```
%----------------------Sea wave energy harvesting SWEH---------
------------
%Name: xxxxxxxxxxxxxxxxxxx
%Subject: Mathematical modeling using state-space companion-form
(radiation)
% with excitation forces calculation submodel-regular waves
%-------------------------------------------------
runcounter=2;
if runcounter==1 clear; runcounter=1;
end
%put it 1 in the 1st run only (or when changing wind speed)
%so the radiation coefficients are calculated for only once to
save next
%runs time write_excel=0;
%put it 1 if the results are required to be written in excel
sheet
%specified clc;close all;

%1)Data:
dia=0.8;    %the buoy's diameter (in m) r=dia/2;    %the buoy's
radius (inrtia m) d=6;  %the buoy's draft (in m)
h=20;   %the water depth (in m)

ru=1030;    %density of sea water (in kg/m^3) g=9.81;
%gravitational acceleration (in m/s)

%1.1)mass:
m=8000; %the mass of the buoy (in kg)
%1.2)damping:
n_bext=1; %number of required calculations for different damping
values if n_bext==1
bext=1E3; else
bext=[linspace(0.1E3,1E3,0.5*n_bext)
linspace(1E3,5E3,0.5*n_bext)]; %the external PTO damping (in
N.s/m)
end
%1.3)stiffness:
k_buoy=ru*g*0.25*pi*dia^2; %stiffness_buoyancy (in N/m)
%1.4)CVT mechanism and added stiffness& system inertia k_s=8000;
%spring added to the lever mechanism (in N/m) Rp=0.15; %pinion
radius in CVT system (in m)
Ip=2.12;    %buoy-side sheave rotational inertia (in kg.m^2)
% L=4; %lever length (in m)
%-----------------------------------------------------------------
-----------
%2) spectrum
```

```
U_S=12; %wind speed value for power spectral density calculation
(in km/hr)
alpha=8.1E-3; beta=0.74; %P-M spectrum constants
A=alpha*g^2/(2*pi)^4;  %Bretschenider formula constant
%omega_max=0.8772*g/U_S; %Peak frequency at U_S (in rad/sec)
%-----------------------------------------------------------------
-----------
if runcounter==1
%3) calculations from spectrum
%3.1) Enegry period, significant wave height and, available
power calculations (at wind speed U_S)

SR=sqrt((k_s-Ip*omega_e/Rp^2)/(omega_e^2*(m+Ip/Rp^2)-k_buoy));
%required speed ratio to provide the required equivalent
stiffness
% L1=GR*L/(1+GR);   %pivot to buoy distance for this GR
% L2=L/(1+GR);  %pivot to spring distance for this GR

for n=1:n_bext
zeta(n)=bext(n)/(2*sqrt(k_total*(m+mr_e(2)))); %this is the
damping ratio
%-----------------------------------------------------------------
-----------
%3.7) State space matrices (falnes) a=-(m+mr_e(2));
A=[0 0 a1 0 b1;1 0 a2 0 b2;0 1 a3 0 b3;0 0 0 0 1;0 0 1/a
k_total/a bext(n)/a];
B=[0 0 0 0 -1/a];
%C_disp=[0 0 0 1 0]; %this is the output matrix for displacement
%C_vel=[0 0 0 0 1]; %this is the output matrix for velocity

%-----------------------------------------------------------------
-----------

%3.8) Runge-Kutta solution
h_rk=0.05; % time step for RK-4 calculations (in sec) tmax=400;
% this is the total time span for calculation t=0:h_rk:tmax;
size_t=size(t,2);

x1=zeros(1,size_t); x2=zeros(1,size_t); x3=zeros(1,size_t);
x4=zeros(1,size_t);
x5=zeros(1,size_t); %these are the state variables

p_absorbed=zeros(1,size_t);
p_radiated=zeros(1,size_t); %absorbed and radiated power
```

```
for i=1:size_t-1 if n==1
[u_f1_1(i) u_f3_1(i)
u_f5_1(i)]=ExcitationForce(F_1ext,F_3ext,F_5ext,omega_e,t(i));
%all excitation forces for k1
[u_f1_23(i) u_f3_23(i)
u_f5_23(i)]=ExcitationForce(F_1ext,F_3ext,F_5ext,omega_e,t(i)+0.
5*h_rk);
%all excitation forces for k2,k3 [u_f1_4(i) u_f3_4(i)
u_f5_4(i)]=ExcitationForce(F_1ext,F_3ext,F_5ext,omega_e,t(i)+h_r
k); %all excitation forces for k4
end k1=A*[x1(i);x2(i);x3(i);x4(i);x5(i)]+B*[0;0;0;0;u_f3_1(i)];

k2=A*[x1(i)+0.5*h_rk*k1(1);x2(i)+0.5*h_rk*k1(2);x3(i)+0.5*h_rk*k
1(3);x4(i)+0.
5*h_rk*k1(4);x5(i)+0.5*h_rk*k1(5)]+B*[0;0;0;0;u_f3_23(i)];

k3=A*[x1(i)+0.5*h_rk*k2(1);x2(i)+0.5*h_rk*k2(2);x3(i)+0.5*h_rk*k
2(3);x4(i)+0.
5*h_rk*k2(4);x5(i)+0.5*h_rk*k2(5)]+B*[0;0;0;0;u_f3_23(i)];

k4=A*[x1(i)+h_rk*k3(1);x2(i)+h_rk*k3(2);x3(i)+h_rk*k3(3);x4(i)+h
_rk*k3(4);x5( i)+h_rk*k3(5)]+B*[0;0;0;0;u_f3_4(i)];

x1(i+1)=x1(i)+(1/6)*h_rk*(k1(1)+2*(k2(1)+k3(1))+k4(1));
x2(i+1)=x2(i)+(1/6)*h_rk*(k1(2)+2*(k2(2)+k3(2))+k4(2));
x3(i+1)=x3(i)+(1/6)*h_rk*(k1(3)+2*(k2(3)+k3(3))+k4(3));
x4(i+1)=x4(i)+(1/6)*h_rk*(k1(4)+2*(k2(4)+k3(4))+k4(4)); %buoy's
displacement
x5(i+1)=x5(i)+(1/6)*h_rk*(k1(5)+2*(k2(5)+k3(5))+k4(5)); %buoy's
velocity

end

p_absorbed(i+1)=0.5*x5(i+1)^2*bext(n)/1000; %absorbed power (in
kW) p_radiated(i+1)=0.5*x5(i+1)^2*br_e(1)/1000; %radiated power
(in kW) clear k1 k2 k3 k4

z_amplitude(n)=max(x4); %amplitude of buoy's displacement (in m)
zdot_amplitude(n)=max(x5); %amplitude of buoy's velocity (in m)

p_absorbed_avg(n)=max(p_absorbed); %average absorbed power (in
kW) p_radiated_avg(n)=max(p_radiated); %average radiated power
(in kW)
```

```
lambda(n)=p_absorbed_avg(n)/p_available;     %the capture width
(in m) eff(n)=100*lambda(n)/dia; %the power absorption
efficiency of the buoy end
if n_bext>1
plot(bext/1000,z_amplitude,bext/1000,zdot_amplitude,'.');grid;
line('XData', [bext(1)/1000 bext(n_bext)/1000], 'YData',
[0.5*Hsig 0.5*Hsig], 'LineStyle', '--','LineWidth', 2)
title(['Buoy displacement and velocity average amplitude at wind
speed of ' num2str(U_S) ' m/s']);xlabel('PTO damping coefficient
(kN.s/m)'); ylabel('Displacement & velocity amp (m,m/s)');
legend('Displacement average amplitude','Velocity average
amplitude','H significant/2');
figure;



plot(bext/1000,p_absorbed_avg);grid;
line('XData', [bext(1)/1000 bext(n_bext)/1000], 'YData',
[p_available*dia p_available*dia], 'LineStyle', '--
','LineWidth', 2)
title(['Average absorbed power at wind speed of ' num2str(U_S) '
m/sec']);xlabel('PTO damping coefficient (kN.s/m)');
ylabel('Average absorbed power (kW)');
legend('Average absorbed power','Available power/buoy width');
figure;

subplot(2,1,1);plot(bext/1000,lambda);grid;
title(['Buoy capture width (in m) at wind speed of '
num2str(U_S) ' m/sec' '
,Buoy diameter is ' num2str(dia) ' m' ]);xlabel('PTO damping
coefficient (kN.s/m)'); ylabel('Capture width (m)');
subplot(2,1,2);plot(bext/1000,eff);grid;
title(['Absorption efficiency % at wind speed of ' num2str(U_S)
' m/sec']);xlabel('PTO damping coefficient (kN.s/m)');
ylabel('Efficiency %');

%-----------------------------------------------------------------
-----------
else

plot(t,x4,t,x5,'--');grid;
line('XData', [0 t(end)], 'YData', [0.5*Hsig 0.5*Hsig],
'LineStyle', '-- ','LineWidth', 2)
line('XData', [0 t(end)], 'YData', [-0.5*Hsig -0.5*Hsig],
'LineStyle', '-- ','LineWidth', 2)
title(['Buoy displacement and velocity at wind speed of '
num2str(U_S) ' m/s, ' 'Amp.ratio= '
```

```
num2str(max(x4)/(0.5*Hsig))]);xlabel('Time (sec)');
ylabel('Displacement & velocity amp (m,m/s)');
legend('Displacement ','Velocity ','H significant/2'); figure;

plot(t,p_absorbed);grid;
line('XData', [0 t(end)], 'YData', [p_available*dia
p_available*dia], 'LineStyle', '--','LineWidth', 2)
title(['Absorbed power at wind speed of ' num2str(U_S) ' m/sec'
' eff.= ' num2str(max(p_absorbed)/(p_available*dia)*100)
'%']);xlabel('Time (sec)'); ylabel('Average absorbed power
(kW)');
legend('Absorbed power','Available power/buoy width');
end
```

```
%----------------------Sea wave energy harvesting SWEH---------------
------
%Name: xxxxxxxxxxxxxxxxxxx
%Subject: Excitation force amplitudes calculation (Surge_1,
Heave_3,Pitch_5)
%--------------------------------------------------

function [F_1ext F_3ext
F_5ext]=ExcitationForceAmplitudes(Hsig,omega_e,r,d) g=9.81;
%gravitational acceleration (m/sec^2)
ru=1030; %water density (kg/m^3)
Hsig=0.5*Hsig; %as the Hsig in the folrmula below is the
amplitude not the wave height
k=omega_e^2/g;

xi_step=0.1; %step in the following integrations with d(xi) in
both F_3ext and F_5ext calculation
xi=(0.001:xi_step:5); %xi values to be used in the integrations
with d(xi) in both F_3ext and F_5ext calculation

z_step=0.01; %step in the following integration with d(z) in
F_5ext calculation
z=(0:z_step:d); %z values to be used in the integration with
d(z)in F_5ext calculation

P_0amp=-(g*Hsig/omega_e)*1*sqrt(-
1)*sqrt(2/pi)*(besselj(0,k*r)+((besselj(0,k*r+0.01)-
besselj(0,k*r-
0.01))/0.02)*besselh(0,1,k*r)/((besselh(0,1,k*r+0.01)-
besselh(0,1,k*r- 0.01))/0.02));
P_1amp=-(g*Hsig/omega_e)*2*sqrt(-
1)^2*sqrt(2/pi)*(besselj(1,k*r)+((besselj(1,k*r+0.01)-
besselj(1,k*r-
0.01))/0.02)*besselh(1,1,k*r)/((besselh(1,1,k*r+0.01)-
besselh(1,1,k*r- 0.01))/0.02));
for w=1:size(z,2)
F_5ext_int1(w)=(z(w)-d)*exp(-k*z(w));
end

for w=1:size(xi,2)
P_0(w)=P_0amp*exp(-k*d)*k/(xi(w)^2+k^2); P_1(w)=P_1amp*exp(-
k*d)*k/(xi(w)^2+k^2);
F_3ext_int(w)=P_0(w)*besseli(1,xi(w)*r)/(xi(w)*besseli(0,xi(w)*r
));
F_5ext_int2(w)=P_1(w)*besseli(2,xi(w)*r)/(xi(w)*besseli(1,xi(w)*
r));
```

```
end
F_3ext_int_value=xi_step*(sum(F_3ext_int)-0.5*F_3ext_int(1)-
0.5*F_3ext_int(end));
F_5ext_int1_value=z_step*(sum(F_5ext_int1)-0.5*F_5ext_int1(1)-
0.5*F_5ext_int1(end));
F_5ext_int2_value=xi_step*(sum(F_5ext_int2)-0.5*F_5ext_int2(1)-
0.5*F_5ext_int2(end));
%above are the values of integrations (d(xi)) using multiple
trapezoidal rule

F_1ext=-(2*pi*sqrt(-1)*ru*g*Hsig*r/k)*(besselj(1,k*r)-
((besselj(1,k*r+0.01)-besselj(1,k*r-
0.01))/0.02)*besselh(1,1,k*r)/((besselh(1,1,k*r+0.01)-
besselh(1,1,k*r- 0.01))/0.02))*(1-exp(-k*d));
F_3ext=-(2*pi*sqrt(-
1)*ru*omega_e*r*sqrt(2/pi))*F_3ext_int_value;

F_5ext=-(2*pi*sqrt(-1)*ru*g*Hsig*r)*(besselj(1,k*r)-
((besselj(1,k*r+0.01)- besselj(1,k*r-
0.01))/0.02)*besselh(1,1,k*r)/((besselh(1,1,k*r+0.01)-
besselh(1,1,k*r-0.01))/0.02))*F_5ext_int1_value+(pi*sqrt(-
1)*ru*omega_e*r^2*sqrt(2/pi))*F_5ext_int2_value;
%   F_1extm=abs(F_1ext);
%   F_3extm=abs(F_3ext);
%   F_5extm=abs(F_5ext); end
```

```
%----------------------Sea wave energy harvesting SWEH---------
------------
%Name: xxxxxxxxxxxxxxxxxxx
%Subject: Q function minimization for state space matrices
coefficients calculation
%-----------------------------------------------------
function Q = Qfunction(x,br,mr,omega,n_omega)

Q=0;
m=51; % this is the number of summations (Q)
delta_t=0.1; % in sec, this is the time step for summation Q for
i=1:m
tk(i)=(i-1)*delta_t; for k=1:n_omega
fun_he(k)=br(k)*cos(omega(k)*tk(i)); fun_ho(k)=omega(k)*(mr(k)-
mr(end))*sin(omega(k)*tk(i));
end
he(i)=(1/pi)*(omega(end)-omega(1)/n_omega)*(sum(fun_he)-
0.5*fun_he(1)- 0.5*fun_he(end));
%h0(i)=(-1/pi)*(omega(end)-omega(1)/n_omega)*(sum(fun_ho)-
0.5*fun_ho(1)-0.5*fun_ho(end));
hr(i)=2*he(i);
%hr(i)=2*h0(i);
Q=Q+(hr(i)-[0 0 1]*exp([0 0 x(1);1 0 x(2);0 1
x(3)]*tk(i))*[x(4);x(5);x(6)])^2; end
end
```

```
%----------------------Sea wave energy harvesting SWEH---------
------------
%Name: xxxxxxxxxxxxxxxxxxx
%Subject: Radiation coefficients calculation
%-------------------------------------------------
function [mr br]=radiationcoeffs_fun(dia,d,h,omega,n_omega) for
w=1:n_omega
%1)Data:
r=dia/2;    %the buoy's radius (in m) ru=1030;  %density of sea
water (in kg/m^3)
g=9.81; %gravitational acceleration (in m/s)

%2)Data for calculations: Np=50;
Nn=50;  %radiation coefficients formula number of summations

%-----------------------------------------------------------------
----
% calculation of s0,sj:

x=linspace(0,50,20000); for i=1:20000
f1(i)=omega(w)^2-g*x(i)*tanh(x(i)*h);
f2(i)=omega(w)^2+g*x(i)*tan(x(i)*h);
end D1=find(abs(diff(sign(f1)))==2);
D2=find(abs(diff(sign(f2)))==2); s0=x(D1);
sj=x(D2);

%-----------------------------------------------------------------
----
%calculation of L0s0, Lnsj, kn:

Ns0=0.5*(1+sinh(2*s0*h)/(2*s0*h));
L0s0=(Ns0^(-0.5)*(h-d)*s0*sinh(s0*(h-d)))/((h-d)^2*s0^2);

%+++++important note:
%counters can't be zero and both "j" and "l" are changing from 0
to Np so
%the counter "j=1" is reserved for the value "j=0" hence for
loops are
%begining from "j=2"

for j=2:Np+1
Nsj(j)=0.5*(1+sin(2*sj(j)*h)/(2*sj(j)*h)); L0sj(j)=(Nsj(j)^(-
0.5)*(h-d)*sj(j)*sin(sj(j)*(h-d)))/((h-
d)^2*sj(j)^2); %used in gamma calculations & final calculations
end
```

```
L0sj(1)=L0s0; %used in gamma calculations & final calculations
Nsj(1)=Ns0;
%the 1st term of L0sj must be calculated separately as shown
above as it
%has a different rule cointaining "s0" instead of "sj" and
"sinh" function
%instead of "sin"

for n=1:Nn
kn(n)=(n)*pi/(h-d);
I0(n)=besseli(0,kn(n)*r); %I_0(kn*r) mod. bessel of 1st kind
(zero order) I1(n)=besseli(1,kn(n)*r); %I_1(kn*r) mod. bessel of
1st kind (1st order)

for j=1:Np+1
if j>1
Lnsj(n,j)=((-1)^(n)*Nsj(j)^(-0.5)*(h-d)*sj(j)*sin(sj(j)*(h-
d)))/((h- d)^2*sj(j)^2-(n)^2*pi^2);%used in gamma calculations &
final calculations
else
Lnsj(n,1)=((-1)^(n)*Ns0^(-0.5)*(h-d)*s0*sinh(s0*(h-d)))/((h-
d)^2*s0^2+(n)^2*pi^2); %this is Lns0
end
end
end

 %-----------------------------------------------------------------
-----
%calculation of gamma0j:

for j=2:Np+1
k0(j)=besselk(0,sj(j)*r); %k_0(sj*r) mod. bessel of 2nd kind
k0dot(j)=(besselk(0,1.01*sj(j)*r)-
besselk(0,0.99*sj(j)*r))/(2*0.01*sj(j)*r); %k_0 dot(sj*r) 1st
derivative of mod. bessel of 2nd kind (using central distance
differentiation)

g0j(j)=sj(j)*r*k0dot(j)/k0(j); %g0j for epsilon_lj calculation
% Zsj(j)=Nsj(j)^(-0.5)*cos(sj(j)*(-d+h));   %Zsj(z) where -
d<=z<=0 end
%to calculate g0j(1) [@j=0]:
H0=besselh(0,1,s0*r); %Hankel function of zero order and [k=1
(superscript)] H0dot=(besselh(0,1,1.01*s0*r)-
besselh(0,1,0.99*s0*r))/(2*0.01*s0*r);    %1st derivative of
Hankel function
g0j(1)=s0*r*H0dot/H0;
```

```
g00=-r^2/(2*(h-d)^2);
%Zsj(1)=Ns0^(-0.5)*cosh(s0*(-d+h)); %this is Zs0 for n=1:Nn
I0dot(n)=(besseli(0,1.01*kn(n)*r)-
besseli(0,0.99*kn(n)*r))/(2*0.01*kn(n)*r); %I_0 dot(kn*r) 1st
derivative of mod. bessel of 1st kind (using central distance
differentiation)
G0n(n)=-kn(n)*r*I0dot(n)/I0(n); %Gon
end
Delta=eye(Np+1,Np+1); %this is the Kronecker delta

%x0l_sum=zeros(Np+1);
%epslj_sum=zeros(Np+1); for l=2:Np+1
x0l_sum=0; for n=1:Nn
x0l_sum=x0l_sum+2*(-1)^n/(n^2*pi^2)*G0n(n)*Lnsj(n,l); end
x0l(l)=g00*L0sj(l)+x0l_sum; %this calculates the vector x_0l for
j=2:Np+1
epslj_sum=0; for n=1:Nn
epslj_sum=epslj_sum+G0n(n)*Lnsj(n,l)*Lnsj(n,j); end
epslj(l,j)=h/(h-d)*g0j(j)*Delta(l,j)+2*epslj_sum; %this
calculates the elements of epsilon_lj matrix
end
end

%this is the 1st element in x0l vector x0l_sum=0;
for n=1:Nn
Lns0(n)=((-1)^n*Ns0^(-0.5)*(h-d)*s0*sinh(s0*(h-d)))/((h-
d)^2*s0^2+n^2*pi^2);
x0l_sum=x0l_sum+2*(-1)^n/(n^2*pi^2)*G0n(n)*Lns0(n);
end x0l(1)=g00*L0sj(1)+x0l_sum;

%this is for the 1st row    in epslj matrix (l=1) for j=1:Np+1
epslj_sum=0; for n=1:Nn
if j>1 epslj_sum=epslj_sum+G0n(n)*Lns0(n)*Lnsj(n,j); else
epslj_sum=epslj_sum+G0n(n)*Lns0(n)*Lns0(n); end
end
end
epslj(1,j)=h/(h-d)*g0j(j)*Delta(1,j)+2*epslj_sum;
%this is for the 1st column in epslj matrix (j=1) for l=1:Np+1
epslj_sum=0; for n=1:Nn
if l>1 epslj_sum=epslj_sum+G0n(n)*Lnsj(n,l)*Lns0(n); else
epslj_sum=epslj_sum+G0n(n)*Lns0(n)*Lns0(n); end
end
end
```

```
epslj(l,1)=h/(h-d)*g0j(1)*Delta(l,1)+2*epslj_sum;

gamma0j=epslj^-1*transpose(x0l); %gamma0j which is complex
number

%-----------------------------------------------------------------
----
%calculation of added mass and hydrodynamic damping:

radsum1=0; %this is for the 1st summation in rad. coeffs.
equation radsum2=0; %this is for the 2nd summation in rad.
coeffs. equation

for j=1:Np+1
radsum1=radsum1+gamma0j(j)*L0sj(j); rad11(j)=radsum1;
end

radsumsum2=zeros(Nn); for n=1:Nn
for j=1:Np+1
radsumsum2=zeros(Nn);
radsumsum2(n)=radsumsum2(n)+gamma0j(j)*Lnsj(n,j)-(-
1)^n/(n^2*pi^2);
%this is for the inner summation of the 2nd summation in rad.
end
%coeffs. equation
end

radsum2=radsum2+(-1)^n*I1(n)/(n*I0(n))*radsumsum2(n);

rad=(1/3)+(1/8)*(r/(h-d))^2+radsum1+(4*(h-d)/(pi*r))*radsum2;
%rad is the complex value for radiation coeffs. (added mass &
hydrodynamic
%damping)

mr(w)=real(rad)*pi*r^2*(h-d)*ru;    %mr is the added mass
br(w)=imag(rad)*pi*r^2*(h-d)*ru*omega(w); %br is the
hydrodynamic damping

end end
```

```
%----------------------Sea wave energy harvesting SWEH---------
------------
%Name: xxxxxxxxxxxxxxxxxxx
%Subject: Mathematical modeling using state-space companion-form
(radiation)
% with excitation forces calculation submodel-irregular waves
%-------------------------------------------------
runcounter=1;
if runcounter==1 clear; runcounter=1;
end
%put it 1 in the 1st run only (or when changing wind speed)
%so the radiation coefficients are calculated for only once to
save next
%runs time write_excel=0;
%put it 1 if the results are required to be written in excel
sheet specified ddt=1;
clc;close all;

%1)Data:
dia=0.8;    %the buoy's diameter (in m) r=dia/2;    %the buoy's
radius (inrtia m) d=6;  %the buoy's draft (in m)
h=20;   %the water depth (in m)

ru=1030;    %density of sea water (in kg/m^3) g=9.81;
%gravitational acceleration (in m/s)

%1.1)mass:
m=8000; %the mass of the buoy (in kg)
%1.2)damping:
n_bext=40; %number of required calculations for different
damping values if n_bext==1
bext=4E3; else
bext=[linspace(0.1E3,2E3,0.5*n_bext)
linspace(2E3,20E3,0.5*n_bext)];  %the external PTO damping (in
N.s/m)
end
%1.3)stiffness:
k_buoy=ru*g*0.25*pi*dia^2; %stiffness_buoyancy (in N/m)
%1.4)CVT mechanism and added stiffness& system inertia k_s=8000;
%spring added to the lever mechanism (in N/m) Rp=0.15; %pinion
radius in CVT system (in m)
Ip=2.12;    %buoy-side sheave rotational inertia (in kg.m^2)
% L=4; %lever length (in m)
%-------------------------------------------------------------
-----------
%2) spectrum
```

```
U_S=10; %wind speed value for power spectral density calculation
(in km/hr)
alpha=8.1E-3; beta=0.74; %P-M spectrum constants
A=alpha*g^2/(2*pi)^4;  %Bretschenider formula constant
%omega_max=0.8772*g/U_S; %Peak frequency at U_S (in rad/sec)
%-----------------------------------------------------------------
-----------
if runcounter==1
%3) calculations from spectrum
%3.1) Enegry period, significant wave height and, available
power calculations (at wind speed U_S)

B=0.7401*(g/(2*pi*U_S))^4; %Bretschenider formula constant
m0=A/(4*B);  %zeroth spectral moment m1=1.2254*A/(4*B^0.75);
%1st spectral moment m2=1.77245*A/(4*B^0.5); %2nd spectral
moment Hsig=4*sqrt(m0);  %significant wave height Tz=0.7511*B^(-
0.25); %zero crossing period
Te=1.12*Tz; %energy period
omega_e=(1/Te)*2*pi; %energy frequency (in rad/sec)
p_available=0.5*Hsig^2*Te; %power per unit width of wavefront
(in kW)
%-----------------------------------------------------------------
-----------
%3.2) Instantaneous wave height and excitation force calculation

h_rk=0.1; % time step for RK-4 calculations (in sec) tmax=600;
% this is the total time span for calculation t=0:h_rk:tmax;
size_t=size(t,2);
t_half=0:0.5*h_rk:tmax; size_t_half=size(t_half,2); %used for
amplitude and force calculation at double steps of time "t"

fn_spectrum_max=1; % (Hz) the maximum frequency value at which
the spectrum is nearly zero valued beyond
N_spectrum=100; %the number of divisions for the frequency range
of the spectrum
n_spectrum=linspace(1,N_spectrum-1,N_spectrum-1); %the frequency
divisions counter (-1) is to avoid the ln(1)=0 in the
fn_spectrum calculation
%fn_spectrum=(B./log(N_spectrum./n_spectrum)).^0.25; % (Hz) the
frequency division which gives non-periodic results
fn_spectrum=rand(1,N_spectrum-1)*fn_spectrum_max;
fn_spectrum=sort(fn_spectrum, 'ascend');
omegan_spectrum=fn_spectrum.*2*pi; %(rad/sec) the frequency
division which gives non-periodic results

% for i=1:N_spectrum-1
```

```
%   Sn(i)=16*(alpha*g^2/omegan_spectrum(i)^5)*exp(-
beta*(g/(U_S*omegan_spectrum(i)))^4); %power spectral density
% end
for i=2:N_spectrum-1
omegan_spectrum_hat(i)=(omegan_spectrum(i)+omegan_spectrum(i-
1))/2; Sn_hat(i)=(alpha*g^2/omegan_spectrum_hat(i)^5)*exp(-
beta*(g/(U_S*omegan_spectrum_hat(i)))^4)*2*16; %power spectral
density


end H_inst=zeros(1,size_t_half);
H_in=zeros(size_t_half,N_spectrum-1);
F_3ext_in_amp=zeros(size_t_half,N_spectrum-1);
F_3ext_inst=zeros(1,size_t_half);


for jn=2:N_spectrum-1


H_in_amp(jn)=sqrt(Sn_hat(jn)*(omegan_spectrum(jn)-
omegan_spectrum(jn-1)));
%these are the amplitudes of waves being superpositioned to make
the wave at certain time


F_3ext_in_amp(jn)=ExcitationForceAmplitudes(H_in_amp(jn),omegan_
spectrum(jn), r,d);


%these are the amplitudes of wave forces being superpositioned
to make the total excitation force at certain time


end
phin_spectrum=rand(1,N_spectrum-1)*2*pi; %random values for
phase value for in=1:size_t_half


for jn=2:N_spectrum-1
%H_in(in,jn)=sqrt(abs(Sn(jn)-Sn(jn-
1))/2)*cos(0.5*(omegan_spectrum(jn)+omegan_spectrum(jn-
1))*t_half(in)- phin_spectrum(jn));
H_in(in,jn)=H_in_amp(jn)*cos(omegan_spectrum_hat(jn)*t_half(in)-
phin_spectrum(jn));
%these are the combination of waves being superpositioned to
make the wave at certain time


H_inst(in)=H_inst(in)+H_in(in,jn);
%the superpositioned wave height at certain time
F_3ext_inst(in)=F_3ext_inst(in)+real(F_3ext_in_amp(jn)*exp(-
sqrt(-
1)*(omegan_spectrum(jn)*t_half(in))));
```

```
%the superpositioned wave excitation force at certain time

end
%clear phin_spectrum clc;
display(['loop# ' num2str(in) ' out of ' num2str(size_t_half) '-
-' num2str(in*100/size_t_half) ' %' ])
end

% sort_H_inst=sort(H_inst, 'descend'); %to sort the H_inst
descendingly
% H_onethird=mean(sort_H_inst(1:round(size_t_half/3))); %to
calculate the significannt wave height by the one-third maximum
method
H_onethird=rms(H_inst,h_rk/2,t_half(end),max(size(t_half)));
display(['H_sig (spectrum calcuations)/H_onethird (time domain
output)= ' num2str(Hsig/H_onethird)])
sort_F_3ext_inst=sort(F_3ext_inst, 'descend'); %to sort the
F_3ext_inst descendingly
F_3ext_inst_onethird=mean(sort_F_3ext_inst(1:round(size_t_half/3
)));    %to calculate the significannt wave height by the one-
third maximum method
% %3.2) Excitation forces amplitude calculation: (complex
amplitudes)
% [F_1ext F_3ext
F_5ext]=ExcitationForceAmplitudes(Hsig,omega_e,r,d);
%-----------------------------------------------------------------
-----------
%3.3) Radiation coefficients n_omega=20;
omega=linspace(0.1,10,n_omega); %these frequencies are used in
the function calls below

[mr br]=radiationcoeffs_fun(dia2,d,h,omega,n_omega);
%this is a function call for radiation coefficients calculations
%radiation coefficients are calculated for several frequencies
to be used
%in Q function minimization to get the coefficients of state-
space matrices
%-----------------------------------------------------------------
-----------
%3.4) Radiation coefficients at the energy frequency

n_omega1=2; % as it's only one frequency (energy frequency)+ one
frequency representing infinity
omega1=[omega_e 10]; % omega= 10 rad/sec can be used as the
frequency at infinity
```

```
% as after 10 rad/sec, the coefficients becomes stable and don't
change [mr_e
br_e]=radiationcoeffs_fun(dia2,d,h,omega1,n_omega1);
%-----------------------------------------------------------------
-----------
%3.5) Q function minimization using the previously calculated
rad coeffs in (3.2)
ObjectiveFunction = @(x) Qfunction(x,br,mr,omega,n_omega);
X0 = [-1 -15 15 30 -40 30]; %these are initial values for the
pattern search algorithm
[x,fval] = patternsearch(ObjectiveFunction,X0);

a1=x(1); a2=x(2); a3=x(3); %Ap matrix coefficients b1=x(4);
b2=x(5); b3=x(6); %Bp matric coefficients
%-----------------------------------------------------------------
-----------
end
% H_inst_smooth=smooth(H_inst,'rlowess');
% F_3ext_inst_smooth=smooth(F_3ext_inst,'rlowess');

%3.6) Calculation of total equivalent stiffness for buoy and
lever ratio k_total=(m+mr_e(1))*(1*omega_e)^2; %total system
stiffness required to obtain resonance

k_eq=k_total-k_buoy; %equivalent spring stiffness required to
obtain resonance
if k_eq<0
disp('#### WARNING: Tuning for natural frequency at this wind
speed is not possible');
beep() end
%means that the frequency required is less than the minumum
possible
%natural frequency of the system

SR=sqrt((k_s-Ip*omega_e/Rp^2)/(omega_e^2*(m+Ip/Rp^2)-k_buoy));
%required for n=1:n_bext
zeta(n)=bext(n)/(2*sqrt(k_total*(m+mr_e(2)))); %this is the
damping ratio
%-----------------------------------------------------------------
-----------
%3.7) State space matrices (falnes) a=-(m+mr_e(2));
A=[0 0 a1 0 b1;1 0 a2 0 b2;0 1 a3 0 b3;0 0 0 0 1;0 0 1/a
k_total/a bext(n)/a];
B=[0 0 0 0 -1/a];
```

```
%-------------------------------------------------------------------
-----------

%3.8) Runge-Kutta solution

% time step for RK-4 calculations (in sec) {refer to 3.2}
% this is the total time span for calculation {refer to 3.2}

x1=zeros(1,size_t); x2=zeros(1,size_t); x3=zeros(1,size_t);
x4=zeros(1,size_t);
x5=zeros(1,size_t); %these are the state variables

p_absorbed=zeros(1,size_t);
p_radiated=zeros(1,size_t); %absorbed and radiated power


for i=1:size_t-1 if n==1

u_f3_1(i) =F_3ext_inst(2*i-1); %heave excitation forces for k1
u_f3_23(i) =F_3ext_inst(2*i);    %heave excitation forces for
k2,k3 u_f3_4(i) =F_3ext_inst(2*i+1);   %heave excitation forces
for k4

%the counters are (2*i-1),(2*i),(2*i+1) as the heave force are
%calculated at double the main simulation time steps (refer to
3.2) end

k1=A*[x1(i);x2(i);x3(i);x4(i);x5(i)]+B*[0;0;0;0;u_f3_1(i)];

k2=A*[x1(i)+0.5*h_rk*k1(1);x2(i)+0.5*h_rk*k1(2);x3(i)+0.5*h_rk*k
1(3);x4(i)+0.
5*h_rk*k1(4);x5(i)+0.5*h_rk*k1(5)]+B*[0;0;0;0;u_f3_23(i)];

k3=A*[x1(i)+0.5*h_rk*k2(1);x2(i)+0.5*h_rk*k2(2);x3(i)+0.5*h_rk*k
2(3);x4(i)+0.
5*h_rk*k2(4);x5(i)+0.5*h_rk*k2(5)]+B*[0;0;0;0;u_f3_23(i)];

k4=A*[x1(i)+h_rk*k3(1);x2(i)+h_rk*k3(2);x3(i)+h_rk*k3(3);x4(i)+h
_rk*k3(4);x5( i)+h_rk*k3(5)]+B*[0;0;0;0;u_f3_4(i)];

x1(i+1)=x1(i)+(1/6)*h_rk*(k1(1)+2*(k2(1)+k3(1))+k4(1));
x2(i+1)=x2(i)+(1/6)*h_rk*(k1(2)+2*(k2(2)+k3(2))+k4(2));
x3(i+1)=x3(i)+(1/6)*h_rk*(k1(3)+2*(k2(3)+k3(3))+k4(3));
```

```
x4(i+1)=x4(i)+(1/6)*h_rk*(k1(4)+2*(k2(4)+k3(4))+k4(4)); %buoy's
displacement
x5(i+1)=x5(i)+(1/6)*h_rk*(k1(5)+2*(k2(5)+k3(5))+k4(5)); %buoy's
velocity

end

p_absorbed(i+1)=0.5*x5(i+1)^2*bext(n)/1000; %absorbed power (in
kW) p_radiated(i+1)=0.5*x5(i+1)^2*br_e(1)/1000; %radiated power
(in kW) clear k1 k2 k3 k4

z_amplitude(n)=rms(x4,h_rk,t(end),max(size(t))); %amplitude of
buoy's displacement (in m)
zdot_amplitude(n)=rms(x5,h_rk,t(end),max(size(t))); %amplitude
of buoy's velocity (in m)

p_absorbed_avg(n)=rms(p_absorbed,h_rk,t(end),max(size(t)));
%average absorbed power (in kW)
p_radiated_avg(n)=rms(p_radiated,h_rk,t(end),max(size(t)));
%average radiated power (in kW)
lambda(n)=p_absorbed_avg(n)/p_available;    %the capture width
(in m) eff(n)=100*lambda(n)/dia; %the power absorption
efficiency of the buoy end

if n_bext>1
plot(bext/1000,z_amplitude,bext/1000,zdot_amplitude,'.');grid;
line('XData', [bext(1)/1000 bext(n_bext)/1000], 'YData',
[0.5*Hsig 0.5*Hsig], 'LineStyle', '--','LineWidth', 2)
title(['Buoy displacement and velocity RMS amplitude at wind
speed of ' num2str(U_S) ' m/s']);xlabel('PTO damping coefficient
(kN.s/m)'); ylabel('Displacement & velocity amp (m,m/s)');
legend('Displacement average amplitude','Velocity average
amplitude','H significant/2');
figure;



plot(bext/1000,p_absorbed_avg);grid;
line('XData', [bext(1)/1000 bext(n_bext)/1000], 'YData',
[p_available*dia p_available*dia], 'LineStyle', '--
','LineWidth', 2)
title(['Average absorbed power RMS at wind speed of '
num2str(U_S) ' m/sec']);xlabel('PTO damping coefficient
(kN.s/m)'); ylabel('Average absorbed power (kW)');
legend('Average absorbed power','Available power*buoy width');
figure;
```

```
subplot(2,1,1);plot(bext/1000,lambda);grid;
title(['Buoy capture width (in m) RMS at wind speed of '
num2str(U_S) ' m/sec' ' ,Buoy diameter is ' num2str(dia) ' m'
]);xlabel('PTO damping coefficient (kN.s/m)'); ylabel('Capture
width (m)'); subplot(2,1,2);plot(bext/1000,eff);grid;
title(['Absorption efficiency % RMS at wind speed of '
num2str(U_S) ' m/sec']);xlabel('PTO damping coefficient
(kN.s/m)'); ylabel('Efficiency %');

%-----------------------------------------------------------------
-----------
else
plot(t,x4,t,x5,'--');grid;
line('XData', [0 t(end)], 'YData', [0.5*Hsig 0.5*Hsig],
'LineStyle', '-- ','LineWidth', 2)
line('XData', [0 t(end)], 'YData', [-0.5*Hsig -0.5*Hsig],
'LineStyle', '-- ','LineWidth', 2)
title(['Buoy displacement and velocity at wind speed of '
num2str(U_S) ' m/s']);xlabel('Time (sec)'); ylabel('Displacement
& velocity amp (m,m/s)'); legend('Displacement ','Velocity ','H
significant/2');
figure;

subplot(3,1,1);plot(t,x4);grid;
title(['Buoy displacement (in m) at wind speed of ' num2str(U_S)
' m/sec' '
,Buoy diameter is ' num2str(dia) ' m' ]);xlabel('Time (Sec)');
ylabel('Buoy displacement (m)');
subplot(3,1,2);plot(t_half,0.5*H_inst);grid;
title(['Instantaneous wave amplitude at wind speed of '
num2str(U_S) ' m/sec']);xlabel('Time (Sec)'); ylabel('Inst. wave
amp. (m)');
line('XData', [0 t_half(end)], 'YData', [0.5*Hsig 0.5*Hsig],
'LineStyle', '-- ','LineWidth', 1)
line('XData', [0 t_half(end)], 'YData', [-0.5*Hsig -0.5*Hsig],
'LineStyle', '--','LineWidth', 1)
subplot(3,1,3);plot(t_half,F_3ext_inst/1000);grid;
title(['Instantaneous heave excitation force at wind speed of '
num2str(U_S) ' m/sec']);xlabel('Time (Sec)'); ylabel('Inst.
Heave excitation force (kN)');

figure;

plot(t,p_absorbed);grid;
```

```
line('XData', [0 t(end)], 'YData', [p_available*dia
p_available*dia], 'LineStyle', '--','LineWidth', 2)
title(['Absorbed power at wind speed of ' num2str(U_S) '
m/sec']);xlabel('Time (sec)'); ylabel('Average absorbed power
(kW)'); legend('Absorbed power','Available power/buoy width');
end

function [xmax,imax,xmin,imin] = extrema(x)

xmax = [];
imax = [];
xmin = [];
imin = [];

% Vector input? Nt = numel(x);
if Nt ~= length(x)
error('Entry must be a vector.') end

% NaN's:
inan = find(isnan(x)); indx = 1:Nt;
if ~isempty(inan) indx(inan) = [];
x(inan) = []; Nt = length(x);
end

% Difference between subsequent elements: dx = diff(x);

% Is an horizontal line? if ~any(dx)
return end

% Flat peaks? Put the middle element:
a = find(dx~=0);     % Indexes where x changes
lm = find(diff(a)~=1) + 1;  % Indexes where a do not changes
d = a(lm) - a(lm-1);     % Number of elements in the flat peak
a(lm) = a(lm) - floor(d/2);    % Save middle elements
a(end+1) = Nt;

% Peaks?
xa  = x(a); % Serie without flat peaks
b = (diff(xa) > 0); % 1 =>  positive slopes (minima begin)
% 0 =>  negative slopes (maxima begin) xb   = diff(b);  % -1 =>
maxima indexes (but one)
% +1 => minima indexes (but one) imax = find(xb == -1) + 1; %
maxima indexes
imin = find(xb == +1) + 1; % minima indexes imax = a(imax);
imin = a(imin);
```

```
nmaxi = length(imax); nmini = length(imin);

% Maximum or minumim on a flat peak at the ends? if (nmaxi==0)
&& (nmini==0)
if x(1) > x(Nt) xmax = x(1);

imax = indx(1); xmin = x(Nt); imin = indx(Nt);
elseif x(1) < x(Nt) xmax = x(Nt);
imax = indx(Nt); xmin = x(1); imin = indx(1);
end return
end

% Maximum or minumim at the ends? if (nmaxi==0)
imax(1:2) = [1 Nt]; elseif (nmini==0)
imin(1:2) = [1 Nt]; else
if imax(1) < imin(1) imin(2:nmini+1) = imin; imin(1) = 1;
else
imax(2:nmaxi+1) = imax; imax(1) = 1;
end
if imax(end) > imin(end) imin(end+1) = Nt;
else
imax(end+1) = Nt; end
end
xmax = x(imax); xmin = x(imin);

% NaN's:
if ~isempty(inan) imax = indx(imax); imin = indx(imin);
end

% Same size as x:
imax = reshape(imax,size(xmax)); imin =
reshape(imin,size(xmin));

% Descending order:
[temp,inmax] = sort(-xmax); clear temp xmax = xmax(inmax);
imax = imax(inmax); [xmin,inmin] = sort(xmin); imin =
imin(inmin);
```