```
function Turbina() % Main Program
      clc,clear all,close all
      tic
      global MaxIter residual relax BladeElemNumber Power GlobalEfficiency CTl miu OmegaRPM
      global TipSpeedRatio LocalSpeedRatio BladeNumber Sigma r_R c_R a a_ fi beta Radius c_R_Burton
      global CpReq U rho Omega ClDesign alfaDesign TwistIdeal Cl Cd Ct Cn alfa c_R_pratico
      global Cd_parametrico Cl_parametrico Eficiencia_parametrico AirfoilList AirfoilBladeElement

      %Lista de perfis utilizados
      AirfoilList={'FX63137sm','NACA4415','SD2030','SG6042'};
      %Carregar ficheiros com dados aerodinamicos parametrizados da lista de perfis
      load Eficiencia_parametrico.dat
      load Cl_parametrico.dat
      load Cd_parametrico.dat

      %Convergence criteria
      MaxIter=1000;
      residual=1e-4;
      relax=0.5; %Relaxation factor
      %Design Requirements
      BladeElemNumber=20;
      Power=1000; % Project requirement
      GlobalEfficiency=0.8;
      rho=1.225;
      miu=1.8e-5;
      TipSpeedRatio=6;
```

```
        BladeNumber=3;
        CpReq=0.45; % Recomended value from literature [20]
        U=10;
        alfaDesign=6*pi/180;
        ClDesign=1.1;

        IdealRotorWithoutWakeRotation()
        PraticalBladeShape(0.8)
        c_R=c_R_Burton;
        IdealRotorWithWakeRotation(true,true)
        toc
end

function IdealRotorWithoutWakeRotation()
        global BladeElemNumber Power GlobalEfficiency AirfoilBladeElement
        global TipSpeedRatio LocalSpeedRatio BladeNumber r_R c_R fi Radius beta
        global CpReq U rho Omega ClDesign alfaDesign TwistIdeal RadiusRoot
        Radius=sqrt(2*Power/(CpReq*GlobalEfficiency*rho*pi*U^3));
        RadiusRoot=Radius*0.15;
        r_R=linspace(RadiusRoot/Radius,1,BladeElemNumber);
        Omega=TipSpeedRatio*U/Radius
        LocalSpeedRatio=Omega*r_R*Radius/U;
        fi=(2/3)*atan(1./LocalSpeedRatio);
        beta=fi-alfaDesign;
        c_R=8*pi.*r_R.*sin(fi)./(3*BladeNumber*ClDesign.*LocalSpeedRatio);
        TwistIdeal=beta-beta(BladeElemNumber);
```

```
      figure(1),plot(r_R,fi*180/pi),xlabel('r/R'),ylabel('fi[º]'),grid on
      title('Ideal Rotor Without Wake Rotation'),hold on
      legenda=sprintf('Tip Speed Ratio = %d',TipSpeedRatio);
      legend(legenda)
      % figure(2),plot(r_R,c_R),xlabel('r/R'),ylabel('C/R'),grid on
      % title('Ideal Rotor Without Wake Rotation'),hold on,
      legenda=sprintf('Tip Speed Rotaion = %d',TipSpeedRatio);
      legend(legenda)
      AirfoilBladeElement=DeclareElementBladeAirfoil('given')
      fid=fopen('nometabela.dat','w');
      fprintf(fid,'r_R\tc_R\tLocalSpeedRatio\tFi\tBeta\n\n');

      for(i=1:1:length(r_R));
      fprintf(fid,'%.5f\t%.5f\t%.5f\t%.5f\t%.5f\n',r_R(i),c_R(i),LocalSpeedRatio(i),fi(i)*180/pi,beta(i)*180/pi);
      end
      fclose(fid);
end

function PraticalBladeShape(localRelativo)
      global BladeElemNumber r_R c_R fi TipSpeedRatio ClDesign BladeNumber c_R_Burton c_R_pratico
      for i=1:1:BladeElemNumber
            if(r_R(i)>=localRelativo)
                  break
            end
      end
```

```
      c_R=8*pi.*r_R.*(1-cos(fi))./(BladeNumber*ClDesign);
      if(i==BladeElemNumber)
            declive=(c_R(i)-c_R(i-1))/(r_R(i)-r_R(i-1));
      else
            declive=((c_R(i+1)-c_R(i-1))/(r_R(i+1)-r_R(i-1)));
      end
      ordenadaOrigem=c_R(i)-declive*r_R(i);
      c_R_pratico=declive*r_R+ordenadaOrigem;
      c_R_Burton=8/(9*TipSpeedRatio*localRelativo)*(2.-
      ((TipSpeedRatio*r_R)/(TipSpeedRatio*localRelativo)))*(2*pi/(ClDesign*
      TipSpeedRatio*BladeNumber));
      figure,plot(r_R,c_R,':r'),xlabel('r/R'),ylabel('C/R'),grid on
      title('Chord distribution'),hold on
      plot(r_R,c_R_pratico,'b'),xlabel('r/R'),ylabel('c/R pratico'),grid on
      plot(r_R,c_R_Burton,'k')
      legend('c/R Ideal','c/R Burton')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function IdealRotorWithWakeRotation(PerdaPonta,IncluirCd)
      global MaxIter residual relax BladeElemNumber Power GlobalEfficiency
      global CTl VelRelativa miu OmegaRPM RadiusRoot VelTangencial CpReq
      global TipSpeedRatio LocalSpeedRatio BladeNumber Sigma r_R c_R fi beta
      global Radius AirfoilBladeElement alfaDesign ClDesign
      global U rho Omega Cl Cd Ct Cn alfa VelAxial Reynolds
```

```
AirfoilBladeElement=DeclareElementBladeAirfoil('given')
%=====================================================================
aOld=ones(1,BladeElemNumber)*0.0;
aNew=zeros(1,BladeElemNumber);
a_Old=ones(1,BladeElemNumber)*0.0;
a_New=zeros(1,BladeElemNumber);
Cl=zeros(1,BladeElemNumber);
Cd=zeros(1,BladeElemNumber);
Ct=zeros(1,BladeElemNumber);
Cn=zeros(1,BladeElemNumber);
alfa=zeros(1,BladeElemNumber);
CTl=zeros(1,BladeElemNumber);
dT=zeros(1,BladeElemNumber);
dQ=zeros(1,BladeElemNumber);
VelRelativa=zeros(1,BladeElemNumber);
VelAxial=zeros(1,BladeElemNumber);
VelTangencial=zeros(1,BladeElemNumber);
Reynolds=zeros(1,BladeElemNumber);
FactorPrandtl=ones(1,BladeElemNumber);
% r_R=linspace(RadiusRoot/Radius,1,BladeElemNumber);

for i=BladeElemNumber:-1:1
      iterAxial=1;
      while(true) % Convergência Factor inducção Axial
            iterTangencial=1;
            while(true) % Convergência Factor inducção Tangencial
```

```
fi(i)=atan((1-aOld(i))/((1+a_Old(i))*LocalSpeedRatio(i)));
if(fi(i)<0)
      fi(i)=-fi(i);
end
if(PerdaPonta)
      argumento=exp(-(BladeNumber/2*(1-r_R(i))/(r_R(i)*sin(fi(i)))));
      if(argumento>1)
      argumento=1;
      end
      FactorPrandtl(i)=(2/pi)*acos(argumento);
else
      FactorPrandtl(i)=1;
end
alfa(i)=fi(i)-beta(i);
Cl(i)=CalculaCL(i);
c_R(i)=8*pi.*r_R(i)*(1-cos(fi(i)))/(BladeNumber*Cl(i));
Sigma(i)=BladeNumber.*c_R(i)/(2*pi.*r_R(i));
if(IncluirCd)
      Cd(i)=CalculaCD(i);
      Ct(i)=Cl(i)*sin(fi(i))-Cd(i)*cos(fi(i));
      Cn(i)=Cl(i)*cos(fi(i))+Cd(i)*sin(fi(i));
else
      Ct(i)=Cl(i)*sin(fi(i));
      Cn(i)=Cl(i)*cos(fi(i));
end
a_New(i)=1/(((4*FactorPrandtl(i)*sin(fi(i))*cos(fi(i)))/(Sigma(i)*Ct(i)))-1);
```

```
      a_New(i)=a_New(i)*relax+a_Old(i)*(1-relax);
      if(abs(a_New(i)-a_Old(i))<residual)
            break
      end
      a_Old(i)=a_New(i);
      iterTangencial=iterTangencial+1;
      if(iterTangencial>MaxIter)
            fprintf(1,'a_(%d) did not converge!\n',i);
            if(i~=BladeElemNumber)
                  a_Old(i)=a_Old(i+1);
            end
            break
      end
end
if(IncluirCd)
      CTl(i)=Sigma(i)*(1-aOld(i))^2*(Cl(i)*cos(fi(i))+Cd(i)*sin(fi(i)))/(sin(fi(i)))^2;
else
      CTl(i)=Sigma(i)*(1-aOld(i))^2*(Cl(i)*cos(fi(i)))/(sin(fi(i)))^2;
end
if(CTl(i)<0.96)
      aNew(i)=1/(1+(4*FactorPrandtl(i)*(sin(fi(i)))^2/(Sigma(i)*Cn(i))));
else
      aNew(i)=(1/FactorPrandtl(i))*(0.143+sqrt(0.0203-0.6427*(0.889-CTl(i))));
end
aNew(i)=aNew(i)*relax+aOld(i)*(1-relax);
if(abs(aNew(i)-aOld(i))<residual)
```

```
            break
        end
        aOld(i)=aNew(i);
        iterAxial=iterAxial+1;
        if(iterAxial>MaxIter)
            fprintf(1,'a(%d) did not converge!\n',i);
            if(i~=BladeElemNumber)
                aOld(i)=aOld(i+1);
            end
            break
        end
    end
    dr=(1-0.15)/BladeElemNumber;
    Thrust=0;
    Torque=0;
    Cp=0;
    for i=1:1:BladeElemNumber
        dT(i)=FactorPrandtl(i)*rho*U^2*4*aNew(i)*(1-aNew(i))*pi*r_R(i)*Radius*dr;
        dQ(i)=4*FactorPrandtl(i)*rho*U*a_New(i)*(1-aNew(i))*pi*(r_R(i)*Radius)^3*Omega*dr;
        Thrust=Thrust+dT(i);
        Torque=Torque+dQ(i);
        VelAxial(i)=U*(1-(aNew(i)));
        VelTangencial(i)=Omega*r_R(i)*Radius*(1-(a_New(i)));
        VelRelativa(i)=sqrt((VelAxial(i))^2+(VelTangencial(i))^2);
        Reynolds(i)=rho*VelRelativa(i)*c_R(i)*Radius/miu;
    end
```

```
        Thrust=Thrust
        Torque=Torque
        PowerVento=0.5*rho*pi*Radius^2*U^3
        Radius
        Power_real=Torque*Omega
        Cp=Power_real/PowerVento
        OmegaRPM=Omega*30/pi
end
figure(4),plot(r_R,aNew,'k'),xlabel('r/R'),grid on,hold on
plot(r_R,a_New,'b'),xlabel('r/R')
legend('Indução axial','Indução tangencial')
figure(6),plot(r_R,CTl,'r'),xlabel('r/R'),ylabel('Coeficiente de Impulso'),grid on,hold on
figure(7),plot(r_R,alfa*180/pi,'r'),xlabel('r/R'),ylabel('Angulo de Ataque [º]'),grid on,hold on
figure(8),plot(r_R,beta*180/pi,'r'),xlabel('r/R'),grid on,hold on
plot(r_R,fi*180/pi,'b'),xlabel('r/R')
legend('Ângulo de torção','Ângulo do escoamento')
% figure(8),plot(r_R,beta*180/pi,'r'),xlabel('Radius/Radius'),ylabel('Angulo de Pitch [º]'),grid on,hold on
% figure(9),plot(r_R,fi*180/pi,'r'),xlabel('Radius/Radius'),ylabel('Fi [º]'),grid on,hold on
figure(10),plot(r_R,Cl,'r'),xlabel('r/R'),ylabel('Coeficiente de Sustentação'),grid on,hold on
figure(11),plot(r_R,dT,'r'),xlabel('r/R'),ylabel('Impulso do elemento de pá [N.s]'),grid on,hold on
figure(12),plot(r_R,dQ,'r'),xlabel('r/R'),ylabel('Binario do elemento de pá [N.m]'),grid on,hold on
figure(13),plot(r_R,VelAxial,'r'),xlabel('r/R'),ylabel('Velocidade Axial do elemento de pá [m/s]'),grid on,hold
on
figure(14),plot(r_R,VelTangencial,'r'),xlabel('r/R'),ylabel('Velocidade Tangencial do elemento de pá [m/s]'),grid
on,hold on
figure(15),plot(r_R,VelRelativa,'r'),xlabel('r/R'),ylabel('Velocidade Relativa do elemento de pá [m/s]'),grid
on,hold on
```

```
      figure(16),plot(r_R,Reynolds,'r'),xlabel('r/R'),ylabel('Número de Reynolds'),grid on,hold on
      figure(17),plot(r_R,c_R*Radius,'r'),xlabel('r/R'),ylabel('Corda [m]'),grid on,hold on
      figure(18),plot(r_R,FactorPrandtl,'r'),xlabel('r/R'),ylabel('FactorPrandtl'),grid on,hold on
      figure(19),plot(r_R,Cd,'r'),xlabel('r/R'),ylabel('Coeficiente de resistência'),grid on,hold on
      figure(20),plot(r_R,Cl./Cd,'r'),xlabel('r/R'),ylabel('Cl/Cd'),grid on,hold on

      save Ideal c_R beta Radius

      fid=fopen('Tabela.dat','w');
      fprintf(fid,
      'r_R\tbeta\tfi\talfa\tFactorPrandtl\tc_R\tLocalSpeedRatio\tCl\tCd\tCt\tCn\taNew\ta_New\tCTl\tdT\tVelAxial\tVelTan
      gencial\tVelRelat
      iva\tReynolds\n\n');
      for(i=1:1:length(r_R));
            fprintf(fid,
            '%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f
            \t%.5f\n',r_R(i),
            beta(i)*180/pi,fi(i)*180/pi,alfa(i)*180/pi,FactorPrandtl(i),c_R(i),LocalSpeedRatio(i),Cl(i),Cd(i),Ct(i),Cn(
            i),aNew(i),a_New(i),CTl(i),dT(i),VelAxial(i),VelTangencial(i),VelRelativa(i),Reynolds(i));
      end
      fclose(fid);
end

function AirfoilBladeElement=DeclareElementBladeAirfoil(opcao)

%DeclareElementBladeAirfoil(opcao)
%
```

```
%opcao==optimo > a velocidade do elmento de pa é convertida para a
%velocidade equivalente usando a relação entre cordas, a incidência é
%idêntica. Estes 2 parâmetros são usados para encontrar o perfil com maior
%eficiência
%
%opcao==given >
%
      global r_R BladeElemNumber AirfoilList
      syms AirfoilBladeElement
      if(strcmp(opcao,'given'))
            fid=fopen('twistAerodinamica.dat','r');
            if(fid==-1)
                  disp('Erro: Não foi possível encontrar o ficheiro twistAerodinamica.dat')
                  return
            else
                  numeroTorcoes=fscanf(fid,'%d',1);
                  if(numeroTorcoes<1)
                        disp('Erro: Número de torções tem de ser um valor maior que zero!')
                        return
                  end
                  indice=1;
                  while(numeroTorcoes>0)
                        nomePerfil=fscanf(fid,'%s',1);
                        if(numeroTorcoes~=1)
                              r_R_max=fscanf(fid,'%f',1);
                        else
```

```
                        r_R_max=1.0;
                    end
                    while((indice<=BladeElemNumber)&&(r_R(indice)<=r_R_max)) % &&=AND logico
                        AirfoilBladeElement(indice)=nomePerfil;
                        indice=indice+1;
                    end
                    numeroTorcoes=numeroTorcoes-1;
                end
                fclose(fid);
            end
        elseif(strcmp(opcao,'optimo'))
            %AirfoilList={'FX63137sm','NACA4415','SD2030','SG6042'};
            candidato=zeros(1,length(AirfoilList));
            for indice=1:1:BladeElemNumber
                for perfil=1:1:6
                    candidato(perfil)=CalculaEficiencia(indice,perfil);
                end
                [eficienciaMaxima,indicePerfil]=max(candidato);
                AirfoilBladeElement(indice)=char(AirfoilList(indicePerfil));
            end
        end
    end

    function eficic=CalculaEficiencia(indiceElementoPa,indicePerfil)
        global VelRelativa alfa c_R Radius Eficiencia_parametrico
        y=VelRelativa(indiceElementoPa)*c_R(indiceElementoPa)*Radius;
```

```
x=alfa(indiceElementoPa)*180/pi;
incidenciaMin=0;
incidenciaMax=15;
velocidadeMin=5;
velocidadeMax=14;
extrapolar=false;
if(x<incidenciaMin)
      extrapolar=true;
end
if(x>incidenciaMax)
      extrapolar=true;
end
if(y<velocidadeMin)
      extrapolar=true;
end
if(y>velocidadeMax)
      extrapolar=true;
end
if(extrapolar)
      %Superficie Cubica: length(a)=10. A expressao analitica e:
      eficic = a(1)+a(2)*x+a(3)*y+a(4)*x^2+a(5)*x*y+a(6)*y^2+a(7)*x*y^2+a(8)*x^2*y^2+a(9)*x^3+a(10)*y^3;
else
      %interpolar
      nomeFicheiro=strcat('Eficiencia_',AirfoilList(indicePerfil),'.dat');
      matriz=load(char(nomeFicheiro));
      eficic=interp2([0 3 6 9 11 13 15],[5 8 11 14],matriz',x,y,'spline');
```

```
	end
end

function lift=CalculaCL(indiceElementoPa)
	global VelRelativa alfa c_R Radius Cl_parametrico AirfoilBladeElement AirfoilList
	for indicePerfil=1:1:length(AirfoilList)
		if(strcmp(char(AirfoilBladeElement(indiceElementoPa)),AirfoilList(indicePerfil)))
			break
		end
	end
	if(indicePerfil>length(AirfoilList))
		disp('Erro: Um ou mais perfis não pertencem à lista de perfis parametrizada!')
		return
	end
	y=VelRelativa(indiceElementoPa)*c_R(indiceElementoPa)*Radius;
	x=alfa(indiceElementoPa)*180/pi;
	incidenciaMin=0;
	incidenciaMax=15;
	velocidadeMin=5;
	velocidadeMax=14;
	extrapolar=false;
	if(x<incidenciaMin)
		extrapolar=true;
	end
	if(x>incidenciaMax)
		extrapolar=true;
```

```
	end
	if(y<velocidadeMin)
		extrapolar=true;
	end
	if(y>velocidadeMax)
		extrapolar=true;
	end
	if(extrapolar)
		%Superficie Cubica: length(a)=10. A expressao analitica e:
		a=Cl_parametrico(indicePerfil,:);
		lift = a(1)+a(2)*x+a(3)*y+a(4)*x^2+a(5)*x*y+a(6)*y^2+a(7)*x*y^2+a(8)*x^2*y^2+a(9)*x^3+a(10)*y^3;
	else
		%interpolar
		nomeFicheiro=strcat('Cl_',AirfoilList(indicePerfil),'.dat');
		matriz=load(char(nomeFicheiro));
		lift=interp2([0 3 6 9 11 13 15],[5 8 11 14],matriz',x,y,'spline');
	end
end

function drag=CalculaCD(indiceElementoPa)
	global VelRelativa alfa c_R Radius Cd_parametrico AirfoilBladeElement AirfoilList
	for indicePerfil=1:1:length(AirfoilList)
		if(strcmp(char(AirfoilBladeElement(indiceElementoPa)),AirfoilList(indicePerfil)))
			break
		end
	end
```

```
if(indicePerfil>length(AirfoilList))
      disp('Erro: Um ou mais perfis não pertencem à lista de perfis parametrizada!')
      return
end
y=VelRelativa(indiceElementoPa)*c_R(indiceElementoPa)*Radius;
x=alfa(indiceElementoPa)*180/pi;
incidenciaMin=0;
incidenciaMax=15;
velocidadeMin=5;
velocidadeMax=14;
extrapolar=false;
if(x<incidenciaMin)
      extrapolar=true;
end
if(x>incidenciaMax)
      extrapolar=true;
end
if(y<velocidadeMin)
      extrapolar=true;
end
if(y>velocidadeMax)
      extrapolar=true;
end
if(extrapolar)
      %Superficie Cubica: length(a)=10. A expressao analitica e:
      a=Cd_parametrico(indicePerfil,:);
```

```
            drag = a(1)+a(2)*x+a(3)*y+a(4)*x^2+a(5)*x*y+a(6)*y^2+a(7)*x*y^2+a(8)*x^2*y^2+a(9)*x^3+a(10)*y^3;
        else
            %interpolar
            nomeFicheiro=strcat('Cd_',AirfoilList(indicePerfil),'.dat');
            matriz=load(char(nomeFicheiro));
            drag=interp2([0 3 6 9 11 13 15],[5 8 11 14],matriz',x,y,'spline');
        end
    end
```