

```

! *** general support modules ***
!
Module Param_y_Ctes
  Implicit none
  Integer, parameter :: precisionD = 12, precisionID = 8, precisionQ = 24
  Integer, parameter :: kd=selected_real_kind(precisionD),
  KQuad=selected_real_kind(precisionQ)
  Integer, parameter :: KIprec=selected_Int_kind(precisionID)
  ! pi
  Real (kd), parameter :: gpi=3.1415926535897932_kd
  Real (kd), parameter :: InvSqr2Tpi = .3989422804014327_kd
  Real (kd), parameter :: Deg_to_Rad = gpi / 180._kd
  Real (kd), parameter :: edouble = 2.718281828459045_kd
  Real (kd), parameter :: Explim = 708._kd, & ! base 'e'
                                ExplimGauss = 37._kd ! in exp(x*x*.5)
  Integer, parameter :: Explim10 = 307 ! base E (10)
  ! Huge y Tiny
  Real (kd) :: x_huge_tiny
  Real (kd), parameter :: HugeStr = Huge(x_huge_tiny), TinyStr = Tiny(x_huge_tiny), &
                                HugeOp = 1.E+308_kd, TinyOp = 1.E-307_kd

  Integer :: IerrDbg
  Integer :: Uwrfile
  Logical :: LgDbg = .FALSE.
End Module Param_y_Ctes
!
! statistical routines etc.
Module Simul_PV_Wind_Stat_Support
!
  Use Param_y_Ctes
!
  Implicit none
!
  Private
  Public :: Hist_Res, Acumul, Gauss_ms, Histogram, Descrip_Stat, &
          Estandar, Kernel_PDF_Hat, CDF_G01_base, Prob_NoPm
!
  Type Hist_Res
    Integer :: ind
    Real (kd) :: Hist, Arg, CDF, Den, StdErr, HistPct
    Character (len=2) :: Warn
  End Type Hist_Res
! constants of the Gaussian CDF.
  Real(kd), Dimension(5), Parameter ::
    AI = (/ 3.20937758913846947E03_kd, 3.77485237685302021E02_kd, &
           1.13864154151050156E02_kd, 3.16112374387056560E00_kd, &
           1.85777706184603153E-1_kd /), &
  BI = (/ 2.84423683343917062E03_kd, 1.28261652607737228E03_kd, &
          2.44024637934444173E02_kd, 2.36012909523441209E01_kd, &
          1._kd /)
  Real(kd), Dimension(9), Parameter ::
    AII = (/ 1.23033935479799725E03_kd, 2.05107837782607147E03_kd, &
            1.71204761263407058E03_kd, 8.81952221241769090E02_kd, &
            2.98635138197400131E02_kd, 6.61191906371416295E01_kd, &
            8.88314979438837594E00_kd, 5.64188496988670089E-1_kd, &
            2.15311535474403846E-8_kd /), &
    BII = (/ 1.23033935480374942E03_kd, 3.43936767414372164E03_kd, &
            4.36261909014324716E03_kd, 3.29079923573345963E03_kd, &
            1.62138957456669019E03_kd, 5.37181101862009858E02_kd, &
            1.17693950891312499E02_kd, 1.57449261107098347E01_kd, &
            1._kd /)
  Real(kd), Dimension(6), Parameter ::
    AIII = (/ 6.58749161529837803E-4_kd, 1.60837851487422766E-2_kd, &
            1.25781726111229246E-1_kd, 3.60344899949804439E-1_kd, &
            3.05326634961232344E-1_kd, 1.63153871373020978E-2_kd /), &
    BIII = (/ 2.33520497626869185E-3_kd, 6.05183413124413191E-2_kd, &
            5.27905102951428412E-1_kd, 1.87295284992346047E00_kd, &
            2.56852019228982242E00_kd, 1._kd /)
!
  Interface Acumul
    Module Procedure Acumul, Acumul_Vc
  End Interface Acumul

```

```

!
Contains
!
! Generates a vector of random Gauss(mean, sigma) variables.
Function Gauss_ms ( U0ld,
                    &
                    mean, sigma )
Real (kd), Intent(in) :: U0ld(:) ! IN
Real (kd), Dimension(Size(U0ld)) :: Gauss_ms
Real (kd), Optional, Intent(in) :: mean, sigma ! Optionals
Integer :: ii, Nsim
Real (kd) :: Fac1, Fac2, Gauss01(Size(U0ld)), U01(Size(U0ld))
!
! Local vars. & ovfl. corr
Nsim = Size(U0ld)
U01 = U0ld
Where ( U0ld < 1.E-307_kd ) U01(:Nsim) = 1.E-307_kd
! SIMULATION_GAUSS01
Do ii = 1, Nsim-1, 2
  Fac1=Sqrt( -2._kd * Log( U01(ii) ) )
  Fac2=Cos( 6.2831853071795864769_kd * U01(ii+1) )
  Gauss01(ii)=Fac1*Fac2
  If ( U01(ii+1) .LE. .5_kd ) Then
    Gauss01(ii+1)=Fac1*Sqrt(1._kd-Fac2*Fac2)
  Else
    Gauss01(ii+1)=Fac1*(-Sqrt(1._kd-Fac2*Fac2))
  End if
End do
! Mean and variance (first sigma !)
If( Present(sigma) ) Gauss01 = sigma * Gauss01
If( Present(mean) ) Gauss01 = Gauss01 + mean
Gauss_ms = Gauss01
End Function Gauss_ms
!
! CDF_G01 = C.D.F. Gaussian (0,1).
Real(kd) Function CDF_G01_base ( Argd )
Real(kd), intent(in) :: Argd
Integer :: ind
Real(kd) :: Arg, AbArg, Erfc, AA16
Real(kd) :: ArgV(9)
!
! EX.
! Calc. Erfc (Arg / Sqrt(2))
Arg = Argd / Sqrt(2._kd) ; AbArg = Abs(Arg) ; ArgV(1) = 1._kd
INTERVALOS : If ( AbArg .LE. 0.46875_kd ) Then
  ! first interval
  ArgV(2) = Arg * Arg
  Do ind = 3, 5
    ArgV(ind) = ArgV(ind-1) * ArgV(2)
  End do
  Erfc = 1._kd - Arg * (Dot_Product(ArgV(:5), AI) / Dot_Product(ArgV(:5), BI))
Else if ( AbArg .LE. 4._kd ) Then INTERVALOS
  ! second interval
  Do ind = 2, 9
    ArgV(ind) = ArgV(ind-1) * AbArg
  End do
  AA16 = Aint(AbArg*16._kd) / 16._kd
  Erfc = (Dot_Product(ArgV, AII) / Dot_Product(ArgV, BII)) &
    / Exp( AA16*AA16 + (AbArg-AA16)*(AbArg+AA16) )
  If ( Arg .LE. 0._kd ) Erfc = 2._kd - Erfc
Else INTERVALOS
  ! third interval
  CHK_OVFLW : If ( AbArg .GT. 1.E+153_kd ) Then
    Erfc = 0._kd
  Else
    AA16 = Aint(AbArg * 16._kd) / 16._kd
    ArgV(2) = 1._kd / (Arg * Arg)
    Do ind = 3, 6
      ArgV(ind) = ArgV(ind-1) * ArgV(2)
    End do
    Erfc = (1._kd / Sqrt(gpi) - ArgV(2) * (Dot_Product(ArgV(:6), AIII) /
      Dot_Product(ArgV(:6), BIII)) ) &

```

```

/ AbArg / Exp( Min( 708._kd, (AA16*AA16 +
(AbArg-AA16)*(AbArg+AA16)) ) )
End if CHK_OVFLW
If ( Arg .LE. 0._kd ) Erfc = 2._kd - Erfc
End if INTERVALOS
! final calc.
CDF_G01_base = 1._kd - .5_kd * Erfc
End Function CDF_G01_base
!
! CDF_G01 = C.D.F. de una Gaussian (0,1). vec. argument.
Function CDF_G01_Vc ( Argd )
Real(kd), intent(in) :: Argd(:) ! IN
Real(kd), Dimension(Size(Argd)) :: CDF_G01_Vc
! ! Local
Real(kd) :: Arg(Size(Argd)), AbArg(Size(Argd)), Erfc(Size(Argd)), AA16(Size(Argd))
Real(kd) :: ArgV(Size(Argd),1:9)
!
! EX.
! Calc. Erfc (Arg / Sqrt(2))
Arg = Argd / Sqrt(2._kd) ; AbArg = Abs(Arg) ; ArgV(:,1) = 1._kd
INTERVALOS : Where ( AbArg .LE. 0.46875_kd )
! first interval
ArgV(:,2) = Arg * Arg ; ArgV(:,3) = ArgV(:,2) * ArgV(:,2)
ArgV(:,4) = ArgV(:,3) * ArgV(:,2) ; ArgV(:,5) = ArgV(:,4) * ArgV(:,2)
Erfc = 1._kd - Arg * (Matmul(ArgV(:,1:5), AI) / Matmul(ArgV(:,1:5), BI))
Else Where ( AbArg .LE. 4._kd ) INTERVALOS
! second interval
ArgV(:,2) = AbArg ; ArgV(:,3) = ArgV(:,2) * AbArg
ArgV(:,4) = ArgV(:,3) * AbArg ; ArgV(:,5) = ArgV(:,4) * AbArg
ArgV(:,6) = ArgV(:,5) * AbArg ; ArgV(:,7) = ArgV(:,6) * AbArg
ArgV(:,8) = ArgV(:,7) * AbArg ; ArgV(:,9) = ArgV(:,8) * AbArg
AA16 = Aint(AbArg*16._kd) / 16._kd
Erfc = (Matmul(ArgV, AII) / Matmul(ArgV, BII)) &
/ Exp( AA16*AA16 + (AbArg-AA16)*(AbArg+AA16) )
Where ( Arg .LE. 0._kd ) Erfc = 2._kd - Erfc
ElseWhere INTERVALOS
! third interval
CHK_OVFLW : Where ( AbArg .GT. 1.E+153_kd )
Erfc = 0._kd
ElseWhere
AA16 = Aint(AbArg * 16._kd) / 16._kd
ArgV(:,2) = 1._kd / (Arg * Arg) ; ArgV(:,3) = ArgV(:,2) * ArgV(:,2)
ArgV(:,4) = ArgV(:,3) * ArgV(:,2) ; ArgV(:,5) = ArgV(:,4) * ArgV(:,2)
ArgV(:,6) = ArgV(:,5) * ArgV(:,2)
Erfc = (1._kd / Sqrt(gpi) - ArgV(:,2) * Matmul(ArgV(:,1:6), AIII) /
Matmul(ArgV(:,1:6), BIII) ) &
/ AbArg / Exp( Min( 708._kd, (AA16*AA16 +
(AbArg-AA16)*(AbArg+AA16)) ) )
End Where CHK_OVFLW
Where ( Arg .LE. 0._kd ) Erfc = 2._kd - Erfc
End Where INTERVALOS
! final calc.
CDF_G01_Vc = 1._kd - .5_kd * Erfc
End Function CDF_G01_Vc
!
! Histograma, Equally spaced intervals.
Subroutine Histogram ( Xd, Results, Uwrfile, CdfTeor )
Interface ! IN
Function CdfTeor ( Argd )
Use Param_y_Ctes
Real(kd), intent(in) :: Argd(:)
Real(kd), Dimension(Size(Argd)) :: CdfTeor
End Function CdfTeor
End Interface
Integer, Intent(in) :: Uwrfile
Real (kd), Intent(in) :: Xd(:)
Type (Hist_Res), Intent(out), Dimension(:) :: Results ! OUT
Optional :: CdfTeor, Uwrfile ! OPTIONAL
!
Integer :: Nobs, Ninte, ind, HistI(0:Size(Results)),
IndXd(Size(Xd)) ! Local

```

```

Real (kd) :: Rinc, Rnobs, xx(Size(Xd))
Real (kd), Dimension(Size(Results)) :: Hist, Arg, CDF, Den, StdErr, HistPct, Cdfstr
Character :: Warn(Size(Results))*2
!
! EXEC.
! Infrastructure ; Local variables
Nobs = Size(Xd) ; Rnobs = Real(Nobs, kd) ; xx = Xd ; Ninte = Size(Results)
! MAIN BODY
! Interval for each value.
Rinc = (Maxval(xx) - Minval(xx)) / Real(Ninte, kd)
IndXd = Ceiling( Real( (xx - Minval(xx)) / Rinc ) )
! interval arg.
Arg = Acumul( (Minval(xx) + Rinc*.5_kd), Rinc, Ninte )
! Histog. calc.
HistI = 0
Do ind = 1, Nobs
  HistI(IndXd(ind)) = HistI(IndXd(ind)) + 1
End do
! Remaining results
Hist = Real(HistI(1:), kd)
Cdf = Acumul( Hist / Rnobs )
Den = Hist / Rnobs / Rinc
HistPct = Hist / Rnobs
StdErr = Sqrt( Max(Cdf * (1._kd - Cdf) / Rnobs, 0._kd ) )
If ( Present(CdfTeor) ) Then
  Cdfstr = CdfTeor(Arg+Rinc*.5_kd)
  StdErr = Sqrt( Max( Cdfstr*(1._kd-Cdfstr)/Rnobs, 0._kd ) )
  Warn = GAUSS_Bands( Cdf-Cdfstr, StdErr )
Else
  Warn = ' '
End If
! OUTPUT
! Complete output
Results%ind = (/ (ind, ind = 1, Size(Results)) /)
Results%Hist = Hist
Results%Arg = Arg
Results%CDF = CDF
Results%Den = Den
Results%StdErr = StdErr
Results%HistPct = HistPct
Results%Warn = Warn
! write info. if required.
If ( .NOT. Present(Uwrfile) ) Return
Write (UwrFile, '( /A/ )' ) 'HISTOGRAM RESULTS'
Write (UwrFile, '( 1X, A18, 1X, I10, 1/ )' ) ' SAMPLE SIZE =', Nobs
Write (UwrFile,*) ' Ind      Hist      Arg      CDF      Den&
                  &      StdErr      HistPct'
Do ind = 1, Ninte
  write ( UwrFile, '( 1X, I3, E15.7, 2(1X, F10.5), 1X, A2, 3(1X, F10.5))' )
  ind,
    &
    Hist(ind), Arg(ind), CDF(ind), Warn(ind), Den(ind), StdErr(ind),
    HistPct(ind)
End do
End Subroutine Histogram
!
! Histograma, Equally spaced intervals (simplified).
Subroutine Histogram_2 ( Xd, Results, Lims, Uwrfile )
!
Integer, Intent(in) :: Uwrfile ! IN
Real (kd), Intent(in) :: Xd(:), Lims(:)
Type (Hist_Res), Intent(out), Dimension(:) :: Results ! OUT
Optional :: Uwrfile
!
Integer :: Nobs, Ninte, ind, HistI(0:Size(Results)),
IndXd(Size(Xd)) ! Local
Real (kd) :: Rinc, Rnobs, xx(Size(Xd)), LimsL(2)
Real (kd), Dimension(Size(Results)) :: Hist, Arg, CDF, Den, StdErr, HistPct, Cdfstr
Character :: Warn(Size(Results))*2
!
! EXEC.
! Infrastructure (Local variables, optionals, allocation)

```

```

Nobs = Size(Xd) ; Rnobs = Real(Nobs, kd) ; xx = Xd ; LimsL = Lims ; Ninte =
Size(Results)
! MAIN BODY
! Interval for each value.
Rinc = (Lims(2) - Lims(1)) / Real(Ninte-2, kd)
Where ( xx >= LimsL(2) ) IndXd = Ninte
Where ( xx <= LimsL(1) ) IndXd = 1
Where( (xx > LimsL(1)) .AND. (xx < LimsL(2)) ) &
IndXd = 1 + Ceiling( Real( (xx - LimsL(1)) / Rinc ) )
! intervals args.
Arg(2:Ninte) = Acumul( (Lims(1) + Rinc*.5_kd), Rinc, (Ninte-1) )
Arg(1) = Arg(2) - Rinc
! Histog. calc.
HistI = 0
Do ind = 1, Nobs
HistI(IndXd(ind)) = HistI(IndXd(ind)) + 1
End do
! Remaining results
Hist = Real(HistI(1:), kd)
Cdf = Acumul( Hist / Rnobs )
Den = Hist / Rnobs / Rinc
HistPct = Hist / Rnobs
StdErr = Sqrt( Max(Cdf * (1._kd - Cdf) / Rnobs, 0._kd ) )
! OUTPUT
! Complete output
Results%ind = (/ (ind, ind = 1, Size(Results)) /)
Results%Hist = Hist
Results%Arg = Arg
Results%CDF = CDF
Results%Den = Den
Results%StdErr = StdErr
Results%HistPct = HistPct
Results%Warn = Warn
! write info. if demanded.
If ( .NOT. Present(Uwrfile) ) Return
Write (UwrFile, '( /A/ )' ) 'HISTOGRAM RESULTS'
Write (UwrFile, '( 1X, A18, 1X, I10, 1/ )' ) ' SAMPLE SIZE =', Nobs
Write (UwrFile,*) ' Ind Hist Arg CDF Den&
& StdErr HistPct'
Do ind = 1, Ninte
write ( UwrFile, '( 1X, I3, E15.7, 2(1X, F10.5), 1X, A2, 3(1X, F10.5))' )
ind, &
Hist(ind), Arg(ind), CDF(ind), Warn(ind), Den(ind), StdErr(ind),
HistPct(ind)
End do
End Subroutine Histogram_2
!
! Estd, estimator zero mean, and std. e. Std_d ;
Function GAUSS_Bands ( Estd, Std_d )
Real (kd), Intent(in) :: Estd(:), Std_d(:)
Character (len=2), Dimension(Size(Estd)) :: GAUSS_Bands, Work
Optional :: Std_d
Real (kd), Dimension(Size(Estd)) :: Std, Est
!
Work = ' ' ; Est = Estd
If ( Present(Std_d) ) Then; Std = Std_d ; Else ; Std = 1._kd ; End If
Where ( Abs(Est) .GT. (1.65_kd * Std) ) Work = ' * '
Where ( Abs(Est) .GT. (1.96_kd * Std) ) Work = ' *** '
GAUSS_Bands = Work
End Function GAUSS_Bands
!
Function Acumul_Vc ( x )
Real (kd), Intent(in) :: x(:)
Real (kd), Dimension(Size(x)) :: Acumul_Vc
Integer :: ind
Real (kd) :: Work(Size(x))
Work(1) = x(1)
Do ind = 2, Size(x)
Work(ind) = Work(ind-1) + x(ind)
End do
Acumul_Vc = Work

```

```

End Function Acumul_Vc
!
Function Acumul ( X1, Rinc, n )
Integer, Intent(in) :: n
Real (kd), Intent(in) :: X1, Rinc
Real (kd), Dimension(n) :: Acumul
Integer :: ind
Real (kd) :: Work(n)
Work(1) = X1
Do ind = 2, n
    Work(ind) = Work(ind-1) + Rinc
End do
Acumul = Work
End Function Acumul
!
! Xd,( n° obs., n° vars.), In
Subroutine Descrip_Stat ( Xd, & ! IN
                        mean, var, sd, skw, kurt, max_, min_, & ! OUT
                        tmean, sdsd, tskw, tkurt, &
                        tskwmom, tkurtmom ) ! OPTIONAL
Real (kd), Intent(in) :: Xd(:, :)
Real (kd), Dimension(:), Intent(out) :: mean, var, sd, skw, kurt, max_, min_, &
                        tmean, sdsd, tskw, tkurt
Real (kd), Intent(out), Optional :: tskwmom(:), tkurtmom(:)
Integer :: I1, Nvar
Real (kd) :: SqRobs
Real (kd), Dimension(Size(xd,2)) :: VarSkw, VarKurt
Real (kd), Dimension(Size(xd,1),Size(xd,2)) :: X, XmM, XmM2
!
! EX.
! Local variables
X = Xd ; Nvar = Size(xd,2)
! Means
Mean = AverageMat( X, 1 )
! Demean
XmM = DemeanMat ( X, 1 )
! Variances , skewness and kurtosis
XmM2 = XmM * XmM
Var = VarianceMat ( X, 1 )
Sd = Sqrt( Var )
Skw = AverageMat( (XmM2*XmM), 1 ) / (Sd*Var)
Kurt = AverageMat( (XmM2*XmM2), 1 ) / (Var*Var)
! t - ratios
SqRobs = Sqrt(Real(Size(xd,1),kd))
tmean = SqRobs * Mean / Sd
sdsd = Sd / Sqrt(2._kd) / SqRobs
tskw = SqRobs * Skw / Sqrt(6._kd)
tkurt = SqRobs * (Kurt-3._kd) / Sqrt(24._kd)
! máx. min. ;
max_ = Maxval( XmM, Dim=1 )
min_ = Minval( XmM, Dim=1 )
! Optional Output
! t - ratios moments methods. (skw, kurt)
If ( Present(tskwmom) ) Then
    VarSkw = VarianceMat ( (XmM2*XmM), 1 ) / (Sd*Var)**2
    tskwmom = SqRobs * Skw / Sqrt(VarSkw)
End if
If ( Present(tkurtmom) ) Then
    VarKurt = VarianceMat ( (XmM2*XmM2), 1 ) / (Var*Var)**2
    tkurtmom = SqRobs * (Kurt-3._kd) / Sqrt(VarKurt)
End if
End Subroutine Descrip_Stat
!
Function AverageMat ( X, Dim )
Real (kd), Intent(in) :: X(:, :)
Integer, Intent(in) :: Dim
Real (kd), Dimension(Size(X,3-Dim)) :: AverageMat
AverageMat = Sum(X,Dim) / Real( Size(X,Dim), kd )
End Function AverageMat
!
Function DemeanMat ( X, Dim )

```

```

Real (kd), Intent(in) :: X(:, :)
Integer, Intent(in) :: Dim
Real (kd), Dimension(Size(X,1),Size(X,2)) :: DemeanMat
DemeanMat = X - Spread( AverageMat(X, Dim), Dim=Dim, ncopies = Size(x,Dim) )
End Function DemeanMat
!
Function VarianceMat_ ( X, Dim )
Real (kd), Intent(in) :: X(:, :)
Integer, Intent(in) :: Dim
Real (kd), Dimension(Size(X,3-Dim)) :: VarianceMat_
Real (kd) :: Nr
Nr = Real(Size(X,Dim), kd)
!VarianceMat = AverageMat ( X*X, Dim ) - AverageMat ( X, Dim )**2
VarianceMat_ = (Nr/(Nr-1._kd)) * (Sum(X*X,Dim)/Nr - (Sum(X,Dim)/Nr)**2)
End Function VarianceMat_
!
Function VarianceMat ( X, Dim )
Real (kd), Intent(in) :: X(:, :)
Integer, Intent(in) :: Dim
Real (kd), Dimension(Size(X,3-Dim)) :: VarianceMat
Real (kd) :: Nr
Nr = Real(Size(X,Dim), kd)
VarianceMat = (Nr/(Nr-1._kd)) * Sum(DemeanMat ( X, Dim )*DemeanMat ( X, Dim ),Dim)/Nr
End Function VarianceMat
!
Function Estandar ( X )
Real (kd), Intent (in) :: X(:)
Real (kd), Dimension(Size(X)) :: Estandar
Real (kd) :: Nr, Demean(Size(X))
Nr = Real(Size(X), kd)
Demean = X - Sum( X ) / Nr
Estandar = Demean / Sqrt( Sum(Demean**2)/(Nr-1._kd))
End Function Estandar
!Epanechnikov's kernel
Function Kernel_Epanech ( x )
Real (kd), Intent(in) :: x(:)
Real (kd), Dimension(Size(x)) :: Kernel_Epanech
Kernel_Epanech = Merge( 0.33541019662496845_kd-0.06708203932499369_kd*x*x, &
0._kd, Abs(x) < 2.2360679774997897_kd )
End Function Kernel_Epanech
!
! Kernel Pdf estimation.
!
Real (kd) Function Kernel_PDF_Hat ( x, xsmp )
Real (kd), Intent(in) :: x, xsmp(:) ! Arg. Types
Real (kd) :: h, Nr, Var, xw(Size(xsmp)) ! Local Types
!
Nr = Real(Size(xsmp),kd)
Var = Sum( (xsmp - Sum( xsmp ) / Nr)**2 ) / Nr
h = 1.06_kd * Sqrt(Var) / Nr**.2_kd ! Silverman, pg.4
xw =(x-xsmp)/h
Kernel_PDF_Hat = Sum( Merge(
0.33541019662496845_kd-0.06708203932499369_kd*xw*xw, &
0._kd, Abs(xw) < 2.2360679774997897_kd )
)
/ (h*Nr)
End Function Kernel_PDF_Hat
!
! Probability estimated non parametrically.
Real (kd) Function Prob_NoPm ( x, xsmp )
Real (kd), Intent(in) :: x, xsmp(:) ! Arg. Types
Real (kd) :: h, Var, Nr ! Local Types
!
Nr = Real(Size(xsmp),kd)
Var = Sum( (xsmp - Sum( xsmp ) / Nr)**2 ) / Nr
h = 1.06_kd * Sqrt(Var) / Nr**.2_kd
Prob_NoPm = Sum(CDF_G01_Vc((x-xsmp)/h)) / Nr
End Function Prob_NoPm
!
End Module Simul_PV_Wind_Stat_Support
!

```

```

! *** specific support modules ***
!
!   Wind routines.
Module Simul_Wind_specf_Support
!
Use Simul_PV_Wind_Stat_Support
Use Param_y_Ctes
!
Implicit none
!
Private
Public :: StrEx_W, Strlw_W, & ! variables
        Ib_W, &
        DcET_W, EIba_W, &
        ACV_lUSD_W, ACV_SCC_W, ACV_TxCIEA_W, ACV_TxCSS_W, &
        Infrastructure_Wind, Wind_Simul ! subroutines
!
! n° of simulations y life span.
Integer, parameter :: nf = 36, & ! number of simulated periods.
                    ls = 27 !
!
Integer :: ios, & ! 'open' error checking
          ind, t, inf ! other auxiliary indexes.
!
! variables Subroutine Param_Assing
Real (kd) :: ma_s(2), & ! means parameters a.
            sda_s(2), & ! s.d. a's
            sdePTurb_s ! s.d. error ec. module prices.
! variables de la Subroutine Gen_RV_smpf
Real (kd) :: U01_Lahey_s(2+nf), Gauss_01_s(2+nf), &
            ePTurb_s(2015:2050), & ! error, module price
            equation.
            as_s(2) ! simulated values a.
! variables de la Subroutine Model_ecs_smpf
Real (kd), dimension(2015:2050) :: LPTurb ! log module price.
! variables Subroutine Read_Gen_Data
Integer :: LoanM, & !
          ScModel !
Real (kd) :: dep, & !
          PR, & ! Performance Ratio.
          kgCO2, & ! n° de kg. de CO2 por kWh.,
          sdr1, sdr2, & ! Social Discount Rate
          WACC ! Weighted Average Cost of Capital.
Real (kd) :: CfWind(2015:2050+ls), & !
          sdr(2015:2050+ls), & ! Social Discount Rate;
          dsf(2014:2050+ls) !
Logical :: LStr, & ! storage costs
          Prv, & ! generate random parameters.
          Errr, & ! generate random equation errors.
          LgLCE_pdf, & ! pdf LCE
          LgLCE_ACV, & !
          LgModPr, & !
          LgPlwbkd, & !
          WCap, & !
          WCorr !
! variables Subroutine Carbon_Cost
Real (kd) :: ZCC, & ! var aux.
            SCC(2015:2050+ls), & ! Social Cost of Carbon; US$ Ton CO2eq
            TxCIEA(2015:2050+ls), & ! Carbon Tax IEA; US$ Ton CO2eq
            TxCSS(2015:2050+ls) ! Carbon Tax Stiglitz-Stern (repot WB); US$
            Ton CO2eq
! variables Subroutine Cap_and_RMap
Integer :: year(2015:2050) !
Real (kd) :: Ceac, & !
            Ibac ! aux.
Real (kd) :: Cap(1985:2050+ls), & !
            LCap(1985:2050), & !
            DLCap(1986:2050), & !
            DCap(1986:2050), & !
            Ib(1986-ls:2050+ls), & !
            Ib_W(2015:2050), & !

```



```

        Ce(2015:2050+ls)      !
! variables Subroutine Alt_Cap_Rmp
    Real (kd) :: IncRmCap      !
! variables Subroutine Dc_En_AvCV
    Integer, parameter :: hy = 8760 ! 365*24
    Real (kd) :: DcET, DcET_W, & ! Energy generated (discounted): total; Mgw h.
        ACV_1USD_W, & ! Avoided Carbon Value, discounted, valued at 1
        US$.
        ACV_SCC_W, & ! Avoided Carbon Value, discounted, valued at SCC
        ACV_TxCIEA_W, & ! Avoided Carbon Value, discounted, valued at Tx IEA
        ACV_TxCSS_W ! Avoided Carbon Value, discounted, valued at TX
        Stig.-Stern
    Real (kd) :: DcE(2015:2050+ls) ! Energy generated (discounted): annually
! variables Subroutine CapEx_Opex_Simul ( PTurb )
    Real (kd) :: CapOpEx ! total de gastos de capital y O&M,
    descontados.
    Real (kd), Dimension(2015:2050) :: &
        CorrInt, & !
        CapExMd, & ! Cap. expend
        CapEx, & ! Capital expenditure total
        OpEx, & ! Operating expenditure total -
        CapOpExa ! Capital and Op. Expend. en el año t, correspondientes
! variables Subroutine All_LCE
    Real (kd) :: LCE, & ! Levelised Cost of Energy
        LCEa(2015:2050) ! LCE t
! variables subrutina E_Inv_a
    Real (kd), Dimension(2015:2050) :: EIba, EIba_W !
! variables Subroutine Pw_Str_Cts, &
    Real (kd) :: StrEx, StrEx_W, & ! Storage Expenditure,
    total discounted: m. US $
        PStr(2014:2050), & ! $ por w.
        Strlw(2014:2050), Strlw_W(2014:2050), & !
        IbStr(1986-20:2050), & !
        StrExa(2015:2050) !
!
Contains
!
Subroutine Param_Assing
! param assign Energy paper;
ma_s = (/ 2.9666_kd, -0.186_kd /)
sda_s = (/ 0._kd, 0.013_kd /)
sdePTurb_s = 0.09_kd
End Subroutine Param_Assing
!
! Read starting & basic values.
!
Subroutine Read_Gen_Data
    Open (5, IOSTAT=Ios, action='read', status='old', &
        file='C:\lf9571\bin\Aveiro_2019\Data\Basic_data_wind.dat')

    If ( ios .NE. 0 ) STOP '*** INPUT FILE INCORRECTLY OPENED; PROGRAM STOPS ***'
    Read (5, '(3F10.2 /, 2F10.6/, F10.3, I4)') dep, PR, kgCO2, sdr1, sdr2, WACC, LoanM
    Read (5, '(L10)') LStr
    Read (5, '(2L10)') Prv, Errr
    Read (5, '(I10)') ScModel
    Read (5, '(4L10)') LgLCE_pdf, LgLCE_ACV, LgModPr, LgPlwbkd
    Read (5, '(2L10)') WCap, Wcorr
    !
    sdr(2015:2044) = sdr1
    sdr(2045:2050+ls) = sdr2
    dsf(2014) = 1._kd
    Do t = 2015, 2050+ls
        dsf(t) = dsf(t-1) / (1._kd + sdr(t))
    End do
    !
    Do t = 2015, 2050
        CorrInt(t) = WACC * ( 1._kd - 1._kd/(1._kd + sdr(t))*LoanM ) &
            / sdr(t) / ( 1._kd - 1._kd/(1._kd + WACC)*LoanM )
    End Do
    ! Capacity factor (series);

```

```

CfWind(2015) = .28_kd
Do ind = 2016, 2030
    CfWind(ind) = CfWind(ind-1) + 0.002_kd
End Do
CfWind(2031:) = .31_kd
End Subroutine Read_Gen_Data
!
! IEA Rmap, Stiglitz-Stern, survey de JCP
!
Subroutine Carbon_Cost
! *** Social Cost of Carbon, 2015, 6.7, 2050, 14.7_kd ; US $ Ton CO2; JCP, 2016
***
SCC(2015) = 6.7_kd
ZCC = (16.04_kd/7.31_kd)**(1._kd/35._kd)
Do ind = 2016, 2050+1s
    SCC(ind) = SCC(ind-1) * ZCC
End Do
! *** IEA - Rmap Irena) ***
! 2020, 13.5; 2030, 91.5; 2040, 140.0; 2050, 161.0;
ZCC = (13.5_kd - 0.0_kd) / 5._kd
TxCIEA(2015) = 0.0_kd
Do ind = 2016, 2020
    TxCIEA(ind) = TxCIEA(ind-1) + ZCC
End Do
ZCC = (91.5_kd - 13.5_kd) / 10._kd
Do ind = 2021, 2030
    TxCIEA(ind) = TxCIEA(ind-1) + ZCC
End Do
ZCC = (140.0_kd - 91.5_kd) / 10._kd
Do ind = 2031, 2040
    TxCIEA(ind) = TxCIEA(ind-1) + ZCC
End Do
ZCC = (161.0_kd - 140.0_kd) / 10._kd
Do ind = 2041, 2050
    TxCIEA(ind) = TxCIEA(ind-1) + ZCC
End Do
TxCIEA(2051:2050+1s) = TxCIEA(2050)
! *** Stiglitz - Stern. ***
! 2020, 60.0; 2030, 75.0;
ZCC = (60.0_kd - 0.0_kd) / 5._kd
TxCSS(2015) = 0.0_kd
Do ind = 2016, 2020
    TxCSS(ind) = TxCSS(ind-1) + ZCC
End Do
ZCC = (75.0_kd - 60.0_kd) / 10._kd
Do ind = 2021, 2030
    TxCSS(ind) = TxCSS(ind-1) + ZCC
End Do
TxCSS(2031:2050+1s) = TxCSS(2030)
End Subroutine Carbon_Cost
!
! => Mw.
!
Subroutine Cap_and_RMap ( Isel_IEA_Irena, Isel_acc_dly )
Integer, intent(in) :: Isel_IEA_Irena, Isel_acc_dly
! Roadmap
! select caso: IEA (1) o, Irena (2)
If ( Isel_IEA_Irena == 1 ) Then
    Open (11, IOSTAT=Ios, action='read', status='old', &
        file='C:\lf9571\bin\Aveiro_2019\Data\CWh_IEA_GW.dat')
Else if ( Isel_IEA_Irena == 2 ) Then
    Open (11, IOSTAT=Ios, action='read', status='old', &
        file='C:\lf9571\bin\Aveiro_2019\Data\CWh_Irena_GW.dat')
Else
    STOP '*** wrong case IEA/Irena selector value ***'
End If
If ( ios .NE. 0 ) STOP '*** INPUT FILE INCORRECTLY OPENED; PROGRAM STOPS ***'
! read Roadmap ; GW
Do ind = 2015, 2050
    Read(11, '(I4, F13.0)') Year(ind), Cap(ind)
End Do

```

```

!
! historic
Open (16, IOSTAT=Ios, action='read', status='old',                                     &
      file='C:\lf9571\bin\Aveiro_2019\Data\CW_historico_MW.dat')
If ( ios .NE. 0 ) STOP '*** INPUT FILE INCORRECTLY OPENED; PROGRAM STOPS ***'
! read; MW
Do ind = 1985, 2014
  Read(16,'(I4, F13.0)') Year(ind), Cap(ind)
End Do
! conversión a Mgw
Cap(2015:2050) = Cap(2015:2050) * 1000._kd
! *** recalcular el Rmp: acelerado o retrasado ***
Call Alt_Cap_Rmp ( Isel_acc_dly )
Cap(2051:2050+ls) = Cap(2050)
! *** series derivadas ***
LCap = Log(Cap(1985:2050))
DLCap = LCap(1986:2050) - LCap(1985:2049)
DCap = Cap(1986:2050) - Cap(1985:2049)
! inversión Mw.
Ib(1986-ls:1985) = 0._kd
Do t = 1986, 2050
  Ib(t) = Ib(t-ls) + DCap(t)
End do
Ib(2051:2050+ls) = 0._kd
Ib_W(2015:2050) = Ib(2015:2050)
Ceac = 0._kd
Do t = 2015, 2050+ls
  Ibac = Ib(t)
  Do ind = 1, ls-1
    If ( (t-ind) < (2015) ) Exit
    Ibac = Ibac + Ib(t-ind)*dep**ind
  End do
  Ce(t) = Ibac
  Ceac = Ceac + Ce(t)
End do
End Subroutine Cap_and_RMap
!
!
Subroutine Alt_Cap_Rmp ( Isel_acc_dly )
  Integer, intent(in) :: Isel_acc_dly
  !
  ! accelerated deployment
  If ( Isel_acc_dly == 1 ) Then
    IncRmCap = (Cap(2050) - Cap(2015)) / 15._kd
    Do ind = 2016, 2030
      Cap(ind) = Cap(ind-1) + IncRmCap
    End do
    Cap(2031:2050) = Cap(2030)
  ! decelerated deployment
  Else if ( Isel_acc_dly == 2 ) Then
    Cap(2016:2030) = Cap(2015)
    IncRmCap = (Cap(2050) - Cap(2015)) / 20._kd
    Do ind = 2031, 2050
      Cap(ind) = Cap(ind-1) + IncRmCap
    End do
  Else
    STOP '*** wrong case Accelerated/delayed selector value ***'
  End If
End Subroutine Alt_Cap_Rmp
!
! Discounted values: energy generated, and avoided CO2
!
Subroutine Dc_En_AvCV
  ! *** Energy generated (discounted): total, DcET_W, and annually, DcE(t) ***
  Do t = 2015, 2050+ls
    DcE(t) = Ce(t) * CfWind(t) * hy * PR * dsf(t)
  End Do
  ! Mw.h
  DcET = Sum(DcE) ; DcET_W = DcET
  ! *** carbon cost Avoided Carbon Value ; discounted present value ***
  ! valorado a 1 US$.

```

```

ACV_1USD_W = DcET * kgCO2
! valorado al Social Carbon Cost: ACV_SCC
ACV_SCC_W = Sum( DcE(:) * SCC(:) ) * kgCO2
! valorado al Tax de la IEA: ACV_TxCIEA
ACV_TxCIEA_W = Sum( DcE(:) * TxCIEA(:) ) * kgCO2
! valorado al Tax de Stiglitz y cía: ACV_TxCSS
ACV_TxCSS_W = Sum( DcE(:) * TxCSS(:) ) * kgCO2
End Subroutine Dc_En_AvCV
!
! Capital expend. y Operational Expend. para cada año,
!
Subroutine CapEx_Opex_Simul ( PTurb )
Real (kd), Intent(in) :: PTurb(2015:2050) !
!
! *** coste ***
Do t = 2015, 2050
    CapExMd(t) = Ib(t) * PTurb(t)
End Do
! *** Total gastos de capital y O&M descontados ***
CapOpEx = .0_kd
Do t = 2015, 2050
    CapEx(t) = CapExMd(t) * CorrInt(t)
    Opex(t) = 0.035_kd * CapExMd(t) &
        * ( 1._kd - 1._kd/(1._kd + sdr(t))*ls ) / sdr(t)
    CapOpExa(t) = (CapEx(t)+Opex(t))
    CapOpEx = CapOpEx + dsf(t) * CapOpExa(t)
End Do
End Subroutine CapEx_Opex_Simul
!
!
Subroutine Pw_Str_Cts
StrEx = 0._kd
StrExa = 0._kd
Strlw = 0._kd
If ( LStr ) Then
    PStr(2014) = 1.04_kd
    IbStr(1986-20:1985) = 0._kd
    Do t = 1986, 2050
        IbStr(t) = 0.25_kd * DCap(t) + IbStr(t-20)
    End do
    Do t = 2015, 2050
        PStr(t) = PStr(t-1) * 0.96122_kd
        StrExa(t) = IbStr(t) * PStr(t) * CorrInt(t)
        StrEx = StrEx + dsf(t) * StrExa(t)
    End Do
    Strlw = 0.25_kd * PStr
End If
Strlw_W = Strlw
StrEx_W = StrEx
End Subroutine Pw_Str_Cts
!
!
Subroutine All_LCE
LCEa(2015:2050) = (CapOpExa(2015:2050)+StrExa(2015:2050)) /
EIba(2015:2050)
LCE = (CapOpEx + StrEx) / DcET
End Subroutine All_LCE
!
! Mw.h, !
Subroutine E_Inv_a
Real (kd) :: Aux_str
!
Do t = 2015, 2050
    Aux_str = .0_kd
    Do ind = 0, ls-1
        Aux_str = Aux_str + CfWind(t+ind) * dsf(t+ind) * dep**ind
    End Do
    EIba(t) = Aux_str * Ib(t) * hy * PR / dsf(t)
End Do
EIba_W = EIba
End Subroutine E_Inv_a

```

```

!
! a
Subroutine Write_data
! writting file opened in main program; n° 30.
If ( ScModel .EQ. 2) Then
    Write(30, '( /A, 2F12.4)') '#medias ec. precio Turbinas :', ma_s
    Write(30, '(A, 2F12.4)') '#dev. est. parámetros ec. precio Turbinas :',
    sda_s
    Write(30, '(A, F12.4)') '#dev. est. ec. precio Turbinas :',
    sdePTurb_s
Else if ( (ScModel .EQ. 3) .OR. (ScModel .EQ. 4) ) Then
    Write(30, '( /A)') '# LR decreciente no estocástica'
End If
!
Write(30, '(A, 5F12.4)') '#valor inicial en 2014 de Cap', Cap(2014)
Write(30, '(A, 3F14.3)') '#depreciación, Performance Ratio, kgCO2', dep, PR, kgCO2
Write(30, '(A, I6)') '#Life Span de los Turbinas', ls
Write(30, '(A, 2F10.6)') '#Social Discount Rate', sdr1, sdr2
Write(30, '(A, F10.3, I4)') '#WACC, Loan Maturity', WACC, LoanM
Write(30, '(A, L2)') '#Generación aleatoria de los parámetros = ', Prv
Write(30, '(A, L2)') '#Generación aleatoria de los errs. de las ecs. = ',
Errr
Write(30, '(A, L2)') '#Storage costs tenido en cuenta = ', LStr
Write(30, '(A, I6)') '#Modelo Precio Turbinas (1, old; 2, simple; 3, LR dec.; 4, LR
+dec.', ScModel
!
If ( WCap ) Then
    Write(30, '(2/,A/)') '#Capacidad Energética del Roadmap (Gw.): Teórica y
    Efectiva '
    Do t = 2015, 2050+ls
        Write(30, '(I6, 2E14.4)') t, (Cap(t)-Cap(2014))*0.001_kd, Ce(t)*0.001_kd
    End Do
    Write(30, '( /,A/)') '#Capacidad Energética: Histórica y Roadmap (Gw.) '
    Do t = 1985, 2050
        Write(30, '(I6, E14.4)') t, Cap(t)*0.001_kd
    End Do
End If
!
If ( WCorr ) Then
    Write(30, '(2/,A/)') '# factor de corrección para el coste del capital'
    Do t = 2015, 2050
        Write(30, '(E14.6)') CorrInt(t)
    End Do
End If
End Subroutine Write_data
!
! generate random variables.
Subroutine Gen_RV_smpf
Call Random_Number ( U01_Lahey_s )
Gauss_01_s = Gauss_ms( U01_Lahey_s )
If ( .NOT. Prv ) Gauss_01_s(1:2) = 0._kd
If ( .NOT. Errr ) Gauss_01_s(3:2+nf) = 0._kd
as_s = ma_s + sda_s * Gauss_01_s(1:2)
ePTurb_s = sdePTurb_s * Gauss_01_s(3:2+nf)
End Subroutine Gen_RV_smpf
!
! model equations;
! USD/W, y el Cap en MW
Subroutine Model_ecs_smpf (inf)
Integer, Intent(in) :: inf
LPTurb(inf) = ma_s(1) + as_s(2) * LCap(inf) + ePTurb_s(inf)
End Subroutine Model_ecs_smpf
!
!
Subroutine PTurb_LRdec (inf)
Integer, Intent(in) :: inf
Real (kd) :: gaLR, gbLR, ggLR, caj2015
gaLR = 0.03_kd * (1._kd - real((inf-2015),kd)/35._kd)
gbLR = 0.24_kd * (1._kd - real((inf-2015),kd)/35._kd)
ggLR = 0.06_kd * (1._kd - real((inf-2015),kd)/35._kd)
caj2015 = 0._kd

```

```

If ( inf == 2015) caj2015 = 0.09553_kd
LPTurb(inf) = caj2015 + LPTurb(inf-1) - gaLR*DLCap(inf) - gbLR*DLCap(inf-1) -
ggLR*DLCap(inf-2)
End Subroutine PTurb_LRdec
!
Subroutine PTurb_LRdec2 (inf)
Integer, Intent(in) :: inf
Real (kd) :: gaLR, caj2015
gaLR = 0.33_kd * (36._kd/real((inf-2014),kd)-1._kd) / 35._kd
caj2015 = 0._kd
If ( inf == 2015) caj2015 = 0.08008_kd
LPTurb(inf) = caj2015 + LPTurb(inf-1) - gaLR * DLCap(inf-1)
End Subroutine PTurb_LRdec2
!
!
Subroutine PTurb_LRdec2_wind (inf)
Integer, Intent(in) :: inf
Real (kd) :: gaLR
gaLR = 0.186_kd * (36._kd/real((inf-2014),kd)-1._kd) / 35._kd
If ( inf == 2015) then
! esto es para que en 2015, PTurb sea el correcto.
LPTurb(inf) = log(1.750_kd)
Else if ( inf .GT. 2015 ) Then
LPTurb(inf) = LPTurb(inf-1) - gaLR * DLCap(inf-1)
End If
End Subroutine PTurb_LRdec2_wind
!
! Infrastructure Wind
Subroutine Infrastructure_Wind ( Isel_IEA_Irena, Isel_acc_dly )
Integer, intent(in) :: Isel_IEA_Irena, Isel_acc_dly
Call Param_Assing
Call Read_Gen_Data
Call Cap_and_RMap ( Isel_IEA_Irena, Isel_acc_dly )
Call Carbon_Cost
Call Dc_En_AvCV
Call E_Inv_a
Call Pw_Str_Cts
write(*,*) 'PASA 1 Infrastructure_Wind'
! rescale to Gw.
Cap = Cap * .001_kd ; DCap = DCap * .001_kd
Call Write_data
End Subroutine Infrastructure_Wind
!
! 'simul - proper'
Subroutine Wind_Simul ( nsim, PTurb_W, LCEas_W, LCEs_W, CapOpExs_W ) &
Integer, intent(in) :: nsim
Real (kd), Dimension(nsim,2015:2050) :: PTurb_W, LCEas_W
Real (kd), Dimension(nsim) :: LCEs_W, CapOpExs_W
Intent(out) :: PTurb_W, LCEas_W, LCEs_W, CapOpExs_W
!
Do ind = 1, nsim
Call Gen_RV_smpf !
Do inf = 2015, 2050
If ( ScModel .EQ. 2) Then
Call Model_ecs_smpf (inf) !
Else if ( ScModel .EQ. 3) Then
Call PTurb_LRdec (inf) !
Else if ( ScModel .EQ. 4) Then
Call PTurb_LRdec2 (inf) !
End If
End Do
PTurb_W(ind,:) = exp(LPTurb(2015:2050))
!
Call CapEx_Opex_Simul ( PTurb_W(ind,:) )
CapOpExs_W(ind) = CapOpEx
! LCE
Call All_LCE
! $ per Mwh.
LCEas_W(ind,:) = LCEa(:) * 10._kd**6
LCEs_W(ind) = LCE * 10._kd**6

```

```

End Do
End Subroutine Wind_Simul
!
End Module Simul_Wind_specf_Support
!
!   PV routines
!
Module Simul_PV_specf_Support
!
Use Simul_PV_Wind_Stat_Support
Use Param_y_Ctes
!
Implicit none
!
Private
Public :: StrEx_PV, & ! variables
        BoSlw_PV, Isrlw_PV, Strlw_PV, & !
        Ib_PV, &
        DcET_PV, EIba_PV, &
        ACV_lUSD_PV, ACV_SCC_PV, ACV_TxCIEA_PV, ACV_TxCSS_PV, &
        Infrastructure_PV, PV_Simul ! subroutines

!
! n° of simulations y life span.
Integer, parameter :: nf = 36, & ! number of simulated periods.
                    ls = 30    ! v

! datos
Real (kd) :: &
        ma(4), & ! means parameters a.
        cteajPm = 0.26_kd, & !
        mb(5), & ! means parameters b.
        mrho, & ! mean rho.
        sda(4), & ! s.d. a's
        sdb(5), & ! s.d. b's.
        sdrho, & ! s.d. rho
        sdepm, & ! s.d. error ec. module prices.
        sdeps, & ! s.d. error ec. silicon prices.
        Pm0, Ps0 ! last obsv. prices silicon and modules.

! modelo de Pm simplificado.
Real (kd) :: &
        epm_s(2015:2050), & ! error, module price equation.
        as_s(2), & ! simulated values a.
        ma_s(2), & ! means parameters a.
        cteajPm_s, & ! cte. ajuste media ec. Precio módulos.
        sda_s(2), & ! s.d. a's
        sdepm_s ! s.d. error ec. module prices.

!
Real (kd) :: as(4), & ! simulated values a.
             bs(5), & ! simulated values b.
             rhos ! simulated values rho

! starting values, capacity and energy prices.
Real (kd), dimension(2014:2050) :: &
        Pe, LPe, & ! energy prices.
        ! simulated values at 0
        upm, & ! AR error ec. module prices.
        LPs, & ! log silicon price.
        LPm ! log module price.

! simulated values: all variables
Real (kd), dimension(2015:2050) :: &
        eps, & ! error, silicon price equation.
        epm ! error, module price equation.

! random numbers for the simulation.
Real (kd), dimension(2*nf+10) :: U01_Lahey, Gauss_01
Real (kd), dimension(2+nf) :: U01_Lahey_s, Gauss_01_s
!
Integer :: ios, & ! 'open' error checking
          ind, inf, t ! other auxiliary indexes.
!
Character (len=55), Dimension(3) :: InfErr
!
! variables Subroutine Read_Gen_Data

```

```

Integer :: LoanM,          & !
          ScModel          !
Real (kd) :: dep,          & !
          PR,              & ! Performance Ratio.
          kgCO2,           & ! n° de kg. de CO2 por kWh., IEA
          sdr1, sdr2, & ! Social Discount Rate
          WACC             ! Weighted Average Cost of Capital.
Real (kd) :: CfPV(2014:2050+ls), & !
          sdr(2015:2050+ls), & ! Social Discount Rate;
          dsf(2014:2050+ls)    !
Logical :: LStr,           & ! storage costs
          Prv,             & ! generate random parameters.
          Errr,            & ! generate random equation errors.
          WCap             ! Escribir los datos de capital.
! variables Subroutine Carbon_Cost
Real (kd) :: ZCC,          & ! var aux.
          SCC(2015:2050+ls), & ! Social Cost of Carbon; US$ Ton CO2eq
          TxCIEA(2015:2050+ls), & ! Carbon Tax IEA; US$ Ton CO2eq
          TxCSS(2015:2050+ls)    ! Carbon Tax Stiglitz-Stern (repot WB); US$
          Ton CO2eq
! variables Subroutine Cap_and_RMap
Integer :: year(2015:2050) !
Real (kd) :: Ceac,         & !
          Ibac             !
Real (kd) :: Cap(1977:2050+ls), & ! Capital PV
          LCap(1977:2050), & !
          DLCap(1978:2050), & !
          DCap(1978:2050), & !
          Ib(1978-1s:2050+ls), & !
          Ib_PV(2015:2050), & !
          Ce(2015:2050+ls)    !
! variables Subroutine Alt_Cap_Rmp
Real (kd) :: IncRmCap      !
! variables de la Subroutine Dc_En_AvCV
Integer, parameter :: hy = 8760 ! 365*24
Real (kd) :: DcET, DcET_PV, & ! Energy generated (discounted): total; Mgw h.
          ACV_1USD_PV, & ! Avoided Carbon Value, discounted, valued at 1
          US$.
          ACV_SCC_PV, & ! Avoided Carbon Value, discounted, valued at SCC
          ACV_TxCIEA_PV, & ! Avoided Carbon Value, discounted, valued at Tx IEA
          ACV_TxCSS_PV    ! Avoided Carbon Value, discounted, valued at TX
          Stig.-Stern
Real (kd) :: DcE(2015:2050+ls) ! Energy generated (discounted): annually
! variables Subroutine CapEx_Opex_Simul ( Pm ) ; excepto Pm
Real (kd) :: CapOpEx      ! total de gastos de capital y O&M,
          descontados.
Real (kd), Dimension(2015:2050) :: &
          CapExMd, & ! Cap. expend
          CapEx, & ! Capital expenditure total
          OpEx, & ! Operating expenditure total
          CapOpExa ! Capital and Op. Expend. en el año t,
! variables Subroutine CapEx_Opex
Real (kd), Dimension(2015:2050) :: CorrInt, & !
          CapExIsr, & !
          BoSlw_, BoSlw_PV, & ! BoS 1 watt.
          Isrlw, Isrlw_PV, & ! Coste inversor 1
          watt.
          CapExBoS_ ! Cap. expend en el resto
          de capítulos del bos
! variables Subroutine All_LCE
Real (kd) :: LCE, & ! Levelised Cost of Enery
          LCEa(2015:2050) ! LCE t
! variables subrutina E_Inv_a
Real (kd), Dimension(2015:2050) :: EIba, EIba_PV ! Energía total
! variables Subroutine Pw_Str_Cts, &
Real (kd) :: StrEx, StrEx_PV, & ! Storage Expenditure,
          total discounted: m. US $
          PStr(2014:2050), & ! precio en $ por w. de
          almacenamiento.
          Strlw(2014:2050), Strlw_PV(2014:2050), & ! 1 watt. de
          PV

```



```

IbStr(1978-20:2050),           & !
StrExa(2015:2050)             ! Storage Expenditure,
annual: m. US $

```

```

!
Contains
!

```

```

Subroutine Param_Assing

```

```

ma = (/ 4.966_kd, -.411_kd, .274_kd, .208_kd /)
mb = (/ 3.112_kd, 2.722_kd, -.459_kd, 1.067_kd, .528_kd /)
mrho = .637_kd
sda = (/ .341_kd, .0372_kd, .0832_kd, .0828_kd /)
sdb = (/ .777_kd, .833_kd, .081_kd, .26_kd, .116_kd /)
sdrho = .164_kd
sdep_m = .104_kd
sdeps = .198_kd

```

```

ma_s = (/ 3.6025_kd, -.320_kd /)
sda_s = (/ 0.114_kd, 0.016_kd /)
sdep_m_s = 0.22_kd
! extensión referee
sda_s = (/ 0.114_kd, 0.016_kd /) * 1.5_kd
sdep_m_s = 0.22_kd * 1.5_kd

```

```

End Subroutine Param_Assing

```

```

!
! Read starting & basic values.
!

```

```

Subroutine Read_Gen_Data

```

```

Open (20, IOSTAT=Ios, action='read', status='old',           &
file='C:\lf9571\bin\Aveiro_2019\Data\A_Basic_data_PV.dat')

```

```

If ( ios .NE. 0 ) STOP '*** INPUT FILE INCORRECTLY OPENED; PROGRAM STOPS ***'
Read (20, '(3F10.2 /, 2F10.6/, F10.3, I4)') dep, PR, kgCO2, sdr1, sdr2, WACC, LoanM
Read (20, '(L10)') LStr
Read (20, '(2L10)') Prv, Errr
Read (20, '(I10)') ScModel
Read (20, '(L10)') WCap
!
sdr(2015:2044) = sdr1
sdr(2045:2050+ls) = sdr2
dsf(2014) = 1._kd
Do t = 2015, 2050+ls
    dsf(t) = dsf(t-1) / (1._kd + sdr(t))
End do
! Capacity factor (series).
CfPV(2014) = .19_kd
Do ind = 2015, 2030
    CfPV(ind) = CfPV(ind-1) + 0.00375_kd
End Do
CfPV(2031:) = .25_kd
! generate energy prices: all years;
Pe(2014) = 0.8229_kd
Do ind = 2015, 2050
    Pe(ind) = Pe(ind-1) * (1._kd + 0._kd)
End Do
LPe = Log(Pe)
!
Pm0 = 0.705 ; Ps0 = 0.075966_kd ; upm(2014) = 0._kd
LPm(2014) = Log(Pm0)
LPs(2014) = Log(Ps0)

```

```

End Subroutine Read_Gen_Data

```

```

!
! IEA Rmap, Stiglitz-Stern, survey JCP.
!

```

```

Subroutine Carbon_Cost

```

```

! *** Social Cost of Carbon, 2015, 6.7, 2050, 14.7_kd ; creo, US $ Ton CO2; JCP,
2016 ***
SCC(2015) = 6.7_kd
ZCC = (16.04_kd/7.31_kd)**(1._kd/35._kd)
Do ind = 2016, 2050+ls
    SCC(ind) = SCC(ind-1) * ZCC

```

```

End Do
! *** IEA - Rmap Irena) ***
! 2020, 13.5; 2030, 91.5; 2040, 140.0; 2050, 161.0;
ZCC = (13.5_kd - 0.0_kd) / 5._kd
TxCIEA(2015) = 0.0_kd
Do ind = 2016, 2020
    TxCIEA(ind) = TxCIEA(ind-1) + ZCC
End Do
ZCC = (91.5_kd - 13.5_kd) / 10._kd
Do ind = 2021, 2030
    TxCIEA(ind) = TxCIEA(ind-1) + ZCC
End Do
ZCC = (140.0_kd - 91.5_kd) / 10._kd
Do ind = 2031, 2040
    TxCIEA(ind) = TxCIEA(ind-1) + ZCC
End Do
ZCC = (161.0_kd - 140.0_kd) / 10._kd
Do ind = 2041, 2050
    TxCIEA(ind) = TxCIEA(ind-1) + ZCC
End Do
TxCIEA(2051:2050+ls) = TxCIEA(2050)
! *** Stiglitz - Stern. ***
! 2020, 60.0; 2030, 75.0;
ZCC = (60.0_kd - 0.0_kd) / 5._kd
TxCSS(2015) = 0.0_kd
Do ind = 2016, 2020
    TxCSS(ind) = TxCSS(ind-1) + ZCC
End Do
ZCC = (75.0_kd - 60.0_kd) / 10._kd
Do ind = 2021, 2030
    TxCSS(ind) = TxCSS(ind-1) + ZCC
End Do
TxCSS(2031:2050+ls) = TxCSS(2030)
End Subroutine Carbon_Cost
!
! => Mw.
!
Subroutine Cap_and_RMap ( Isel_IEA_Irena, Isel_acc_dly )
Integer, intent(in) :: Isel_IEA_Irena, Isel_acc_dly
! Roadmap
! select caso: IEA (1) o, Irena (2)
If ( Isel_IEA_Irena == 1 ) Then
    Open (23, IOSTAT=Ios, action='read', status='old', &
        file='C:\lf9571\bin\Aveiro_2019\Data\A_IEA_PVRm_2050.dat')
Else if ( Isel_IEA_Irena == 2 ) Then
    Open (23, IOSTAT=Ios, action='read', status='old', &
        file='C:\lf9571\bin\Aveiro_2019\Data\A_Irena_PVRm_2050.dat')
Else
    STOP '*** wrong case IEA/Irena selector value ***'
End If
If ( ios .NE. 0 ) STOP '*** INPUT FILE INCORRECTLY OPENED; PROGRAM STOPS ***'
! read Roadmap ; GW
Do ind = 2015, 2050
    Read(23, '(I4, F13.0)') Year(ind), Cap(ind)
End Do
! conversión a Mgw
Cap(2015:2050) = Cap(2015:2050) * 1000._kd
Cap(2051:2050+ls) = Cap(2050)
! *** Cap. histórico ***
Cap(1977:2014) = (/ 0.6_kd, 1.6_kd,
3.0_kd, 6.3_kd, 11.7_kd, 19.4_kd, 33.9_kd,
&
51.4_kd, 70.8_kd, 91.8_kd, 116.7_kd, 148.2_kd, 186.1_kd, 228.8_kd,
&
276.9_kd, 331.1_kd, 386.8_kd, 447.8_kd, 519.3_kd, 601.9_kd, 716.0_kd,
&
850.8_kd, 1026.3_kd, 1278.3_kd, 1631.2_kd, 2136.1_kd, 2811.3_kd, 3861.2_kd,
&

```

```

5268.9_kd,7253.4_kd,10353.5_kd,15845.3_kd,23758.5_kd,41160.8_kd,
&
64740.2_kd,90801.9_kd,123802.0_kd,157760.9_kd /)
! *** recalcular Rmp: ***
Call Alt_Cap_Rmp ( Isel_acc_dly )
! *** series derivadas ***
LCap = Log(Cap(1977:2050))
DLCap = LCap(1978:2050) - LCap(1977:2049)
DCap = Cap(1978:2050) - Cap(1977:2049)
! Mw.
Ib(1978-ls:1977) = 0._kd
Do t = 1978, 2050
    Ib(t) = Ib(t-ls) + DCap(t)
End do
Ib(2051:2050+ls) = 0._kd
Ib_PV(2015:2050) = Ib(2015:2050)
Ceac = 0._kd
Do t = 2015, 2050+ls
    Ibac = Ib(t)
    Do ind = 1, ls-1
        If ( (t-ind) < (2015) ) Exit
        Ibac = Ibac + Ib(t-ind)*dep**ind
    End do
    Ce(t) = Ibac
    Ceac = Ceac + Ce(t)
End do
End Subroutine Cap_and_RMap
!
!
Subroutine Alt_Cap_Rmp ( Isel_acc_dly )
Integer, intent(in) :: Isel_acc_dly
!
! accelerated deployment
If ( Isel_acc_dly == 1 ) Then
    IncRmCap = (Cap(2050) - Cap(2015)) / 15._kd
    Do ind = 2016, 2030
        Cap(ind) = Cap(ind-1) + IncRmCap
    End do
    Cap(2031:2050) = Cap(2030)
! decelerated deployment
Else if ( Isel_acc_dly == 2 ) Then
    Cap(2016:2030) = Cap(2015)
    IncRmCap = (Cap(2050) - Cap(2030)) / 20._kd
    Do ind = 2031, 2050
        Cap(ind) = Cap(ind-1) + IncRmCap
    End do
Else
    STOP ' *** wrong case Accelerated/delayed selector value *** '
End If
End Subroutine Alt_Cap_Rmp
!
! Discounted values: energy generated, and avoided CO2
!
Subroutine Dc_En_AvCV
! *** Energy generated (discounted): total, DcET, and annually, DcE(t) ***
DcE = Ce * (CfPV(2015:) * dsf(2015:)) * hy * PR
! Mw.h
DcET = Sum(DcE) ; DcET_PV = DcET
! *** carbon cost Avoided Carbon Value ; discounted present value ***
ACV_lUSD_PV = DcET * kgCO2
ACV_SCC_PV = Sum( DcE * SCC ) * kgCO2
ACV_TxCIEA_PV = Sum( DcE * TxCIEA ) * kgCO2
ACV_TxCSS_PV = Sum( DcE * TxCSS ) * kgCO2
End Subroutine Dc_En_AvCV
!
! Capital expend. y Operational Expend. para cada año,
!
Subroutine CapEx_Opex_Simul ( Pm )
Real (kd), Intent(in) :: Pm(2015:2050) !
!
! *** coste módulos PV ***

```

```

CapExMd = Ib(2015:2050) * Pm
! *** Total gastos capital O&M descontados ***
CapOpEx = .0_kd
Do t = 2015, 2050
    CapEx(t) = (CapExMd(t) + CapExIsr(t) + CapExBoS_(t)) * CorrInt(t)
    ! Opex
    Opex(t) = 0.015_kd * CapEx(t) &
              * ( 1._kd - 1._kd/(1._kd + sdr(t))*ls ) / sdr(t)
    CapOpExa(t) = (CapEx(t)+Opex(t))
    CapOpEx = CapOpEx + dsf(t) * CapOpExa(t)
End Do
End Subroutine CapEx_Opex_Simul
!
!
Subroutine CapEx_Opex
    Real (kd) :: Resid,      & ! BoS Residencial.
                  Comm,      & ! BoS Comercial.
                  Util,      & ! BoS Utilities.
                  TasaBoS_   ! Breyer.

    !
    ! *** Inversores: LR del 20% (Breyer) ***
    ! precio 2014, 0.132 USD p.w.; LR del 20% con la capacidad.
    Do t = 2015, 2050
        Isrlw(t) = 0.132_kd * (Cap(2014)/Cap(t))*0.322_kd
        CapExIsr(t) = Ib(t) * Isrlw(t)
    End Do
    ! *** Resto BoS (BoS_) por sectores ***
    Util = 0.408_kd ! USD p.w.
    Comm = 0.612_kd ! USD p.w.; 50% más que util.
    Resid = 0.8976_kd ! USD p.w.; 120% más que util.
    TasaBoS_ = .9809_kd
    Do t = 2015, 2050
        Resid = Resid * TasaBoS_
        Comm = Comm * TasaBoS_
        Util = Util * TasaBoS_
        BoSlw_(t) = .25_kd*Resid + .25*Comm + .5*Util
        CapExBoS_(t) = Ib(t) * BoSlw_(t)
    End Do
    BoSlw_PV = BoSlw_
    ! *** Total gastos de capital y O&M descontados ***
    Do t = 2015, 2050
        CorrInt(t) = WACC * ( 1._kd - 1._kd/(1._kd + sdr(t))*LoanM ) &
                      / sdr(t) / ( 1._kd - 1._kd/(1._kd + WACC)*LoanM )
    End Do
End Subroutine CapEx_Opex
!
!
Subroutine Pw_Str_Cts
    StrEx = 0._kd
    StrExa = 0._kd
    Strlw = 0._kd
    If ( LStr ) Then
        PStr(2014) = 1.04_kd
        IbStr(1978-20:1977) = 0._kd
        Do t = 1978, 2050
            IbStr(t) = 0.25_kd * DCap(t) + IbStr(t-20)
        End do
        Do t = 2015, 2050
            PStr(t) = PStr(t-1) * 0.96122_kd
            StrExa(t) = IbStr(t) * PStr(t) * CorrInt(t)
        End Do
        StrEx = Sum( dsf(2015:2050) * StrExa )
        Strlw = 0.25_kd * PStr
    End If
    StrEx_PV = StrEx
    Strlw_PV = Strlw
End Subroutine Pw_Str_Cts
!
!
Subroutine All_LCE
    LCEa(2015:2050) = (CapOpExa(2015:2050)+StrExa(2015:2050)) / EIba(2015:2050)

```

```

LCE = (CapOpEx + StrEx) / DcET
End Subroutine All_LCE
!
! Mw.h,
!
Subroutine E_Inv_a
  Real (kd) :: Aux_str
  !
  Do t = 2015, 2050
    Aux_str = .0_kd
    Do ind = 0, ls-1
      Aux_str = Aux_str + CfpV(t+ind) * dsf(t+ind) * dep**ind
    End Do
    EIba(t) = Aux_str * Ib(t) * hy * PR / dsf(t)
  End Do
  EIba_PV = EIba
End Subroutine E_Inv_a
!
!
Subroutine Write_data
  ! the 'write file' is opened in the main program: number, 30.
  If ( ScModel .EQ. 1) Then
    Write(30, '( /A, 4F12.4)') '#medias ec. precio módulos :', ma
    Write(30, '(A, F12.4)') '#cte.de ajuste ec. precio módulos :', cteajPm
    Write(30, '(A, 5F12.4)') '#medias ec. precio silicio :', mb
    Write(30, '(A, F12.4)') '#rho . ec. precio módulos :', mrho
    Write(30, '(A, 4F12.4)') '#desviación estándar parámetros ec. precio módulos
    :', sda
    Write(30, '(A, 5F12.4)') '#desviación estándar parámetros ec. precio silicio
    :', sdb
    Write(30, '(A, F12.4)') '#desviación estándar rho :', sdrho
    Write(30, '(A, F12.4)') '#desviación std. ec. precio módulos :', sdepm
    Write(30, '(A, F12.4)') '#desviación std. ec. precio silicio :', sdeps
  Else if ( ScModel .EQ. 2) Then
    Write(30, '( /A, 2F12.4)') '#medias ec. precio módulos :', ma_s
    Write(30, '(A, 2F12.4)') '#dev. est. parámetros ec. precio módulos :', sda_s
    Write(30, '(A, F12.4)') '#dev. est. ec. precio módulos :', sdepm_s
  Else if ( (ScModel .EQ. 3) .OR. (ScModel .EQ. 4) ) Then
    Write(30, '( /A)') '# LR decreciente no estocástica'
  End If
  !
  Write(30, '(A, 5F12.4)') '#valores iniciales en 2014 de Cap, Pe, Pm, Ps, upm
  :', &
    Cap(2014), Pe(2014), Pm0, Ps0, upm(2014)
  Write(30, '(A, 3F14.3)') '#depreciación, Performance Ratio, kgCO2 :', dep, PR,
  kgCO2
  Write(30, '(A, I6)') '#Life Span de los módulos :', ls
  Write(30, '(A, 2F10.6)') '#Social Discount Rate :', sdr1, sdr2
  Write(30, '(A, F10.3, I4)') '#WACC, Loan Maturity :', WACC, LoanM
  Write(30, '(A, L2)') '#Generación aleatoria de los parámetros = ', Prv
  Write(30, '(A, L2)') '#Generación aleatoria de los errs. de las ecs. = ',
  Errr
  Write(30, '(A, L2)') '#Storage costs tenido en cuenta = ', LStr
  Write(30, '(A, I6)') '#Modelo Precio módulos (1, old; 2, simple; 3, LR dec.; 4, LR
  +dec. :', ScModel
  !
  If ( WCap ) Then
    Write(30, '(2/,A/)') '#Capacidad Energética del Roadmap (Gw.): Teórica y
    Efectiva '
    Do t = 2015, 2050+ls
      Write(30, '(I6, 2E14.4)') t, (Cap(t)-Cap(2014))*0.001_kd, Ce(t)*0.001_kd
    End Do
    Write(30, '(/,A/)') '#Capacidad Energética: Histórica y Roadmap (Gw.) '
    Do t = 1977, 2050
      Write(30, '(I6, E14.4)') t, Cap(t)*0.001_kd
    End Do
  End If
End Subroutine Write_data
!
! generate random variables.
Subroutine Gen_RV

```

```

Call Random_Number ( U01_Lahey )
Gauss_01 = Gauss_ms( U01_Lahey )
If ( .NOT. Prv ) Gauss_01(1:10) = 0._kd
If ( .NOT. Errr ) Gauss_01(11:10+2*nf) = 0._kd
as = ma + sda * Gauss_01(1:4)
bs = mb + sdb * Gauss_01(5:9)
rhos = mrho + sdrho * Gauss_01(10)
eps = sdeps * Gauss_01(11:10+nf)
epm = sdepm * Gauss_01(11+nf:10+2*nf)
End Subroutine Gen_RV
!
! model equations.
Subroutine Model_ecs (inf)
Integer, Intent(in) :: inf
LPs(inf) = bs(1) + bs(2)*DLCap(inf) + bs(3)*LCap(inf-1) + bs(4)*LPe(inf) &
+ bs(5)*LPs(inf-1) + eps(inf)
upm(inf) = mrho*upm(inf-1) + epm(inf)
LPm(inf) = cteajPm &
+ as(1) + as(2)*LCap(inf-1) + as(3)*(LPs(inf) - LPs(inf-1)) &
+ as(4)*LPs(inf-1) + upm(inf)
End Subroutine Model_ecs
!
! generate random variables.
Subroutine Gen_RV_smpf
Call Random_Number ( U01_Lahey_s )
Gauss_01_s = Gauss_ms( U01_Lahey_s )
If ( .NOT. Prv ) Gauss_01_s(1:2) = 0._kd
If ( .NOT. Errr ) Gauss_01_s(3:2+nf) = 0._kd
as_s = ma_s + sda_s * Gauss_01_s(1:2)
epm_s = sdepm_s * Gauss_01_s(3:2+nf)
End Subroutine Gen_RV_smpf
!
! model equations; simplificada
Subroutine Model_ecs_smpf (inf)
Integer, Intent(in) :: inf
LPm(inf) = as_s(1) + as_s(2) * LCap(inf) + epm_s(inf)
End Subroutine Model_ecs_smpf
!
!
Subroutine Pm_LRdec (inf)
Integer, Intent(in) :: inf
Real (kd) :: gaLR, gbLR, ggLR, caj2015
gaLR = .03_kd * (1._kd - real((inf-2015),kd)/35._kd)
gbLR = .24_kd * (1._kd - real((inf-2015),kd)/35._kd)
ggLR = .06_kd * (1._kd - real((inf-2015),kd)/35._kd)
caj2015 = 0._kd
If ( inf == 2015 ) caj2015 = 0.09553_kd
LPm(inf) = caj2015 + LPm(inf-1) - gaLR*DLCap(inf) - gbLR*DLCap(inf-1) -
ggLR*DLCap(inf-2)
End Subroutine Pm_LRdec
!
!
Subroutine Pm_LRdec2 (inf)
Integer, Intent(in) :: inf
Real (kd) :: gaLR, caj2015
gaLR = 0.33_kd * (36._kd/real((inf-2014),kd)-1._kd) / 35._kd
caj2015 = 0._kd
If ( inf == 2015 ) caj2015 = 0.08008_kd
LPm(inf) = caj2015 + LPm(inf-1) - gaLR * DLCap(inf-1)
End Subroutine Pm_LRdec2
!
! Infrastructure PV
Subroutine Infrastructure_PV ( Isel_IEA_Irena, Isel_acc_dly )
Integer, intent(in) :: Isel_IEA_Irena, Isel_acc_dly
Call Param_Assing
Call Read_Gen_Data
Call Cap_and_RMap ( Isel_IEA_Irena, Isel_acc_dly )
Call Carbon_Cost
Call Dc_En_AvCV
Call CapEx_Opex
Call E_Inv_a

```

```

    Call Pw_Str_Cts
    write(*,*) 'PASA 1 Infrastructure_PV'
    ! rescale to Gw.
    Cap = Cap * .001_kd ; DCap = DCap * .001_kd
    Call Write_data
End Subroutine Infrastructure_PV
!
! 'simul - proper'
Subroutine PV_Simul ( nsim,                                     &
                    Pm, LCEas_PV, LCEs_PV, CapOpExs_PV )
    Integer, intent(in) :: nsim
    Real (kd), Dimension(nsim,2015:2050) :: Pm, LCEas_PV
    Real (kd), Dimension(nsim) :: LCEs_PV, CapOpExs_PV
    Intent(out) :: Pm, LCEas_PV, LCEs_PV, CapOpExs_PV
    !
    Do ind = 1, nsim
        Call Gen_RV_smpf !
        Do inf = 2015, 2050
            If ( ScModel .EQ. 2) Then
                Call Model_ecs_smpf (inf)
            Else if ( ScModel .EQ. 3) Then
                Call Pm_LRdec (inf)
            Else if ( ScModel .EQ. 4) Then
                Call Pm_LRdec2 (inf)
            End If
        End Do
        Pm(ind,:) = exp(LPm(2015:2050))
        !
        Call CapEx_Opex_Simul ( Pm(ind,:) )
        CapOpExs_PV(ind) = CapOpEx
        ! LCE .
        Call All_LCE
        ! $ per Mwh.
        LCEas_PV(ind,:) = LCEa(:) * 10._kd**6
        LCEs_PV(ind) = LCE * 10._kd**6
    End Do
End Subroutine PV_Simul
!
End Module Simul_PV_specf_Support
!
! *****
! *** main PV simulation program. ***
! *****
!
Program Simul_PV_Wind_Irena_Iea
!
Use Param_y_Ctes
! statistics, general purpose
Use Simul_PV_Wind_Stat_Support
! Wind
Use Simul_Wind_specf_Support
! PV
Use Simul_PV_specf_Support
!
Implicit none
!
Integer, parameter :: nsim = 100000, nf = 36
Integer :: ios, & !
    Isel_IEA_Irena, & ! selector de caso: 1, IEA; 2, Irena
    Isel_acc_dly & ! selector de caso: 1, despliegue acelerado; 2, retardado.
Logical :: LgLCE_pdf, LgACV, LgCapExs, LgCapCst, LgClWbkd
!
! all simulated values.
! annual
Real (kd), dimension(nsim,2015:2050) :: PTurb, LCEas_W, & ! Simulated: Wind,
    Pm, LCEas_PV, & ! Simulated: PV,
    LCEas, & ! Total,
    Pm_Turb_w & ! coste 1W
! aggregated
Real (kd), dimension(nsim) :: LCEs_W, CapOpExs_W, & ! Simulated: Wind,

```

```

LCEs_PV, CapOpExs_PV,      & ! Simulated: PV,
LCEs, CapOpExs,           & ! Total
LCEas_v                   ! auxiliar

! otras auxiliares.
Real (kd), dimension(2015:2050) :: w_Ib_W, w_Ib_PV      !
!
Real (kd) :: StrEx
!
! internal variables required to calculate all derived stats. measures.
! instrumental vector to call the histogram and Descrip_Stat.
Real (kd) :: Pmv(nsim), dummy, LCEm(nsim,1), CapOpExm(nsim,1)
! histogram output.
Real (kd), dimension(2015:2050) :: mean, var, sd, skw, kurt, max_, min_,      &
tmean, sdsd, tskw, tkurt, tskwmom, tkurtmom
Real (kd), dimension(1) :: mean1, var1, sd1, skw1, kurt1, max_1, min_1,      &
tmean1, sdsd1, tskw1, tkurt1, tskwmom1,
tkurtmom1
! for the calculations of adjusted densities.
Real (kd) :: Lower, Upper, Rinc
Real (kd), dimension(:, :), allocatable :: Arg_str, PDFsm_str, CDFsm_str
Real (kd), dimension(:, :), allocatable :: Arg_str_v, PDFsm_str_v, CDFsm_str_v
!
Integer :: ind, inf,      & ! índices
rng, nsimw, intp, nintp, nint_, ind80(1), ind90(1), ind95(1)
Integer, parameter :: nres = 500      ! dim. Results hist.
!
Type (Hist_Res), Dimension(nres, 2015:2050) :: Results
Type (Hist_Res), Dimension(nres) :: Results1
!
Character (Len=10), Dimension(36) :: Color
!
! EXEC.
!
! read case select, and calculations to be performed.
Open (40, IOSTAT=Ios, action='read', status='old', file='C:\lf9571\bin\Aveiro_2019&
&\Data\calc_sel2.dat')

If ( ios .NE. 0 ) STOP '*** MAIN PROGRAM INPUT FILE INCORRECTLY OPENED; PROGRAM STOPS ***'
Read (40, '(2I10)') Isel_IEA_Irena, Isel_acc_dly
Read (40, '(5L10)') LgLCE_pdf, LgACV, LgCapExs, LgCapCst, LgClWbkd
! write case select and n° simulations
Open (30, IOSTAT=Ios, action='write', status='new',      &
file='C:\lf9571\bin\Aveiro_2019\Results\V2_Results_generic.dat')

If ( ios .NE. 0 ) STOP '*** OUTPUT FILE INCORRECTLY OPENED; PROGRAM STOPS ***'
Write(30, '(/A, I4)') '# *** CASO: 1, IEA ; 2, Irena : ', Isel_IEA_Irena
Write(30, '(/A, I4)') '# *** Roadmap (1,acelerado; 2,retrasado) : ', Isel_acc_dly
Write(30, '(/A, I8)') '#n° de simulaciones = ', nsim
!
!Call Random_seed ( put = (/ 27, 28, 52, 631 /) )
Call Random_seed ( put = (/ 2, 7/ ) )
!
! Wind
Write (30, '(/A)') '# ***** Parámetros Wind *****'
Call Infrastructure_Wind ( Isel_IEA_Irena, Isel_acc_dly )
Call Wind_Simul ( nsim,      &
PTurb, LCEas_W, LCEs_W, CapOpExs_W ) ! output
!
! PV
Write (30, '(/A)') '# ***** Parámetros PV *****'
Call Infrastructure_PV ( Isel_IEA_Irena, Isel_acc_dly )
Call PV_Simul ( nsim,      &
Pm, LCEas_PV, LCEs_PV, CapOpExs_PV ) ! output
!
CapOpExs = CapOpExs_PV + CapOpExs_W
StrEx = StrEx_PV + StrEx_W
!
! LCE
LCEs = LCEs_W * (DcET_W / (DcET_W + DcET_PV)) + LCEs_PV * (DcET_PV / (DcET_W + DcET_PV))

```



```

Do inf = 2015, 2050
    LCEas(:,inf) = LCEas_W(:,inf) * (EIba_W(inf) / (EIba_W(inf) + EIba_PV(inf))) &
                  + LCEas_PV(:,inf) * (EIba_PV(inf) / (EIba_W(inf) + EIba_PV(inf)))
End Do
!
! coste lW
w_Ib_W = Ib_W / (Ib_W + Ib_PV) ; w_Ib_PV = Ib_PV / (Ib_W + Ib_PV)
Do inf = 2015, 2050
    Pm_Turb_w(:,inf) = Pm(:,inf) * w_Ib_PV(inf) + PTurb(:,inf) * w_Ib_W(inf)
End Do
!
write(*,*) 'PASA 10 main program'
!
! ***** ANALIZE SIMULATIONS *****
!
! ++++++ Energy generated (total discounted; ¿units?) ++++++
Write (30, '(2/, 3(A/), /A, 3E14.5, /A, 2F14.6)') &
    '*****' , &
    '***** ENERGY GENERATED *****' , &
    '*****' , &
    &
    ' Total, PV, Wind: ', (DcET_W+DcET_PV), DcET_PV, DcET_W, &
    ' PV(%), Wind(%): ', (DcET_PV/(DcET_W+DcET_PV)), (DcET_W/ (DcET_W+DcET_PV))
!
! ++++++ LCEa (USD/MWh) ++++++
Write (30, '(2/, 3(A/), A, /)') &
    '#*****' , &
    '#***** ANNUAL LCE (means; US $ per Mwh.) *****' , &
    '#*****' , &
    '# Total PV Wind'
Do inf = 2015, 2050
    Write (30, '(I6, 3E14.5)') inf, Sum(LCEas(:,inf))/nsim,
    Sum(LCEas_PV(:,inf))/nsim, &
    Sum(LCEas_W(:,inf))/nsim
End Do
!
! ++++++ LCE (USD/MWh) ++++++
Write (30, '(2/, 2(A/), A)') &
    '#*****' , &
    '#***** TOTAL, PV & WIND, LCE (US$/MWh): SUMMARY STATISTICS *****' , &
    '#*****'
Call LCE_basic_stats ( LCEs_W, ' Wind' )
Call LCE_basic_stats ( LCEs_PV, ' PV ' )
Call LCE_basic_stats ( LCEs, 'Total' )
!
! ++++++ LCE (USD/MWh) all years ++++++
Write (30, '(2/, 2(A/), A)') &
    '#*****'
    '*****' , &
    '#***** TOTAL LCE (US$/MWh): SUMMARY STATISTICS; ALL YEARS: 2015 - 2050'
    '*****' , &
    '#*****'
    '*****'
Write (30, '(/A/)') '# MEAN, VARIANCE, SKW, KURT, INT: 2.5%, 50%, 80%, 90%, 95%, 97.5%'
Do inf = 1, nf
    Call LCE_Years_basic_stats ( LCEas(:,(2014+inf)), nsim, 1, inf )
End Do
!
! ++++++ stat. p.d.f. ++++++
! esta es previa para todas las densidades.
Call Set_Intervals
!
Write (30, '(2/, 3(A/), A)') &
    '#*****'
    , &
    '# p.d.f. & C.d.f LCE (US$/MWh): TOTAL, PV, WIND (Smoothed, NOT
    standardised)', &

```

```

!#*****
, &
'# Total PV Wind'
Write(30, '( /A/ )' ) '# ***** NOT standardised
***** !
If ( LgLCE_pdf ) Then
    Call PDF_CDF_pdf ( LCEs, Arg_str_v, PDFsm_str_v, CDFsm_str_v, nintp )
    Arg_str(:,1) = Arg_str_v; PDFsm_str(:,1) = PDFsm_str_v; CDFsm_str(:,1) =
    CDFsm_str_v
    Call PDF_CDF_pdf ( LCEs_PV, Arg_str_v, PDFsm_str_v, CDFsm_str_v, nintp )
    Arg_str(:,2) = Arg_str_v; PDFsm_str(:,2) = PDFsm_str_v; CDFsm_str(:,2) =
    CDFsm_str_v
    Call PDF_CDF_pdf ( LCEs_W, Arg_str_v, PDFsm_str_v, CDFsm_str_v, nintp )
    Arg_str(:,3) = Arg_str_v; PDFsm_str(:,3) = PDFsm_str_v; CDFsm_str(:,3) =
    CDFsm_str_v
End If
Do ind = -nintp, nintp, 5
    Write ( 30, '(1X, I4, 9E12.4)' ) ind, Arg_str(ind,1), PDFsm_str(ind,1),
    CDFsm_str(ind,1), &
    Arg_str(ind,2), PDFsm_str(ind,2),
    CDFsm_str(ind,2), &
    Arg_str(ind,3), PDFsm_str(ind,3),
    CDFsm_str(ind,3)
End do
!
Write(30, '( /A/ )' ) '# ***** Standardised
***** !
If ( LgLCE_pdf ) Then
    Call PDF_CDF_pdf ( Estandar(LCEs), Arg_str_v, PDFsm_str_v, CDFsm_str_v, nintp )
    Arg_str(:,1) = Arg_str_v; PDFsm_str(:,1) = PDFsm_str_v
    Call PDF_CDF_pdf ( Estandar(LCEs_PV), Arg_str_v, PDFsm_str_v, CDFsm_str_v, nintp )
    Arg_str(:,2) = Arg_str_v; PDFsm_str(:,2) = PDFsm_str_v
    Call PDF_CDF_pdf ( Estandar(LCEs_W), Arg_str_v, PDFsm_str_v, CDFsm_str_v, nintp )
    Arg_str(:,3) = Arg_str_v; PDFsm_str(:,3) = PDFsm_str_v
End If
Do ind = -nintp, nintp, 5
    Write ( 30, '(1X, I4, 6E14.4)' ) ind, Arg_str(ind,1), PDFsm_str(ind,1), &
    Arg_str(ind,2), PDFsm_str(ind,2), Arg_str(ind,3), PDFsm_str(ind,3)
End do
!
Write (30, '(2/, 3(A/), A)' ) &
!#*****
***** !, &
!#*** p.d.f. LCE (US$/MWh): TOTAL 2020, 2030, 2040 ,2050 (Smoothed, Not
standardised) *** !, &
!#*****
***** !, &
'# 2020 2030 2040
2050'
Write(30, '( /A/ )' ) '# ***** NOT standardised ***** !
! cálculos escribir resultados.
Do inf = 6, nf, 10
    LCEas_v = LCEas(:,2014+inf)
    Call PDF_CDF_pdf ( LCEas_v, Arg_str_v, PDFsm_str_v, CDFsm_str_v, nintp )
    Arg_str(:,inf) = Arg_str_v; PDFsm_str(:,inf) = PDFsm_str_v
End Do
! plot n 2D
Do ind = -nintp, nintp, 5
    Write ( 30, '(1X, I4, 8E12.4)' ) ind, Arg_str(ind,6),
    PDFsm_str(ind,6), &
    Arg_str(ind,16), PDFsm_str(ind,16), Arg_str(ind,26), PDFsm_str(ind,26), &
    Arg_str(ind,36), PDFsm_str(ind,36)
End do
! plot en 3D
Write(30, '( /A/ )' ) '# ***** adaptados para un plot en 3D con gnuplot
***** !
! first define colors ...
Color(6) = " 0xff0000" ; Color(16) = " 0xffff00" ; Color(26) = " 0x00ffff" ;

```

```

Color(36) = " 0x0000ff"
Do inf = 6, nf, 10
  Write(30, '()')
  Do ind = -nintp, nintp, 5
    Write ( 30, '(3E12.4, A)' ) Arg_str(ind,inf), PDFsm_str(ind,inf),
      (2014._kd+inf), Color(inf)
  End do
End do

!
Write(30, '(/A/)' ) '# ***** Standardised
***** !

! cálculos resultados.
Do inf = 6, nf, 10
  LCEas_v = LCEas(:,2014+inf)
  Call PDF_CDF_pdf ( Estandar(LCEas_v), Arg_str_v, PDFsm_str_v, CDFsm_str_v, nintp )
  Arg_str(:,inf) = Arg_str_v; PDFsm_str(:,inf) = PDFsm_str_v
End Do

! plot en 2D
Do ind = -nintp, nintp, 5
  Write ( 30, '(1X,I4,8E12.4)' ) ind, Arg_str(ind,6),
    PDFsm_str(ind,6), &
    Arg_str(ind,16), PDFsm_str(ind,16), Arg_str(ind,26), PDFsm_str(ind,26), &
    Arg_str(ind,36), PDFsm_str(ind,36)
End do

! plot en 3D
Write(30, '(/A/)' ) '# ***** adaptados para un plot en 3D con gnuplot
***** !

Do inf = 6, nf, 10
  Write(30, '()')
  Do ind = -nintp, nintp, 5
    Write ( 30, '(3E12.4)' ) Arg_str(ind,inf), PDFsm_str(ind,inf), (2014._kd+inf)
  End do
End do

!
! ++++++ Avoided Carbon Values (US. $ tr.) ++++++
! en el programa se calcula en $: dividiendo por 10**12, salen trs. US$
If ( LgACV
)
Write (30, '(2/, 3(A/), A,
/4E14.4)' )

! *****
**! , &
! ***** AVOIDED CARBON VALUES ($ tr.)
***** ! , &

! *****
**! , &
!# Valued at: 1 US$, Social Carbon Cost, Tax IEA, Tax
S.S.', &
(ACV_1USD_W+ACV_1USD_PV)/10.**12_kd,
(ACV_SCC_W+ACV_SCC_PV)/10.**12_kd, &
(ACV_TxCIEA_W+ACV_TxCIEA_PV)/10.**12_kd,
(ACV_TxCSS_W+ACV_TxCSS_PV)/10.**12_kd
!
! ++++++ CapOpEx (US. $ tr.) ++++++
If ( LgCapExs ) Then
  Write (30, '(2/, 3(A/))' ) &

  ! ***** !
  &
  ! ***** CAPOPEX (all; tr. US$): SUMMARY STATISTICS
  ***** ! , &

  ! ***** !

  Call CapOpEx_basic_stas
End If

!
If ( LgACV .AND. LgCapExs
)
&

```

```

Write (30, '(2/, 3(A/), /A,
/4E14.4)')

'#####'
&
'#####
CAPITAL COSTS - AVOIDED CARBON VALUES (US$ tr.)
*****'
&
'#####'
&
'# Valued at: 1 US$, Social Carbon Cost, Tax IEA, Tax
S.S.'
&
mean1-(ACV_1USD_W+ACV_1USD_PV)/10.**12_kd,
mean1-(ACV_SCC_W+ACV_SCC_PV)/10.**12_kd, &
mean1-(ACV_TxCIEA_W+ACV_TxCIEA_PV)/10.**12_kd,
mean1-(ACV_TxCSS_W+ACV_TxCSS_PV)/10.**12_kd
!
! ++++++ ev. precio del capital (1W) en el tiempo. ++++++
If ( LgCapCst ) Then
Write (30, '(2/,
3(A/))')
&
'#####
**'
&
'##### CAPITAL PRICES (USD/1W)
*****'
&
'#####
**'
Call Capital_cost
End if
!
! ++++++ breakdown del coste de 1w ++++++
If ( LgClWbkd ) Then
Write ( 30, '(2/, 3(A/), A,
/)' )
&
'#####
**'
&
'##### TOTAL COSTS USD/W: TIMELINE BREAKDOWN
*****'
&
'#####
**'
&
'# PV cap.(Mod+inv) PV BoS PV Str W cap. W BoS W Str'
Do inf = 2015, 2050
If ( (inf-2015) == ((inf-2015)/5)*5 ) Then
Write (30, '(I4,6F12.6,I6)') inf,
(Sum(Pm(:,inf))/nsim+Isrlw_PV(inf))*w_Ib_PV(inf), &
BoSlw_PV(inf)*w_Ib_PV(inf),
&
Strlw_PV(inf)*w_Ib_PV(inf),
&
0.7_kd*(Sum(PTurb(:,inf))/nsim)*w_Ib_W(inf),
&
0.3_kd*(Sum(PTurb(:,inf))/nsim)*w_Ib_W(inf),
&
Strlw_W(inf)*w_Ib_W(inf), inf
Else
Write (30, '(I4, 6F12.6)') inf,
(Sum(Pm(:,inf))/nsim+Isrlw_PV(inf))*w_Ib_PV(inf), &
BoSlw_PV(inf)*w_Ib_PV(inf),
&
Strlw_PV(inf)*w_Ib_PV(inf),
&

```

```

                                0.7_kd*(Sum(PTurb(:,inf))/nsim)*w_Ib_W(inf),
                                &
                                0.3_kd*(Sum(PTurb(:,inf))/nsim)*w_Ib_W(inf),
                                &
                                Strlw_W(inf)*w_Ib_W(inf)

        End if
    End Do
End if
!
CONTAINS
!
!
! intervalos;
Subroutine Set_Intervals
    rng = 3          ! de momento un intervalo grande; luego mas fino, 1
    intp = 600       ! intp/rng que sea exacto
    nintp = intp / rng
    nint_ = 2 * nintp + 1
    Allocate ( Arg_str(-nintp:nintp,nf), PDFsm_str(-nintp:nintp,nf),
    CDFsm_str(-nintp:nintp,nf), &
                Arg_str_v(-nintp:nintp), PDFsm_str_v(-nintp:nintp),
                CDFsm_str_v(-nintp:nintp) )
End Subroutine Set_Intervals
!
Subroutine PDF_CDF_pdf ( LCEs, Arg, PDFsm, CDFsm, nintp )
    Integer, intent(in) :: nintp
    Real (kd), intent(in) :: LCEs(:)
    Real (kd), intent(out) :: Arg(-nintp:nintp), PDFsm(-nintp:nintp), CDFsm(-nintp:nintp)
    Integer :: ind
    Real (kd) :: LCEs_str(Size(LCEs))
    ! smooth
    LCEs_str = LCEs
    Where ( Abs( Estandar ( LCEs_str ) ) > 9._kd ) LCEs_str = 0._kd
    Where ( Abs( Estandar ( LCEs_str ) ) > 9._kd ) LCEs_str = 0._kd
    ! arguments for the p.d.f.
    Rinc = (Maxval(LCEs_str) - Minval(LCEs_str)) / Real((2*nintp+1), kd)
    Arg = Acumul( (Minval(LCEs_str) + Rinc*.5_kd), Rinc, (2*nintp+1) )
    ! p.d.f. & C.d.f.
    PDFsm(-nintp) = Kernel_PDF_Hat ( Arg(-nintp), LCEs_str )
    CDFsm(-nintp) = PDFsm(-nintp) * Rinc
    Do ind = -nintp+1, nintp
        PDFsm(ind) = Kernel_PDF_Hat ( Arg(ind), LCEs_str )
        CDFsm(ind) = CDFsm(ind-1) + PDFsm(ind) * Rinc
    End Do
    CDFsm = CDFsm / CDFsm(nintp)
End Subroutine PDF_CDF_pdf
!
! Cap, Ope, y Str expenditures, basic stats
Subroutine CapOpEx_basic_stas
    ! +++ CapOpEx (US. $ tr.) +++
    ! basic stats., median, and tail probs.
    CapOpExs = (CapOpExs + StrEx) / 10.**6_kd
    CapOpExm(:,1) = CapOpExs
    Call Descrip_Stat ( CapOpExm, & ! IN
                        mean1, var1, sd1, skw1, kurt1, max_1, min_1, tmean1, sdsd1, tskw1,
                        tkurt1 )
    Write (30, '(//A)') 'MEAN, VARIANCE, SKW, KURT & MAX. VALUES 50%, 80%, 90%, 95%'
    Call Histogram ( CapOpExs, Results1 )
    ind80 = minloc(abs(Results1%CDF-.80_kd))
    ind90 = minloc(abs(Results1%CDF-.90_kd))
    ind95 = minloc(abs(Results1%CDF-.95_kd))
    Write (30, '(8E11.3)') mean1, var1, skw1, kurt1, &
                        Results1(minloc(abs(Results1%CDF-.50_kd)))%Arg, &
                        Results1(ind80)%Arg, Results1(ind90)%Arg, Results1(ind95)%Arg
    ! catastrophe analysis: CapOpStrEx > max. CapOpStrEx 80%, 90%, 95%.
    Write (30, '(//A)') ' CapOpStrEx EXPECT. / ( CapOpStrEx > CapOpStrEx max. al 80%,
    90%, 95%)'
    Write (30, '(//3E12.4)') &
        Sum( Results1(ind80(1)+1:nres)%HistPct * Results1(ind80(1)+1:nres)%Arg ) /

```

```

        .2_kd, &
        Sum( Results1(ind90(1)+1:nres)%HistPct * Results1(ind90(1)+1:nres)%Arg ) /
        .1_kd, &
        Sum( Results1(ind95(1)+1:nres)%HistPct * Results1(ind95(1)+1:nres)%Arg ) / .05_kd
End Subroutine CapOpEx_basic_stas
!
Subroutine Capital_cost
    Call Descrip_Stat ( Pm_Turb_w, & ! IN
                        mean, var, sd, skw, kurt, max_, min_, tmean, sdsd, tskw, tkurt, & !
                        OUT
                        tskwmom, tkurtmom ) ! OPTIONAL
!
Write (30, '(A/A/)' ) '# MEANS, VARIANCES & MAX. VALUES 50%, 80%, 90%, 95%', &
                        '# ALL YEARS: 2015-2050'
Do inf = 2015, 2050
    Pmv = Pm_Turb_w(:,inf)
    Call Histogram ( Pmv , Results(:,inf) )
    Write (30, '(I6, 6E11.3)' ) inf, mean(inf), var(inf), &
                                                Results(minloc(abs(Results(:,inf)%CDF-.50_kd)),inf)%Arg, &
                                                Results(minloc(abs(Results(:,inf)%CDF-.80_kd)),inf)%Arg, &
                                                Results(minloc(abs(Results(:,inf)%CDF-.90_kd)),inf)%Arg, &
                                                Results(minloc(abs(Results(:,inf)%CDF-.95_kd)),inf)%Arg
End Do
End Subroutine Capital_cost
!
Subroutine LCE_basic_stats ( LCEs, Energy )
    Real (kd), intent(in) :: LCEs(:)
    Character (len=5), intent(in) :: Energy
    ! +++ LCE (US. $ millions) +++
    ! basic stats., median, and tail probs.
Write (30, '(2/,3A)' ) ' ***** ', Energy, ' ***** '
LCEm(:,1) = LCEs
Call Descrip_Stat ( LCEm, & ! IN
                    mean1, var1, sd1, skw1, kurt1, max_1, min_1, tmean1, sdsd1, tskw1,
                    tkurt1 )
Write (30, '(/A/)' ) 'MEAN, VARIANCE, SKW, KURT & MAX. VALUES 50%, 80%, 90%, 95%'
Call Histogram ( LCEs, Results1 )
ind80 = minloc(abs(Results1%CDF-.80_kd))
ind90 = minloc(abs(Results1%CDF-.90_kd))
ind95 = minloc(abs(Results1%CDF-.95_kd))
Write (30, '(8E11.3)' ) mean1, var1, skw1, kurt1, &
                        Results1(minloc(abs(Results1%CDF-.50_kd)))%Arg, &
                        Results1(ind80)%Arg, Results1(ind90)%Arg, Results1(ind95)%Arg
! catastrophe analysis: LCE > max. LCE 80%, 90%, 95%.
Write (30, '(/A/A/)' ) ' LCE EXPECT. / ( LCE > LCE max. at 80%, 90%,
95%)'
Write (30, '(3E12.4)' ) &
Sum( Results1(ind80(1)+1:nres)%HistPct * Results1(ind80(1)+1:nres)%Arg ) / .2_kd, &
Sum( Results1(ind90(1)+1:nres)%HistPct * Results1(ind90(1)+1:nres)%Arg ) / .1_kd, &
Sum( Results1(ind95(1)+1:nres)%HistPct * Results1(ind95(1)+1:nres)%Arg ) / .05_kd
!write(*,*) 'PASA 20'
End Subroutine LCE_basic_stats
!
Subroutine LCE_Years_basic_stats ( LCEm, dm1, dm2, ind )
    Integer, intent(in) :: dm1, dm2, ind
    Real (kd), intent(in) :: LCEm(dm1,dm2)
    ! basic stats., median, and tail probs.
    Call Descrip_Stat ( LCEm, & ! IN
                        mean1, var1, sd1, skw1, kurt1, max_1, min_1, tmean1, sdsd1, tskw1,
                        tkurt1 )
    LCEas_v = LCEm(:,1)
    Call Histogram ( LCEas_v, Results1 )
    Write (30, '(I6, 10E11.3)' ) ind, mean1, var1, skw1, kurt1, &

```

```

Results1(minloc(abs(Results1%CDF-.025_kd)))%Arg, &
Results1(minloc(abs(Results1%CDF-.500_kd)))%Arg,
&
Results1(minloc(abs(Results1%CDF-.800_kd)))%Arg, &
Results1(minloc(abs(Results1%CDF-.900_kd)))%Arg, &
Results1(minloc(abs(Results1%CDF-.950_kd)))%Arg, &

Results1(minloc(abs(Results1%CDF-.975_kd)))%Arg

```

```

End Subroutine LCE_Years_basic_stats
!
End Program Simul_PV_Wind_Irena_Iea
!

```