_1cw_MR)*((1/4)*mu_MR*beta_1sw_MR*(-eps_MR^2+1)-
(1/4)*mu_MR*(1-eps_MR)^2*beta_1sw_MR+(2/3)*beta_0_MR+(1/4)*mu_MR*(-
eps_MR^2+1)*q_hw_MR)+((1/4)*K_1_MR*beta_1sw_MR-
(1/4)*theta_1sw_MR)*((3/4)*mu_MR*(1-eps_MR)^2*beta_1cw_MR+(-
eps_MR^2+1)*(mu_z_MR-lambda_0_MR+(1/4)*beta_1cw_MR*mu_MR)+(3/4)*mu_MR*(-
eps_MR^2+1)*p_hw_MR)+(mu_z_MR-lambda_0_MR)*eps_MR*(1-eps_MR)*beta_1cw_MR-
(3/4)*beta_1cw_MR*(mu_z_MR-lambda_0_MR)*(-eps_MR^2+1)+(1/4)*(2/3-
eps_MR)*beta_0_MR*beta_1sw_MR-(1/6)*q_hw_MR*beta_0_MR-(1/4)*p_hw_MR*(-
2*eps_MR^2+2)*(mu_z_MR-lambda_0_MR)+(1/4)*mu_MR*(eps_MR*(1-eps_MR)*(-
beta_1cw_MR^2+beta_1sw_MR^2)+(1/4)*(1-eps_MR)^2*(-
beta_1cw_MR^2+beta_1sw_MR^2)-(-(1/2)*eps_MR^2+1/2)*(-
(5/2)*beta_1cw_MR^2+(1/2)*beta_1sw_MR^2-
2*beta_0_MR^2)+(1/4)*beta_1cw_MR*(-
eps_MR^2+1)*p_hw_MR+(1/4)*beta_1sw_MR*(-eps_MR^2+1)*q_hw_MR));
    Y_W_MR   =  (1/2)*n_b_MR*rho*a_MR*c_MR*R_MR^3*Omega_MR^2*(-(-
(1/4)*beta_0_MR*K_1_MR+(1/4)*theta_0_MR)*(-(eps_MR-
2/3)*beta_1sw_MR+(2/3)*beta_1sw_MR+3*beta_0_MR*(-
eps_MR^2+1)*mu_MR+2*beta_1sw_MR*(1-eps_MR)*mu_MR^2-(2/3)*q_hw_MR)-
(1/4)*theta_t_MR*(-((2/3)*eps_MR-
1/2)*beta_1sw_MR+(1/2)*beta_1sw_MR+2*beta_0_MR*mu_MR+beta_1sw_MR*(-
eps_MR^2+1)*mu_MR^2-(1/2)*q_hw_MR)-((1/4)*K_1_MR*beta_1cw_MR-
(1/4)*theta_1cw_MR)*((mu_z_MR-lambda_0_MR)*(-eps_MR^2+1)+mu_MR*(-
(5/4)*beta_1cw_MR*(-eps_MR^2+1)+(1/4)*(1-
```

```
eps_MR)^2*beta_1cw_MR)+(1/4)*mu_MR*(-eps_MR^2+1)*p_hw_MR)-
((1/4)*K_1_MR*beta_1sw_MR-(1/4)*theta_1sw_MR)*(-(2/3)*beta_0_MR+mu_MR*(-
(7/4)*beta_1sw_MR*(-eps_MR^2+1)-(1/4)*(1-
eps_MR)^2*beta_1sw_MR+(1/4)*q_hw_MR)-2*mu_MR^2*beta_0_MR*(1-eps_MR))-
(1/2)*(mu_z_MR-lambda_0_MR)*(1-eps_MR)^2*beta_1sw_MR-
(1/6)*beta_0_MR*p_hw_MR-(1/2)*beta_0_MR*(1/3-(1/2)*eps_MR)*beta_1cw_MR-
(1/4)*beta_1sw_MR*(mu_z_MR-lambda_0_MR)*(-eps_MR^2+1)+(1/4)*q_hw_MR*(-
2*eps_MR^2+2)*(mu_z_MR-lambda_0_MR)-(1/4)*mu_MR*(6*beta_0_MR*(mu_z_MR-
lambda_0_MR)*(1-eps_MR)-(1/2)*beta_1cw_MR*beta_1sw_MR*(-eps_MR^2+1)-
(1/2)*(1-eps_MR)^2*beta_1cw_MR*beta_1sw_MR+(5/4)*beta_1sw_MR*(-
eps_MR^2+1)*p_hw_MR+(7/4)*beta_1cw_MR*(-
eps_MR^2+1)*q_hw_MR)+mu_MR^2*beta_0_MR*beta_1cw_MR*(1-eps_MR));
    M_W_MR  = (1/2)*n_b_MR*(-K_beta_MR*beta_1cw_MR-
eps_MR*R_MR*M_beta_MR*beta_1cw_MR*Omega_MR^2/g)-
(1/2)*n_b_MR*I_beta_MR*Omega_MR^2*gamma_MR*eps_MR*(-(K_1_MR*beta_1cw_MR-
theta_1cw_MR)*(1/6+(1/8)*mu_MR^2*(1-eps_MR))-(1/4)*beta_0_MR*(-
eps_MR^2+1)*mu_MR-(1/8)*beta_1sw_MR*(1-eps_MR)*mu_MR^2-(1/6-
(1/4)*eps_MR)*beta_1sw_MR+(1/6)*q_hw_MR);
    L_W_MR  = (1/2)*n_b_MR*(-K_beta_MR*beta_1sw_MR-
eps_MR*R_MR*M_beta_MR*beta_1sw_MR*Omega_MR^2/g)-
(1/2)*n_b_MR*I_beta_MR*Omega_MR^2*gamma_MR*eps_MR*((1/2)*mu_MR*(-
eps_MR^2+1)*(-K_1_MR*beta_0_MR+theta_0_MR)-(K_1_MR*beta_1sw_MR-
theta_1sw_MR)*(1/6+(3/8)*mu_MR^2*(1-
eps_MR))+(1/3)*mu_MR*theta_t_MR+(1/2)*(mu_z_MR-lambda_0_MR)*mu_MR*(1-
eps_MR)-(1/8)*mu_MR^2*(1-eps_MR)*beta_1cw_MR+(1/6-
(1/4)*eps_MR)*beta_1cw_MR+(1/6)*p_hw_MR);
    Q_MR    =
(1/2)*n_b_MR*rho*a_MR*c_MR*R_MR^4*Omega_MR^2*((1/4)*delta*(1+mu_MR^2*(-
eps_MR^2+1))/a_MR-(-K_1_MR*beta_0_MR+theta_0_MR)*((1/3)*mu_z_MR-
(1/3)*lambda_0_MR-
(1/4)*mu_MR*eps_MR*beta_1cw_MR+(1/6)*p_hw_MR*mu_MR)+(K_1_MR*beta_1cw_MR-
theta_1cw_MR)*(-(1/8-(1/6)*eps_MR)*beta_1sw_MR-(1/6)*beta_0_MR*mu_MR-
(1/16)*beta_1sw_MR*(-
eps_MR^2+1)*mu_MR^2+(1/8)*q_hw_MR)+(K_1_MR*beta_1sw_MR-
theta_1sw_MR)*((1/8-(1/6)*eps_MR)*beta_1cw_MR+(-
(1/2)*eps_MR^2+1/2)*((1/2)*(mu_z_MR-lambda_0_MR)*mu_MR-
(1/8)*beta_1cw_MR*mu_MR^2)+(1/8)*p_hw_MR)-theta_t_MR*((1/4)*mu_z_MR-
(1/4)*lambda_0_MR-(1/6)*mu_MR*eps_MR*beta_1cw_MR)-(-
(1/2)*eps_MR^2+1/2)*(-(mu_z_MR-lambda_0_MR)*mu_MR*beta_1cw_MR+(mu_z_MR-
lambda_0_MR)^2-
mu_MR*eps_MR*beta_0_MR*beta_1sw_MR+mu_MR^2*((1/2)*beta_0_MR^2+(3/8)*beta_
1cw_MR^2+(1/8)*beta_1sw_MR^2))-(1/3)*mu_MR*beta_0_MR*beta_1sw_MR-
q_hw_MR*(-(1/3)*beta_0_MR*mu_MR-(1/4-(1/3)*eps_MR)*beta_1sw_MR)-
p_hw_MR*(1/4-(1/3)*eps_MR)*beta_1cw_MR-(1/8)*q_hw_MR^2-(1/8)*p_hw_MR^2-
(1/4-
(2/3)*eps_MR+(1/2)*eps_MR^2)*((1/2)*beta_1sw_MR^2+(1/2)*beta_1cw_MR^2));
   % forces and moments at CG

    F_H      =    rh_k(-beta_w)*[-H_W_MR;Y_W_MR;-T_MR];        % cN, p14
    F_B      =    rh_j(gamma_s)*F_H;

    M_H      =    rh_k(-beta_w)*[L_W_MR;M_W_MR;Q_MR];              % cN, p14
    M_B      =    rh_j(gamma_s)*M_H;
```

```
    F_CG    =   F_B;
    M_CG    =   M_B + cross(r_H,F_B);

    v_i_MR  =   lambda_0_MR*Omega_MR*R_MR;


    % salida (6+6+8 elementos)
    beta_1c_MR=0;
    beta_1s_MR=0;
    out = [beta_0_MR; beta_1cw_MR; beta_1sw_MR; beta_w; beta_1c_MR; 
beta_1s_MR; ...
            H_W_MR; Y_W_MR; T_MR; L_W_MR; M_W_MR; Q_MR; ...
            F_CG; M_CG; chi_MR; v_i_MR];

end
```

C.4: Rotor principal (girodino)

```
function out = ral_fm_main_rotor(commands, lv_B, ar_B, d_ar_B, 
loc_pr_mi_MR, P, f_Om)

    global pr_mi_MR;

    % parameters adaptation

    gTOL        =   P.gtol;
    rho         =   P.rho;
    g           =   P.g;

    omega_ref_MR=   P.Omega_MR;
    Omega_MR    =   P.Omega_MR;
    R_MR        =   P.R_MR;

    a_MR        =   P.a_MR;
    c_MR        =   P.c_MR;
    s_MR        =   P.sol_MR;
    n_b_MR      =   P.n_b_MR;
    gamma_MR    =   P.gamma_MR;
    theta_t_MR  =   P.theta_t_MR;

    K_beta_MR   =   P.K_beta_MR;
    I_beta_MR   =   P.I_beta_MR;
    M_beta_MR   =   P.M_beta_MR;
    eps_MR      =   P.eps_MR;
    K_1_MR      =   P.K_1_MR;
    gamma_s     =   P.gamma_s_MR;
    P2_MR       =   P.P2_MR;

    r_H         =   [-(P.STA_H-P.STA_CG); P.BL_H-P.BL_CG ; -(P.WL_H-
P.WL_CG)];


    % ralentizacion rotores
```

```
    omega_ref_MR =    f_Om*P.Omega_MR;
    Omega_MR     =    f_Om*P.Omega_MR;


    % input adaptation
    theta_0_MR  =       commands(1);
    theta_1s_MR =       commands(2);
    theta_1c_MR =       commands(3);

    % hub velocity and rates in bH
    lv_H        =   rh_j(-gamma_s)*(lv_B + cross(ar_B,r_H));
    ar_H        =   rh_j(-gamma_s)*ar_B;
    d_ar_H      =   rh_j(-gamma_s)*d_ar_B;
    % hub-wind angle, velocity and rate
    if  (abs(lv_H(1))<gTOL)&&(abs(lv_H(2))<gTOL)
        beta_w  = 0.0;
    else
        beta_w  =   atan2(lv_H(2),lv_H(1));
    end
    lv_W        =   rh_k(beta_w)*lv_H;
    ar_W        =   rh_k(beta_w)*ar_H;
    d_ar_W      =   rh_k(beta_w)*d_ar_H;

    % hub-wind adimensional angular rate and acceleration
    p_hw_MR     =   ar_W(1)/omega_ref_MR;
    q_hw_MR     =   ar_W(2)/omega_ref_MR;
    d_p_hw_MR   =   d_ar_W(1)/omega_ref_MR^2;
    d_q_hw_MR   =   d_ar_W(2)/omega_ref_MR^2;

    % advance and vertical velocity ratio
    mu_MR       =   lv_W(1)/(omega_ref_MR*R_MR);
    mu_z_MR         =   lv_W(3)/(omega_ref_MR*R_MR);

    % pitch angles in bW (pHFD, p199)
    theta_1sw_MR   =   theta_1c_MR*sin(beta_w)+theta_1s_MR*cos(beta_w);
    theta_1cw_MR   =   theta_1c_MR*cos(beta_w)-theta_1s_MR*sin(beta_w);


    % iteration for lambda_0 and C_T

    k_11    =   P2_MR + (gamma_MR*K_1_MR*mu_MR^2/4) * (1/2 - eps_MR +
eps_MR^2/2) ;
    k_12    =   -gamma_MR*mu_MR/4 * (eps_MR/2 - eps_MR^2) ;
    k_13    =   -gamma_MR*K_1_MR*mu_MR/4 *(2/3 - eps_MR) ;
    k_21    =   -gamma_MR*mu_MR/2*(1/3 - eps_MR/2) ;
    k_22    =   P2_MR - 1 + (gamma_MR*K_1_MR*mu_MR^2/8)*(1/2 - eps_MR +
eps_MR^2/2) ;
    k_23    =   gamma_MR/2*(1/4 -2*eps_MR/3 + eps_MR^2/2) +
gamma_MR*mu_MR^2/8*(1/2 - eps_MR + eps_MR^2/2) ;
    k_31    =   -gamma_MR*K_1_MR*mu_MR/2*(2/3 - eps_MR) ;
    k_32    =   -gamma_MR/2*(1/4 -2*eps_MR/3 + eps_MR^2/2) +
gamma_MR*mu_MR^2/8*(1/2 - eps_MR + eps_MR^2/2) ;
```

```
    k_33    =   P2_MR - 1 + (3*gamma_MR*K_1_MR*mu_MR^2/8)*(1/2 - eps_MR +
eps_MR^2/2) ;
    K       =   Omega_MR^2*[k_11, k_12, k_13; k_21, k_22, k_23; k_31,
k_32, k_33];

    lambda_0_MR =   loc_pr_mi_MR;

    F1      =   -
M_beta_MR/(I_beta_MR*Omega_MR^2)+(1/2)*theta_0_MR*gamma_MR*(1/4-
(1/3)*eps_MR+(1/2)*mu_MR^2*(1/2-
eps_MR+(1/2)*eps_MR^2))+(1/2)*theta_t_MR*gamma_MR*(1/5-
(1/4)*eps_MR+(1/2)*mu_MR^2*(1/3-
(1/2)*eps_MR))+(1/2)*theta_1sw_MR*gamma_MR*mu_MR*(1/3-
(1/2)*eps_MR)+(1/2)*(mu_z_MR-lambda_0_MR)*gamma_MR*(1/3-
(1/2)*eps_MR)+(1/8)*p_hw_MR*gamma_MR*mu_MR*(2/3-eps_MR);
    F2      =   -M_beta_MR/(I_beta_MR*Omega_MR^2)-
(1/2)*theta_1cw_MR*gamma_MR*(1/4-(1/3)*eps_MR+(1/4)*mu_MR^2*(1/2-
eps_MR+(1/2)*eps_MR^2))-d_q_hw_MR-
2*p_hw_MR*(1+eps_MR*R_MR*M_beta_MR/(g*I_beta_MR))-
(1/2)*q_hw_MR*gamma_MR*(1/4-(1/3)*eps_MR);
    F3      =   -(1/2)*theta_0_MR*gamma_MR*mu_MR*(2/3-eps_MR)-
(1/2)*theta_t_MR*gamma_MR*mu_MR*(1/2-(2/3)*eps_MR)-
(1/2)*theta_1sw_MR*gamma_MR*(1/4-(1/3)*eps_MR+(3/4)*mu_MR^2*(1/2-
eps_MR+(1/2)*eps_MR^2))-(1/2)*(mu_z_MR-lambda_0_MR)*gamma_MR*mu_MR*(1/2-
eps_MR+(1/2)*eps_MR^2)-d_p_hw_MR-(1/2)*p_hw_MR*gamma_MR*(1/4-
(1/3)*eps_MR)+2*q_hw_MR*(1+eps_MR*R_MR*M_beta_MR/(g*I_beta_MR));
    f       =   Omega_MR^2*[F1;F2;F3];
    Beta_m          =   K\f;
    beta_0_MR       =   Beta_m(1);
    beta_1cw_MR     =   -Beta_m(2);
    beta_1sw_MR     =   -Beta_m(3);

    ct_MR       =   a_MR*s_MR/2*((1/2)*(mu_z_MR-lambda_0_MR)*(-
eps_MR^2+1)+theta_0_MR*(1/3+(1/2)*mu_MR^2*(1-
eps_MR))+theta_t_MR*(1/4+(1/4)*mu_MR^2*(-eps_MR^2+1))-
(1/2)*(K_1_MR*beta_1sw_MR-theta_1sw_MR)*mu_MR*(-eps_MR^2+1)-
beta_0_MR*K_1_MR*(1/3+(1/2)*mu_MR^2*(1-eps_MR))-
(1/2)*beta_1cw_MR*mu_MR*eps_MR*(1-eps_MR)+(1/4)*p_hw_MR*mu_MR*(1-
eps_MR)^2);
    Lambda      =   mu_MR^2+(lambda_0_MR-mu_z_MR)^2;
    h_i         =   -(2*lambda_0_MR*sqrt(Lambda)-
ct_MR)*Lambda/(2*Lambda^(3/2)+a_MR*s_MR/4*Lambda-ct_MR*(mu_z_MR-
lambda_0_MR));
    i_iter = 1;
    while abs(h_i/lambda_0_MR)>gTOL && i_iter<50
        lambda_0_MR =   lambda_0_MR+0.6*h_i;
        F1          =   -
M_beta_MR/(I_beta_MR*Omega_MR^2)+(1/2)*theta_0_MR*gamma_MR*(1/4-
(1/3)*eps_MR+(1/2)*mu_MR^2*(1/2-
eps_MR+(1/2)*eps_MR^2))+(1/2)*theta_t_MR*gamma_MR*(1/5-
(1/4)*eps_MR+(1/2)*mu_MR^2*(1/3-
(1/2)*eps_MR))+(1/2)*theta_1sw_MR*gamma_MR*mu_MR*(1/3-
(1/2)*eps_MR)+(1/2)*(mu_z_MR-lambda_0_MR)*gamma_MR*(1/3-
(1/2)*eps_MR)+(1/8)*p_hw_MR*gamma_MR*mu_MR*(2/3-eps_MR);
        F2          =   -M_beta_MR/(I_beta_MR*Omega_MR^2)-
(1/2)*theta_1cw_MR*gamma_MR*(1/4-(1/3)*eps_MR+(1/4)*mu_MR^2*(1/2-
eps_MR+(1/2)*eps_MR^2))-d_q_hw_MR-
```

```
2*p_hw_MR*(1+eps_MR*R_MR*M_beta_MR/(g*I_beta_MR))-
(1/2)*q_hw_MR*gamma_MR*(1/4-(1/3)*eps_MR);
        F3          =   -(1/2)*theta_0_MR*gamma_MR*mu_MR*(2/3-eps_MR)-
(1/2)*theta_t_MR*gamma_MR*mu_MR*(1/2-(2/3)*eps_MR)-
(1/2)*theta_1sw_MR*gamma_MR*(1/4-(1/3)*eps_MR+(3/4)*mu_MR^2*(1/2-
eps_MR+(1/2)*eps_MR^2))-(1/2)*(mu_z_MR-lambda_0_MR)*gamma_MR*mu_MR*(1/2-
eps_MR+(1/2)*eps_MR^2)-d_p_hw_MR-(1/2)*p_hw_MR*gamma_MR*(1/4-
(1/3)*eps_MR)+2*q_hw_MR*(1+eps_MR*R_MR*M_beta_MR/(g*I_beta_MR));
        f           =   Omega_MR^2*[F1;F2;F3];
        Beta_m      =   K\f;
        beta_0_MR   =   Beta_m(1);
        beta_1cw_MR =   -Beta_m(2);
        beta_1sw_MR =   -Beta_m(3);
        ct_MR       =   a_MR*s_MR/2*((1/2)*(mu_z_MR-lambda_0_MR)*(-
eps_MR^2+1)+theta_0_MR*(1/3+(1/2)*mu_MR^2*(1-
eps_MR))+theta_t_MR*(1/4+(1/4)*mu_MR^2*(-eps_MR^2+1))-
(1/2)*(K_1_MR*beta_1sw_MR-theta_1sw_MR)*mu_MR*(-eps_MR^2+1)-
beta_0_MR*K_1_MR*(1/3+(1/2)*mu_MR^2*(1-eps_MR))-
(1/2)*beta_1cw_MR*mu_MR*eps_MR*(1-eps_MR)+(1/4)*p_hw_MR*mu_MR*(1-
eps_MR)^2);
        Lambda      =   mu_MR^2+(lambda_0_MR-mu_z_MR)^2;
         h_i        =   -(2*lambda_0_MR*sqrt(Lambda)-
ct_MR)*Lambda/(2*Lambda^(3/2)+a_MR*s_MR/4*Lambda-ct_MR*(mu_z_MR-
lambda_0_MR));
         i_iter=i_iter+1;
    end
    if i_iter>=50
        lambda_0_MR =   lambda_0_MR+0.3*h_i;
        'peta_MR'
    end
    pr_mi_MR      =   lambda_0_MR;
    %lambda_0_MR_fin =  pr_mi_MR

% ¿es necesario comprobar coincidencia valores de CT y T?

   % wake angle and linear inflow in bW (pHFD, p121)
    chi_MR         =   atan2 (mu_MR,lambda_0_MR-mu_z_MR);
    delta       = 0.009+0.3*(6*ct_MR/(a_MR/s_MR))^2;
    T_MR    =
(1/2)*n_b_MR*rho*a_MR*c_MR*R_MR^3*Omega_MR^2*((1/2)*(mu_z_MR-
lambda_0_MR)*(-eps_MR^2+1)+theta_0_MR*(1/3+(1/2)*mu_MR^2*(1-
eps_MR))+theta_t_MR*(1/4+(1/4)*mu_MR^2*(-eps_MR^2+1))-
(1/2)*(K_1_MR*beta_1sw_MR-theta_1sw_MR)*mu_MR*(-eps_MR^2+1)-
beta_0_MR*K_1_MR*(1/3+(1/2)*mu_MR^2*(1-eps_MR))-
(1/2)*beta_1cw_MR*mu_MR*eps_MR*(1-eps_MR)+(1/4)*p_hw_MR*mu_MR*(1-
eps_MR)^2);
    H_W_MR  =
(1/2)*n_b_MR*rho*a_MR*c_MR*R_MR^3*Omega_MR^2*((1/2)*delta*mu_MR*(-
eps_MR^2+1)/a_MR-(-(1/4)*beta_0_MR*K_1_MR+(1/4)*theta_0_MR)*(2*(mu_z_MR-
lambda_0_MR)*mu_MR*(1-eps_MR)-(eps_MR-
2/3)*beta_1cw_MR+(2/3)*beta_1cw_MR+(2/3)*p_hw_MR)-
(1/4)*theta_t_MR*((mu_z_MR-lambda_0_MR)*mu_MR*(-eps_MR^2+1)-
((2/3)*eps_MR-
```

```
1/2)*beta_1cw_MR+(1/2)*beta_1cw_MR+(1/2)*p_hw_MR)+((1/4)*K_1_MR*beta_1cw_
MR-(1/4)*theta_1cw_MR)*((1/4)*mu_MR*beta_1sw_MR*(-eps_MR^2+1)-
(1/4)*mu_MR*(1-eps_MR)^2*beta_1sw_MR+(2/3)*beta_0_MR+(1/4)*mu_MR*(-
eps_MR^2+1)*q_hw_MR)+((1/4)*K_1_MR*beta_1sw_MR-
(1/4)*theta_1sw_MR)*((3/4)*mu_MR*(1-eps_MR)^2*beta_1cw_MR+(-
eps_MR^2+1)*(mu_z_MR-lambda_0_MR+(1/4)*beta_1cw_MR*mu_MR)+(3/4)*mu_MR*(-
eps_MR^2+1)*p_hw_MR)+(mu_z_MR-lambda_0_MR)*eps_MR*(1-eps_MR)*beta_1cw_MR-
(3/4)*beta_1cw_MR*(mu_z_MR-lambda_0_MR)*(-eps_MR^2+1)+(1/4)*(2/3-
eps_MR)*beta_0_MR*beta_1sw_MR-(1/6)*q_hw_MR*beta_0_MR-(1/4)*p_hw_MR*(-
2*eps_MR^2+2)*(mu_z_MR-lambda_0_MR)+(1/4)*mu_MR*(eps_MR*(1-eps_MR)*(-
beta_1cw_MR^2+beta_1sw_MR^2)+(1/4)*(1-eps_MR)^2*(-
beta_1cw_MR^2+beta_1sw_MR^2)-(-(1/2)*eps_MR^2+1/2)*(-
(5/2)*beta_1cw_MR^2+(1/2)*beta_1sw_MR^2-
2*beta_0_MR^2)+(1/4)*beta_1cw_MR*(-
eps_MR^2+1)*p_hw_MR+(1/4)*beta_1sw_MR*(-eps_MR^2+1)*q_hw_MR));
    Y_W_MR  = (1/2)*n_b_MR*rho*a_MR*c_MR*R_MR^3*Omega_MR^2*(-(-
(1/4)*beta_0_MR*K_1_MR+(1/4)*theta_0_MR)*(-(eps_MR-
2/3)*beta_1sw_MR+(2/3)*beta_1sw_MR+3*beta_0_MR*(-
eps_MR^2+1)*mu_MR+2*beta_1sw_MR*(1-eps_MR)*mu_MR^2-(2/3)*q_hw_MR)-
(1/4)*theta_t_MR*(-((2/3)*eps_MR-
1/2)*beta_1sw_MR+(1/2)*beta_1sw_MR+2*beta_0_MR*mu_MR+beta_1sw_MR*(-
eps_MR^2+1)*mu_MR^2-(1/2)*q_hw_MR)-((1/4)*K_1_MR*beta_1cw_MR-
(1/4)*theta_1cw_MR)*((mu_z_MR-lambda_0_MR)*(-eps_MR^2+1)+mu_MR*(-
(5/4)*beta_1cw_MR*(-eps_MR^2+1)+(1/4)*(1-
eps_MR)^2*beta_1cw_MR)+(1/4)*mu_MR*(-eps_MR^2+1)*p_hw_MR)-
((1/4)*K_1_MR*beta_1sw_MR-(1/4)*theta_1sw_MR)*(-(2/3)*beta_0_MR+mu_MR*(-
(7/4)*beta_1sw_MR*(-eps_MR^2+1)-(1/4)*(1-
eps_MR)^2*beta_1sw_MR+(1/4)*q_hw_MR)-2*mu_MR^2*beta_0_MR*(1-eps_MR))-
(1/2)*(mu_z_MR-lambda_0_MR)*(1-eps_MR)^2*beta_1sw_MR-
(1/6)*beta_0_MR*p_hw_MR-(1/2)*beta_0_MR*(1/3-(1/2)*eps_MR)*beta_1cw_MR-
(1/4)*beta_1sw_MR*(mu_z_MR-lambda_0_MR)*(-eps_MR^2+1)+(1/4)*q_hw_MR*(-
2*eps_MR^2+2)*(mu_z_MR-lambda_0_MR)-(1/4)*mu_MR*(6*beta_0_MR*(mu_z_MR-
lambda_0_MR)*(1-eps_MR)-(1/2)*beta_1cw_MR*beta_1sw_MR*(-eps_MR^2+1)-
(1/2)*(1-eps_MR)^2*beta_1cw_MR*beta_1sw_MR+(5/4)*beta_1sw_MR*(-
eps_MR^2+1)*p_hw_MR+(7/4)*beta_1cw_MR*(-
eps_MR^2+1)*q_hw_MR)+mu_MR^2*beta_0_MR*beta_1cw_MR*(1-eps_MR));
    M_W_MR  = (1/2)*n_b_MR*(-K_beta_MR*beta_1cw_MR-
eps_MR*R_MR*M_beta_MR*beta_1cw_MR*Omega_MR^2/g)-
(1/2)*n_b_MR*I_beta_MR*Omega_MR^2*gamma_MR*eps_MR*(-(K_1_MR*beta_1cw_MR-
theta_1cw_MR)*(1/6+(1/8)*mu_MR^2*(1-eps_MR))-(1/4)*beta_0_MR*(-
eps_MR^2+1)*mu_MR-(1/8)*beta_1sw_MR*(1-eps_MR)*mu_MR^2-(1/6-
(1/4)*eps_MR)*beta_1sw_MR+(1/6)*q_hw_MR);
    L_W_MR  = (1/2)*n_b_MR*(-K_beta_MR*beta_1sw_MR-
eps_MR*R_MR*M_beta_MR*beta_1sw_MR*Omega_MR^2/g)-
(1/2)*n_b_MR*I_beta_MR*Omega_MR^2*gamma_MR*eps_MR*((1/2)*mu_MR*(-
eps_MR^2+1)*(-K_1_MR*beta_0_MR+theta_0_MR)-(K_1_MR*beta_1sw_MR-
theta_1sw_MR)*(1/6+(3/8)*mu_MR^2*(1-
eps_MR))+(1/3)*mu_MR*theta_t_MR+(1/2)*(mu_z_MR-lambda_0_MR)*mu_MR*(1-
eps_MR)-(1/8)*mu_MR^2*(1-eps_MR)*beta_1cw_MR+(1/6-
(1/4)*eps_MR)*beta_1cw_MR+(1/6)*p_hw_MR);
    Q_MR    =
(1/2)*n_b_MR*rho*a_MR*c_MR*R_MR^4*Omega_MR^2*((1/4)*delta*(1+mu_MR^2*(-
eps_MR^2+1))/a_MR-(-K_1_MR*beta_0_MR+theta_0_MR)*((1/3)*mu_z_MR-
(1/3)*lambda_0_MR-
(1/4)*mu_MR*eps_MR*beta_1cw_MR+(1/6)*p_hw_MR*mu_MR)+(K_1_MR*beta_1cw_MR-
theta_1cw_MR)*(-(1/8-(1/6)*eps_MR)*beta_1sw_MR-(1/6)*beta_0_MR*mu_MR-
(1/16)*beta_1sw_MR*(-
eps_MR^2+1)*mu_MR^2+(1/8)*q_hw_MR)+(K_1_MR*beta_1sw_MR-
```

```
theta_1sw_MR)*((1/8-(1/6)*eps_MR)*beta_1cw_MR+(-
(1/2)*eps_MR^2+1/2)*((1/2)*(mu_z_MR-lambda_0_MR)*mu_MR-
(1/8)*beta_1cw_MR*mu_MR^2)+(1/8)*p_hw_MR)-theta_t_MR*((1/4)*mu_z_MR-
(1/4)*lambda_0_MR-(1/6)*mu_MR*eps_MR*beta_1cw_MR)-(-
(1/2)*eps_MR^2+1/2)*(-(mu_z_MR-lambda_0_MR)*mu_MR*beta_1cw_MR+(mu_z_MR-
lambda_0_MR)^2-
mu_MR*eps_MR*beta_0_MR*beta_1sw_MR+mu_MR^2*((1/2)*beta_0_MR^2+(3/8)*beta_
1cw_MR^2+(1/8)*beta_1sw_MR^2))-(1/3)*mu_MR*beta_0_MR*beta_1sw_MR-
q_hw_MR*(-(1/3)*beta_0_MR*mu_MR-(1/4-(1/3)*eps_MR)*beta_1sw_MR)-
p_hw_MR*(1/4-(1/3)*eps_MR)*beta_1cw_MR-(1/8)*q_hw_MR^2-(1/8)*p_hw_MR^2-
(1/4-
(2/3)*eps_MR+(1/2)*eps_MR^2)*((1/2)*beta_1sw_MR^2+(1/2)*beta_1cw_MR^2));
   % forces and moments at CG

    F_H     =   rh_k(-beta_w)*[-H_W_MR;Y_W_MR;-T_MR];       % cN, p14
    F_B     =   rh_j(gamma_s)*F_H;

    M_H     =   rh_k(-beta_w)*[L_W_MR;M_W_MR;Q_MR];             % cN, p14
    M_B     =   rh_j(gamma_s)*M_H;

    F_CG    =   F_B;
    M_CG    =   M_B + cross(r_H,F_B);

    v_i_MR  =   lambda_0_MR*Omega_MR*R_MR;

    % salida
    beta_1c_MR=0;
    beta_1s_MR=0;
    out = [beta_0_MR; beta_1cw_MR; beta_1sw_MR; beta_w; beta_1c_MR;
beta_1s_MR; ...
            H_W_MR; Y_W_MR; T_MR; L_W_MR; M_W_MR; Q_MR; ...
            F_CG; M_CG; chi_MR; v_i_MR];

end
```

C.5: Rotor de cola (helicóptero)

```
function out = UH60_fm_tail_rotor(commands, lv_B, ar_B, chi_MR, v_i_MR,
loc_pr_mi_TR, P)

    global pr_mi_TR;

   % parameters adaptation
    gTOL        =   P.gtol;
    rho         =   P.rho;
    %g           =   P.g;
```

```
    omega_ref_TR=   P.Omega_TR;
    Omega_TR    =   P.Omega_TR;
    R_TR        =   P.R_TR;

    a_TR        =   P.a_TR;
    c_TR        =   P.c_TR;
    s_TR        =   P.sol_TR;
    n_b_TR      =   P.n_b_TR;
    gamma_TR    =   P.gamma_TR;
    theta_t_TR  =   P.theta_t_TR;
    beta_0_TR   =   P.beta_00_TR;
    K_1_TR      =   P.K_1_TR;
    K_ang       =   P.K_ang;
    r_TR        =   [-(P.STA_TR-P.STA_CG); P.BL_TR-P.BL_CG  ; -(P.WL_TR-
P.WL_CG)];

    % input adaptation
    theta_TR    =       commands(4);


   % wake induced velocity at tail rotor
    chi_dg = [0, 20, 70, 100];
    ki = [0.4, 1.6, 2.35, 1.35];
    chi_test = (180/pi)*chi_MR;
    k_i_TR = interp1 (chi_dg, ki, chi_test,'linear','extrap');


    if v_i_MR>=0
        w_i_TR   = k_i_TR*v_i_MR;
    else
        w_i_TR   = 0.0;
    end


    % velocity at tail rotor in bTR (parallel to bB)
    lv_TR   =   lv_B + cross(ar_B,r_TR) + [0; 0; -w_i_TR];
    lv_CTR  =   rh_i(pi/2-K_ang)*lv_TR;
    ar_CTR  =   rh_i(pi/2-K_ang)*ar_B;

        % aerodynamic angles
    V_CTR = norm(lv_CTR);
    if V_CTR<gTOL
        beta_CTR = 0.0;
    else
        beta_CTR   = atan2(lv_CTR(2),lv_CTR(1));
    end

    % angular rates in bTRW   NO NORMALIZADOS
    p_hw_CTR     =   ar_CTR(1)*cos(beta_CTR)+ar_CTR(2)*sin(beta_CTR);
    q_hw_CTR     =   ar_CTR(2)*cos(beta_CTR)-ar_CTR(1)*sin(beta_CTR);

    % advance and vertical velocity ratio
    mu_CTR       =   sqrt(lv_CTR(1)^2+lv_CTR(2)^2)/(omega_ref_TR*R_TR);
    mu_z_CTR     =   lv_CTR(3)/(omega_ref_TR*R_TR);
```

```
    % iteration for lambda_0 and C_T
    % initial values

    lambda_0_TR =  loc_pr_mi_TR;

    D_TR        =   1-
(1/4)*mu_CTR^4+K_1_TR^2*(1+(1/2)*mu_CTR^2)*(1+(3/2)*mu_CTR^2);
%not included in iteration loop
    f1_TR       =   (4/3)*mu_CTR*beta_0_TR-
16*p_hw_CTR/(gamma_TR*Omega_TR)-q_hw_CTR/Omega_TR;  %not included in
iteration loop
    f2_TR       =
(8/3)*K_1_TR*mu_CTR*beta_0_TR+16*q_hw_CTR/(gamma_TR*Omega_TR)-
mu_CTR*((8/3)*theta_TR+2*theta_t_TR+2*mu_z_CTR-2*lambda_0_TR)-
p_hw_CTR/Omega_TR;
    beta_1cw_TR =   -(K_1_TR*(1+(3/2)*mu_CTR^2)*f1_TR-
(1+(1/2)*mu_CTR^2)*f2_TR)/D_TR;
    beta_1sw_TR =   -((1-
(1/2)*mu_CTR^2)*f1_TR+K_1_TR*(1+(1/2)*mu_CTR^2)*f2_TR)/D_TR;
    ct_TR       =   a_TR*s_TR/2*((1/2)*(mu_z_CTR-
lambda_0_TR)+theta_TR*(1/3+(1/2)*mu_CTR^2)+(1/4)*theta_t_TR*(1+mu_CTR^2)-
(1/2)*mu_CTR*K_1_TR*beta_1sw_TR-
beta_0_TR*(1/3+(1/2)*mu_CTR^2)*K_1_TR+(1/4)*mu_CTR*p_hw_CTR/Omega_TR);
    Lambda      =   mu_CTR^2+(lambda_0_TR-mu_z_CTR)^2;
    h_i         =   -(2*lambda_0_TR*sqrt(Lambda)-
ct_TR)*Lambda/(2*Lambda^(3/2)+a_TR*s_TR/4*Lambda-ct_TR*(mu_z_CTR-
lambda_0_TR));
    i_iter= 1;
    while abs(h_i/lambda_0_TR)>gTOL && i_iter<50
        lambda_0_TR   =    lambda_0_TR+0.6*h_i;
        f2_TR       =
(8/3)*K_1_TR*mu_CTR*beta_0_TR+16*q_hw_CTR/(gamma_TR*Omega_TR)-
mu_CTR*((8/3)*theta_TR+2*theta_t_TR+2*mu_z_CTR-2*lambda_0_TR)-
p_hw_CTR/Omega_TR;
        beta_1cw_TR =   -(K_1_TR*(1+(3/2)*mu_CTR^2)*f1_TR-
(1+(1/2)*mu_CTR^2)*f2_TR)/D_TR;
        beta_1sw_TR =   -((1-
(1/2)*mu_CTR^2)*f1_TR+K_1_TR*(1+(1/2)*mu_CTR^2)*f2_TR)/D_TR;
        ct_TR       =   a_TR*s_TR/2*((1/2)*(mu_z_CTR-
lambda_0_TR)+theta_TR*(1/3+(1/2)*mu_CTR^2)+(1/4)*theta_t_TR*(1+mu_CTR^2)-
(1/2)*mu_CTR*K_1_TR*beta_1sw_TR-
beta_0_TR*(1/3+(1/2)*mu_CTR^2)*K_1_TR+(1/4)*mu_CTR*p_hw_CTR/Omega_TR);
        Lambda      =   mu_CTR^2+(lambda_0_TR-mu_z_CTR)^2;
        h_i         =   -(2*lambda_0_TR*sqrt(Lambda)-
ct_TR)*Lambda/(2*Lambda^(3/2)+a_TR*s_TR/4*Lambda-ct_TR*(mu_z_CTR-
lambda_0_TR));
         i_iter=i_iter+1;
    end
    if i_iter>=50
        lambda_0_TR =   lambda_0_TR+0.3*h_i;
        'peta_TR'
    end
    pr_mi_TR =   lambda_0_TR;
```

```
     % forces and torque
    delta    =   0.009+0.3*(6*ct_TR/(a_TR/s_TR))^2;
    T_CTR    =   2*rho*a_TR*c_TR*R_TR^3*Omega_TR^2*((1/2)*mu_z_CTR-
(1/2)*lambda_0_TR+theta_TR*(1/3+(1/2)*mu_CTR^2)+(1/4)*theta_t_TR*(mu_CTR^
2+1)-(1/2)*mu_CTR*K_1_TR*beta_1sw_TR-
beta_0_TR*(1/3+(1/2)*mu_CTR^2)*K_1_TR+(1/4)*mu_CTR*p_hw_CTR/Omega_TR);
% ¿es necesario comprobar coincidencia valores ?
    Q_CTR    =
2*rho*a_TR*c_TR*R_TR^4*Omega_TR^2*((1/4)*delta*(mu_CTR^2+1)/a_TR-
(1/3)*(mu_z_CTR-lambda_0_TR)*(-K_1_TR*beta_0_TR+theta_TR)-
(1/4)*theta_t_TR*(mu_z_CTR-lambda_0_TR)-K_1_TR*(-beta_1cw_TR*(-
(1/8)*beta_1sw_TR-(1/6)*mu_CTR*beta_0_TR-(1/16)*beta_1sw_TR*mu_CTR^2)-
beta_1sw_TR*((1/8)*beta_1cw_TR+(1/4)*mu_CTR*(mu_z_CTR-lambda_0_TR)-
(1/16)*beta_1cw_TR*mu_CTR^2))-(1/2)*(mu_z_CTR-
lambda_0_TR)^2+(1/2)*(mu_z_CTR-lambda_0_TR)*mu_CTR*beta_1cw_TR-
(1/2)*mu_CTR^2*((1/2)*beta_0_TR^2+(3/8)*beta_1cw_TR^2+(1/8)*beta_1sw_TR^2
)-(1/3)*mu_CTR*beta_0_TR*beta_1sw_TR-(1/8)*beta_1cw_TR^2-
(1/8)*beta_1sw_TR^2-p_hw_CTR*((1/6)*mu_CTR*(-K_1_TR*beta_0_TR+theta_TR)-
(1/8)*K_1_TR*beta_1sw_TR+(1/4)*beta_1cw_TR)/omega_ref_TR-q_hw_CTR*(-
(1/8)*K_1_TR*beta_1cw_TR-(1/4)*beta_1sw_TR-
(1/3)*mu_CTR*beta_0_TR)/omega_ref_TR-(1/8)*p_hw_CTR^2/Omega_TR^2-
(1/8)*q_hw_CTR^2/Omega_TR^2);
    H_WCTR  =
2*rho*a_TR*c_TR*R_TR^3*Omega_TR^2*((1/2)*delta*mu_CTR/a_TR-(1/4)*(-
K_1_TR*beta_0_TR+theta_TR)*(2*mu_CTR*(mu_z_CTR-
lambda_0_TR)+(4/3)*beta_1cw_TR)-(1/4)*theta_t_TR*(mu_CTR*(mu_z_CTR-
lambda_0_TR)+beta_1cw_TR)-(1/4)*K_1_TR*(-(2/3)*beta_0_TR*beta_1cw_TR-
beta_1sw_TR*(mu_CTR*beta_1cw_TR+mu_z_CTR-lambda_0_TR))-(3/4)*(mu_z_CTR-
lambda_0_TR)*beta_1cw_TR+(1/6)*beta_0_TR*beta_1sw_TR+(1/4)*mu_CTR*(beta_0
_TR^2+beta_1cw_TR^2)-p_hw_CTR/Omega_TR*(-
(1/6)*K_1_TR*beta_0_TR+(1/6)*theta_TR+(1/8)*theta_t_TR-
(3/16)*mu_CTR*K_1_TR*beta_1sw_TR+(1/2)*mu_z_CTR-(1/2)*lambda_0_TR-
(1/16)*beta_1cw_TR*mu_CTR)-q_hw_CTR/Omega_TR*(-
(1/16)*K_1_TR*beta_1cw_TR*mu_CTR+(1/6)*beta_0_TR-
(1/16)*beta_1sw_TR*mu_CTR));
    Y_WCTR  =   2*rho*a_TR*c_TR*R_TR^3*Omega_TR^2*(-(1/4)*(-
K_1_TR*beta_0_TR+theta_TR)*((4/3)*beta_1sw_TR+3*mu_CTR*beta_0_TR+2*beta_1
sw_TR*mu_CTR^2)-
(1/4)*theta_t_TR*(mu_CTR^2*beta_1sw_TR+2*mu_CTR*beta_0_TR+beta_1sw_TR)+(1
/4)*K_1_TR*(-beta_1cw_TR*(-mu_CTR*beta_1cw_TR+mu_z_CTR-lambda_0_TR)-
beta_1sw_TR*(-(2/3)*beta_0_TR-2*beta_1sw_TR*mu_CTR-
2*beta_0_TR*mu_CTR^2))-(3/4)*(mu_z_CTR-lambda_0_TR)*beta_1sw_TR-
(1/6)*beta_0_TR*beta_1cw_TR-(1/4)*mu_CTR*(6*beta_0_TR*(mu_z_CTR-
lambda_0_TR)-
beta_1cw_TR*beta_1sw_TR)+mu_CTR^2*beta_0_TR*beta_1cw_TR+p_hw_CTR*(-
(1/16)*K_1_TR*beta_1cw_TR*mu_CTR-(1/6)*beta_0_TR-
(5/16)*beta_1sw_TR*mu_CTR)/Omega_TR+q_hw_CTR*(-
(1/6)*K_1_TR*beta_0_TR+(1/6)*theta_TR+(1/8)*theta_t_TR-
(1/16)*mu_CTR*K_1_TR*beta_1sw_TR+(1/2)*mu_z_CTR-(1/2)*lambda_0_TR-
(7/16)*beta_1cw_TR*mu_CTR)/Omega_TR);

   % forces and moments at CG

    F_CTR    =   rh_k(-beta_CTR)*[-H_WCTR;Y_WCTR;-T_CTR];
    F_TR     =   rh_i(K_ang-pi/2)*F_CTR;
```

```
    F_CG     =    F_TR;
    M_CG     =    rh_i(K_ang-pi/2)*[0;0;Q_CTR] + cross(r_TR,F_TR);


   v_i_TR  =   lambda_0_TR*Omega_TR*R_TR;


    out = [F_CG; M_CG; v_i_TR;...
        H_WCTR; Y_WCTR; T_CTR; 0; 0; Q_CTR; ...
        beta_CTR; beta_1cw_TR; beta_1sw_TR ] ;

end
```

C.6: Rotor de cola (girodino)

```
function out = ral_UH60_fm_tail_rotor(commands, lv_B, ar_B, chi_MR,
v_i_MR, loc_pr_mi_TR, P, f_Om)

    global pr_mi_TR;

   % parameters adaptation
    gTOL        =   P.gtol;
    rho         =   P.rho;
    %g           =   P.g;

    %omega_ref_TR=   P.Omega_TR;
    Omega_TR    =   P.Omega_TR;
    R_TR        =   P.R_TR;

    a_TR        =   P.a_TR;
    c_TR        =   P.c_TR;
    s_TR        =   P.sol_TR;

    n_b_TR      =   P.n_b_TR;
    gamma_TR    =   P.gamma_TR;
    theta_t_TR  =   P.theta_t_TR;
    beta_0_TR   =   P.beta_00_TR;
    K_1_TR      =   P.K_1_TR;
    K_ang       =   P.K_ang;
    r_TR        =   [-(P.STA_TR-P.STA_CG); P.BL_TR-P.BL_CG  ; -(P.WL_TR-
P.WL_CG)];

    % input adaptation
    theta_TR    =       commands(4);

    % ralentizacion rotor
    omega_ref_TR =   f_Om*P.Omega_TR;


   % wake induced velocity at tail rotor

    chi_dg = [0, 20, 70, 100];
    ki = [0.4, 1.6, 2.35, 1.35];
    chi_test = (180/pi)*chi_MR;
    k_i_TR = interp1 (chi_dg, ki, chi_test,'linear','extrap');

    %k_i_TR = 1.8;

    if v_i_MR>=0
        w_i_TR   = k_i_TR*v_i_MR;
    else
        w_i_TR   = 0.0;
    end
  % fprintf('hola1');
    % velocity at tail rotor in bTR (parallel to bB)
    lv_TR   =   lv_B + cross(ar_B,r_TR) + [0; 0; -w_i_TR];
    lv_CTR  =   rh_i(pi/2-K_ang)*lv_TR;
    ar_CTR  =   rh_i(pi/2-K_ang)*ar_B;
```

```
        % aerodynamic angles
    V_CTR = norm(lv_CTR);
    if V_CTR<gTOL
        beta_CTR = 0.0;
    else
        beta_CTR   = atan2(lv_CTR(2),lv_CTR(1));
    end

    % angular rates in bTRW   NO NORMALIZADOS
    p_hw_CTR     =   ar_CTR(1)*cos(beta_CTR)+ar_CTR(2)*sin(beta_CTR);
    q_hw_CTR     =   ar_CTR(2)*cos(beta_CTR)-ar_CTR(1)*sin(beta_CTR);

    % advance and vertical velocity ratio
    mu_CTR       =   sqrt(lv_CTR(1)^2+lv_CTR(2)^2)/(omega_ref_TR*R_TR);
    mu_z_CTR     =   lv_CTR(3)/(omega_ref_TR*R_TR);

    % iteration for lambda_0 and C_T
    % initial values

    lambda_0_TR =  loc_pr_mi_TR;
   % lambda_0_TR_init =  lambda_0_TR
  % fprintf('hola3');
    D_TR        =   1-
(1/4)*mu_CTR^4+K_1_TR^2*(1+(1/2)*mu_CTR^2)*(1+(3/2)*mu_CTR^2);
%not included in iteration loop
    f1_TR       =   (4/3)*mu_CTR*beta_0_TR-
16*p_hw_CTR/(gamma_TR*Omega_TR)-q_hw_CTR/Omega_TR;  %not included in
iteration loop
    f2_TR       =
(8/3)*K_1_TR*mu_CTR*beta_0_TR+16*q_hw_CTR/(gamma_TR*Omega_TR)-
mu_CTR*((8/3)*theta_TR+2*theta_t_TR+2*mu_z_CTR-2*lambda_0_TR)-
p_hw_CTR/Omega_TR;
    beta_1cw_TR =   -(K_1_TR*(1+(3/2)*mu_CTR^2)*f1_TR-
(1+(1/2)*mu_CTR^2)*f2_TR)/D_TR;
    beta_1sw_TR =   -((1-
(1/2)*mu_CTR^2)*f1_TR+K_1_TR*(1+(1/2)*mu_CTR^2)*f2_TR)/D_TR;
    ct_TR       =   a_TR*s_TR/2*((1/2)*(mu_z_CTR-
lambda_0_TR)+theta_TR*(1/3+(1/2)*mu_CTR^2)+(1/4)*theta_t_TR*(1+mu_CTR^2)-
(1/2)*mu_CTR*K_1_TR*beta_1sw_TR-
beta_0_TR*(1/3+(1/2)*mu_CTR^2)*K_1_TR+(1/4)*mu_CTR*p_hw_CTR/Omega_TR);
    Lambda      =   mu_CTR^2+(lambda_0_TR-mu_z_CTR)^2;
    h_i         =   -(2*lambda_0_TR*sqrt(Lambda)-
ct_TR)*Lambda/(2*Lambda^(3/2)+a_TR*s_TR/4*Lambda-ct_TR*(mu_z_CTR-
lambda_0_TR));
    i_iter= 1;
    while abs(h_i/lambda_0_TR)>gTOL && i_iter<50
        lambda_0_TR    =    lambda_0_TR+0.6*h_i;
        f2_TR       =
(8/3)*K_1_TR*mu_CTR*beta_0_TR+16*q_hw_CTR/(gamma_TR*Omega_TR)-
mu_CTR*((8/3)*theta_TR+2*theta_t_TR+2*mu_z_CTR-2*lambda_0_TR)-
p_hw_CTR/Omega_TR;
```

```
        beta_1cw_TR =   -(K_1_TR*(1+(3/2)*mu_CTR^2)*f1_TR-
(1+(1/2)*mu_CTR^2)*f2_TR)/D_TR;
        beta_1sw_TR =   -((1-
(1/2)*mu_CTR^2)*f1_TR+K_1_TR*(1+(1/2)*mu_CTR^2)*f2_TR)/D_TR;
        ct_TR       =   a_TR*s_TR/2*((1/2)*(mu_z_CTR-
lambda_0_TR)+theta_TR*(1/3+(1/2)*mu_CTR^2)+(1/4)*theta_t_TR*(1+mu_CTR^2)-
(1/2)*mu_CTR*K_1_TR*beta_1sw_TR-
beta_0_TR*(1/3+(1/2)*mu_CTR^2)*K_1_TR+(1/4)*mu_CTR*p_hw_CTR/Omega_TR);
        Lambda      =   mu_CTR^2+(lambda_0_TR-mu_z_CTR)^2;
        h_i         =   -(2*lambda_0_TR*sqrt(Lambda)-
ct_TR)*Lambda/(2*Lambda^(3/2)+a_TR*s_TR/4*Lambda-ct_TR*(mu_z_CTR-
lambda_0_TR));
         i_iter=i_iter+1;
    end
    if i_iter>=50
        lambda_0_TR =   lambda_0_TR+0.3*h_i;
        'peta_TR'
    end
    pr_mi_TR =   lambda_0_TR;



    % forces and torque
    delta   =   0.009+0.3*(6*ct_TR/(a_TR/s_TR))^2;
    T_CTR   =   2*rho*a_TR*c_TR*R_TR^3*Omega_TR^2*((1/2)*mu_z_CTR-
(1/2)*lambda_0_TR+theta_TR*(1/3+(1/2)*mu_CTR^2)+(1/4)*theta_t_TR*(mu_CTR^
2+1)-(1/2)*mu_CTR*K_1_TR*beta_1sw_TR-
beta_0_TR*(1/3+(1/2)*mu_CTR^2)*K_1_TR+(1/4)*mu_CTR*p_hw_CTR/Omega_TR);
% ¿es necesario comprobar coincidencia valores ?
    Q_CTR   =
2*rho*a_TR*c_TR*R_TR^4*Omega_TR^2*((1/4)*delta*(mu_CTR^2+1)/a_TR-
(1/3)*(mu_z_CTR-lambda_0_TR)*(-K_1_TR*beta_0_TR+theta_TR)-
(1/4)*theta_t_TR*(mu_z_CTR-lambda_0_TR)-K_1_TR*(-beta_1cw_TR*(-
(1/8)*beta_1sw_TR-(1/6)*mu_CTR*beta_0_TR-(1/16)*beta_1sw_TR*mu_CTR^2)-
beta_1sw_TR*((1/8)*beta_1cw_TR+(1/4)*mu_CTR*(mu_z_CTR-lambda_0_TR)-
(1/16)*beta_1cw_TR*mu_CTR^2))-(1/2)*(mu_z_CTR-
lambda_0_TR)^2+(1/2)*(mu_z_CTR-lambda_0_TR)*mu_CTR*beta_1cw_TR-
(1/2)*mu_CTR^2*((1/2)*beta_0_TR^2+(3/8)*beta_1cw_TR^2+(1/8)*beta_1sw_TR^2
)-(1/3)*mu_CTR*beta_0_TR*beta_1sw_TR-(1/8)*beta_1cw_TR^2-
(1/8)*beta_1sw_TR^2-p_hw_CTR*((1/6)*mu_CTR*(-K_1_TR*beta_0_TR+theta_TR)-
(1/8)*K_1_TR*beta_1sw_TR+(1/4)*beta_1cw_TR)/omega_ref_TR-q_hw_CTR*(-
(1/8)*K_1_TR*beta_1cw_TR-(1/4)*beta_1sw_TR-
(1/3)*mu_CTR*beta_0_TR)/omega_ref_TR-(1/8)*p_hw_CTR^2/Omega_TR^2-
(1/8)*q_hw_CTR^2/Omega_TR^2);
    H_WCTR  =
2*rho*a_TR*c_TR*R_TR^3*Omega_TR^2*((1/2)*delta*mu_CTR/a_TR-(1/4)*(-
K_1_TR*beta_0_TR+theta_TR)*(2*mu_CTR*(mu_z_CTR-
lambda_0_TR)+(4/3)*beta_1cw_TR)-(1/4)*theta_t_TR*(mu_CTR*(mu_z_CTR-
lambda_0_TR)+beta_1cw_TR)-(1/4)*K_1_TR*(-(2/3)*beta_0_TR*beta_1cw_TR-
beta_1sw_TR*(mu_CTR*beta_1cw_TR+mu_z_CTR-lambda_0_TR))-(3/4)*(mu_z_CTR-
lambda_0_TR)*beta_1cw_TR+(1/6)*beta_0_TR*beta_1sw_TR+(1/4)*mu_CTR*(beta_0
_TR^2+beta_1cw_TR^2)-p_hw_CTR/Omega_TR*(-
(1/6)*K_1_TR*beta_0_TR+(1/6)*theta_TR+(1/8)*theta_t_TR-
(3/16)*mu_CTR*K_1_TR*beta_1sw_TR+(1/2)*mu_z_CTR-(1/2)*lambda_0_TR-
(1/16)*beta_1cw_TR*mu_CTR)-q_hw_CTR/Omega_TR*(-
(1/16)*K_1_TR*beta_1cw_TR*mu_CTR+(1/6)*beta_0_TR-
(1/16)*beta_1sw_TR*mu_CTR));
```

```
    Y_WCTR   =   2*rho*a_TR*c_TR*R_TR^3*Omega_TR^2*(-(1/4)*(-
K_1_TR*beta_0_TR+theta_TR)*((4/3)*beta_1sw_TR+3*mu_CTR*beta_0_TR+2*beta_1
sw_TR*mu_CTR^2)-
(1/4)*theta_t_TR*(mu_CTR^2*beta_1sw_TR+2*mu_CTR*beta_0_TR+beta_1sw_TR)+(1
/4)*K_1_TR*(-beta_1cw_TR*(-mu_CTR*beta_1cw_TR+mu_z_CTR-lambda_0_TR)-
beta_1sw_TR*(-(2/3)*beta_0_TR-2*beta_1sw_TR*mu_CTR-
2*beta_0_TR*mu_CTR^2))-(3/4)*(mu_z_CTR-lambda_0_TR)*beta_1sw_TR-
(1/6)*beta_0_TR*beta_1cw_TR-(1/4)*mu_CTR*(6*beta_0_TR*(mu_z_CTR-
lambda_0_TR)-
beta_1cw_TR*beta_1sw_TR)+mu_CTR^2*beta_0_TR*beta_1cw_TR+p_hw_CTR*(-
(1/16)*K_1_TR*beta_1cw_TR*mu_CTR-(1/6)*beta_0_TR-
(5/16)*beta_1sw_TR*mu_CTR)/Omega_TR+q_hw_CTR*(-
(1/6)*K_1_TR*beta_0_TR+(1/6)*theta_TR+(1/8)*theta_t_TR-
(1/16)*mu_CTR*K_1_TR*beta_1sw_TR+(1/2)*mu_z_CTR-(1/2)*lambda_0_TR-
(7/16)*beta_1cw_TR*mu_CTR)/Omega_TR);

   % forces and moments at CG

    F_CTR    =   rh_k(-beta_CTR)*[-H_WCTR;Y_WCTR;-T_CTR];
    F_TR     =   rh_i(K_ang-pi/2)*F_CTR;

    F_CG     =   F_TR;
    M_CG     =   rh_i(K_ang-pi/2)*[0;0;Q_CTR] + cross(r_TR,F_TR);


   v_i_TR   =   lambda_0_TR*Omega_TR*R_TR;
   % fprintf('hola6');
    %out = [F_CG; M_CG; v_i_TR];
    out = [F_CG; M_CG; v_i_TR;...
        H_WCTR; Y_WCTR; T_CTR; 0; 0; Q_CTR; ...
        beta_CTR; beta_1cw_TR; beta_1sw_TR ] ;

      % % fprintf('hola7');
end
```

C.7: Fuselaje

```
function out = UH60_fm_fuselage (lv_B, chi_MR, v_i_MR, P)

    % parameters adaptation

    gTOL     =   P.gtol;
    rho      =   P.rho;

    r_FU         =   [-(P.STA_ACF-P.STA_CG); P.BL_ACF-P.BL_CG; -(P.WL_ACF-
P.WL_CG)];
```

```
    % wake induced velocity over fuselage
    x1  = 70;
    y1  = 1.12;
    x2  = 100;
    y2  = 0.60;

    b=(y2-y1)/(x2-1/2*x1-x2^2/(2*x1));
    a=-b/(2*x1);
    c=y1-1/2*b*x1;

    chi_test = (180/pi)*chi_MR;
    k_i_FU =y1*(chi_test<=x1)+(a*chi_test^2+b*chi_test+c)*(chi_test>x1);

    if v_i_MR>=0  % if lambda_0>=0
        w_i_FU = k_i_FU*v_i_MR;
    else
        w_i_FU = 0.0;
    end
    lv_FU = lv_B + [0; 0; -w_i_FU];


    % velocity and angles at fuselage (ar_B effects, ignored)

    V_FU            =   norm(lv_FU);
    V_xz            =   sqrt((lv_FU(1))^2+(lv_FU(3))^2);
    alpha_FU        =   atan2(lv_FU(3),abs(lv_FU(1))) ;
    %alpha_FU       =   atan2(lv_FU(3),lv_FU(1)) ;
    alpha_FUd       =   180/pi*alpha_FU;
    if V_xz<gTOL
        beta_FU   =     0.0;
    else
        beta_FU     =     atan2(lv_FU(2),V_xz);
    end
    beta_FUd        =     180/pi*beta_FU;
        psi_FU          =     -beta_FU;
    psi_FUd         =     -beta_FUd;


   % forces at CP in local wind axes

    D_q     =   90.0555*(sin(alpha_FU))^2-
41.5604*cos(alpha_FU)+2.94684*cos(4*psi_FU)-103.141*cos(2*psi_FU)-
0.535350e-6*psi_FUd^4+160.2049;
    L_q     =   29.3616*sin(alpha_FU)+43.4680*sin(2*alpha_FU)-
81.8924*(sin(alpha_FU))^2-84.1469*cos(alpha_FU)-0.821406e-
1*psi_FUd+3.00102*sin(4*psi_FU)+0.0323477*psi_FUd^2+85.3496;
    Y_q     =   35.3999*sin(psi_FU)+71.8019*sin(2*psi_FU)-
8.04823*sin(4*psi_FU)-0.980257e-12;
    M_q     =
2.37925*alpha_FUd+728.026*sin(2*alpha_FU)+426.760*(sin(alpha_FU))^2+348.0
72*cos(alpha_FU)-510.581*(cos(psi_FU))^3+56.111;
    if (0<=abs(psi_FUd) && abs(psi_FUd)<10) %pi/18
        ell_q   = 0.0;
    elseif (10<=abs(psi_FUd) && abs(psi_FUd)<25)   %5*pi/36
        ell_q = psi_FUd/abs(psi_FUd)*(455.707*(cos(psi_FU))^4-428.639);
    elseif (25<=abs(psi_FUd) && abs(psi_FUd)<=90) %pi/2
```

```
        ell_q = 614.797*sin(psi_FU)+psi_FUd/abs(psi_FUd)*(-
47.7213*cos(4*psi_FU)-290.504*(cos(psi_FU))^3+735.507*(cos(psi_FU))^4-
669.266);
    else
        fprintf('otra vez a jugar con los rangos. Para ell_FU');
    end
    if (0<=abs(psi_FUd) && abs(psi_FUd)<20) %pi/9
        N_q = -278.133*sin(2*psi_FU)+422.644*sin(4*psi_FU)-1.83172;
    elseif (20<=abs(psi_FUd) && abs(psi_FUd)<=90) %90°
        N_q = 220.0*sin(2*psi_FU)+sign(psi_FUd)*(671.0*(cos(psi_FU))^4-
429.0);
    else
        fprintf('otra vez a jugar con los rangos. Para N_FU');
    end

    P_dyn = 1/2*rho*V_FU^2;
    l_esc1      =   0.3048;     % m/ft

    D_FU        =   P_dyn*D_q*(l_esc1)^2;
    L_FU        =   P_dyn*L_q*(l_esc1)^2;
    Y_FU        =   P_dyn*Y_q*(l_esc1)^2;
    M_FU        =   P_dyn*M_q*(l_esc1)^3;
    ell_FU      =   P_dyn*ell_q*(l_esc1)^3;
    N_FU        =   P_dyn*N_q*(l_esc1)^3;


    % forces and moments at CG

    F_CG = rh_j(alpha_FU)*rh_k(-beta_FU)*[-D_FU; Y_FU; -L_FU];
    M_CG = rh_j(alpha_FU)*rh_k(-beta_FU)*[ell_FU; M_FU; N_FU] +
cross(r_FU,F_CG);

    %out = [F_CG; M_CG];
    out = [F_CG; M_CG; ...
        alpha_FU; beta_FU;...
         D_FU; Y_FU; L_FU; ell_FU; M_FU; N_FU];
        %           D_q; Y_q; L_q; ell_q; M_q; N_q];   % para funciones
de comprobacion
end
```

C.8: Estabilizador vertical

```
function out = cN_fm_ver_tail (lv_B, ar_B, chi_MR, v_i_MR, v_i_TR, P)

    % parameters adaptation

    gTOL         =    P.gtol;
    rho          =    P.rho;

    r_VT         =    [-(P.STA_VT-P.STA_CG); P.BL_VT-P.BL_CG; -(P.WL_VT-
P.WL_CG)];

    S_VT         =    P.S_VT;
    i_VT         =    P.alpha_i_VT;
    AR           =    P.AR_VT;
    CLM          =    P.CLM_VT;
    Lambda_VT    =    P.LAMBDA_VT;
    nu           =    P.nu_VT;
    kv_VT        =    P.kv_VT;

    % MR wake induced velocity at vertical tail
    chi_dg = [0, 20, 70, 100];
    ki = [0.4, 1.6, 2.35, 1.35];
    chi_test = (180/pi)*chi_MR;
    kv_VT_MR = interp1 (chi_dg, ki, chi_test,'linear','extrap');


    w_i_VT       =    kv_VT_MR*v_i_MR;

    % TR wake induced velocity at vertical tail
    v_i_VT       =    kv_VT*v_i_TR;

    % velocity at vertical tail
    lv_VT = lv_B + cross(ar_B,r_VT) + [0; v_i_VT; -w_i_VT];

    % aerodynamic angles
    V_VT = norm(lv_VT);
    alpha_VT   = atan2(lv_VT(2),lv_VT(1)) + i_VT;
    if V_VT<gTOL
        beta_VT = 0.0;
    else
        beta_VT    = asin(lv_VT(3)/V_VT);
    end

    % corrected lift curve slope (en vez de la formula de cN se usa la de
Helmbold (pHFD, lecture3, p19)
    a_VT         =
(cos(beta_VT+Lambda_VT))^2*(pi*AR/(1+sqrt(1+AR^2/4)));

    % conditions  at  the  stall
    alpha_s      =    CLM/a_VT;
    if (alpha_s>pi/4)
        alpha_s =    pi/4;
        CLM    =    a_VT*pi/4;
    end
```

```
    alpha_1     =   1.2*alpha_s;

    % alpha shift (0,2*pi)
    alpha_VT    =  mod(alpha_VT,2*pi);

    % angle of attack for expressions
    if (0<=alpha_VT && alpha_VT<pi/2)
       alpha_i =   alpha_VT;
    elseif (pi/2<=alpha_VT && alpha_VT<pi)
        alpha_i =   pi-alpha_VT;
    elseif (pi<=alpha_VT && alpha_VT<3*pi/2)
        alpha_i =   alpha_VT-pi;
    elseif (3*pi/2<=alpha_VT && alpha_VT<=2*pi)
        alpha_i =   2*pi-alpha_VT;
    else
        fprintf('que pasa')
    end

    % lift coefficient-step1
    if (0<=alpha_i && alpha_i<alpha_s)
        C_L_0   =   a_VT*alpha_i;
    elseif (alpha_s<=alpha_i && alpha_i<alpha_1)
        C_L_0   =   CLM - a_VT*(alpha_i-alpha_s);
    elseif (alpha_1<=alpha_i && alpha_i<=pi/2)
        C_L_0   =   0.8*CLM*(1-((alpha_i-alpha_1)/(pi/2-alpha_1))^2) ; %
NO SÉ SI FALTA ALGO: FOTOCOPIA BORROSA
    else
        fprintf('sin valor C_L_0')
    end

    % drag coefficient-step1
    if (0<=alpha_i && alpha_i<0.35)
        C_D_P   =   0.009 + 0.11*alpha_i^2;
    elseif (0.35<=alpha_i && alpha_i<=pi/2)
        C_D_P   =   -0.1254 + 0.09415*alpha_i +
0.977525*(sin(alpha_i)^2);
    end

    % lift coefficient-step2
    if (0<=alpha_VT && alpha_VT<pi/2)
        C_L     =   C_L_0;
    elseif (pi/2<=alpha_VT && alpha_VT<=pi)
        C_L     =   -0.8*C_L_0;
    elseif (pi<=alpha_VT && alpha_VT<3*pi/2)
        C_L     =   0.8*C_L_0;
    elseif (3*pi/2<=alpha_VT && alpha_VT<=2*pi)
        C_L     =   -C_L_0;
    else
         fprintf('sin valor C_L');
    end

    % drag coefficient-step2
    C_D     =   C_D_P + C_L^2/(0.8*pi*AR);
```

```
    % forces at VT in local wind axes

    L_VT = (1/2)*rho*V_VT^2*S_VT*C_L*nu;
    D_VT = (1/2)*rho*V_VT^2*S_VT*C_D*nu;

    % forces and moments at CG

    F_CG = rh_k(-(alpha_VT-i_VT))*rh_j(beta_VT)*[-D_VT; -L_VT; 0];
    M_CG = cross(r_VT,F_CG);

    %out = [F_CG; M_CG];
     out = [F_CG; M_CG; ...
        alpha_VT; beta_VT;...
        D_VT; L_VT; 0; 0; 0; 0];

end
```

C.9: Estabilizador horizontal

```
function out = UH60_fm_hor_tail(lv_B, ar_B, chi_MR, v_i_MR, i_HT, P)

    % parameters adaptation
    gTOL     =   P.gtol;
    rho      =   P.rho;

    r_HT     =   [-(P.STA_HT-P.STA_CG); P.BL_HT-P.BL_CG; -(P.WL_HT-
P.WL_CG)];
    %BL discrepa con VAC,p27; cN,p29 desprecia p;
    S_HT     =   P.S_HT;
    %i_HT     =   P.alpha_i_HT;
    AR       =   P.AR_HT;
    CLM      =   P.CLM_HT;
    nu       =   P.nu_HT;
    kv_HTa    =   P.kv_HT;

    % MR wake induced velocity at horizontal tail (VAC,p31; cN, p33)
    kv_HTb      =   1.299+0.671*chi_MR-1.172*chi_MR^2+0.351*chi_MR^3;

    chi_dg = [0, 20, 70, 100];
    ki = [0.4, 1.6, 2.35, 1.35];
    chi_test = (180/pi)*chi_MR;
    kv_HTc = interp1 (chi_dg, ki, chi_test,'linear','extrap');

    w_i_HT   =   kv_HTc*v_i_MR;

    % velocity at horizontal tail
    lv_HT = lv_B + cross(ar_B,r_HT) + [0; 0; -w_i_HT];

    % aerodynamic angles
    V_HT = norm(lv_HT);
```

```
    alpha_HT  = atan2(lv_HT(3),lv_HT(1)) + i_HT;
    if abs(V_HT)<gTOL
        beta_HT = 0.0;
    else
        beta_HT   = asin(lv_HT(2)/V_HT);
    end

    % corrected lift curve slope
    % (en vez de la formula de cN,p29 se usa la de Helmbold (pFD,
lecture3, p19)
    a_HT        =    (cos(beta_HT))^2*(pi*AR/(1+sqrt(1+AR^2/4)));

    % conditions  at  the  stall
    alpha_s     =    CLM/a_HT;
    if (alpha_s>pi/4)
        alpha_s =    pi/4;
        CLM     =    a_HT*pi/4;
    end
    alpha_1     =    1.2*alpha_s;

    % alpha shift (0,2*pi)
    alpha_HT    =   mod(alpha_HT,2*pi);

    % angle of attack for expressions
    if (0<=alpha_HT && alpha_HT<pi/2)
       alpha_i =    alpha_HT;
    elseif (pi/2<=alpha_HT && alpha_HT<pi)
        alpha_i =    pi-alpha_HT;
    elseif (pi<=alpha_HT && alpha_HT<3*pi/2)
        alpha_i =    alpha_HT-pi;
    elseif (3*pi/2<=alpha_HT && alpha_HT<=2*pi)
        alpha_i =    2*pi-alpha_HT;
    else
        fprintf('que pasa')
    end

    % lift coefficient-step1
    if (0<=alpha_i && alpha_i<alpha_s)
        C_L_0    =    a_HT*alpha_i;
    elseif (alpha_s<=alpha_i && alpha_i<alpha_1)
        C_L_0    =    CLM - a_HT*(alpha_i-alpha_s);
    elseif (alpha_1<=alpha_i && alpha_i<=pi/2)
        C_L_0    =    0.8*CLM*(1-((alpha_i-alpha_1)/(pi/2-alpha_1))^2) ; %
NO SÉ SI FALTA ALGO: FOTOCOPIA BORROSA
    else
        fprintf('sin valor C_L_0')
    end

    % drag coefficient-step1
    if (0<=alpha_i && alpha_i<0.35)
        C_D_P    =    0.009 + 0.11*alpha_i^2;
    elseif (0.35<=alpha_i && alpha_i<=pi/2)
```

```
        C_D_P    =    -0.1254 + 0.09415*alpha_i +
0.977525*(sin(alpha_i)^2);
    end

    % lift coefficient-step2
    if (0<=alpha_HT && alpha_HT<pi/2)
        C_L      =    C_L_0;
    elseif (pi/2<=alpha_HT && alpha_HT<=pi)
        C_L      =    -0.8*C_L_0;
    elseif (pi<=alpha_HT && alpha_HT<3*pi/2)
        C_L      =    0.8*C_L_0;
    elseif (3*pi/2<=alpha_HT && alpha_HT<=2*pi)
        C_L      =    -C_L_0;
    else
         fprintf('sin valor C_L');
    end

    % drag coefficient-step2
    C_D      =    C_D_P + C_L^2/(0.8*pi*AR);

    % forces at HT in local wind axes
    L_HT = (1/2)*rho*V_HT^2*S_HT*C_L*nu;
    D_HT = (1/2)*rho*V_HT^2*S_HT*C_D*nu;

    % forces and moments at CG
    % cN erróneo, como se pd comprobar en Howlett
    F_CG = rh_j(alpha_HT-i_HT)*rh_k(-beta_HT)*[-D_HT; 0; -L_HT];
    M_CG = cross(r_HT,F_CG);

    %out = [F_CG; M_CG];
    out = [F_CG; M_CG; ...
        alpha_HT; beta_HT;...
        D_HT; 0; L_HT; 0; 0; 0];

end
```

C.10: Propulsor

```
function out = ral_modUH60_pr (lv_B, ar_B, chi_MR, v_i_MR, Va, beta_prop,
P, f_Om)


% parameters adaptation
    gTOL     =    P.gtol;
    %rho      =    P.rho;

    % velocity at propeller
    LT_PR2    =    P.LT_PR2;
    LT_PR3    =    P.LT_PR3;
    r_PR     =    [-(P.STA_PR-P.STA_CG); P.BL_PR-P.BL_CG; -(P.WL_PR-
P.WL_CG)];


    chi_dg = [0, 20, 70, 100];
```

```
 ki = [0.4, 1.6, 2.35, 1.35];
 chi_test = (180/pi)*chi_MR;
 k_i_PR = interp1 (chi_dg, ki, chi_test,'linear','extrap');
 if v_i_MR>=0
     w_i_PR    = k_i_PR*v_i_MR;
 else
     w_i_PR    = 0.0;
 end

 lv_PR = lv_B + cross(ar_B,r_PR) + [0; 0; -w_i_PR];
 Vpr   = lv_PR(1);
 % aerodynamic angles
 V_PR = norm(lv_PR);
 alpha_PR   = atan2(lv_PR(3),lv_PR(1));
 if abs(V_PR)<gTOL
     beta_PR = 0.0;
 else
     beta_PR    = asin(lv_PR(2)/V_PR);
 end



if Va<P.Va_start
     F_CG = [0;0;0];
     M_CG = [0;0;0];
     t = 0.0;
     q = 0.0;
elseif (Va>=P.Va_start && f_Om==1)
     % tabla 2D table looking up
     T = LT_PR2.T;
     Q = LT_PR2.Q;
     VA = 0.5144*LT_PR2.Va_kn;
     mBETA = (pi/180)*LT_PR2.mTHETA_dg_02;
     t = interp2(VA, mBETA, T, Vpr, beta_prop, 'cubic');
     q = interp2(VA, mBETA, Q, Vpr, beta_prop, 'cubic');
else
  % tabla 3D
     T = LT_PR3.T;
     Q = LT_PR3.Q;
     VA = 0.5144*LT_PR3.Va_kn;
     mBETA = (pi/180)*LT_PR3.mTHETA_dg_02;
     F_OM  = LT_PR3.F_OM;
     t = interpn(VA, mBETA, F_OM, T, Vpr, beta_prop, f_Om, 'linear');
     q = interpn(VA, mBETA, F_OM, Q, Vpr, beta_prop, f_Om, 'linear');
end

% forces and moments at CG
 F_CG = [t; 0; 0];
 M_CG = [q; 0; 0] + cross(r_PR,F_CG);


out = [F_CG; M_CG; ...
```

```
            alpha_PR; beta_PR;...
            t; 0; 0; q; 0; 0];


end
```

C.11: Ala derecha

```
function out = modUH60_rw (lv_B, ar_B, chi_MR, v_i_MR, P)

    % parameters adaptation
    gTOL     =   P.gtol;
    rho      =   P.rho;

    r_RW     =   [-(P.STA_RW-P.STA_CG); P.BL_RW-P.BL_CG; -(P.WL_RW-
P.WL_CG)];
    S_RW     =   P.S_RW;
    MAC_RW   =   P.MAC_RW;
    i_RW     =   P.alpha_i_RW;
    LT_RW    =   P.LT_wn;


    % wake induced velocity over fuselage

    x1   = 70;
    y1   = 1.12;
    x2   = 100;
    y2   = 0.60;

    b=(y2-y1)/(x2-1/2*x1-x2^2/(2*x1));
    a=-b/(2*x1);
    c=y1-1/2*b*x1;

    chi_test = (180/pi)*chi_MR;
    k_i_RW =y1*(chi_test<=x1)+(a*chi_test^2+b*chi_test+c)*(chi_test>x1);



    if v_i_MR>=0  % if lambda_0>=0
        w_i_RW = k_i_RW*v_i_MR;
    else
        w_i_RW = 0.0;
    end
    lv_RW = lv_B + [0; 0; -w_i_RW];

    % aerodynamic angles
    V_RW = norm(lv_RW);
    alpha_RW   = atan2(lv_RW(3),lv_RW(1)) + i_RW;
    alpha_RWd    = alpha_RW*(180/pi);
    beta_RW = 0;

    % forces at RW in local wind axes
    coefs = interp1(LT_RW(:,1), LT_RW(:,2:4), alpha_RWd, 'PCHIP');
    L_RW = 1.0*(1/2)*rho*V_RW^2*S_RW*coefs(1);
```

```
    D_RW = 1.0*(1/2)*rho*V_RW^2*S_RW*coefs(2);
    m_RW = 1.0*(1/2)*rho*V_RW^2*S_RW*MAC_RW*coefs(3);

    %     % forces and moments at CG
    F_CG = rh_j(alpha_RW-i_RW)*rh_k(-beta_RW)*[-D_RW; 0; -L_RW];
    M_CG = rh_j(alpha_RW-i_RW)*rh_k(-beta_RW)*[0; m_RW;
0]+cross(r_RW,F_CG);

    out = [F_CG; M_CG; ...
        alpha_RW; beta_RW;...
        D_RW; 0; L_RW; 0; m_RW; 0];

end
```

C.12: Ala izquierda

```
function out = modUH60_lw(lv_B, ar_B, chi_MR, v_i_MR, P)

    % parameters adaptation
    gTOL     =   P.gtol;
    rho      =   P.rho;

    r_LW    =   [-(P.STA_LW-P.STA_CG); P.BL_LW-P.BL_CG; -(P.WL_LW-
P.WL_CG)];
    S_LW     =   P.S_LW;
    MAC_LW   =   P.MAC_LW;
    i_LW     =   P.alpha_i_LW;
    LT_LW    = P.LT_wn;


    % wake induced velocity over fuselage

    x1   = 70;
    y1   = 1.12;
    x2   = 100;
    y2   = 0.60;

    b=(y2-y1)/(x2-1/2*x1-x2^2/(2*x1));
    a=-b/(2*x1);
    c=y1-1/2*b*x1;

    chi_test = (180/pi)*chi_MR;
    k_i_LW =y1*(chi_test<=x1)+(a*chi_test^2+b*chi_test+c)*(chi_test>x1);
```

ETSEIB

```
    if v_i_MR>=0  % if lambda_0>=0
        w_i_LW = k_i_LW*v_i_MR;
    else
        w_i_LW = 0.0;
    end
    lv_LW = lv_B + [0; 0; -w_i_LW];

    % aerodynamic angles
    V_LW = norm(lv_LW);
    alpha_LW  = atan2(lv_LW(3),lv_LW(1)) + i_LW;
    alpha_LWd   = alpha_LW*(180/pi);

    beta_LW = 0;

    % forces at RW in local wind axes
    coefs = interp1(LT_LW(:,1), LT_LW(:,2:4), alpha_LWd, 'pchip');
    L_LW = 1.0*(1/2)*rho*V_LW^2*S_LW*coefs(1);
    D_LW = 1.0*(1/2)*rho*V_LW^2*S_LW*coefs(2);
    m_LW = 1.0*(1/2)*rho*V_LW^2*S_LW*MAC_LW*coefs(3);

    %     % forces and moments at CG
    F_CG = rh_j(alpha_LW-i_LW)*rh_k(-beta_LW)*[-D_LW; 0; -L_LW];
    M_CG = rh_j(alpha_LW-i_LW)*rh_k(-beta_LW)*[0; m_LW;
0]+cross(r_LW,F_CG);


   out = [F_CG; M_CG; ...
        alpha_LW; beta_LW;...
        D_LW; 0; L_LW; 0; m_LW; 0];

end
```

Anexo D: Algoritmos de optimización

D.1: General helicóptero (v1)

```
function calculo_general_v1_beta_phi(varargin)

    %clc
    %clear

    % se guarda path original
    path_or     = path;

    % establecimiento directorios
    fp_rt       = directorio_raiz();
    fp_pr       = [fp_rt, '\00_previos'];
    fp_mdc      = [fp_rt, '\01_modelos_dinamicos\00_comunes'];
    fp_mdh      = [fp_rt, '\01_modelos_dinamicos\01_helicoptero'];
    fp_ac       = [fp_rt, '\02_algoritmos_calculo\01_helicoptero'];
    fp_vs       = [fp_rt, '\03_visualizacion_resultados'];
    fp_rs       = [fp_rt,
'\03_visualizacion_resultados\01_datos_helicoptero\01_ultimo_calculo'];

    addpath (genpath(fp_pr), fp_mdc, fp_mdh, fp_ac, fp_vs, '-begin');



    %carga parametros y constantes conversion
    run constantes_y_conversiones.m
    run UH60_params.m



    % vuelo compensado horizontal; rango de velocidades de estudio
    if nargin<2                         % para total calculo v1
        Va_rg_kh    = 0:20:260;
        Va_rg_kh(1) = 5;
        n_v         = size(Va_rg_kh,2);
        rg_kh       = 1:n_v;
        Va_kh_mn    = zeros(1,n_v);
        com_ang_mn  = zeros(6,n_v);
        geom_ang_mn = zeros(4,n_v);
    elseif nargin==3                    % para pasos iniciales v2
        Va_rg_kh    = 0:10:260;
        Va_rg_kh(1) = 5;
        n_v         = size(Va_rg_kh,2);
        rg_kh       = varargin{2}:varargin{3};
        Va_kh_mn    = zeros(1,n_v);
        com_ang_mn  = zeros(6,n_v);
        geom_ang_mn = zeros(4,n_v);
```

```
    end
    gamma       = 0.0;
    R           = 1e308;        %Inf;
    crear_carpeta_resultados (fp_rs, Va_rg_kh);


    % velocidad de cambio de modelo (54 kn)
    if nargin<1                                         % cambio phi-beta
        Va_kh_sw    = 54.0*v_esc2;      %km/h
    else                                                 % solo se anula
beta
        Va_kh_sw    = varargin{1};      %km/h
    end


    % deflacion del estabilizador horizontal (valores Howlett)
    Va_kn_ho    = [0.5, 20, 40, 60, 100, 140];          %   nudos
    Va_ho       = P.v_esc1*Va_kn_ho;                      %  m/s
    i_HT_rg     = [0.680550000000000,    0.680550000000000,
0.602665492046573,    0.335510072408958,    0.029521554212139,  -
0.005681703794443]; %rad


    for i_v = rg_kh

        Va_kh   = Va_rg_kh(i_v);
        Va      = Va_kh/3.6;                %(m/s)
        i_HT    = interp1 (Va_ho, i_HT_rg, Va, 'pchip'); %rad

        % fprintf('se empieza con Va_kh %f \n \n', Va_kh);

        if Va_kh<Va_kh_sw
            beta     = 0.0;
        else
            phi     = 0.0;
        end

        % switch parametros trim
        trim_pars = 1;

        % valores iniciales
        vars0            = [Va; gamma; R;  0; 0; 0;  0.0; 0.0; 0; 0;
i_HT; 0];

        % ETAPA 01
        fm_act          = [1; 1; 0; 1; 0; 0];  % Fg, MR, FU
        cf_wght         = [0;   1; 0; 1;  0; 0; 0;  0; 0; 0];  %Fx, Fz
        opt_sw          = [0; 0; 0;     0; 0; 0;    1; 1; 0; 0; 0; 0];  %
theta_0, theta_1s
        trim_pars       = [''];

        out = trim_01 (P, pr_mi_MR ,pr_mi_TR, vars0, opt_sw, fm_act,
cf_wght, trim_pars);
        vars_lon = out(1:12);
        fprintf('FINAL ETAPA 1: long_00\n')
```

```
        % ETAPA 02
        fm_act          = [1; 1; 0; 1; 1; 0];  % Fg, MR, FU + HT
        cf_wght         = [0;   1; 0; 1;  0; 0; 0;  0; 1; 0];  %Fx, Fz,
My
        opt_sw          = [0; 0; 0;     1; 0; 0;    1; 1; 0; 0; 0; 0];  %
theta_0, theta_1s, alpha
        trim_pars       = [''];

        out = trim_01 (P, pr_mi_MR ,pr_mi_TR, vars_lon, opt_sw, fm_act,
cf_wght, trim_pars);
        vars_lon = out(1:12);
        fprintf('FINAL ETAPA 2: long_01\n')

        % ETAPA 03
        fm_act          = [0; 1; 1; 0; 0; 0];  %  MR, TR
        cf_wght         = [0;   0; 0; 0;  0; 0; 0;  0; 0; 1];  % Mz
        opt_sw          = [0; 0; 0;     0; 0; 0;    0; 0; 0; 1; 0; 0];  %
theta_TR
        trim_pars       = [''];

        out = trim_01 (P, pr_mi_MR ,pr_mi_TR, vars0, opt_sw, fm_act,
cf_wght, trim_pars);
        vars_lat = out(1:12);
        fprintf('FINAL ETAPA 3: lat_00\n')


        % ETAPA 04
        fm_act          = [1; 1; 1; 1; 0; 1];  %  Fg, MR, TR, FU, VT
        cf_wght         = [0;   0; 1; 0;  0; 0; 0;  1; 0; 1];  % Fy, Mx,
Mz
        trim_pars       = [''];
        if Va_kh<Va_kh_sw
            opt_sw          = [0; 0; 0;     0; 0; 1;    0; 0; 1; 1; 0;
0];  % theta_1c, theta_TR, phi
        else
            opt_sw          = [0; 0; 0;     0; 1; 0;    0; 0; 1; 1; 0;
0];  % theta_1c, theta_TR, beta
        end
        out = trim_01 (P, pr_mi_MR ,pr_mi_TR, vars_lat, opt_sw, fm_act,
cf_wght, trim_pars);
        vars_lat = out(1:12);
        fprintf('FINAL ETAPA 4: lat_01\n')

        % previo alternante
        if Va_kh<Va_kh_sw
            vars = [Va; gamma; R;       vars_lon(4); 0; vars_lat(6);
vars_lon(7); vars_lon(8); vars_lat(9); vars_lat(10); i_HT; 0];
        else
            vars = [Va; gamma; R;       vars_lon(4); vars_lat(5); 0;
vars_lon(7); vars_lon(8); vars_lat(9); vars_lat(10); i_HT; 0];
        end
```

```
        cf_wght          = [0;    1; 1; 1;   0; 0; 0;   1; 1; 1];   % Fx, Fy,
Fz, Mx, My, Mz
        fm_act           = [1; 1; 1; 1; 1; 1];
        out = cf_01  (P, pr_mi_MR ,pr_mi_TR, vars, fm_act, cf_wght, 1);
        prev_FM = norm(out(2:3))


    i_iter          = 0;
    term_cond   = 0;
    %prev_FM            =   norm(FM);

    while (term_cond==0)

        i_iter      = i_iter+1;
        trim_pars   = 1 - (Va_kh<20);

    % long

        fm_act           = [1; 1; 1; 1; 1; 1];  %  Fg, MR, TR, FU, HT, VT
        cf_wght          = [0;    2.5; 1; 1;   0; 0; 0;   1; 1.7; 1.5];  %
Fx, Fy, Fz, Mx, My, Mz
        opt_sw           = [0; 0; 0;     1; 0; 0;     1; 1; 0; 0; 0; 0];  %
theta_1c, theta_TR, phi

        out = trim_02 (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw, fm_act,
cf_wght, trim_pars);
        vars = out(1:12);
        FM_lon = norm(out(14:15));

    %      fprintf('FINAL long\n')


    % lat

        fm_act           = [1; 1; 1; 1; 1; 1];  %  Fg, MR, TR, FU, HT, VT
        cf_wght          = [0;    2.5; 1; 1;   0; 0; 0;   1; 1.7; 1.5];  %
Fx, Fy, Fz, Mx, My, Mz
        if Va_kh<Va_kh_sw
            opt_sw           = [0; 0; 0;     0; 0; 1;     0; 0; 1; 1; 0;
0];  % theta_1c, theta_TR, phi
        else
            opt_sw           = [0; 0; 0;     0; 1; 0;     0; 0; 1; 1; 0;
0];  % theta_1c, theta_TR, beta
        end

        out = trim_02 (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw, fm_act,
cf_wght, trim_pars);
        vars = out(1:12);
        FM_lat = norm(out(14:15));

    %   fprintf('FINAL lat\n');


        mn_mean = 0.5*(FM_lat+FM_lon);
        test_delta = abs(mn_mean-prev_FM)/prev_FM;
```

```
        i_iter_max = 700;
        if i_iter>=i_iter_max
            term_cond = 1;
        elseif test_delta <1e-6
            term_cond = 2;
        elseif FM_lat<10 && FM_lon<10
            term_cond = 3;
        end

        prev_FM     = mn_mean

    end

      fprintf('FINAL ETAPA 05, tras iteración %d, con test_delta = %f
\n', i_iter, test_delta);


        m_fp =  strcat (fp_rs, '\Va_kh_', num2str(Va_kh));



        opt_sw          = [0; 0; 0;      1; 1; 1;    1; 1; 1; 1; 0; 0];
        fm_act          = ones(6, 1);
        cf_wght        = ones(10, 1);

        % % vars        = [Va; gamma; R;    alpha; beta; phi;   theta_0;
theta_1s; theta_1c; theta_TR; i_HT; 0];
        trim_vars = gen_trim_aa_cm (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw,
fm_act, cf_wght, 1, m_fp);


        % calculo theta
        alpha       = trim_vars(4);
        beta        = trim_vars(5);
        phi         = trim_vars(6);
        u           = Va*cos(alpha)*cos(beta);
        v           = Va*sin(beta);
        w           = Va*sin(alpha)*cos(beta);
        myfun       = @(th) Va*sin(gamma) - sin(th)*u +
sin(phi)*cos(th)*v + cos(phi)*cos(th)*w;
        th0         = alpha;
        theta       = fzero (myfun, th0);

        Va_kh_mn(i_v) = Va_kh;
        com_ang_mn(1:6,i_v) = trim_vars (7:12);
        geom_ang_mn(1:4,i_v)= [alpha, beta, theta, phi]';

        fprintf('hecho con Va_kh %f \n \n', Va_kh);
```

```
    end

    save (strcat (fp_rs, '\results_mn.mat'), 'Va_kh_mn', 'com_ang_mn',
'geom_ang_mn');


    % se restaura path original
    path(path_or);

end



function fp = directorio_raiz()

    % path_or     = path;
    % path_00     = strsplit(path_or,';');
    % aux_01      = char(path_00(1,1));
    aux_01      = mfilename('fullpath');
    div         = 'matlab_ordenado';
    path_01     = strsplit(aux_01,div);
    aux_02      = char(path_01(1,1));

    fp          = [aux_02 , div];

end


function crear_carpeta_resultados (fp_in, v_rg)

    fp1         = fp_in;
    if exist (fp1, 'dir');
        rmdir (fp1, 's');
    end
    mkdir  (fp1);
    n_v         = size (v_rg,2);
    for i_v = 1:n_v
        mkdir  ([fp1,'\Va_kh_',num2str(v_rg(i_v))]);
    end

end
```

D.2: General helicóptero incremental (v2)

```
function calculo_general_v2_phi

    % velocidad de cambio de modelo, se anula
    Va_kh_sw    = 1000;

    % puntos de partida
    i_v1      = 13; % 120 km/h
```

```
    i_v2        = 14; % 130 km/h



    % valores iniciales con algoritmo general v1
    calculo_general_v1_beta_phi(Va_kh_sw, i_v1, i_v2);                %
se anula cambio modelo; solo se anula beta en todo el rango

    % se guarda path original
    path_or       = path;

    % establecimiento directorios
    fp_rt         = directorio_raiz();
    fp_pr         = [fp_rt, '\00_previos'];
    fp_mdc        = [fp_rt, '\01_modelos_dinamicos\00_comunes'];
    fp_mdh        = [fp_rt, '\01_modelos_dinamicos\01_helicoptero'];
    fp_ac         = [fp_rt, '\02_algoritmos_calculo\01_helicoptero'];
    fp_vs         = [fp_rt, '\03_visualizacion_resultados'];
    fp_rs         = [fp_rt,
'\03_visualizacion_resultados\01_datos_helicoptero\01_ultimo_calculo'];

    addpath (genpath(fp_pr), fp_mdc, fp_mdh, fp_ac, fp_vs, '-begin');



    %carga parametros y constantes conversion
    run constantes_y_conversiones.m
    run UH60_params.m

    %carga valores de pasos iniciales
    gamma         = 0.0;
    R             = 1e308;%Inf;
    Va_rg_kh      = 0:10:260;
    Va_rg_kh(1)   = 5;
  %  n_v           = size(Va_rg_kh,2);
    load (strcat (fp_rs, '\results_mn.mat'));

    % deflacion del estabilizador horizontal (valores Howlett)
    Va_kn_ho      = [0.5, 20, 40, 60, 100, 140];           %   nudos
    Va_ho         = P.v_esc1*Va_kn_ho;                       %  m/s
    i_HT_rg       = [0.680550000000000,    0.680550000000000,
0.602665492046573,    0.335510072408958,    0.029521554212139,  -
0.005681703794443]; %rad


    % cálculo de velocidades inferiores a las de partida


     for i_v = (i_v1-1):-1:1

       % valores impuestos
```

```
        Va_kh    = Va_rg_kh(i_v);
        Va       = Va_kh/3.6;                %(m/s)
        i_HT     = interp1 (Va_ho, i_HT_rg, Va, 'pchip'); %rad

        fprintf('se empieza con Va_kh %f \n \n', Va_kh);

        % switch parametros trim
        trim_pars = 1;

       %valores impuestos por iteracion
        vars = zeros(12,1);

        vars (1:3) =    [Va; gamma; R];
        vars (11:12) =  [i_HT; 0];


        % para el resto de variables, se toman las soluciones de  Va
siguientes  como punto de partida

        Va_kh_pr_01 = Va_rg_kh(i_v+1);
        m_fp_pr_01 =  strcat (fp_rs, '\Va_kh_', num2str(Va_kh_pr_01));
        Va_kh_pr_02 = Va_rg_kh(i_v+2);
        m_fp_pr_02 =  strcat (fp_rs, '\Va_kh_', num2str(Va_kh_pr_02));

        S = load (strcat(m_fp_pr_01, '\test_comm.txt'), '-ascii');
        comm_pr_01 = S(end,:);
        S = load (strcat(m_fp_pr_02, '\test_comm.txt'), '-ascii');
        comm_pr_02 = S(end,:);
        vars(7:10) = comm_pr_01(2:5) + (comm_pr_01(2:5)-comm_pr_02(2:5));

        S = load (strcat(m_fp_pr_01, '\test_fm.txt'), '-ascii');
        fm_pr_01 = S(end,:);
        S = load (strcat(m_fp_pr_02, '\test_fm.txt'), '-ascii');
        fm_pr_02 = S(end,:);
        vars(4:6) = fm_pr_01(1:3) + (fm_pr_01(1:3)-fm_pr_02(1:3));

            % se retoma ETAPA 05
    cf_wght          = [0;   1; 1; 1;  0; 0; 0;  1; 1; 1];  % Fx, Fy, Fz,
Mx, My, Mz
    fm_act           = [1; 1; 1; 1; 1; 1];
    out = cf_01  (P, pr_mi_MR ,pr_mi_TR, vars, fm_act, cf_wght, 1);
    prev_FM = norm(out(2:3))


i_iter          = 0;
term_cond    = 0;
while (term_cond==0)

    i_iter      = i_iter+1;
    trim_pars   = 1 - (Va_kh<20);

    % long

    fm_act           = [1; 1; 1; 1; 1; 1];
```

```
    cf_wght          = [0;   2.5; 1; 1;  0; 0; 0;  1; 1.7; 1.5];  % Fx,
Fy, Fz, Mx, My, Mz
%     cf_wght          = [0;   1; 1; 1;  0; 0; 0;  1; 1; 1];  % Fx, Fy,
Fz, Mx, My, Mz
    opt_sw          = [0; 0; 0;     1; 0; 0;    1; 1; 0; 0; 0; 0];  %
theta_1c, theta_TR, phi

    out = trim_02 (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw, fm_act, cf_wght,
trim_pars);
    vars = out(1:12);
    FM_lon = norm(out(14:15));

%      fprintf('FINAL long\n')


% lat

    fm_act          = [1; 1; 1; 1; 1; 1];
    cf_wght          = [0;   2.5; 1; 1;  0; 0; 0;  1; 1.7; 1.5];  % Fx,
Fy, Fz, Mx, My, Mz
%     cf_wght          = [0;   1; 1; 1;  0; 0; 0;  1; 1; 1];  % Fx, Fy,
Fz, Mx, My, Mz
    if Va_kh<Va_kh_sw
        opt_sw          = [0; 0; 0;     0; 0; 1;    0; 0; 1; 1; 0; 0];  %
theta_1c, theta_TR, phi
    else
        opt_sw          = [0; 0; 0;     0; 1; 0;    0; 0; 1; 1; 0; 0];  %
theta_1c, theta_TR, beta
    end

    out = trim_02 (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw, fm_act, cf_wght,
trim_pars);
    vars = out(1:12);
    FM_lat = norm(out(14:15));

%   fprintf('FINAL lat\n');


    mn_mean = 0.5*(FM_lat+FM_lon);
    test_delta = abs(mn_mean-prev_FM)/prev_FM;


    i_iter_max = 700;
    if i_iter>=i_iter_max
        term_cond = 1;
    elseif test_delta <1e-6
        term_cond = 2;
    elseif FM_lat<5 && FM_lon<5 % FM_lat<15 && FM_lon<15
        term_cond = 3;
    end
```

```
    prev_FM     = mn_mean

end

  fprintf('FINAL ETAPA 05, tras iteración %d, con test_delta = %f \n',
i_iter, test_delta);

    m_fp =  strcat (fp_rs, '\Va_kh_', num2str(Va_kh));

    opt_sw          = [0; 0; 0;     1; 1; 1;    1; 1; 1; 1; 0; 0];
    fm_act          = ones(6, 1);
    cf_wght        = ones(10, 1);

    % % vars          = [Va; gamma; R;    alpha; beta; phi;   theta_0;
theta_1s; theta_1c; theta_TR; i_HT; 0];
    trim_vars = gen_trim_aa_cm (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw,
fm_act, cf_wght, 1, m_fp);


    % calculo theta
    alpha       = trim_vars(4);
    beta        = trim_vars(5);
    phi         = trim_vars(6);
    u           = Va*cos(alpha)*cos(beta);
    v           = Va*sin(beta);
    w           = Va*sin(alpha)*cos(beta);
    myfun       = @(th) Va*sin(gamma) - sin(th)*u + sin(phi)*cos(th)*v +
cos(phi)*cos(th)*w;
    th0         = alpha;
    theta       = fzero (myfun, th0);

    Va_kh_mn(i_v) = Va_kh;
    com_ang_mn(1:6,i_v) = trim_vars (7:12);
    geom_ang_mn(1:4,i_v)= [alpha, beta, theta, phi]';

    fprintf('hecho con Va %f \n \n', Va_kh);

    save (strcat (fp_rs, '\results_mn.mat'), 'Va_kh_mn', 'com_ang_mn',
'geom_ang_mn');

     end


    % cálculo de velocidades superiores a las de partida


    for i_v = (i_v2+1):1:27

       % valores impuestos
        Va_kh   = Va_rg_kh(i_v);
        Va      = Va_kh/3.6;                %(m/s)
        i_HT    = interp1 (Va_ho, i_HT_rg, Va, 'pchip'); %rad

        fprintf('se empieza con Va_kh %f \n \n', Va_kh);
```

```
        % switch parametros trim
        trim_pars = 1;

       %valores impuestos por iteracion
        vars = zeros(12,1);

        vars (1:3) =    [Va; gamma; R];
        vars (11:12) =  [i_HT; 0];


        % para el resto de variables, se toman las soluciones de Va
anteriores como punto de partida

        Va_kh_pr_01 = Va_rg_kh(i_v-1);
        m_fp_pr_01 =  strcat (fp_rs, '\Va_kh_', num2str(Va_kh_pr_01));
        Va_kh_pr_02 = Va_rg_kh(i_v-2);
        m_fp_pr_02 =  strcat (fp_rs, '\Va_kh_', num2str(Va_kh_pr_02));

        S = load (strcat(m_fp_pr_01, '\test_comm.txt'), '-ascii');
        comm_pr_01 = S(end,:);
        S = load (strcat(m_fp_pr_02, '\test_comm.txt'), '-ascii');
        comm_pr_02 = S(end,:);
        vars(7:10) = comm_pr_01(2:5) + (comm_pr_01(2:5)-comm_pr_02(2:5));

        S = load (strcat(m_fp_pr_01, '\test_fm.txt'), '-ascii');
        fm_pr_01 = S(end,:);
        S = load (strcat(m_fp_pr_02, '\test_fm.txt'), '-ascii');
        fm_pr_02 = S(end,:);
        vars(4:6) = fm_pr_01(1:3) + (fm_pr_01(1:3)-fm_pr_02(1:3));



                % se retoma ETAPA 05
    cf_wght          = [0;   1; 1; 1;  0; 0; 0;  1; 1; 1];  % Fx, Fy, Fz,
Mx, My, Mz
    fm_act           = [1; 1; 1; 1; 1; 1];
    out = cf_01  (P, pr_mi_MR ,pr_mi_TR, vars, fm_act, cf_wght, 1);
    prev_FM = norm(out(2:3))


i_iter          = 0;
term_cond   = 0;
while (term_cond==0)

    i_iter      = i_iter+1;
    trim_pars   = 1 - (Va_kh<20);

    % long

    fm_act           = [1; 1; 1; 1; 1; 1];
```

```
    cf_wght         = [0;   2.5; 1; 1;   0; 0; 0;   1; 1.7; 1.5];  % Fx,
Fy, Fz, Mx, My, Mz
%     cf_wght         = [0;   1; 1; 1;   0; 0; 0;   1; 1; 1];  % Fx, Fy,
Fz, Mx, My, Mz
    opt_sw          = [0; 0; 0;     1; 0; 0;     1; 1; 0; 0; 0; 0];  %
theta_1c, theta_TR, phi

    out = trim_02 (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw, fm_act, cf_wght,
trim_pars);
    vars = out(1:12);
    FM_lon = norm(out(14:15));

%      fprintf('FINAL long\n')


% lat

    fm_act          = [1; 1; 1; 1; 1; 1];
    cf_wght         = [0;   2.5; 1; 1;   0; 0; 0;   1; 1.7; 1.5];  % Fx,
Fy, Fz, Mx, My, Mz
%     cf_wght         = [0;   1; 1; 1;   0; 0; 0;   1; 1; 1];  % Fx, Fy,
Fz, Mx, My, Mz
    if Va_kh<Va_kh_sw
        opt_sw          = [0; 0; 0;     0; 0; 1;     0; 0; 1; 1; 0; 0];  %
theta_1c, theta_TR, phi
    else
        opt_sw          = [0; 0; 0;     0; 1; 0;     0; 0; 1; 1; 0; 0];  %
theta_1c, theta_TR, beta
    end

    out = trim_02 (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw, fm_act, cf_wght,
trim_pars);
    vars = out(1:12);
    FM_lat = norm(out(14:15));

%   fprintf('FINAL lat\n');


    mn_mean = 0.5*(FM_lat+FM_lon);
    test_delta = abs(mn_mean-prev_FM)/prev_FM;


    i_iter_max = 700;
    if i_iter>=i_iter_max
        term_cond = 1;
    elseif test_delta <1e-6
        term_cond = 2;
    elseif FM_lat<5 && FM_lon<5 % FM_lat<15 && FM_lon<15
        term_cond = 3;
    end

    prev_FM     = mn_mean

end
```

```
  fprintf('FINAL ETAPA 05, tras iteración %d, con test_delta = %f \n',
i_iter, test_delta);

    m_fp =  strcat (fp_rs, '\Va_kh_', num2str(Va_kh));

    opt_sw          = [0; 0; 0;     1; 1; 1;    1; 1; 1; 1; 0; 0];
    fm_act          = ones(6, 1);
    cf_wght        = ones(10, 1);

    % % vars          = [Va; gamma; R;    alpha; beta; phi;   theta_0;
theta_1s; theta_1c; theta_TR; i_HT; 0];
    trim_vars = gen_trim_aa_cm (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw,
fm_act, cf_wght, 1, m_fp);


    % calculo theta
    alpha       = trim_vars(4);
    beta        = trim_vars(5);
    phi         = trim_vars(6);
    u           = Va*cos(alpha)*cos(beta);
    v           = Va*sin(beta);
    w           = Va*sin(alpha)*cos(beta);
    myfun       = @(th) Va*sin(gamma) - sin(th)*u + sin(phi)*cos(th)*v +
cos(phi)*cos(th)*w;
    th0         = alpha;
    theta       = fzero (myfun, th0);

    Va_kh_mn(i_v) = Va_kh;
    com_ang_mn(1:6,i_v) = trim_vars (7:12);
    geom_ang_mn(1:4,i_v)= [alpha, beta, theta, phi]';

    fprintf('hecho con Va %f \n \n', Va_kh);

    save (strcat (fp_rs, '\results_mn.mat'), 'Va_kh_mn', 'com_ang_mn',
'geom_ang_mn');

    end

    % se restaura path original
    path(path_or);



end



function fp = directorio_raiz()

    % path_or     = path;
```

```
    % path_00     = strsplit(path_or,';');
    % aux_01      = char(path_00(1,1));
    aux_01      = mfilename('fullpath');
    div         = 'matlab_ordenado';
    path_01     = strsplit(aux_01,div);
    aux_02      = char(path_01(1,1));

    fp          = [aux_02 , div];

end
```

D.3: Función de trimado 1

```
function out = trim_01 (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw, fm_act,
cf_wght, trim_pars)

% vars = [TV; AA; CM] = (Va, gamma, R;  alpha, beta, phi; commands, i_HT,
beta_prop)

    tau = 0.5;
    sigma = 0.5;
    n_iter_max = 4;
    n_tau_max= 400;
    delta = 0.7;
    J0_obj = 1e-8;
    inc = 1e-6;


i_n = 1;
term_cond = 0;
k = 1;
J0_ref = Inf;
% fprintf('empezamos a pensar \n');

while (term_cond==0)

    cc =  cf_01  (P, pr_mi_MR ,pr_mi_TR, vars, fm_act, cf_wght, 0);
    J0  = cc(1);
    inc = min([1e-6, 0.01*k*norm(J0)]);

    dJ = zeros(12,1);
    for i_sw = 1:12
        if opt_sw(i_sw)==1
            inc_vars = vars;
            inc_inp = vars(i_sw)+inc;
%             % se limita entrada saturada
%             if 6<i_sw && i_sw<11
%                 inc_inp = inc_inp.*(P.theta_min(i_sw-
6)<inc_inp).*(inc_inp<P.theta_max(i_sw-6)) + P.theta_max(i_sw-
6).*(P.theta_max(i_sw-6)<inc_inp) + P.theta_min(i_sw-
6).*(inc_inp<P.theta_min(i_sw-6));
%             end
            inc_vars(i_sw) = inc_inp;
```

```
            J_i  = cf_01 (P, pr_mi_MR ,pr_mi_TR, inc_vars, fm_act,
cf_wght, 0);
            dJ(i_sw) = (1/inc)*(J_i-J0);
        end
    end
    dJn = dJ/norm(dJ);

    k = 1;
    cc = cf_01 (P, pr_mi_MR ,pr_mi_TR, vars-k*dJn, fm_act, cf_wght, 0);
    i_tau = 0;
%  fprintf('fuera J0: %f, cc(1): %f\n',J0, cc(1) );
    while cc(1)>=J0-sigma*k*norm(dJ) && i_tau<5001
        k = tau*k;
        cc = cf_01 (P, pr_mi_MR ,pr_mi_TR, vars-k*dJn, fm_act, cf_wght,
0);
        i_tau = i_tau+1;
%       fprintf('dentro J0: %f, cc(1): %f\n',J0, cc(1) );
    end
    if i_tau>n_tau_max   %esto va muy lento
        term_cond = 5;
    end

    vars    = vars-delta*k*dJn;

    if J0<J0_obj
        term_cond = 1;
    end
    %if abs(J0_ref-J0)/J0_ref<1e-7 %J0 no decrece lo suficiente
    %   term_cond = 2;
%   end
    if J0>J0_ref
        term_cond = 3;
    end
    if i_n>n_iter_max  %esto va muy lento
        term_cond = 4;
    end

    i_n = i_n+1;
    J0_ref = J0;
%   if mod(i_n,25)==0
%       fprintf('iteracion: %d  -  J0: %f; k = %f ; inc = %f \n', i_n,
J0, k, inc);
%   end

end

    % fprintf('campana y se acabo. Motivo: %d \n', term_cond);
    % fprintf('iteracion: %d  -  J0: %f; k = %f ; inc = %f \n', i_n, J0,
k, inc);
    cc = cf_01 (P, pr_mi_MR ,pr_mi_TR, vars, fm_act, cf_wght, 1);
    out = [vars; cc; term_cond];
```

```
end
```

D.4: Función de trimado 2

```
function out = trim_02 (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw, fm_act,
cf_wght, trim_pars)

% vars = [TV; AA; CM] = (Va, gamma, R;  alpha, beta, phi; commands, i_HT,
beta_prop)


    if trim_pars==0
        tau = 0.3;
        sigma = 0.3;
        delta = 0.4;
    else
        tau = 0.5;
        sigma = 0.5;
        delta = 0.6;
    end
    n_iter_max = 2;
    n_tau_max= 400;
    J0_obj = 1e-8;





i_n = 1;
term_cond = 0;
k=1;
J0_ref = Inf;
% fprintf('empezamos a pensar \n');

while (term_cond==0)

    cc =  cf_01  (P, pr_mi_MR ,pr_mi_TR, vars, fm_act, cf_wght, 0);
    J0   = cc(1);
%     inc = min([1e-6, J0^(1.8/1)]);
inc = min([1e-6, k*norm(J0)]);

    dJ = zeros(12,1);
    for i_sw = 1:12
        if opt_sw(i_sw)==1
            inc_vars = vars;
            inc_inp = vars(i_sw)+inc;
%             % se limita entrada saturada
%             if 6<i_sw && i_sw<12
%                 inc_inp = inc_inp.*(P.theta_min(i_sw-
6)<inc_inp).*(inc_inp<P.theta_max(i_sw-6)) + P.theta_max(i_sw-
6).*(P.theta_max(i_sw-6)<inc_inp) + P.theta_min(i_sw-
6).*(inc_inp<P.theta_min(i_sw-6));
%             end
             inc_vars(i_sw) = inc_inp;
            J_i  = cf_01 (P, pr_mi_MR ,pr_mi_TR, inc_vars, fm_act,
cf_wght, 0);
```

```
            dJ(i_sw) = (1/inc)*(J_i-J0);
        end
    end
    dJn = dJ/norm(dJ);

    k   = 1;
    cc = cf_01 (P, pr_mi_MR ,pr_mi_TR, vars-k*dJn, fm_act, cf_wght, 0);
    i_tau = 0;
%  fprintf('fuera J0: %f, cc(1): %f\n',J0, cc(1) );
    while cc(1)>=J0-sigma*k*norm(dJ) && i_tau<5001
        k = tau*k;
        cc = cf_01 (P, pr_mi_MR ,pr_mi_TR, vars-k*dJn, fm_act, cf_wght,
0);
        i_tau = i_tau+1;
%       fprintf('dentro J0: %f, cc(1): %f\n',J0, cc(1) );
    end
    if i_tau>n_tau_max    %esto va muy lento
        term_cond = 5;
    end

    vars    = vars-delta*k*dJn;

    if J0<J0_obj
        term_cond = 1;
    end
    %if abs(J0_ref-J0)/J0_ref<1e-7 %J0 no decrece lo suficiente
    %   term_cond = 2;
%   end
    if J0>J0_ref
        term_cond = 3;
    end
    if i_n>n_iter_max    %esto va muy lento
        term_cond = 4;
    end

    i_n = i_n+1;
    J0_ref = J0;
%   if mod(i_n,25)==0
%       fprintf('iteracion: %d  -  J0: %f; k = %f ; inc = %f \n', i_n,
J0, k, inc);
%   end

end

% fprintf('campana y se acabo. Motivo: %d \n', term_cond);
% fprintf('iteracion: %d  -  J0: %f; k = %f ; inc = %f \n', i_n, J0, k,
inc);
    cc = cf_01 (P, pr_mi_MR ,pr_mi_TR, vars, fm_act, cf_wght, 1);
    out = [vars; cc; term_cond];

end
```

D.5: Función de coste general

```
function out = gen_cf  (P, pr_mi_MR ,pr_mi_TR, m_inputs, fm_act, cf_wght, 
out_flag, fp) % inputs = [TV; AA; CM] = (Va, gamma, R;  alpha, beta, phi; 
commands, i_HT, beta_prop)

% global pr_mi_MR;
% global pr_mi_TR;

% desmigado entradas
Va           = m_inputs(1);
gamma        = m_inputs(2);
R            = m_inputs(3);
alpha        = m_inputs(4);
beta         = m_inputs(5);
phi          = m_inputs(6);
cmds         = m_inputs(7:12);


% valor objetivo de estados
dx_t = [-Va*sin(gamma); 0;0;0; 0;0;Va/R*cos(gamma); 0;0;0];


% velocidades actuales
u            = Va*cos(alpha)*cos(beta);
v            = Va*sin(beta);
w            = Va*sin(alpha)*cos(beta);
%theta         = alpha+gamma;
myfun = @(th) Va*sin(gamma) - sin(th)*u + sin(phi)*cos(th)*v + 
cos(phi)*cos(th)*w;
th0   = alpha;
theta = fzero (myfun, th0);

p            = -Va/R*sin(theta);
q            = Va/R*sin(phi)*cos(theta);
r            = Va/R*cos(phi)*cos(theta);

lv_B         = [u; v; w];
ar_B         = [p; q; r];
d_ar_B       = [0;0;0];

% FM de helicoptero

Fg           = P.mass*P.g*[-sin(theta); cos(theta)*sin(phi); 
cos(theta)*cos(phi); 0; 0; 0]  ;

out_MR   = cN_fm_main_rotor(cmds, lv_B, ar_B, d_ar_B, pr_mi_MR, P);
chi_MR   = out_MR(19);
v_i_MR   = out_MR(20);

out_TR   = UH60_fm_tail_rotor(cmds, lv_B, ar_B, chi_MR, v_i_MR, pr_mi_TR, 
P);
v_i_TR   = out_TR(7);
```

```
out_FU  = UH60_fm_fuselage (lv_B, chi_MR, v_i_MR, P);
out_HT  = UH60_fm_hor_tail(lv_B, ar_B, chi_MR, v_i_MR, cmds(5), P);
out_VT  = cN_fm_ver_tail (lv_B, ar_B, chi_MR, v_i_MR, v_i_TR, P);

%out_RW  = modUH60_rw (lv_B, ar_B, chi_MR, v_i_MR, P);
%out_LW  = modUH60_lw (lv_B, ar_B, chi_MR, v_i_MR, P);
%out_PR  = modUH60_pr (lv_B, ar_B, chi_MR, v_i_MR, Va, beta_prop, P);

FM           = [Fg, out_MR(13:18), out_TR(1:6), out_FU(1:6), out_HT(1:6),
out_VT(1:6)]*fm_act; %+out_RW(1:6)+out_LW(1:6)+out_PR(1:6);


% estados actuales
fx       = FM(1);
fy       = FM(2);
fz       = FM(3);
ell      = FM(4);
m        = FM(5);
n        = FM(6);

Gamma=P.Ixx*P.Izz-P.Ixz^2;
Gamma1=(1/Gamma)*P.Ixz*(P.Ixx-P.Iyy+P.Izz);
Gamma2=(1/Gamma)*(P.Izz*(P.Izz-P.Iyy)+P.Ixz^2);
Gamma3=(1/Gamma)*P.Izz;
Gamma4=(1/Gamma)*P.Ixz;
Gamma5=(1/P.Iyy)*(P.Izz-P.Ixx);
Gamma6=(1/P.Iyy)*(P.Ixz);
Gamma7=(1/Gamma)*(P.Ixx*(P.Ixx-P.Iyy)+P.Ixz^2);
Gamma8=(1/Gamma)*P.Ixx;

pddot = -sin(theta)*u + sin(phi)*cos(theta)*v + cos(phi)*cos(theta)*w;

udot = r*v - q*w + fx/P.mass;
vdot = p*w - r*u + fy/P.mass;
wdot = q*u - p*v + fz/P.mass;

phidot      = p + q*sin(phi)*tan(theta) + r*cos(phi)*tan(theta);
thetadot    = q*cos(phi) - r*sin(phi);
psidot      = q*sin(phi)/cos(theta) + r*cos(phi)/cos(theta);

pdot = Gamma1*p*q - Gamma2*q*r + Gamma3*ell + Gamma4*n;
qdot = Gamma5*p*r - Gamma6*(p^2-r^2) + (1/P.Iyy)*m;
rdot = Gamma7*p*q - Gamma1*q*r + Gamma4*ell + Gamma8*n;

% derivadas actuales y funcion coste
dx_i = [pddot; udot; vdot; wdot; phidot; thetadot; psidot; pdot; qdot;
rdot];
J = norm(cf_wght.*(dx_i-dx_t))^2;

if out_flag==1
```

```
    out = [J; dx_i];
    % preparacion fichero
    aux = '%20.6f %20.6f %20.6f %20.6f %20.6f %20.6f %20.6f %20.6f %20.6f
%20.6f ';
    %format_fm = strcat (aux, aux, aux, aux, aux, aux, aux, aux, aux,
aux, aux, aux, aux, aux, '%20.6f %20.6f \n');
    format_fm = strcat (aux, aux, aux, aux, aux, aux, aux, aux, aux, aux,
'\n');
    fid_fm = fopen(strcat(fp, '\test_fm.txt'),'a');
    todo = [alpha; beta; phi; ...
% 1 + 1 + 1  (3)
                    theta; lv_B; ar_B; ...
% 1 + 3 + 3  (7)
                    out_MR; out_TR; out_FU; out_HT; out_VT; ...
% 20 + 16 + 14 + 14 + 14 (78)
    %                out_RW; out_LW; out_PR; ...
% 14 + 14 + 14  (42);
                    Fg; FM];
% (12)
    fprintf(fid_fm,format_fm,todo);
    fclose(fid_fm);
else
    out = J;
end

end
```

D.6: Función de coste 1

```
function out = cf_01  (P, pr_mi_MR ,pr_mi_TR, m_inputs, fm_act, cf_wght,
out_sw) % inputs = [TV; AA; CM] = (Va, gamma, R;  alpha, beta, phi;
commands, i_HT, beta_prop)

% desmigado entradas
Va          = m_inputs(1);
gamma       = m_inputs(2);
R           = m_inputs(3);
alpha       = m_inputs(4);
beta        = m_inputs(5);
phi         = m_inputs(6);
cmds        = m_inputs(7:12);


% velocidades actuales
u           = Va*cos(alpha)*cos(beta);
v           = Va*sin(beta);
w           = Va*sin(alpha)*cos(beta);
%theta       = alpha+gamma;
myfun = @(th) Va*sin(gamma) - sin(th)*u + sin(phi)*cos(th)*v +
cos(phi)*cos(th)*w;
th0   = alpha;
theta = fzero (myfun, th0);

p           = -Va/R*sin(theta);
q           = Va/R*sin(phi)*cos(theta);
r           = Va/R*cos(phi)*cos(theta);
```

```
lv_B        = [u; v; w];
ar_B        = [p; q; r];
d_ar_B      = [0;0;0];

% FM de helicoptero

Fg          = P.mass*P.g*[-sin(theta); cos(theta)*sin(phi);
cos(theta)*cos(phi); 0; 0; 0]  ;

out_MR  = cN_fm_main_rotor(cmds, lv_B, ar_B, d_ar_B, pr_mi_MR, P);
chi_MR  = out_MR(19);
v_i_MR  = out_MR(20);

out_TR  = UH60_fm_tail_rotor(cmds, lv_B, ar_B, chi_MR, v_i_MR, pr_mi_TR,
P);
v_i_TR  = out_TR(7);

out_FU  = UH60_fm_fuselage (lv_B, chi_MR, v_i_MR, P);
out_HT  = UH60_fm_hor_tail(lv_B, ar_B, chi_MR, v_i_MR, cmds(5), P);
out_VT  = cN_fm_ver_tail (lv_B, ar_B, chi_MR, v_i_MR, v_i_TR, P);

%out_RW  = modUH60_rw (lv_B, ar_B, chi_MR, v_i_MR, P);
%out_LW  = modUH60_lw (lv_B, ar_B, chi_MR, v_i_MR, P);
%out_PR  = modUH60_pr (lv_B, ar_B, chi_MR, v_i_MR, Va, beta_prop, P);

FM          = [Fg, out_MR(13:18), out_TR(1:6), out_FU(1:6), out_HT(1:6),
out_VT(1:6)]*fm_act; %+out_RW(1:6)+out_LW(1:6)+out_PR(1:6);
mFM         = [0; FM(1:3); 0; 0; 0; FM(4:6)];
sum_F       = sqrt(FM(1)^2+FM(2)^2+FM(3)^2);
sum_M       = sqrt(FM(4)^2+FM(5)^2+FM(6)^2);

% funcion coste y output
J = sqrt(sum(cf_wght.*(mFM.^2)));

    if out_sw==1
        out         = [J; sum_F; sum_M];
    else
        out = J;
    end

end
```

D. 7. General girodino (v1)

```
function calculo_general_v1_beta_phi(varargin)

    %clc
```

```
    %clear

    % se guarda path original
    path_or     = path;

    % establecimiento directorios
    fp_rt       = directorio_raiz();
    fp_pr       = [fp_rt, '\00_previos'];
    fp_mdc      = [fp_rt, '\01_modelos_dinamicos\00_comunes'];
    fp_mdh      = [fp_rt, '\01_modelos_dinamicos\02_girodino'];
    fp_ac       = [fp_rt, '\02_algoritmos_calculo\02_girodino'];
    fp_vs       = [fp_rt, '\03_visualizacion_resultados'];
    fp_rs       = [fp_rt,
'\03_visualizacion_resultados\02_datos_girodino\01_ultimo_calculo'];

    addpath (genpath(fp_pr), fp_mdc, fp_mdh, fp_ac, fp_vs, '-begin');



    %carga parametros y constantes conversion
    run constantes_y_conversiones.m
    run modUH60_params.m




    % vuelo compensado horizontal; rango de velocidades de estudio
    if nargin<2                         % para total calculo v1
        Va_rg_kh    = 0:10:380;
        Va_rg_kh(1) = 5;
        n_v         = size(Va_rg_kh,2);
        rg_kh       = 1:n_v;
        Va_kh_mn    = zeros(1,n_v);
        com_ang_mn  = zeros(6,n_v);
        geom_ang_mn = zeros(4,n_v);
    elseif nargin==3                    % para pasos iniciales v2
        Va_rg_kh    = 0:10:380;
        Va_rg_kh(1) = 5;
        n_v         = size(Va_rg_kh,2);
        rg_kh       = varargin{2}:varargin{3};
        Va_kh_mn    = zeros(1,n_v);
        com_ang_mn  = zeros(6,n_v);
        geom_ang_mn = zeros(4,n_v);
    end
    gamma       = 0.0;
    R           = 1e308;        %Inf;
    crear_carpeta_resultados (fp_rs, Va_rg_kh);


    % velocidad de cambio de modelo (54 kn)
    if nargin<1                                         % cambio phi-beta
        Va_kh_sw    = 54.0*v_esc2;      %km/h
    else                                                 % solo se anula
beta
        Va_kh_sw    = varargin{1};      %km/h
    end
```

```
    % deflacion del estabilizador horizontal (valores Howlett)
    Va_kn_ho    = [0.5, 20, 40, 60, 100, 140];          %   nudos
    Va_ho       = (P.v_esc1)*Va_kn_ho;                      %  m/s
    i_HT_rg     = [0.680550000000000,    0.680550000000000,
0.602665492046573,    0.335510072408958,    0.029521554212139,   -
0.005681703794443]; %rad
    i_HT_max    = i_HT_rg(end);
    Va_HT_max   = Va_ho(end);


    % programa cuadrático propulsor
    m1   = P.m_beta;
    V_ign    = 119*0.5144;
    beta_ign      = 14.1*(pi/180); % rad
    V_end   =   200*0.5144;
    beta_end     = 32.2*(pi/180);

    b2           = (beta_end-beta_ign-
m1*V_end^2/(2*V_ign)+V_ign*m1/2)/(V_end-1/2*V_ign-V_end^2/(2*V_ign));
    a2           = (m1-b2)/(2*V_ign);
    c2           = (beta_ign-1/2*V_ign*(b2+m1));




    for i_v = rg_kh

        Va_kh    = Va_rg_kh(i_v);
        Va       = Va_kh/3.6;                %(m/s)
        fprintf('se empieza con Va_kh %f \n \n', Va_kh);

        %variables fijadas
        i_HT         = (Va<=Va_HT_max).*(interp1 (Va_ho, i_HT_rg, Va,
'pchip')) + (Va_HT_max<Va).*(Va-Va_HT_max)./(Va_rg_kh(n_v)./3.6-
Va_HT_max).*(i_HT_max-0.0349); %rad
        beta_prop    = (a2*Va.^2+b2*Va+c2).*(Va>V_ign);

        if Va_kh<Va_kh_sw
            beta     = 0.0;
        else
            phi      = 0.0;
        end

        % switch parametros trim
        trim_pars = 1;

        % valores iniciales
        vars0             = [Va; gamma; R;  0; 0; 0;  0.0; 0.0; 0; 0;
i_HT; 0];
```

```
        % ETAPA 01
        fm_act          = [1; 1; 0; 1; 0; 0; 1; 1; 1];  % Fg, MR, FU, RW,
LW, PR
        cf_wght         = [0;   1; 0; 1;  0; 0; 0;  0; 0; 0];  %Fx, Fz
        opt_sw          = [0; 0; 0;     0; 0; 0;    1; 1; 0; 0; 0; 0];  %
theta_0, theta_1s
        trim_pars       = [''];

        out = trim_01 (P, pr_mi_MR ,pr_mi_TR, vars0, opt_sw, fm_act,
cf_wght, trim_pars);
        vars_lon = out(1:12);
        fprintf('FINAL ETAPA 1: long_00\n')

        % ETAPA 02
        fm_act          = [1; 1; 0; 1; 1; 0; 1; 1; 1];  % Fg, MR, FU +
HT, RW, LW, PR
        cf_wght         = [0;   1; 0; 1;  0; 0; 0;  0; 1; 0];  %Fx, Fz,
My
        opt_sw          = [0; 0; 0;     1; 0; 0;    1; 1; 0; 0; 0; 0];  %
theta_0, theta_1s, alpha
        trim_pars       = [''];

        out = trim_01 (P, pr_mi_MR ,pr_mi_TR, vars_lon, opt_sw, fm_act,
cf_wght, trim_pars);
        vars_lon = out(1:12);
        fprintf('FINAL ETAPA 2: long_01\n')

        % ETAPA 03
        fm_act          = [0; 1; 1; 0; 0; 0; 0; 0; 0];  %  MR, TR
        cf_wght         = [0;   0; 0; 0;  0; 0; 0;  0; 0; 1];  % Mz
        opt_sw          = [0; 0; 0;     0; 0; 0;    0; 0; 0; 1; 0; 0];  %
theta_TR
        trim_pars       = [''];

        out = trim_01 (P, pr_mi_MR ,pr_mi_TR, vars0, opt_sw, fm_act,
cf_wght, trim_pars);
        vars_lat = out(1:12);
        fprintf('FINAL ETAPA 3: lat_00\n')


        % ETAPA 04
        fm_act          = [1; 1; 1; 1; 0; 1; 1; 1; 1];  %  Fg, MR, TR,
FU, VT, RW, LW, PR
        cf_wght         = [0;   0; 1; 0;  0; 0; 0;  1; 0; 1];  % Fy, Mx,
Mz
        trim_pars       = [''];
        if Va_kh<Va_kh_sw
            opt_sw          = [0; 0; 0;     0; 0; 1;    0; 0; 1; 1; 0;
0];  % theta_1c, theta_TR, phi
        else
            opt_sw          = [0; 0; 0;     0; 1; 0;    0; 0; 1; 1; 0;
0];  % theta_1c, theta_TR, beta
        end
        out = trim_01 (P, pr_mi_MR ,pr_mi_TR, vars_lat, opt_sw, fm_act,
cf_wght, trim_pars);
```

```
        vars_lat = out(1:12);
        fprintf('FINAL ETAPA 4: lat_01\n')

        % previo alternante
        if Va_kh<Va_kh_sw
            vars = [Va; gamma; R;       vars_lon(4); 0; vars_lat(6);
vars_lon(7); vars_lon(8); vars_lat(9); vars_lat(10); i_HT; 0];
        else
            vars = [Va; gamma; R;       vars_lon(4); vars_lat(5); 0;
vars_lon(7); vars_lon(8); vars_lat(9); vars_lat(10); i_HT; 0];
        end
        cf_wght         = [0;   1; 1; 1;  0; 0; 0;  1; 1; 1];  % Fx, Fy,
Fz, Mx, My, Mz
        fm_act          = [1; 1; 1; 1; 1; 1; 1; 1; 1];
        out = cf_01  (P, pr_mi_MR ,pr_mi_TR, vars, fm_act, cf_wght, 1);
        prev_FM = norm(out(2:3))


    i_iter          = 0;
    term_cond   = 0;
    %prev_FM            =   norm(FM);

    while (term_cond==0)

        i_iter      = i_iter+1;
        trim_pars   = 1 - (Va_kh<20);

    % long

        fm_act          = [1; 1; 1; 1; 1; 1; 1; 1; 1];  %  Fg, MR, TR,
FU, HT, VT, RW, LW, PR
        cf_wght         = [0;   2.5; 1; 1;  0; 0; 0;  1; 1.7; 1.5];  %
Fx, Fy, Fz, Mx, My, Mz
        opt_sw          = [0; 0; 0;     1; 0; 0;    1; 1; 0; 0; 0; 0];  %
theta_1c, theta_TR, phi

        out = trim_02 (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw, fm_act,
cf_wght, trim_pars);
        vars = out(1:12);
        FM_lon = norm(out(14:15));

    %      fprintf('FINAL long\n')


    % lat

        fm_act          = [1; 1; 1; 1; 1; 1; 1; 1; 1];  %  Fg, MR, TR,
FU, HT, VT, RW, LW, PR
        cf_wght         = [0;   2.5; 1; 1;  0; 0; 0;  1; 1.7; 1.5];  %
Fx, Fy, Fz, Mx, My, Mz
        if Va_kh<Va_kh_sw
```

```
            opt_sw          = [0; 0; 0;     0; 0; 1;    0; 0; 1; 1; 0;
0];  % theta_1c, theta_TR, phi
        else
            opt_sw          = [0; 0; 0;     0; 1; 0;    0; 0; 1; 1; 0;
0];  % theta_1c, theta_TR, beta
        end

        out = trim_02 (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw, fm_act,
cf_wght, trim_pars);
        vars = out(1:12);
        FM_lat = norm(out(14:15));

    %   fprintf('FINAL lat\n');


        mn_mean = 0.5*(FM_lat+FM_lon);
        test_delta = abs(mn_mean-prev_FM)/prev_FM;


        i_iter_max = 700;
        if i_iter>=i_iter_max
            term_cond = 1;
        elseif test_delta <1e-6
            term_cond = 2;
        elseif FM_lat<10 && FM_lon<10
            term_cond = 3;
        end

        prev_FM     = mn_mean

    end

      fprintf('FINAL ETAPA 05, tras iteración %d, con test_delta = %f
\n', i_iter, test_delta);


        m_fp =  strcat (fp_rs, '\Va_kh_', num2str(Va_kh));



        opt_sw          = [0; 0; 0;     1; 1; 1;    1; 1; 1; 1; 0; 0];
        fm_act          = ones(9, 1);
        cf_wght        = ones(10, 1);

        % % vars        = [Va; gamma; R;    alpha; beta; phi;   theta_0;
theta_1s; theta_1c; theta_TR; i_HT; 0];
        trim_vars = gen_trim_aa_cm (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw,
fm_act, cf_wght, 1, m_fp);


        % calculo theta
        alpha       = trim_vars(4);
        beta        = trim_vars(5);
        phi         = trim_vars(6);
```

```
        u           = Va*cos(alpha)*cos(beta);
        v           = Va*sin(beta);
        w           = Va*sin(alpha)*cos(beta);
        myfun       = @(th) Va*sin(gamma) - sin(th)*u +
sin(phi)*cos(th)*v + cos(phi)*cos(th)*w;
        th0         = alpha;
        theta       = fzero (myfun, th0);

        Va_kh_mn(i_v) = Va_kh;
        com_ang_mn(1:6,i_v) = trim_vars (7:12);
        geom_ang_mn(1:4,i_v)= [alpha, beta, theta, phi]';

        fprintf('hecho con Va_kh %f \n \n', Va_kh);

    end

    save (strcat (fp_rs, '\results_mn.mat'), 'Va_kh_mn', 'com_ang_mn',
'geom_ang_mn');


    % se restaura path original
    path(path_or);

end



function fp = directorio_raiz()

    % path_or      = path;
    % path_00      = strsplit(path_or,';');
    % aux_01       = char(path_00(1,1));
    aux_01      = mfilename('fullpath');
    div         = 'matlab_ordenado';
    path_01     = strsplit(aux_01,div);
    aux_02      = char(path_01(1,1));

    fp          = [aux_02 , div];

end


function crear_carpeta_resultados (fp_in, v_rg)

    fp1         = fp_in;
    if exist (fp1, 'dir');
        rmdir (fp1, 's');
    end
    mkdir  (fp1);
    n_v         = size (v_rg,2);
    for i_v = 1:n_v
```

```
        mkdir  ([fp1,'\Va_kh_',num2str(v_rg(i_v))]);
    end

end
```

D. 8. General girodino incremental (v2)

```
function calculo_general_v2_phi_reestructurado

    % velocidad de cambio de modelo, se anula
    Va_kh_sw    = 1000;

    % puntos de partida
%     i_v1      = 8; % 140 km/h
%     i_v2      = 9; % 160 km/h
    i_v1      = 15; % 140 km/h
    i_v2      = 16; % 160 km/h


    % valores iniciales con algoritmo general v1
    calculo_general_v1_beta_phi(Va_kh_sw, i_v1, i_v2);              %
se anula cambio modelo; solo se anula beta en todo el rango

    % se guarda path original
    path_or     = path;

    % establecimiento directorios
    fp_rt       = directorio_raiz();
    fp_pr       = [fp_rt, '\00_previos'];
    fp_mdc      = [fp_rt, '\01_modelos_dinamicos\00_comunes'];
    fp_mdh      = [fp_rt, '\01_modelos_dinamicos\02_girodino'];
    fp_ac       = [fp_rt, '\02_algoritmos_calculo\02_girodino'];
    fp_vs       = [fp_rt, '\03_visualizacion_resultados'];
    fp_rs       = [fp_rt,
'\03_visualizacion_resultados\02_datos_girodino\01_ultimo_calculo'];

    addpath (genpath(fp_pr), fp_mdc, fp_mdh, fp_ac, fp_vs, '-begin');



    %carga parametros y constantes conversion
    run constantes_y_conversiones.m
    run modUH60_params.m

    %carga valores de pasos iniciales
    gamma       = 0.0;
    R           = 1e308;%Inf;
    Va_rg_kh    = 0:10:380;
    n_v         = size (Va_rg_kh,2);
    Va_rg_kh(1) = 5;
%   load (strcat (fp_rs, '\results_mn.mat'));
```

```
    % cálculo de velocidades inferiores a las de partida


     for i_v = (i_v1-1):-1:1

        Va_kh    = Va_rg_kh(i_v);
        Va       = Va_kh/3.6;                   %(m/s)
        fprintf('se empieza con Va_kh %f \n \n', Va_kh);

        vars = valores_iniciales_01 (P, Va, gamma, R, Va_rg_kh, n_v, i_v,
fp_rs);
        calculo_principal (P, pr_mi_MR ,pr_mi_TR, vars, Va_kh, Va_kh_sw,
i_v, fp_rs);

        fprintf('hecho con Va %f \n \n', Va_kh);


     end


    % cálculo de velocidades superiores a las de partida

    for i_v = (i_v2+1):1:n_v

    %    load (strcat (fp_rs, '\results_mn.mat'), 'Va_kh_mn',
'com_ang_mn', 'geom_ang_mn');

       % valores impuestos
        Va_kh    = Va_rg_kh(i_v);
        Va       = Va_kh/3.6;                   %(m/s)
        fprintf('se empieza con Va_kh %f \n \n', Va_kh);

        vars = valores_iniciales_02 (P, Va, gamma, R, Va_rg_kh, n_v, i_v,
fp_rs);
        calculo_principal (P, pr_mi_MR ,pr_mi_TR, vars, Va_kh, Va_kh_sw,
i_v, fp_rs);

        fprintf('hecho con Va %f \n \n', Va_kh);


    end


    % se restaura path original
    path(path_or);

end
```

```
function fp = directorio_raiz()

    % path_or      = path;
    % path_00      = strsplit(path_or,';');
    % aux_01       = char(path_00(1,1));
    aux_01       = mfilename('fullpath');
    div          = 'matlab_ordenado';
    path_01      = strsplit(aux_01,div);
    aux_02       = char(path_01(1,1));

    fp           = [aux_02 , div];

end

function vars = valores_iniciales_01 (P, Va, gamma, R, Va_rg_kh, n_v,
i_v, fp_rs)

    % deflacion del estabilizador horizontal (valores Howlett)
    Va_kn_ho     = [0.5, 20, 40, 60, 100, 140];           %   nudos
    Va_ho        = (P.v_esc1)*Va_kn_ho;                     %  m/s
    i_HT_rg      = [0.680550000000000,    0.680550000000000,
0.602665492046573,    0.335510072408958,    0.029521554212139,   -
0.005681703794443]; %rad
    i_HT_max     = i_HT_rg(end);
    Va_HT_max    = Va_ho(end);


    % programa cuadrático propulsor
    m1    =      P.m_beta;
    V_ign     =  119*0.5144;
    beta_ign     = 14.1*(pi/180); % rad
    V_end    =   200*0.5144;
    beta_end     = 32.2*(pi/180);

    b2             = (beta_end-beta_ign-
m1*V_end^2/(2*V_ign)+V_ign*m1/2)/(V_end-1/2*V_ign-V_end^2/(2*V_ign));
    a2             = (m1-b2)/(2*V_ign);
    c2             = (beta_ign-1/2*V_ign*(b2+m1));


    %variables fijadas
    %  i_HT     = interp1 (Va_ho, i_HT_rg, Va, 'pchip'); %rad
    %i_HT          = (Va<=Va_HT_max).*(interp1 (Va_ho, i_HT_rg, Va,
'pchip')) + (Va_HT_max<Va).*i_HT_max; %rad
    i_HT         = (Va<=Va_HT_max).*(interp1 (Va_ho, i_HT_rg, Va,
'pchip')) + (Va_HT_max<Va).*(Va-Va_HT_max)./(Va_rg_kh(n_v)./3.6-
Va_HT_max).*(i_HT_max-0.0349); %rad

    beta_prop = (a2*Va.^2+b2*Va+c2).*(Va>V_ign);

    % switch parametros trim
    trim_pars = 1;
```

```
   %valores impuestos por iteracion
    vars = zeros(12,1);

    vars (1:3) =    [Va; gamma; R];
    vars (11:12) =  [i_HT; beta_prop];


    % para el resto de variables, se toman las soluciones de  Va
siguientes  como punto de partida

    Va_kh_pr_01 = Va_rg_kh(i_v+1);
    m_fp_pr_01 =  strcat (fp_rs, '\Va_kh_', num2str(Va_kh_pr_01));
    Va_kh_pr_02 = Va_rg_kh(i_v+2);
    m_fp_pr_02 =  strcat (fp_rs, '\Va_kh_', num2str(Va_kh_pr_02));

    S = load (strcat(m_fp_pr_01, '\test_comm.txt'), '-ascii');
    comm_pr_01 = S(end,:);
    S = load (strcat(m_fp_pr_02, '\test_comm.txt'), '-ascii');
    comm_pr_02 = S(end,:);
    vars(7:10) = comm_pr_01(2:5) + (comm_pr_01(2:5)-comm_pr_02(2:5));

    S = load (strcat(m_fp_pr_01, '\test_fm.txt'), '-ascii');
    fm_pr_01 = S(end,:);
    S = load (strcat(m_fp_pr_02, '\test_fm.txt'), '-ascii');
    fm_pr_02 = S(end,:);
    vars(4:6) = fm_pr_01(1:3) + (fm_pr_01(1:3)-fm_pr_02(1:3));

end

function vars = valores_iniciales_02 (P, Va, gamma, R, Va_rg_kh, n_v,
i_v, fp_rs)

     Va_kh    = Va_rg_kh(i_v);

  % deflacion del estabilizador horizontal (valores Howlett)
    Va_kn_ho     = [0.5, 20, 40, 60, 100, 140];          %   nudos
    Va_ho        = (P.v_esc1)*Va_kn_ho;                      %  m/s
    i_HT_rg      = [0.680550000000000,    0.680550000000000,
0.602665492046573,    0.335510072408958,    0.029521554212139,   -
0.005681703794443]; %rad
    i_HT_max     = i_HT_rg(end);
    Va_HT_max    = Va_ho(end);


    % programa cuadrático propulsor
    m1    =      P.m_beta;
    V_ign     =   119*0.5144;
    beta_ign     = 14.1*(pi/180); % rad
    V_end    =   200*0.5144;
    beta_end     = 32.2*(pi/180);
```

```
    b2           = (beta_end-beta_ign-
m1*V_end^2/(2*V_ign)+V_ign*m1/2)/(V_end-1/2*V_ign-V_end^2/(2*V_ign));
    a2           = (m1-b2)/(2*V_ign);
    c2           = (beta_ign-1/2*V_ign*(b2+m1));



       %variables fijadas
%i_HT        = (Va<=Va_HT_max).*(interp1 (Va_ho, i_HT_rg, Va, 'pchip')) +
(Va_HT_max<Va).*i_HT_max; %rad
%        i_HT    = interp1 (Va_ho, i_HT_rg, Va, 'pchip'); %rad
  i_HT        = (Va<=Va_HT_max).*(interp1 (Va_ho, i_HT_rg, Va, 'pchip'))
+ (Va_HT_max<Va).*(Va-Va_HT_max)./(Va_rg_kh(n_v)./3.6-
Va_HT_max).*(i_HT_max-0.0349); %rad
        beta_prop = (a2*Va.^2+b2*Va+c2).*(Va>V_ign);

        % switch parametros trim
        trim_pars = 1;

       %valores impuestos por iteracion
        vars = zeros(12,1);

        vars (1:3) =    [Va; gamma; R];
        vars (11:12) =  [i_HT; beta_prop];


        % para el resto de variables, se toman las soluciones de Va
anteriores como punto de partida


            if Va_kh<280
                Va_kh_pr_01 =  Va_rg_kh(i_v-1);
                m_fp_pr_01 =  strcat (fp_rs, '\Va_kh_',
num2str(Va_kh_pr_01));
                Va_kh_pr_02 =  Va_rg_kh(i_v-2);
                m_fp_pr_02 =  strcat (fp_rs, '\Va_kh_',
num2str(Va_kh_pr_02));

                S = load (strcat(m_fp_pr_01, '\test_comm.txt'), '-
ascii');
                comm_pr_01 = S(end,:);
                S = load (strcat(m_fp_pr_02, '\test_comm.txt'), '-
ascii');
                comm_pr_02 = S(end,:);
                vars(7:10) = comm_pr_01(2:5) + (comm_pr_01(2:5)-
comm_pr_02(2:5));

                S = load (strcat(m_fp_pr_01, '\test_fm.txt'), '-ascii');
                fm_pr_01 = S(end,:);
                S = load (strcat(m_fp_pr_02, '\test_fm.txt'), '-ascii');
                fm_pr_02 = S(end,:);
                vars(4:6) = fm_pr_01(1:3) + (fm_pr_01(1:3)-
fm_pr_02(1:3));
            else
```

```
                load (strcat (fp_rs, '\results_mn.mat'), 'Va_kh_mn',
'com_ang_mn', 'geom_ang_mn');
                %aux = interp1(Va_kh_mn(:,(end-5):end)',
[geom_ang_mn(:,(end-5):end)',com_ang_mn(:,(end-5):end)'], Va_kh
,'pchip');
                aux = interp1(Va_kh_mn(:,(i_v-6):(i_v-1))',
[geom_ang_mn(:,(i_v-6):(i_v-1))',com_ang_mn(:,(i_v-6):(i_v-1))'], Va_kh
,'pchip');
                vars(7:10) = aux(5:8)';
                vars(4:6) = [aux(1), aux(2), aux(4)];
            end


end

function calculo_principal (P, pr_mi_MR ,pr_mi_TR, vars, Va_kh, Va_kh_sw,
i_v, fp_rs)




     % se retoma ETAPA 05
        cf_wght         = [0;   1; 1; 1;  0; 0; 0;  1; 1; 1];  % Fx, Fy,
Fz, Mx, My, Mz
        fm_act          = [1; 1; 1; 1; 1; 1; 1; 1; 1];
        out = cf_01  (P, pr_mi_MR ,pr_mi_TR, vars, fm_act, cf_wght, 1);
        prev_FM = norm(out(2:3))


        i_iter          = 0;
        term_cond   = 0;
        while (term_cond==0)

            i_iter      = i_iter+1;
            trim_pars   = 1 - (Va_kh<20);

            % long

            fm_act          = [1; 1; 1; 1; 1; 1; 1; 1; 1];  %  Fg, MR,
TR, FU, HT, VT, RW, LW, PR
            cf_wght         = [0;   2.5; 1; 1;  0; 0; 0;  1; 1.7; 1.5];
% Fx, Fy, Fz, Mx, My, Mz
        %     cf_wght         = [0;   1; 1; 1;  0; 0; 0;  1; 1; 1];  %
Fx, Fy, Fz, Mx, My, Mz
            opt_sw          = [0; 0; 0;     1; 0; 0;    1; 1; 0; 0; 0;
0];  % theta_1c, theta_TR, phi

            out = trim_02 (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw, fm_act,
cf_wght, trim_pars);
            vars = out(1:12);
            FM_lon = norm(out(14:15));
```

```
        %      fprintf('FINAL long\n')


        % lat

            fm_act          = [1; 1; 1; 1; 1; 1; 1; 1; 1];  %  Fg, MR,
TR, FU, HT, VT, RW, LW, PR
            cf_wght         = [0;   2.5; 1; 1;  0; 0; 0;  1; 1.7; 1.5];
% Fx, Fy, Fz, Mx, My, Mz
        %     cf_wght         = [0;   1; 1; 1;  0; 0; 0;  1; 1; 1];   %
Fx, Fy, Fz, Mx, My, Mz
            if Va_kh<Va_kh_sw
                opt_sw          = [0; 0; 0;     0; 0; 1;    0; 0; 1; 1;
0; 0];  % theta_1c, theta_TR, phi
            else
                opt_sw          = [0; 0; 0;     0; 1; 0;    0; 0; 1; 1;
0; 0];  % theta_1c, theta_TR, beta
            end

            out = trim_02 (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw, fm_act,
cf_wght, trim_pars);
            vars = out(1:12);
            FM_lat = norm(out(14:15));

        %   fprintf('FINAL lat\n');


            mn_mean = 0.5*(FM_lat+FM_lon);
            test_delta = abs(mn_mean-prev_FM)/prev_FM;


            i_iter_max = 700;
            if i_iter>=i_iter_max
                term_cond = 1;
            elseif test_delta <1e-6
                term_cond = 2;
            elseif FM_lat<15 && FM_lon<15 % FM_lat<5 && FM_lon<5
                term_cond = 3;
            end

            prev_FM     = mn_mean

        end

        fprintf('FINAL ETAPA 05, tras iteración %d, con test_delta = %f
\n', i_iter, test_delta);

        m_fp =  strcat (fp_rs, '\Va_kh_', num2str(Va_kh));

        opt_sw          = [0; 0; 0;     1; 1; 1;    1; 1; 1; 1; 0; 0];
        fm_act          = ones(9, 1);
        cf_wght        = ones(10, 1);

        % % vars        = [Va; gamma; R;    alpha; beta; phi;   theta_0;
theta_1s; theta_1c; theta_TR; i_HT; 0];
```

```
        trim_vars = gen_trim_aa_cm (P, pr_mi_MR ,pr_mi_TR, vars, opt_sw,
fm_act, cf_wght, 1, m_fp);


        % calculo theta

        Va           = vars (1);
        gamma        = vars(2);
        R            = vars (3);

        alpha        = trim_vars(4);
        beta         = trim_vars(5);
        phi          = trim_vars(6);
        u            = Va*cos(alpha)*cos(beta);
        v            = Va*sin(beta);
        w            = Va*sin(alpha)*cos(beta);
        myfun        = @(th) Va*sin(gamma) - sin(th)*u +
sin(phi)*cos(th)*v + cos(phi)*cos(th)*w;
        th0          = alpha;
        theta        = fzero (myfun, th0);


        load(strcat (fp_rs, '\results_mn.mat'), 'Va_kh_mn', 'com_ang_mn',
'geom_ang_mn');
        Va_kh_mn(i_v) = Va_kh;
        com_ang_mn(1:6,i_v) = trim_vars(7:12);
        geom_ang_mn(1:4,i_v)= [trim_vars(4); trim_vars(5); theta;
trim_vars(6)];
        save (strcat (fp_rs, '\results_mn.mat'), 'Va_kh_mn',
'com_ang_mn', 'geom_ang_mn');


end
```