```
'**************************************************************
'
' CLEANest method for period analysis (Foster G., AJ Vol 109, Nr 4, April 1995
'                                      "The cleanest Fourier spectrum")
'
' CLEANest is an effective method for removing false peaks from a power spectrum.
' It allows to describe and detect multiperiodic signals. It can be
' considered as an extension of the DCDFT method.
'
'
' Copyright 2005 The American Association of Variable Star Observers
'
' This code was partially funded by a Small Research Grant from the
'American Astronomical Society.
'
' The code is released for your use under the GPL 2.0 license. You can
'use it for noncommercial use with proper attribution and if you
'send a copy of your program to the AAVSO (aavso@aavso.org).
'
' Full license text is here: http://www.gnu.org/licenses/gpl.txt
'
'
'
'
'
'
'
'
'
'
'
'
'
'
'
'
'
'
'
'''''''''**************************************************************
Public NbrFrequencies As Integer
Dim PeakMatrix(NumberOfSignificantPeriods) As PeakMatElem  'Matrix with most significant peaks
Dim nDim As Integer, Poly As Integer
Const NMAXDIM = 50
Const NBEST = 20
Dim DVEC(NMAXDIM) As Double  ' covariant vector
Dim DCOEF(NMAXDIM) As Double 'contravariant vector
Dim DMAT(NMAXDIM, NMAXDIM) As Double
Dim FreqToTest(NBEST) As Double
Dim DtZero As Double, DtScale As Double, DtCorr As Double, variance_time As Double, DFourAmp2
As Double

Private Function GetTVec(jd As Double) As Double
   GetTVec = jd - Int(a(1).jd)
End Function

Private Function TVEC(jd As Double) As Double
   TVEC = GetTVec(jd) + DtCorr
End Function

Private Sub FunctionSpaceProjection(NbrFreq As Integer, power As Double, amp2 As Double)
'perform function space projection and return Power = Theta
   On Error GoTo ErrorHandler
   Dim PowOfTime() As Double
   Dim RadianFreq() As Double
   Dim CosMatrix() As Double
   Dim SinMatrix() As Double
   Dim n As Integer, n1 As Integer, n2 As Integer, nf As Integer, nf2 As Integer, nDim As
Integer
   Dim NP As Integer
   Dim dPower As Double
   Dim Weight As Double, Amp As Double
   Dim DT As Double, dx As Double, dphase As Double

   ReDim PowOfTime(Poly): ReDim RadianFreq(NbrFreq): ReDim CosMatrix(NbrFreq): ReDim
SinMatrix(NbrFreq)
   nDim = Poly + 2 * NbrFreq '+ NBIAS
   Weight = 0
   For n1 = 0 To nDim
        DVEC(n1) = 0
        For n2 = 0 To nDim
            DMAT(n1, n2) = 0
        Next n2
   Next n1
   For nf = 1 To NbrFreq
        RadianFreq(nf) = 2 * PI * FreqToTest(nf) * DtScale
        For nf2 = nf + 1 To NbrFreq
            If Abs(FreqToTest(nf) - FreqToTest(nf2)) < 0.00000001 Then
               dPower = 0: power = 0: amp2 = 0
               Exit Sub
            End If
        Next nf2
   Next nf
   PowOfTime(0) = 1
   For n = 1 To BigN
            Weight = Weight + 1
            DT = TVEC(a(n).jd)
            DT = (DT - DtZero) / DtScale
            dx = a(n).mag
            ' Compute powers of time
            For NP = 1 To Poly
                PowOfTime(NP) = PowOfTime(NP - 1) * DT
            Next NP
            ' Compute trig functions
            For nf = 1 To NbrFreq
                dphase = RadianFreq(nf) * DT: CosMatrix(nf) = Cos(dphase): SinMatrix(nf) =
Sin(dphase)
            Next nf
            ' Compute matrix coefficients for polynomials
            For NP = 0 To Poly
                DMAT(0, NP) = DMAT(0, NP) + PowOfTime(NP)
                If NP > 0 Then DMAT(NP, Poly) = DMAT(NP, Poly) + PowOfTime(NP) *
PowOfTime(Poly)
                DVEC(NP) = DVEC(NP) + dx * PowOfTime(NP)
                n2 = Poly
                ' Compute matrix coeff for products of polynomials with trig functions
                For nf = 1 To NbrFreq
                    n2 = n2 + 2
                    DMAT(NP, n2 - 1) = DMAT(NP, n2 - 1) + PowOfTime(NP) * CosMatrix(nf)
                    DMAT(NP, n2) = DMAT(NP, n2) + PowOfTime(NP) * SinMatrix(nf)
                Next nf
            Next NP
            ' Compute matrix values for products of trig functions
            n1 = Poly
            For nf = 1 To NbrFreq
                n2 = n1: n1 = n1 + 2
                DVEC(n1 - 1) = DVEC(n1 - 1) + dx * CosMatrix(nf)
                DVEC(n1) = DVEC(n1) + dx * SinMatrix(nf)
                For nf2 = nf To NbrFreq
                    n2 = n2 + 2
                    DMAT(n1 - 1, n2 - 1) = DMAT(n1 - 1, n2 - 1) + CosMatrix(nf) *
CosMatrix(nf2)
                    DMAT(n1 - 1, n2) = DMAT(n1 - 1, n2) + CosMatrix(nf) * SinMatrix(nf2)
                    DMAT(n1, n2 - 1) = DMAT(n1, n2 - 1) + SinMatrix(nf) * CosMatrix(nf2)
                    DMAT(n1, n2) = DMAT(n1, n2) + SinMatrix(nf) * SinMatrix(nf2)
                Next nf2
            Next nf
    Next n
    For n1 = 1 To Poly - 1
        For n2 = n1 To Poly - 1
            DMAT(n1, n2) = DMAT(n1 - 1, n2 + 1)
        Next n2
    Next n1
    For n1 = 0 To nDim
        DVEC(n1) = DVEC(n1) / Weight
        For n2 = n1 To nDim
            DMAT(n1, n2) = DMAT(n1, n2) / Weight
        Next n2
    Next n1
    DMAT(0, 0) = 1
    For n1 = 1 To nDim
        For n2 = 0 To n1 - 1
            DMAT(n1, n2) = DMAT(n2, n1)
        Next n2
    Next n1
    InvertMatrix DMAT, nDim
    amp2 = 0
    For n1 = 0 To nDim
        DCOEF(n1) = 0
        For n2 = 0 To nDim
            DCOEF(n1) = DCOEF(n1) + DMAT(n1, n2) * DVEC(n2)
        Next n2
        amp2 = amp2 + DCOEF(n1) * DVEC(n1)
    Next n1
    amp2 = amp2 - mean_mag ^ 2
    If amp2 < 0 Then amp2 = 0
    dPower = 0
    If nDim > 0 Then dPower = (BigN - 1) * amp2 / variance_mag / nDim
    ' Compute FOURIER power, amp^2
    power = (BigN - 1) * (amp2 - DFourAmp2)  'DFPOW
    power = power / (variance_mag - DFourAmp2) / 2
ErrorHandler:
End Sub

Private Sub CLEANest_Resolve(ddr As Double, ddp As Double)
   On Error GoTo ErrorHandler
   Dim Nexp As Integer
   If ddr = 0 Then Exit Sub
   Nexp = 0
   If ddr < 1 Then
      Do Until ddr > 1
          ddr = ddr * 10: Nexp = Nexp - 1
      Loop
   Else
      Do Until ddr < 10
          ddr = ddr / 10: Nexp = Nexp + 1
      Loop
   End If
   Select Case ddr
       Case 1 To 2
           ddr = 1
       Case 2 To 5
           ddr = 2
       Case Else
           ddr = 5
   End Select
   ddr = ddr * 10 ^ Nexp: ddp = ddp / ddr: ddp = ddr * Int(ddp + 0.5)
ErrorHandler:
End Sub

Public Sub SLICK_AskFreqRange()
   On Error GoTo ErrorHandler
   Dim Precision As Integer, temp As String, leftval As Double, rightval As Double
   CLEANestParamSetFormVisible = True
   CLEANestParamSetForm.SystemGeneratedClick = True
   CLEANestParamSetForm.ShowPeriodOptions = False
   CLEANestParamSetForm.CLEANest_AskFreqRangeMode = True
   CLEANestParamSetForm.PerWinSourceForAIs Me
   CLEANestParamSetForm.Show
   If TimeData Then
      CLEANestParamSetForm.Option2.value = True
      leftval = 1 / finalfreq: rightval = 1 / firstfreq
      If leftval > rightval Then Swap leftval, rightval
   Else
      leftval = firstfreq: rightval = finalfreq
      CLEANestParamSetForm.Option1.value = True
   End If
   Precision = FloatingPointResolution(rightval - leftval) + 1
   RoundValueToExactDecimals leftval, Precision, temp
   CLEANestParamSetForm.Text1.Text = Trim(temp)
   RoundValueToExactDecimals rightval, Precision, temp
   CLEANestParamSetForm.Text2.Text = Trim(temp)
   CLEANestParamSetForm.Text3.Text = resolution
   If Val(CLEANestParamSetForm.Text1.Text) = 0 Then CLEANestParamSetForm.Text1.Text = "0.01"
   If Val(CLEANestParamSetForm.Text2.Text) = 0 Then CLEANestParamSetForm.Text2.Text = "0.01"
   CLEANestParamSetForm.SystemGeneratedClick = False
ErrorHandler:
End Sub

Public Sub CloseCLEANestParamSetForm()
   On Error GoTo ErrorHandler
   CLEANestForm.HideFixedPeriodForm
   If CLEANestParamSetFormVisible Then
      CLEANestParamSetFormVisible = False
      CLEANestParamSetForm.Hide
   End If
ErrorHandler:
End Sub

Public Sub UpdateOverlays()
   If Me.ModelFunctionVisible Then ParentObsWin.UpdateOverlay ModelFunctionId,
CLEANestForm.ModelFunctionColor, CLEANestForm.ModelFunctionLineWidth
   If Me.ResidualsVisible Then ParentObsWin.UpdateOverlay ResidualsId,
CLEANestForm.ResidualsColor, CLEANestForm.ResidualsLineWidth
End Sub

Public Sub RefreshOverlayInfo()
   'will check if the Overlays (mentioned in CLEANest) are still alive. Could have been
deleted
   'through the parent ObsWin Overlays form
   If Not ParentObsWin.OverlayExists(Me.ResidualsId) Then
      Me.ResidualsVisible = False
   End If
   If Not ParentObsWin.OverlayExists(Me.ModelFunctionId) Then
      Me.ModelFunctionVisible = False
   End If
End Sub

Public Sub CLEANestAddFixedFrequency(FreqVal As Double)
   On Error GoTo ErrorHandler
   Dim i As Integer, foo As Double
   For i = NumberOfSignificantPeriods To 2 Step -1
       PeakMatrix(i).freq = PeakMatrix(i - 1).freq
       PeakMatrix(i).power = PeakMatrix(i - 1).power
       PeakMatrix(i).Visible = PeakMatrix(i - 1).Visible
       PeakMatrix(i).detailed_info = False
   Next i
   PeakMatrix(1).freq = FreqVal
   PeakMatrix(1).Visible = False
   'NbrFrequencies = 1
   FreqToTest(NbrFrequencies) = FreqVal
   'FunctionSpaceProjection nbrfrequencies, PeakMatrix(1).power, foo
   CLEANest_CalculateDetailedPeakInfo NbrFrequencies, PeakMatrix(1).power, foo
   PublishPeriods
ErrorHandler:
End Sub

Public Sub CLEANest_SLICKScan() 'perform SLICK using Foster CLEANest approach
   On Error GoTo ErrorHandler
   Dim power As Double, amp2 As Double
   CLEANestForm.SetCommandButtonsVisibility False
   CLEANest_CalculateDetailedPeakInfo NbrFrequencies, power, amp2
   DFourAmp2 = amp2
   NbrFrequencies = NbrFrequencies + 1
   CLEANest_InitProgressBar
   PerWinForm.P_XAxisMin = firstfreq
   PerWinForm.P_XAxisMax = finalfreq
   PerWinForm.UpdatePerWinForm
   Core_Period_Analysis 0, 0
   DFourAmp2 = 0
   If Not AnalysisCanceled Then
      Me.ShowCLEANestForm
      CLEANestForm.PublishedPeriods
   End If
ErrorHandler:
If CLEANestFormVisible Then CLEANestForm.SetCommandButtonsVisibility True
End Sub

Private Sub CLEANest_InitProgressBar()
   ProgressBarForm.SetParentPerWinForm Me, PerWinForm
   ProgressBarForm.ProgressBar1.Min = firstfreq
   ProgressBarForm.ProgressBar1.Max = finalfreq
   ProgressBarForm.PermLabel = ""
   ProgressBarForm.CancelAnalysisButton.Caption = "Cancel"
   ProgressBarForm.PermLabel = ""
   Me.AnalysisCanceled = False
   ProgressBarForm.Show
End Sub
```

```
Public Sub CLEANest_CLEANestScan() 'perform multi period scan using Foster CLEANest approach
   On Error GoTo ErrorHandler
   Dim n As Integer, power As Double, Dango As Double, dBPower As Double
   Dim dLPower As Double, dlFreq As Double, amp2 As Double
   Dim dTest() As Double, dres() As Double, Nvary As Integer, counter As Long
   Dim nSofar As Integer, nV As Integer, nVLast As Integer, nChange As Integer
   ReDim dTest(NbrFrequencies): ReDim dres(NbrFrequencies)
   counter = 0: CLEANestForm.Enabled = False
   MainForm.MousePointer = vbHourglass
   Dango = 1 / Sqr(12 * variance_time) / 4
   For n = 1 To NbrFrequencies
       dTest(n) = 1 / FreqToTest(n)
       dres(n) = (Dango * dTest(n) ^ 2) / 10
       CLEANest_Resolve dres(n), dTest(n)
   Next n
   Nvary = NbrFrequencies            ' number VARIABLE frequ/per
   dBPower = 0
   ' Perform multi-scan. STEP 1 : COMPUTE BASE LEVEL
   For n = 1 To NbrFrequencies
       FreqToTest(n) = 1 / dTest(n)
   Next n
   CLEANestForm.CLEANestProgressLabel.Caption = "Computing base level..."
   CLEANestForm.CLEANestProgressLabel.Visible = True
   CLEANestForm.Refresh
   FunctionSpaceProjection NbrFrequencies, power, amp2
   dBPower = power     ' set base level for power
   If dBPower = 0 Then dBPower = 1
   nSofar = 0: nChange = 0: nV = 0: nVLast = 0         ' last changed frequ
   ' STEP 2. REFINE THE PERIODS
   Do
      If nChange < 0 And nVLast > 0 Then
         Swap nV, nVLast
      Else
         If nChange < 0 Then nVLast = nV
         nV = nV + 1
         If nV > Nvary Then nV = 1
      End If
      nChange = 0     ' init to NO CHANGE
      ' STEP 3. TEST HIGHER PERIODS
      Do
         counter = counter + 1
         dTest(0) = dTest(nV) + dres(nV)
         FreqToTest(nV) = 1 / dTest(0)
         If (counter Mod 10) = 0 Then
            CLEANestForm.CLEANestProgressLabel.Caption = "Testing periods [" +
Trim(Str(counter)) + "]"
            CLEANestForm.Refresh
         End If
         FunctionSpaceProjection NbrFrequencies, power, amp2
         If power > dBPower Then ' if better then
            dBPower = power      ' save new ampl.
            dTest(nV) = dTest(0) ' save new per.
            nChange = -1         ' mark CHANGED
            nSofar = -1          ' mark CHANGED
         Else
            power = 0
         End If
      Loop Until power < dBPower
      If nChange = 0 Then
         ' STEP 4. TEST LOWER PERIODS
         Do
            counter = counter + 1
            dTest(0) = dTest(nV) - dres(nV)
            FreqToTest(nV) = 1 / dTest(0)
            If (counter Mod 10) = 0 Then
               CLEANestForm.CLEANestProgressLabel.Caption = "Testing periods [" +
Trim(Str(counter)) + "]"
               CLEANestForm.Refresh
            End If
            FunctionSpaceProjection NbrFrequencies, power, amp2
            If power > dBPower Then ' if better
               dBPower = power      ' save ampl.
               dTest(nV) = dTest(0) ' save per.
               nSofar = -1          ' mark CHANGED
               nChange = -1         ' mark CHANGED
            Else
```

```
               power = 0
            End If
         Loop Until power < dBPower
      End If
      FreqToTest(nV) = 1 / dTest(nV)
      nSofar = nSofar + 1
   Loop Until nSofar >= Nvary
   ' Save best set to table
   dLPower = dBPower
   For n = 1 To NbrFrequencies
       dlFreq = 1 / dTest(n)
       AddToPeakTable dlFreq, dLPower 'this is a significant peak, so at it to the table
   Next n
   CLEANest_CalculateDetailedPeakInfo NbrFrequencies, dBPower, amp2
   CLEANestForm.CLEANestProgressLabel.Visible = False
   Me.ShowCLEANestForm
ErrorHandler:
   CLEANestForm.Enabled = True
   MainForm.MousePointer = vbNormal
End Sub

Private Sub CLEANestSmooth(Dtime As Double, Dmag As Double) 'Compute value of Model function
   On Error GoTo ErrorHandler
   Dim DT As Double, NP As Integer, n2 As Integer, dphase As Double, nf As Integer
   DT = (Dtime - DtZero) / DtScale
   Dmag = DCOEF(0)
   For NP = 1 To Poly
       Dmag = Dmag + DCOEF(NP) * DT ^ NP
   Next NP
   n2 = Poly
   For nf = 1 To NbrFrequencies
       n2 = n2 + 2
       dphase = 2 * PI * FreqToTest(nf) * DtScale * DT
       Dmag = Dmag + DCOEF(n2 - 1) * Cos(dphase)
       Dmag = Dmag + DCOEF(n2) * Sin(dphase)
   Next nf
ErrorHandler:
End Sub

Public Sub CLEANest_WriteResidualsToFile(FileName As String)
   On Error GoTo ErrorHandler
   Dim DT As Double, dx As Double, power As Double, amp2 As Double, n As Long, temp As String
   Open FileName For Output As #1
   ' Compute coefficients
   FunctionSpaceProjection NbrFrequencies, power, amp2
   ' Compute residuals
   For n = 1 To BigN
          DT = TVEC(a(n).jd)
          CLEANestSmooth DT, dx
          RoundValueToExactDecimals a(n).jd, 4, temp
          Print #1, Trim(temp) + vbTab;
          RoundValueToExactDecimals a(n).mag - dx, 4, temp
          Print #1, Trim(temp) + vbTab;
          RoundValueToExactDecimals a(n).mag, 4, temp
          Print #1, Trim(temp) + vbTab;
          RoundValueToExactDecimals dx, 4, temp
          Print #1, Trim(temp)
   Next n
ErrorHandler:
   Close #1
End Sub

Public Sub ShowModelFunction(vColor As Long, vLineWidth As Integer, vPerWinID As String)
   On Error GoTo ErrorHandler
   Dim DT As Double, dx As Double, power As Double, amp2 As Double, n As Long
   Dim x() As Double, y() As Double
   ' Calculate model function that fits the selected periods
   FunctionSpaceProjection NbrFrequencies, power, amp2
   ReDim x(BigN + 1): ReDim y(BigN + 1)
   For n = 1 To BigN
       DT = TVEC(a(n).jd)
       CLEANestSmooth DT, dx
       x(n) = a(n).jd: y(n) = dx
   Next n
   'Then store the model function as an Overlay for ObsWin
   ParentObsWin.ShowModelFunction x, y, BigN + 1, vColor, vLineWidth, vPerWinID,
ModelFunctionId 'returns unique id of model function
```

```
   ModelFunctionVisible = True
ErrorHandler:
End Sub

Public Sub HideModelFunction()
   On Error GoTo ErrorHandler
   If ModelFunctionVisible Then
      ParentObsWin.HideModelFunction ModelFunctionId
      ModelFunctionVisible = False
   End If
ErrorHandler:
End Sub

Public Sub ShowResiduals(vColor As Long, vLineWidth As Integer, vPerWinID As String)
   On Error GoTo ErrorHandler
   Dim DT As Double, dx As Double, power As Double, amp2 As Double, n As Long
   Dim x() As Double, y() As Double
   ' Calculate residuals using the selected periods
   FunctionSpaceProjection NbrFrequencies, power, amp2
   ReDim x(BigN + 1): ReDim y(BigN + 1)
   For n = 1 To BigN
       DT = TVEC(a(n).jd)
       CLEANestSmooth DT, dx
       x(n) = a(n).jd: y(n) = a(n).mag - dx + mean_mag
   Next n
   'Then store the model function as an Overlay for ObsWin
   ParentObsWin.ShowResiduals x, y, BigN + 1, vColor, vLineWidth, vPerWinID, ResidualsId
'returns unique id of residuals
   ResidualsVisible = True
ErrorHandler:
End Sub

Public Sub HideResiduals()
   On Error GoTo ErrorHandler
   If ResidualsVisible Then
      ParentObsWin.HideResiduals ResidualsId
      ResidualsVisible = False
   End If
ErrorHandler:
End Sub

Public Sub CLEANest_CalculateDetailedPeakInfo(NbrFrequencies As Integer, power As Double, amp2
As Double)
   On Error GoTo ErrorHandler
   Dim n As Long, temp As String, i As Integer, dd As Double
   Dim nb As Integer, na As Integer, DT As Double, dres As Double, dz As Double, dalpha As
Double, dbeta As Double
   Dim dper As Double, dsigfre As Double, dsigper As Double, dphase As Double, sdv As Double
   ' Compute coefficients
   FunctionSpaceProjection NbrFrequencies, power, amp2
   nb = Poly
   For i = 1 To NbrFrequencies
      nb = nb + 2: na = nb - 1
      dd = DCOEF(na) ^ 2 + DCOEF(nb) ^ 2
      PeakMatrix(i).ampl = Sqr(dd)
      PeakMatrix(i).sIni = DCOEF(nb)
      PeakMatrix(i).cosi = DCOEF(na)
   Next i
   DT = a(BigN).jd - a(1).jd
   dres = stdev_mag ^ 2
   dres = dres * ((BigN - 1) - (2 * power))
   dres = dres / (BigN - 1 - (3 * NbrFrequencies))
   If dres < 0 Then dres = 0
   dres = SquareRoot(dres): dz = 2 / BigN: dalpha = dres * SquareRoot(dz)
   dz = 6 / BigN: dbeta = dres * SquareRoot(dz) / DT / PI
   dalpha = 2 * dalpha: dbeta = 2 * dbeta
   For i = 1 To NbrFrequencies
       dper = 1 / PeakMatrix(i).freq
       dsigfre = dbeta / PeakMatrix(i).ampl
       dsigper = dsigfre * dper * dper
       If PeakMatrix(i).cosi <> 0 Then
          dz = -PeakMatrix(i).sIni / PeakMatrix(i).cosi
          dphase = Atn(dz)
          dphase = arctan(-PeakMatrix(i).sIni, PeakMatrix(i).cosi)
          dphase = dphase / 2 / PI
       ElseIf PeakMatrix(i).sIni > 0 Then
          dphase = -0.25
       Else
          dphase = 0.25
       End If
       If PeakMatrix(i).cosi < 0 Then dphase = dphase + 0.5
       If dphase < 0 Then dphase = dphase + 1
       PeakMatrix(i).freqerr = Abs(dsigfre)
       PeakMatrix(i).pererr = Abs(dsigper)
       PeakMatrix(i).amplerr = Abs(dalpha)
       PeakMatrix(i).Phase = dphase
       PeakMatrix(i).detailed_info = True
   Next i
ErrorHandler:
End Sub

Private Sub CLEANest_calculate_theta(f As Double, theta As Double)
   On Error GoTo ErrorHandler
   'For a given frequency f, calculate the Theta value using the DCDFT method of Foster
   Dim na As Integer, nb As Integer, dd As Double, amp2 As Double, Damp As Double
   Static Dlamp As Double, Dllamp As Double, dlFreq As Double, dLPower As Double 'these static
vars are used to keep previous peak values. When looking for peaks, you only know a value was
a peak, when you just passed it
   FreqToTest(NbrFrequencies) = f
   FunctionSpaceProjection NbrFrequencies, theta, amp2  'Fourier transform. Theta contains
Power value
   Damp = Sqr(2 * (amp2 - DFourAmp2))
   na = Poly + 1: nb = na + 1
   dd = Sqr(DCOEF(na) ^ 2 + DCOEF(nb) ^ 2)  'Ampl
   If Damp < Dlamp And Dlamp >= Dllamp Then AddToPeakTable dlFreq, dLPower    'this is a
significant peak, so at it to the table
   Dllamp = Dlamp: Dlamp = Damp: dlFreq = f: dLPower = theta
   'Theta = dd  'plot amplitude
ErrorHandler:
End Sub

Public Sub InitCommonCLEANestVariables()
   'initialise common variables
   Dim dtspan As Double, x As Double, tresolv As Double, dd As Long
   Poly = 0: NbrFrequencies = 1: DFourAmp2 = 0
End Sub

Public Sub Cleanest_Analysis(f1 As Double, f0 As Double, df As Double)
   On Error GoTo ErrorHandler
   AnalysisMethod = cCleanestAnalysis
   CommonPeriodAnalysisInitialiser f1, f0, df, True
   InitCommonCLEANestVariables
   Core_Period_Analysis 0, 0
ErrorHandler:
End Sub
```