Apéndice B

Código MATLAB

En este apéndice se explican los códigos de MATLAB más importantes que se el DVD.

Para obtener una simulación con flujo impuesto U, abrir el archivo ./MATLAB/Vibracion con modos naturales con fluido y gravedad/Flujo impuesto/fixedflow.m y modificar los valores de las variables en la cabecera del programa. Posteriormente ejecutar y esperar que terminen los calculos.

Para obtener una simulación con presión impuesta con velocidad equivalente U_{eq}^{equiv} dada, hacer lo mismo que en el caso anterior con el archivo: ./MATLAB/Vibracion con modos naturales con fluido y gravedad/Presion impuesta/presion.m

B.1. Simulación

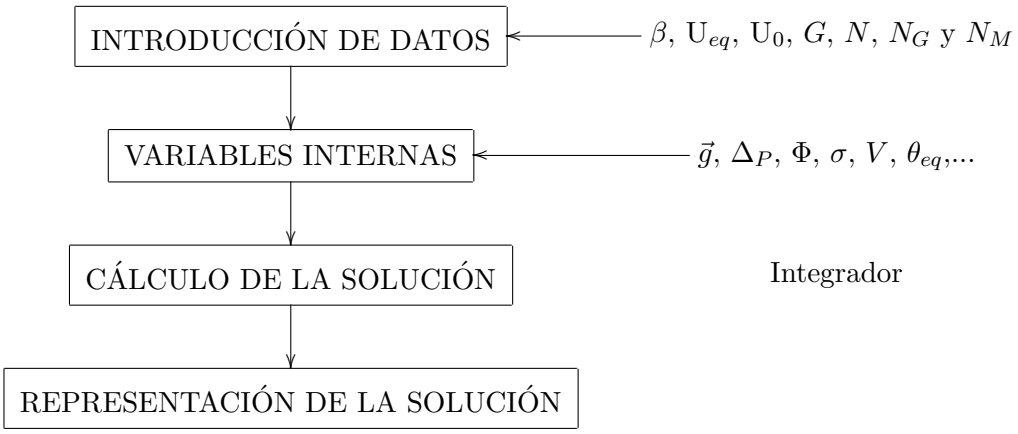
Para obtener una simulación con flujo impuesto U, abrir el archivo ./MATLAB/Simulacion/Flujo impuesto/fixedflow.m y modificar los valores de las variables en la cabecera del programa. Posteriormente ejecutar y esperar que terminen los calculos.

Para obtener una simulación con presión impuesta con velocidad equivalente U_{eq}^{equiv} dada, hacer lo mismo que en el caso anterior con el archivo: ./MATLAB/Simulacion/Presion impuesta/presion.m

B.1.1. Flujo impuesto

La simulación con flujo impuesto U sigue el siguiente esquema

Programa principal sigue el siguiente esquema



Programa principal:

```
clc,        clear,       close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% INTRODUCCION DE DATOS %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%discretisation
N=100;          NG=4;          NM=4;

%fluid flow
beta=0.2;              U=13.5;
%gravity
G=0;

%%NO MODIFICAR A PARTIR DE AQUI

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% CALCULO DE VARIABLES INTERNAS %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tetaG=-pi/2;
g=G*[cos(tetaG);sin(tetaG)];

%%%% discretizacion
epsilon1=5e-1;

%%%% vector tiempo
```

```
inct=1e-2 ; tf=6; tspan=0:inct:tf;

%%%% matriz FI
FI=calcFI(N,NG);
FI_gorro=[FI,zeros(size(FI));zeros(size(FI)),FI];

%%%% incremento de presion necesario
[teta_eq,teta_prima_eq]=gravity(N,G,tetaG,beta,U);
%%%%H=-sum( sin(teta_eq) )/N;   inc_P=.5*beta*U^2-beta*G*H;

%%%% matrices
L=tril(ones(N))-.5*eye(N);  A=(L'*L)/(N^3);
K=eye(N)-diag(ones(N-1,1),-1);  K=K+K';K(N,N)=1;K(1,1)=1;   K=K*N;
D=L'*ones(N,1)/N; g=G*[cos(tetaG);sin(tetaG)];

%%%% problema autovalores
[CAMBIO,sigma_,VV]=calcEIG(N,NG,NM,teta_eq,beta,U,G,tetaG,FI);

%%%% perturbacion del modo inestable
if max(real(sigma_))<1e-10,
    disp('No es inestable'),
    y0(2*NM-1:2*NM,1)=epsilon1;
else
J=1;    while sigma_(J)<1e-10, J=J+1; end
y0=zeros(2*NM,1); y0(J)=epsilon1; y0(J+1)=epsilon1;
end

%%%% Energia inicial
E0=0;
ppp=CAMBIO*y0;
teta(1:N,1)=ppp(1:N,1);
tetap(1:N,1)=ppp(1+N:2*N,1);   Angle=teta*ones(1,N);
M=A.*cos(Angle-Angle');
E0=E0+.5*teta'*(K)*teta;
n=[-sin(teta),cos(teta)];   tau=[cos(teta),sin(teta)];
v_p=(L/N)*([tetap,tetap].*n);   v_f=(L/N)*([tetap,tetap].*n)+U*tau;
for i=1:N,
    norm_v_p(i,1)=norm(v_p(i,:));
    norm_v_f(i,1)=norm(v_f(i,:));
```

```
end,
E0=E0+.5*(1-beta)*(norm_v_p'*norm_v_p )/N;
E0=E0+.5*beta*(norm_v_f'*norm_v_f )/N;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% CALCULO DE LA SOLUCION %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[T,Y_] = ODE15s(@fixedflowdiff,tspan,[y0;E0],[],N,NG,NM,A,L,K,D,...
    beta,g,CAMBIO,teta_eq,U,FI);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% PLOTEO DE LA SOLUCION %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
save ws,
representa(N,NG,NM,T,Y_,VV,FI,teta_eq,inct,sigma_,...
CAMBIO,A,K,L,U,beta,g)
```

Integración numérica (llamada al integrador):

```
function yp=fixedflowdiff(t,y,N,NG,NM,A,L,K,D,beta,g,CAMBIO,teta_eq,U,FI)

%%%%% RENOMBRAR VARIABLES
y_=y(1:(2*NM),1);
p=CAMBIO(1:2*N,1:2*NM)*y_;
teta(1:N,1)=p(1:N,1)+teta_eq;
tetap(1:N,1)=p(1+N:2*N,1);
THETA=[teta;tetap];

%%%%% CALCULO DE MATRICES EN THETA EN EL INSTANTE
Angle=teta*ones(1,N);Angle=Angle-Angle';%Angle(i,j)=Angle_i-Angle_j;
COS=cos(Angle); SIN=sin(Angle);
n=[-sin(teta),cos(teta)]; tau=[cos(teta),sin(teta)]; %B=L'-L;
TAU=[cos(teta(N,1)),sin(teta(N,1))];
%% ecuacion del sistema
Qg=(1/N)*[D,D].*n*g;         E=ones(1,N)/N;
Cu=2*L'.*COS/(N^2); Qf=beta*(U^2)*(1/N)*(SIN(:,N));
M=A.*COS; C=(A.*SIN)*diag(tetap);
Qg=(1/N)*[D,D].*n*g;
```

```
%%%%% EN FORMA DE STATE-SPACE EN [ALPHA,U]
MM=[eye(size(M)),zeros(size(M));zeros(size(M)),M];
AA=[zeros(size(M)),eye(size(M));-K,-(C+beta*U*Cu)];
Qpipe=[zeros(size(Qf));Qf+Qg];
Mass= CAMBIO'*MM*CAMBIO;
QQ=CAMBIO'*[AA*THETA+Qpipe];

alp_p=Mass\QQ;

E=ones(1,N)/N;
Mfp=beta*E*( (L/N).*SIN);     Cfp=-beta*E*((L/N).*COS*diag(tetap));
%P=Mfp*FI*yp(Modes+1:2*Modes,1)+Cfp*tetap;
P=[Cfp,Mfp]*CAMBIO*alp_p-0*beta*E*tau*g;

BENJAMIN=1;  %%RESUELVE ENERGIA DE BENJAMIN si BENJAMIN=1, sino las dadas
%%en el texto.
if BENJAMIN==1,
    EE=-beta*U*((norm(n'*tetap/N)^2)+U*TAU*(n'*tetap/N));
else
EE=-beta*U*(.5*(norm(n'*tetap/N)^2)+U*TAU*(n'*tetap/N))+...
    U*P+(1-beta)*E*(L/N)*(tetap.*(n*g))+0*U*beta*E*tau*g;
end

yp=[alp_p;EE];
```

El resto de archios son para calcular la forma de equilibrio.

B.1.2. Presión impuesta

El programa es el mismo pero introduciendo la ecuación (2.27) y el término en $\dot{U}$ de la ecuación (2.26).

```
clc,        clear,       close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% INTRODUCCION DE DATOS %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%discretisation
N=100;          NG=3;          NM=3;
```

```
%fluid flow
beta=0.2;               U0=0;    U=16;
%gravity
G=20;

%%NO MODIFICAR A PARTIR DE AQUI

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% CALCULO DE VARIABLES INTERNAS %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
alpha=0; reg=2;
tetaG=-pi/2;
g=G*[cos(tetaG);sin(tetaG)];

%%%% discretizacion
epsilon1=5e-1;

%%%% vector tiempo
inct=1e-2 ; tf=5; tspan=0:inct:tf;

%%%% matriz FI
FI=calcFI(N,NG);
FI_gorro=[FI,zeros(size(FI));zeros(size(FI)),FI];

%%%% incremento de presion necesario
[teta_eq,teta_prima_eq]=gravity(N,G,tetaG,beta,U);
H=-sum( sin(teta_eq) )/N;   inc_P=.5*beta*U^2-beta*G*H;

%%%% matrices
L=tril(ones(N))-.5*eye(N);  A=(L'*L)/(N^3);
K=eye(N)-diag(ones(N-1,1),-1);  K=K+K';K(N,N)=1;K(1,1)=1;   K=K*N;
D=L'*ones(N,1)/N; g=G*[cos(tetaG);sin(tetaG)];

%%%% problema autovalores
[CAMBIO_,sigma_,VV]=calcEIG(N,NG,NM,teta_eq,beta,U,G,tetaG,FI);
CAMBIO=[CAMBIO_,zeros(2*N,1);zeros(1,2*NM),1];

%%%% problema autovalores
```

```
E0=.5*beta*U^2;

%%%% perturbacion del modo inestable
if max(real(sigma_))<1e-10,
    disp('No es inestable'),
    y0(2*NM-1:2*NM,1)=epsilon1;
else
J=1;    while sigma_(J)<1e-10, J=J+1; end
y0=zeros(2*NM,1); y0(J)=epsilon1; y0(J+1)=epsilon1;
end

%%%% Energia inicial
E0=0;
ppp=CAMBIO_*y0;
teta(1:N,1)=ppp(1:N,1);
tetap(1:N,1)=ppp(1+N:2*N,1);   Angle=teta*ones(1,N);
M=A.*cos(Angle-Angle');
E0=E0+.5*teta'*(K)*teta;
n=[-sin(teta),cos(teta)];   tau=[cos(teta),sin(teta)];
v_p=(L/N)*([tetap,tetap].*n);
v_f=(L/N)*([tetap,tetap].*n)+U0*tau;
for i=1:N,
    norm_v_p(i,1)=norm(v_p(i,:));
    norm_v_f(i,1)=norm(v_f(i,:));
end,
E0=E0+.5*(1-beta)*(norm_v_p'*norm_v_p )/N;
E0=E0+.5*beta*(norm_v_f'*norm_v_f )/N;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% CALCULO DE LA SOLUCION %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[T,Y_] = ODE15s(@presiondiff,tspan,[y0;U0;E0],[],N,NG,NM,A,L,K,D,...
    beta,g,alpha,reg,CAMBIO,teta_eq,inc_P);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% PLOTEO DE LA SOLUCION %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
save ws,
```

```
representa(N,NG,NM,T,Y_,VV,FI,teta_eq,inct,sigma_,CAMBIO_,A,K,L,U,beta,g)
```

Integración numérica (llamada al integrador):

```
function yp=presiondiff(t,y,N,NG,NM,A,L,K,D,...
beta,g,alpha,reg,CAMBIO,teta_eq,inc_P)
%%%%% RENOMBRAR VARIABLES
y_=y(1:(2*NM),1);
U=y((2*NM+1),1);
p=CAMBIO(1:2*N,1:2*NM)*y_;
teta(1:N,1)=p(1:N,1)+teta_eq;
tetap(1:N,1)=p(1+N:2*N,1);
THETA=[teta;tetap];

%%%%% CALCULO DE MATRICES EN THETA EN EL INSTANTE
Angle=teta*ones(1,N);Angle=Angle-Angle';%Angle(i,j)=Angle_i-Angle_j;
COS=cos(Angle); SIN=sin(Angle);
n=[-sin(teta),cos(teta)]; tau=[cos(teta),sin(teta)]; %B=L'-L;
TAU=[cos(teta(N,1)),sin(teta(N,1))];
%% ecuacion del sistema
Qg=(1/N)*[D,D].*n*g;         E=ones(1,N)/N;
Cu=2*L'.*COS/(N^2); Qf=beta*(U^2)*(1/N)*(SIN(:,N));
M=A.*COS; C=(A.*SIN)*diag(tetap);
Qg=(1/N)*[D,D].*n*g;
Mpf=-beta*(L'.*SIN/N)*(E');
%% ecuacion del fluido
Mfp=beta*E*( (L/N).*SIN);
Cfp=-beta*E*((L/N).*COS)*diag(tetap);
P=inc_P-(alpha*(U^(reg-1))+.5*beta*U)*U;
Qfluid=P+beta*E*tau*g-Cfp*tetap;

%%%%% EN FORMA DE STATE-SPACE EN [ALPHA,U]
MM=[eye(size(M)),zeros(size(M));zeros(size(M)),M];
MMpf=[zeros(N,1);Mpf];
MMfp=[zeros(1,N),Mfp];
AA=[zeros(size(M)),eye(size(M));-K,-(C+beta*U*Cu)];
Qpipe=[zeros(size(Qf));Qf+Qg];
Mass= CAMBIO'*[MM,MMpf;MMfp,beta]*CAMBIO;
QQ=CAMBIO'*[AA*THETA+Qpipe;Qfluid];
```

```
yp=Mass\QQ;


EE=-beta*U*(.5*(norm(n'*tetap/N)^2)+U*TAU*(n'*tetap/N))+...
U*P+E*(L/N)*(tetap.*(n*g))+U*beta*E*tau*g;
yp((2*NM+2),1)=EE;
```

Las funciones básicas vienen dadas por la ecuación (2.24) aunque para valores muy altos de λ hay problemas numéricos que se resuelven fácilemente dividiendo numerador y denominador por la exponencial e^{λ}. calcFI.m calcula la matrix Φ correctamente.

```
function FI_=calcFI(N,NG)

lambda(1:4)=[1.875104, 4.694091, 7.854757, 10.99554];
for i=5:NG,    lambda(i)=(i-.5)*pi; end
for i=1:NG
    a=lambda(i); C=(cosh(a)+cos(a))/(sinh(a)+sin(a));
    f=(1+exp(-2*a)+2*exp(-a)*cos(a))/(1-exp(-2*a)+2*exp(-a)*sin(a));
    for j=1:N
        xi=(j-.5)/N;
        FI_(j,i)=.5*(exp(a*xi)*(1-f)-exp(-a*xi)*(1+f))+sin(a*xi)+f*cos(a*xi);
    end
end,
```

B.2. Solución de equilibrio

En la carpeta *./MATLAB/Estabilidad/gravedad pi* se encuentra los archivos donde se calcula la formas de equilibrio no rectas.

gravedaddiff.m integra solo con la fuerza de la gravedad desde $\theta(0)$

```
function yp=gravedaddiff(xi,y,thetaG,G)

yp(1,1)=y(2,1);
yp(2,1)=-(1-xi)*G*sin( thetaG-y(1,1) );
```

gravedadfluidodiff.m integra con la fuerza de la gravedad y con fluido desde $\theta(1)$

```
function yp=gravedadfluidodiff(xi,y,thetaG,G,betaU2,d)

theta=y(1,1);
yp(1,1)=y(2,1);
yp(2,1)=-xi*G*sin( thetaG-theta ) + betaU2*sin(d-theta);
```

gravedad.m calcula la deformada bajo gravedad G

```
clc,close all,clear all

thetaG=pi;
G_=0:.5:40
clear G_, G_=30

N=100;  XI_=[0:1/N:1]'; z(1:N+1,1)=1;

for j=1:length(G_)
G=G_(j)

A=fsolve(@(x) objetivo(x,thetaG,G,XI_,N),-10)

[XI,Y]= ode45(@gravedaddiff,XI_,A*[0;1],[],thetaG,G);
figure(1), pause(.025)
x=cumsum(cos(Y(1:1+N,1))/N);y=cumsum(sin(Y(1:1+N,1))/N);

tauN_(j)=Y(N+1,1);
xtip(j)=x(length(x));ytip(j)=y(length(y)); ztip(j)=G;

figure(1), hold on
plot3(x,y,G*z,'k'),axis([-1 1 -1 1 0 G_(length(G_))]),axis square
end
figure(1), hold on,
plot3(xtip,ytip,ztip,'k'),plot3([0 0],[0 0],[0 G_(length(G_))],'k')

figure(2), plot3(x,y,'k'),axis([-1 1 -1 1 0 G_(length(G_))]), axis square

save('map2','G_','tauN_','N')
```

gravedadfluido.m calcula la deformada bajo gravedad G y flujo βU

```
load map2
G=30;           tauN=-interp1(G_,tauN_,G);z(1:N+1,1)=1;

thetaG=pi;
N=100;  XI_=[0:1/N:1]';

incd=.02
d=-(tauN*[incd:incd:1]);

LD=length(d)
for jj=1:LD, jj/LD
    j=LD-(jj-1);

    betaU2(j)=fsolve(@(x) objetivo1(x,d(j),thetaG,G,XI_,N),15);
    [XI,Y_(:,:,j)]= ode45(@gravedadfluidodiff,...
XI_,d(j)*[1;0],[],thetaG,G,betaU2(j),d(j));
    for i=1:2,  for k=1:length(XI)
    Y(k,i,j)=-Y_(length(XI)+1-k,i,j);
    end,        end,

    x=cumsum(cos(Y(:,1,j))/N);y=cumsum(sin(Y(:,1,j))/N);
    xtip(j)=x(length(x));ytip(j)=y(length(y)); ztip(j)=G;
    figure(1), hold off, plot(x,y,'k'),
    axis([-1 1 -1 1]), title(num2str(betaU2(j)))
end
```

las funciones *objetivo.m* y *objetivo1.m* son las condiciones de contorno que se imponen por el método de shooting, $\theta'(1) = 0$ y $\theta(0) = 0$.

objetivo.m:

```
function M=objetivo(a,thetaG,G,XI_,N)

[XI,Y]= ode45(@gravedaddiff,XI_,a*[0;1],[],thetaG,G);
M=Y(1+N,2);
```

objetivo1.m:

```
function ang0=objetivo1(betaU2,d,thetaG,G,XI_,N)
```

```
[XI,Y]= ode45(@gravedadfluidodiff,XI_,d*[1;0],[],thetaG,G,betaU2,d);
ang0=Y(1+N,1);
```

B.3. Estabilidad

El siguiente programa calcula la estabilidad en el caso $\theta_G = \pi$ tanto para flujo impuesto U como para diferencia de presión impuesta ΔP.

```
function EstabPiGrecto(G)

%%%clc,
%%%clear all,
load map2,
%%%%close all

%%%% PARAMETERS

%discretisation
Modes=15;  NG=Modes;

N=100;  XI_=[0:1/N:1]';
thetaG=pi;

%%%%G_=7.6:.001:7.7;
%%G=45;

incbeta=.0025;  beta_=.0025:.0025:1;
UU=0:.1:25;

L=(tril(ones(N))-.5*eye(N))/N;  A=L'*L;
K=eye(N)-diag(ones(N-1,1),-1);  K=K+K';K(N,N)=1;K(1,1)=1;   K=K*(N^2);
D=L'*ones(N,1);
Ku1(1:N,N)=1; Ku1=Ku1-eye(N);

Z=zeros(Modes); I=eye(Modes); I2=eye(2*Modes);  ZZ=zeros(Modes,1);

lambda(1:4)=[1.875104, 4.694091, 7.854757, 10.99554];
for i=5:Modes,    lambda(i)=(i-.5)*pi; end
```

```
for i=1:Modes,  a=lambda(i); %%%C=(cosh(a)+cos(a))/(sinh(a)+sin(a));
    f=(1+exp(-2*a)+2*exp(-a)*cos(a))/(1-exp(-2*a)+2*exp(-a)*sin(a));
    for n=1:N
        xi=(n-.5)/N;
        FI(n,i)=.5*(exp(a*xi)*(1-f)-exp(-a*xi)*(1+f))+sin(a*xi)+f*cos(a*xi);
    end
end

display(['Con G=',num2str(G),'y thetaG=',num2str(thetaG)])
for dd=1:length(UU), u=dd;
%%for dd=1:length(G_), u=dd; G=G_(u)
    teta_eq(1:N,1)=0;

    Angle=teta_eq*ones(1,N);Angle=Angle-Angle';%Angle(i,j)=Angle_i-Angle_j;
    COS=cos(Angle); SIN=sin(Angle);

    M=A.*COS;
    Cu=2*L'.*COS;
    Ku=diag(COS(:,N))*Ku1;
    KG=diag(D.*cos(teta_eq-thetaG ));

    Fc=-SIN(:,N);
    Mf=ones(1,N)*( L.*SIN)/N;
    KGf=sin(teta_eq'-thetaG)/N;
    Frel=-L.*SIN*ones(N,1);

    M_=FI'*M*FI;    Cu_=FI'*Cu*FI;      Ku_=FI'*Ku*FI;
    K_=FI'*K*FI;    KG_=FI'*KG*FI;

    Fc_=FI'*Fc;     Mf_=Mf*FI;          KGf_=KGf*FI;
    Frel_=FI'*Frel;


    %%%%beta_=0;
for bet=1:length(beta_), beta=beta_(bet);
    [bet/length(beta_),u/length(UU)]
    U_=UU(u);
    %%%U_=0;
```

```
    Mass=M_;
    Damping=  (   beta*  U_    *Cu_);
    Stiffness=(G*KG_+K_+beta*( U_^2 )*Ku_);

    AAf=[Z,I,ZZ;...
        -Stiffness,-Damping,-2*beta*U_*Fc_;...
        -beta*G*KGf_,ZZ',-beta*U_];
    MMf=[I,Z,ZZ;...
        Z,Mass,beta*Frel_;...
        ZZ',beta*Mf_,beta];

    MM=[I,Z;Z,Mass];
    AA=[Z,I;-Stiffness,-Damping];

    [VV_1,DD_1]=eig(AA,MM); sigma_1=diag(DD_1);
    U(bet,u)=U_;
    [f1(u,bet),pos]=max(real(sigma_1));
    omega1(u,bet)=sigma_1(pos);

    [VV_1,DD_2]=eig(AAf,MMf); sigma_2=diag(DD_2);
    U(bet,u)=U_;
    [f2(u,bet),pos]=max(real(sigma_2));
    omega2(u,bet)=sigma_1(pos);

    clear Mass MM Damping Stiffness AA VV_1 DD_1 VV_2 DD_2 sigma_1
end
end

[X,YY]=meshgrid(beta_,ones(length(UU),1));

% % figure(1), hold on,contour(X,U',f1,[0 0],'k'),
% % xlabel('$$\beta$$','Interpreter','Latex','FontSize',16)
% % ylabel('$$\textsc{U}$$','Interpreter','Latex','FontSize',16)

save(['EstabPiG',num2str(G),'recto'],'X','U',...
'f1','omega1','f2','omega2','Modes','N','beta_');
```

Para deformadas curvas, se calcula dentro del bucle y se recorre el ángulo del extremo libre.