ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

ANEXOS

SOFTWARE DE CÁLCULO Y OPTIMIZACIÓN DE CIMENTACIONES POR PILOTES

ANEXO I

Nº TFG: 423.15.5

Autor: David Ostáriz Falo

Director: José Ángel Pérez Benedicto

Fecha: 28-6-2016

INDICE DE CONTENIDO

1. INTRODUCCIÓN 3

2. VARIABLES RECURRENTES Y FUNCIONES BÁSICAS 4

3. FUNCIONES BÁSICAS Y ESTRUCTURALES 11

3.1. TENSIÓN EFECTIVA 11

3.2. RESISTENCIA UNITARIA POR PUNTA 12

3.2.1. Resistencia unitaria por punta a corto plazo 12

3.2.2. Resistencia unitaria por punta a largo plazo 18

3.3. RESISTENCIA UNITARIA POR FUSTE A CORTO PLAZO 23

3.3.1. Resistencia unitaria por fuste a corto plazo-CTE 25

3.3.2. Resistencia unitaria por fuste a corto plazo-GCOC 26

3.3.3. Resistencia unitaria por fuste a corto plazo-ROM 28

3.3.4. Resistencia unitaria por fuste a corto plazo-GMOC 29

3.4. RESISTENCIA UNITARIA POR FUSTE A LARGO PLAZO 30

3.4.1. Resistencia unitaria por fuste a largo plazo-CTE 32

3.4.2. Resistencia unitaria por fuste a largo plazo-GCOC 33

3.4.3. Resistencia unitaria por fuste a largo plazo-ROM 35

3.4.4. Resistencia unitaria por fuste a largo plazo-GMOC 36

3.5. CARGA MÁXIMA SOBRE PILOTE 37

3.6. CARGA MÍNIMA SOBRE PILOTE 38

3.7. CORTANTE MÁXIMO SOBRE PILOTE 38

3.8. PESO DEL ENCEPADO 38

3.9. PESO PROPIO DEL PILOTE 39

3.10. FRICCIÓN NEGATIVA 40

3.11. TOPE ESTRUCTURAL 40

3.11.1. Tope estructural-micropilotes 40

3.12. DIMENSIONAMIENTO DE LONGITUD 41

3.13. COMPROBACIÓN DEL EFECTO GRUPO 42

3.14. COMPROBACIÓN ARRANQUE 44

3.15. ASIENTO DE PILOTES-ASIENTO INDIVIDUAL 46

3.16. ASIENTO-DE PILOTES-EFECTO GRUPO 47

3.17. INTERPOLACIONES DE SONDEOS 48
3.17.1. Dos sondeos 50
3.17.2. Tres sondeos 51
4. FUNCIÓN DE COSTE 53
5. FUNCIONES DE OPTIMIZACIÓN 56
5.1. ARRANQUE AISLADO 56
5.2. PLANTA-UN PILOTE 56
5.3. PLANTA- DOS PILOTES 60
5.4. CORRECCION DISTORSIONES 64
5.4.1. Corrección- un pilote 64
5.4.2. Corrección- dos pilotes 72

1. INTRODUCCIÓN

En este anexo se ha introducido el código relativo a cálculos estructurales, de coste y optimización que conforman la infraestructura del del programa. Se han omitido todos aquellos relativos a gestión de datos, representaciones gráficas y control de formularios ya que, a pesar de suponer la mayoría del volumen del programa, no se consideran relevantes en cuanto al aspecto técnico y, por tanto, no han sido referenciados en la memoria.

Comienza con la presentación de las variables representativas, dado que al ser llamadas en múltiples ocasiones en el código es conveniente conocer su significado.

Posteriormente se presentan las funciones y las macros que realizan los procedimientos que se han expuesto en la memoria.

2. VARIABLES RECURRENTES Y FUNCIONES DE LLAMADA

```
Public capacidad_portante As Single
'Valor de la capacidad portante en MPa. Se extrae del formulario de Datos Generales
Public normativa As String
'Nombre de la normativa que regirá los cálculos. Se extrae del formulario de Datos Generales
Public n_estratos As Integer
'Número de estratos de terreno. Se extrae del formulario de datos del terreno
Public listaestratos() As String
'Matriz de Datos del Terreno
'0      1       2       3               4                   5
'Nombre Tipo    Espesor Densidad seca   Densidad saturada   Ángulo de rozamiento
'6          7                               8                           9
'Cohesión   Resistencia a corte sin drenaje Coef. de empuje al reposo   Áng roz pilote-terreno
'10                                 11              12  13  14  15  16
'Resistencia a compresión simple    fd/a1·a2·a3     s   a   RQD E   v(poisson)
Public listaestratos2() As Single
'Segunda matriz de datos del terreno
'0              1                           2
'Prof.acumulada Tension efectiva inferior   Tensión efectiva media

Public nselect As Integer
Public n_freatico As Single
'Profundidad a la que se encuentra el nivel freático. Se extrae del formulario de datos del terreno
Public ejecucion As String
'Procedimiento de ejecución. Se extrae del formulario de datos generales
Public disposiciones() As String
'Matriz de disposiciones: representa el número de pilotes y las coordenadas locales de cada pilote
'Se genera y edita en el formulario de encepados
Public n_encepados As Integer
Public disposiciones2() As Single
'Contiene datos característicos de las disposiciones que se emplean en otros cálculos
Public encepado() As String
'Matriz de un encepado. Se genera en el formulario de nuevo encepado
Public arranques() As String
'Matriz de arranques. Se obtiene en el formulario de arranques y cargas
Public material As String
```

```
'Material de los pilotes. Se obtiene del formulario de datos generales
Public listapilotes() As String
'Matriz de pilotes, se obtiene en el formulario de tablas de pilotes
Public tipoencepados_carga As Integer
'Tipoencepados: 1-encepados rígidos  2-encepados de canto mínimo 3-no considerar
Public npilotes As Integer
Public n_arranques As Integer
Public nsondeos As Integer
Public Situacion_proyecto As String
'Situación de proyecto
Public coef_seguridad As Single
'Coeficiente de seguridad según la situación y la normativa
'Se obtiene del formulario de datos generales
Public arranques2() As String
'Matriz secundaria de arranques. En ella se guardan los datos de cálculo
'0       1           2   3       4       5          6       7       8
'Pilote  Encepado    K   Nmax    Nencep  Nroz_neg   Npil    Ntot    L
'9      10      11      12
'Qp,cp  Qf,cp   Qp,lp   Qf,lp
Public punzonamiento_arcillas As Boolean
'Consideración de punzonamientos de arcillas
Public Fuste_suelos As Boolean
'Consideración de resistencia por fuste de suelos
Public Punta_suelos As Boolean
'Consideración de resistencia por punta en suelos
Public Fuste_roca As Boolean
'Consideración de resistencia por fuste de roca
Public rozamiento_negativo As Boolean
'Consideración de rozamiento negativo
'Parámetros de cálculo de micropilotes
Public Re As Single
Public fe As Single
Public R As Single
Public fu As Single


'Precios
Public precio_hormigon As Single
Public precio_hormigon_encepado As Single
Public precio_acero As Single
Public precio_madera As Single
Public precio_armaduras As Single
Public precio_S235 As Single
Public precio_S275 As Single
Public precio_S355 As Single
Public precio_S420 As Single
```

```
Public precio_S460 As Single

Public precio_H25 As Single
Public precio_H30 As Single
Public precio_H35 As Single
Public precio_H40 As Single
Public precio_H45 As Single
Public precio_H50 As Single

Public precio_B400 As Single
Public precio_B500 As Single


'Características de materiales
Public modulo_elastico_material As Single
Public densidad_hormigon As Single
Public densidad_acero As Single
Public densidad_madera As Single
Public distorsion_angular As Single
Public modulo_hormigon As Single
Public modulo_acero As Single
Public modulo_madera As Single


'Separaciones de ejes para iteraciones
Public separacion_ejes_maxima As Integer
Public separacion_ejes_minima As Integer
Public separacion_ejes_incremento As Single
Public separacion_ejes_fija As Boolean


Public diametro_efecto_grupo_individual As Boolean
'Consideración del diámetro del pilote individual para la zona activa del grupo
Public fila_informe As Integer
Public efecto_grupo_coeficiente_eficiencia As Boolean
'Cálculo del efecto grupo mediante el coeficiente de eficiencia
Public tipo_perforacion As String
'Tipo de perforación para función de coste

Public armado_continuo As Boolean
'Consideración de armado contínuo a lo largo de todo el pilote
Public longitud_armado_d As Integer
'Longitud armada
Public longitud_armado_l As Single
'longitud de armado en metros
Public longitud_armado_en_diametros As Boolean
```

```
'Consideración de diámetros como medida de longitud


Public cuantia_encepado As Single
'Cuantía de armaduras en encepado
Public cuantia_pilotes As Single
'Cuantía de armadura en pilotes

Public arranques_optimizacion() As Single
'Tabla auxiliar para datos de optimización

Public distorsiones_angulares() As String
'Tabla de distorsiones angulares

Public espesor_estratos_sondeo() As Single
'Tabla de espesores de estratos según sondeos

Public profundidades_estratos_sondeos() As Single
'Tabla de profundidades de estratos en los sondeos
Public profundidades_estratos_arranques() As Single
'Tabla de profundidades de estratos en los arranques

Public lista_sondeos() As String
'Identificador      x    y
Public n_sondeos As Integer


Public tipo_interpolacion As Integer
'0 si es con el mismo sondeo
'1 si es con el sondeo más cercano
'2 si es con los dos más cercanos
'3 si es con los tres más cercanos

Public armaduras_tubulares() As String
'Tabla de armaduras tubulares




'Se definen una serie de funciones destinadas a extraer los valores de las tablas
'y así facilitar la labor de codificar


Public Function tipoestrato(ByVal estrato As Integer) As String
```

```
    tipoestrato = Módulo1.listaestratos(estrato, 1)
End Function

Public Function Esp(ByVal estrato As Integer) As Single
    Esp = CSng(Módulo1.listaestratos(estrato, 2))
End Function

Public Function Dseca(ByVal estrato As Integer) As Single
    Dseca = CSng(Módulo1.listaestratos(estrato, 3))
End Function

Public Function Dsat(ByVal estrato As Integer) As Single
    Dsat = CSng(Módulo1.listaestratos(estrato, 4))
End Function

Public Function angroz(ByVal estrato As Integer) As Single
    angroz = CSng(Módulo1.listaestratos(estrato, 5)) * 3.1416 / 180
End Function

Public Function c(ByVal estrato As Integer) As Single
    c = CSng(Módulo1.listaestratos(estrato, 6))
End Function

Public Function cu(ByVal estrato As Integer) As Single
    cu = CSng(Módulo1.listaestratos(estrato, 7))
End Function

Public Function ko(ByVal estrato As Integer) As Single
    ko = CSng(Módulo1.listaestratos(estrato, 8))
End Function

Public Function angroz2(ByVal estrato As Integer) As Single
    angroz2 = CSng(Módulo1.listaestratos(estrato, 9)) * 3.1416 / 180
End Function

Public Function qu(ByVal estrato As Integer) As Single
    qu = CSng(Módulo1.listaestratos(estrato, 10))
End Function

Public Function factordiaclas(ByVal estrato As Integer) As Single
    factordiaclas = CSng(Módulo1.listaestratos(estrato, 11))
End Function

Public Function s(ByVal estrato As Integer) As Single
    s = CSng(Módulo1.listaestratos(estrato, 12))
End Function
```

```
Public Function a(ByVal estrato As Integer) As Single
    a = CSng(Módulo1.listaestratos(estrato, 13))
End Function

Public Function RQD(ByVal estrato As Integer) As Single
    RQD = CSng(Módulo1.listaestratos(estrato, 14))
End Function
Public Function E(estrato As Integer) As Single
    E = CSng(Módulo1.listaestratos(estrato, 15))
End Function

Public Function v(estrato As Integer) As Single
    v = CSng(Módulo1.listaestratos(estrato, 16))
End Function

Public Function hini(ByVal estrato As Integer) As Single
    If estrato = 1 Then
        hini = 0
    Else
        hini = Módulo1.listaestratos2(estrato - 1, 0)
    End If
End Function

Public Function hfinal(ByVal estrato As Integer) As Single
    hfinal = Módulo1.listaestratos2(estrato, 0)
End Function

Public Function tenini(ByVal estrato As Integer) As Single
    If estrato = 1 Then
        tenini = 0
    Else
        tenini = Módulo1.listaestratos2(estrato - 1, 1)
    End If
End Function

Public Function tenfin(ByVal estrato As Integer) As Single
    tenfin = Módulo1.listaestratos2(estrato, 1)
End Function

Public Function tenmed(ByVal estrato As Integer) As Single
    tenmed = Módulo1.listaestratos2(estrato, 2)
End Function

Public Function areal(ByVal pilote As Integer) As Single
    areal = CSng(Módulo1.listapilotes(pilote, 1)) / 10000
```

```
End Function

Public Function aencerr(ByVal pilote As Integer) As Single
    aencerr = CSng(Módulo1.listapilotes(pilote, 2)) / 10000
End Function

Public Function perimetro(ByVal pilote As Integer) As Single
    perimetro = CSng(Módulo1.listapilotes(pilote, 3)) / 100
End Function

Public Function diam(ByVal pilote As Integer) As Single
    diam = CSng(Módulo1.listapilotes(pilote, 4)) / 100
End Function
Public Function xarranque(arranque As Integer) As Single
    xarranque = Módulo1.arranques(arranque, 1)
End Function
Public Function yarranque(arranque As Integer) As Single
    yarranque = Módulo1.arranques(arranque, 2)
End Function
Public Function axil(arranque As Integer) As Single
    axil = Módulo1.arranques(arranque, 3)
End Function
Public Function momentox(arranque As Integer) As Single
    momentox = Módulo1.arranques(arranque, 4)
End Function
Public Function momentoy(arranque As Integer) As Single
    momentoy = Módulo1.arranques(arranque, 5)
End Function
Public Function cortantex(arranque As Integer) As Single
    cortantex = Módulo1.arranques(arranque, 6)
End Function
Public Function cortantey(arranque As Integer) As Single
    cortantey = Módulo1.arranques(arranque, 3)
End Function
Public Function torsor(arranque As Integer) As Single
    torsor = Módulo1.arranques(arranque, 3)
End Function
```

3. FUNCIONES BÁSICAS Y ESTRUCTURALES

3.1. TENSIÓN EFECTIVA

```
      Public Sub tension_efectiva_tabla()

          'Calcula la profundidad acumulada
          For i = 1 To 20
              If Módulo1.listaestratos(i, 0) = "" Then Exit For
              Módulo1.listaestratos2(i,  0)  =  CSng(Módulo1.listaestratos(i,  2))  +  Módu-
lo1.listaestratos2(i - 1, 0)
          Next

          'Calcula las tensiones efectivas del terreno en el límite inferior de cada estra-
tos
          For i = 1 To n_estratos
              prof = listaestratos2(i, 0)

              If prof <= n_freatico Then
                  If i = 1 Then
                      listaestratos2(i, 1) = Esp(i) * Dseca(i)
                  Else
                      listaestratos2(i, 1) = listaestratos2(i - 1, 1) + Esp(i) * Dseca(i)
                  End If
              Else
                  If listaestratos2(i - 1, 0) < n_freatico Then
                      listaestratos2(i, 1) = listaestratos2(i - 1, 1) + (n_freatico - lis-
taestratos2(i - 1, 0)) * Dseca(i) + (prof - n_freatico) * (Dsat(i) - 1)
                  Else
                      listaestratos2(i, 1) = listaestratos(i - 1, 1) + Esp(i) * (Dsat(i) -
1)
                  End If
              End If
          Next
          'Calcula las tensiones medias de cada estrato, incluso en aquellos en los que se
encuentra el nivel freático
          For i = 1 To n_estratos
              prof2 = listaestratos2(i, 0)
              prof1 = listaestratos2(i - 1, 0)
              ten1 = listaestratos2(i - 1, 1)
              ten2 = listaestratos2(i, 1)
              If prof2 <= n_freatico Then
                  If i = 1 Then
                      listaestratos2(i, 2) = ten2 / 2
                  Else
                      listaestratos2(i, 2) = (ten1 + ten2) / 2
                  End If
              Else
                  If prof1 < n_freatico Then
                      esp1 = n_freatico - prof1
                      esp2 = prof2 - n_freatico
                      listaestratos2(i, 2) = (esp1 * (ten1 + 0.5 * esp1 * Dseca(i)) + esp2
* (ten2 - 0.5 * esp2 * Dsat(i))) / Esp(i)
                  Else
                      listaestratos(i, 2) = (ten1 + ten2) / 2
                  End If
              End If
          Next
      End Sub
```

```
Public Function ten_efec(ByVal h As Single) As Single
'Calcula la tensión efectiva a una profundidad determinada
For i = 1 To n_estratos
    If h > hini(i) And h < hfinal(i) Then
        If Módulo1.n_freatico > hini(i) And Módulo1.n_freatico < hfinal(i) Then
            If h > Módulo1.n_freatico Then
                ten_efec = tenini(i) + (Módulo1.n_freatico - hini(i)) * Dseca(i) + (h
- Módulo1.n_freatico) * (Dsat(i) - 1)
            Else
                ten_efec = tenini(i) + (h - hini(i)) * Dseca(i)
            End If
        Else
        abc1 = tenini(i)
        abc2 = tenfin(i)
        abc3 = Esp(i)
            ten_efec = tenini(i) + (tenfin(i) - tenini(i)) * (h - hini(i)) / Esp(i)
        End If
    End If
Next
End Function
```

3.2. RESISTENCIA UNITARIA POR PUNTA

3.2.1. Resistencia unitaria por punta a corto plazo

```
Public Function Qp_cp(longitud, pil) As Single
'Determina la resistencia por punta a corto plazo
For i = 1 To n_estratos
    altura = hini(i)
    altura2 = hfinal(i)
    If longitud > hini(i) And longitud < hfinal(i) Then

            Select Case Módulo1.normativa

            Case Is = "GCOC"

                Select Case tipoestrato(i)

                Case Is = "Rellenos"
                    Qp_cp = 0
                Case Is = "Granular"

                    'Factor de diámetro
                    If diam(pil) > 1 Then
                        fd = 2 / 3
                    Else
                        fd = 1 - diam(pil) / 3
```

```
                        End If

                        'Coeficiente de corrección
                        If Módulo1.ejecucion = "Pilotes perforados" Then
                            Ncorrec = 1
                        ElseIf Módulo1.ejecucion = "Pilotes hincados" Then
                            Ncorrec = 2
                        End If

                        'Resistencia por punta a corto plazo en T/m2
                        Qp = Ncorrec * (1.5 * fd * ten_efec(longitud) + 9 * fd * c(i) *
0.1)
                        Qp_cp = Qp * aencerr(pil)

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_cp = 0

                        If Módulo1.Punta_suelos = False Then Qp_cp = 0

                    Case Is = "Cohesivo"

                        'Factor de diámetro
                        If diam(pil) > 1 Then
                            fd = 2 / 3
                        Else
                            fd = 1 - diam(pil) / 3
                        End If

                        'Resistencia por punta a corto plazo en T/m2
                        Qp = 1.5 * fd * ten_efec(longitud) + 9 * fd * c(i) * 0.1
                        Qp_cp = Qp * aencerr(pil)

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_cp = 0
                        If Módulo1.Punta_suelos = False Then Qp_cp = 0

                    Case Is = "Roca"

                        'Factor de empotramiento
                        df = 2

                        'Resistencia por punta

                        Qp = 100 * 2 * df * factordiaclas(i) * (qu(i) / 100) ^ 0.5
                        If Qp > 2000 Then Qp = 2000
                        Qp_cp = Qp * aencerr(pil)
```

```
                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_cp = 0


                    End Select

                Case Is = "CTE-SE-C"

                    Select Case tipoestrato(i)

                    Case Is = "Rellenos"
                        Qp_cp = 0
                    Case Is = "Granular"

                        'Parámetros
                        If Módulo1.ejecucion = "Pilotes perforados" Then
                            fp = 2.5
                        Else
                            fp = 3
                        End If

                        'Resistencia por punta

                        Nq = Exp(3.1416 * Tan(angroz(i))) * (1 + Sin(angroz(i))) / (1 -
Sin(angroz(i)))
                        Qp = fp * ten_efec(longitud) * Nq

                        If Qp > 2000 Then Qp = 2000

                        Qp_cp = Qp * aencerr(pil)

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_cp = 0
                        If Módulo1.Punta_suelos = False Then Qp_cp = 0

                    Case Is = "Cohesivo"
                        'Resistencias a corto plazo
                        Qp = 0.9 * cu(i)
                        Qp_cp = Qp * aencerr(pil)

                        If (longitud - hini(i)) < 2 * diam(pil) Or (hfinal(i) - longitud)
< 2 * diam(pil) Then Qp_cp = 0
                        If Módulo1.Punta_suelos = False Then Qp_cp = 0
                    Case Is = "Roca"
                        If a(i) / s(i) < 0.02 Then
```

```
                            Qp = (3 + s(i) / (1000 * diam(pil))) * qu(i) * diam(pil) * 3
/ (10 * (1 + 300 * a(i) / s(i)) ^ 0.5)
                        Else
                            Qp = (3 + s(i) / (1000 * diam(pil))) * qu(i) * diam(pil) * 3
/ (10 * (1 + 300 * 0.02) ^ 0.5)
                        End If

                        Qp_cp = Qp * aencerr(pil) * Módulo1.coef_seguridad

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_cp = 0

                    End Select

                Case Is = "ROM 05 05"

                    Select Case tipoestrato(i)

                    Case Is = "Rellenos"
                        Qp_cp = 0
                    Case Is = "Granular"

                        'Factor de diámetro

                        If diam(pil) > 0.9 Then
                            fd = 0.7
                        Else
                            fd = 1 - diam(pil) / 3
                        End If

                        'Factores de corrección

                        If Módulo1.ejecucion = "Pilotes hincados" Then
                            Ncorrec = 1
                        Else
                            Ncorrec = 0.5
                        End If

                        'Resistencia por punta

                        Nq = Exp(3.1416 * Tan(angroz(i))) * (1 + Sin(angroz(i))) / (1 -
Sin(angroz(i)))
                        Qp = 3 * fd * Ncorrec * ten_efec(longitud - hini(i)) * Nq
                        If Qp > 2000 Then Qp = 2000
                        Qp_cp = Qp * aencerr(pil)
```

```
                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_cp = 0
                        If Módulo1.Punta_suelos = False Then Qp_cp = 0

                    Case Is = "Cohesivo"

                        'Resistencia por punta a corto plazo

                        If diam(pil) <= 1 Then
                            Qp = 0.1 * (9 - 3 * diam(pil)) * cu(i)
                        Else
                            Qp = 0.6 * cu(i)
                        End If

                        Qp_cp = Qp * aencerr(pil)

                        If (longitud - hini(i)) < 4 * diam(pil) Or (hfinal(i) - longitud)
< 2 * diam(pil) Then Qp_cp = 0
                        If Módulo1.Punta_suelos = False Then Qp_cp = 0

                    Case Is = "Roca"

                        If 2 * (s(i) * 0.001 / diam(pil)) ^ 0.5 < 0.2 * (RQD(i) / di-
am(pil)) ^ 0.5 Then
                            fd = 2 * (s(i) * 0.001 / diam(pil)) ^ 0.5
                        Else
                            fd = 0.2 * (RQD(i) / (diam(pil))) ^ 0.5
                        End If

                        If fd > 1 Then fd = 1
                        'Resistencia por punta

                        Qp = 2 * 100 * ((0.01 * qu(i)) ^ 0.5) * fd * factordiaclas(i) *
3.4

                        Qp_cp = Qp * aencerr(pil)

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_cp = 0


                    End Select

                Case Is = "GMOC (Micropilotes)"

                    Select Case tipoestrato(i)
```

```
                    Case Is = "Rellenos"
                        Qp_cp = 0
                    Case Is = "Granular"
                        'Resistencia por fuste a corto plazo en T/m2
                        tf = 0.1 * c(i) + ko(i) * Tan(angroz2(i)) * ten_efec(longitud)
                        Qp_cp = 0.15 * tf * areal(pil)

                        If Módulo1.Punta_suelos = False Then Qp_cp = 0

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_cp = 0
                        If Módulo1.Punta_suelos = False Then Qp_cp = 0

                    Case Is = "Cohesivo"

                        'Resistencia por fuste a corto plazo en T/m2
                        tf = 0.1 * cu(estrato)

                        Qp_cp = 0.15 * tf * areal(pil)

                        If Módulo1.Punta_suelos = False Then Qp_cp = 0

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_cp = 0
                        If Módulo1.Punta_suelos = False Then Qp_cp = 0

                    Case Is = "Roca"

                        Qp_cp = 0.07 * qu(i) * areal(pil) * Módulo1.coef_seguridad
                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_cp = 0

                    End Select

                End Select

            Exit For
        End If
    Next

    End Function
```

3.2.2. Resistencia unitaria por punta a largo plazo

```
                        'Resistencia por punta a largo plazo en T/m2
                        Nq = 1.5 * Exp(3.1416 * Tan(angroz(i))) * fd * (1 +
Sin(angroz(i))) / (1 - Sin(angroz(i)))
                        Nc = (Nq - 1) / Tan(angroz(i))
                        Qp = Ncorrec * (Nq * ten_efec(hini(i) + 6 * diam(pil)) + Nc * 0.1
* c(i))
                        Qp_lp = Qp * aencerr(pil)

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_lp = 0
                        If Módulo1.Punta_suelos = False Then Qp_lp = 0
                    Case Is = "Cohesivo"

                        'Factor de diámetro
                        If diam(pil) > 1 Then
                            fd = 2 / 3
                        Else
                            fd = 1 - diam(pil) / 3
                        End If

                        'Resistencia por punta a largo plazo en T/m2
                        Nq = 1.5 * Exp(3.1416 * Tan(angroz(i))) * fd * (1 +
Sin(angroz(i))) / (1 - Sin(angroz(i)))
                        Nc = (Nq - 1) / Tan(angroz(i))
                        Qp = (Nq * ten_efec(hini(i) + 6 * diam(pil)) + Nc * 0.1 * c(i)) *
2
                        Qp_lp = Qp * aencerr(pil)

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_lp = 0
                        If Módulo1.Punta_suelos = False Then Qp_lp = 0

                    Case Is = "Roca"

                        'Factor de empotramiento
                        df = 2

                        'Resistencia por punta
```

```
                        Qp = 100 * 2 * df * factordiaclas(i) * (qu(i) / 100) ^ 0.5
                        If Qp > 2000 Then Qp = 2000

                        Qp_lp = Qp * aencerr(pil)

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_lp = 0


                    End Select

                Case Is = "CTE-SE-C"

                    Select Case tipoestrato(i)

                    Case Is = "Rellenos"
                        Qp_lp = 0
                    Case Is = "Granular"

                        'Parámetros
                        If Módulo1.ejecucion = "Pilotes perforados" Then
                            fp = 2.5
                        Else
                            fp = 3
                        End If

                        'Resistencia por punta

                        Nq = Exp(3.1416 * Tan(angroz(i))) * (1 + Sin(angroz(i))) / (1 -
Sin(angroz(i)))
                        Qp = fp * ten_efec(longitud) * Nq

                        If Qp > 2000 Then Qp = 2000

                        Qp_lp = Qp * aencerr(pil)

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_lp = 0
                        If Módulo1.Punta_suelos = False Then Qp_lp = 0

                    Case Is = "Cohesivo"

                        If Módulo1.ejecucion = "Pilotes perforados" Then
                            fp = 2.5
                        Else
                            fp = 3
```

```
                        End If

                        'Resistencia por punta

                        Nq = Exp(3.1416 * Tan(angroz(i))) * (1 + Sin(angroz(i))) / (1 -
Sin(angroz(i)))
                        Qp = fp * ten_efec(longitud) * Nq

                        If Qp > 2000 Then Qp = 2000

                        Qp_lp = Qp * aencerr(pil)

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_lp = 0
                        If Módulo1.Punta_suelos = False Then Qp_lp = 0

                    Case Is = "Roca"

                        If a(i) / s(i) < 0.02 Then
                            Qp = (3 + s(i) / (1000 * diam(pil))) * qu(i) * diam(pil) * 3
/ (10 * (1 + 300 * a(i) / s(i)) ^ 0.5)
                        Else
                            Qp = (3 + s(i) / (1000 * diam(pil))) * qu(i) * diam(pil) * 3
/ (10 * (1 + 300 * 0.02) ^ 0.5)
                        End If

                        Qp_lp = Qp * aencerr(pil) * Módulo1.coef_seguridad

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_lp = 0

                    End Select

                Case Is = "ROM 05 05"

                    Select Case tipoestrato(i)

                    Case Is = "Rellenos"
                        Qp_lp = 0
                    Case Is = "Granular"

                        'Factor de diámetro

                        If diam(pil) > 0.9 Then
                            fd = 0.7
                        Else
```

```
                    fd = 1 - diam(pil) / 3
                End If

                'Factores de corrección

                If Módulo1.ejecucion = "Pilotes hincados" Then
                    Ncorrec = 1
                Else
                    Ncorrec = 0.5
                End If

                'Resistencia por punta

                Nq = Exp(3.1416 * Tan(angroz(i))) * (1 + Sin(angroz(i))) / (1 -
Sin(angroz(i)))
                Qp = 3 * fd * Ncorrec * ten_efec(longitud) * Nq
                If Qp > 2000 Then Qp = 2000
                Qp_lp = Qp * aencerr(pil)

                If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_lp = 0
                If Módulo1.Punta_suelos = False Then Qp_lp = 0

            Case Is = "Cohesivo"

                'Resistencia por punta a largo plazo

                Nq = (1 + Sin(angroz(i))) * Exp(3.1416 * Tan(angroz(i))) / (1 -
Sin(angroz(i)))
                Nc = (Nq - 1) / Tan(angroz(i))

                If diam(pil) > 1 Then
                    fd = 0.7
                Else
                    fd = 1 - diam(pil) / 3
                End If

                Qp = (3 * ten_efec(longitud) * Nq + 3 * c(i) * Nc) * fd

                If Módulo1.ejecucion = "Pilotes excavados" Then
                    Qp = 0.5 * Qp
                End If

                Qp_lp = Qp * aencerr(pil)
```

```
                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_lp = 0
                        If Módulo1.Punta_suelos = False Then Qp_lp = 0

                    Case Is = "Roca"

                        If 2 * (s(i) * 0.001 / diam(pil)) ^ 0.5 < 0.2 * (RQD(i) / di-
am(pil)) ^ 0.5 Then
                            fd = 2 * (s(i) * 0.001 / diam(pil)) ^ 0.5
                        Else
                            fd = 0.2 * (RQD(i) / (diam(pil))) ^ 0.5
                        End If

                        If fd > 1 Then fd = 1

                        'Resistencia por punta

                        Qp = 2 * 100 * ((0.01 * qu(i)) ^ 0.5) * fd * factordiaclas(i) *
3.4

                        If Qp > 1500 Then Qp = 1500

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_lp = 0

                        Qp_lp = aencerr(CInt(pil)) * Qp

                    End Select

                Case Is = "GMOC (Micropilotes)"

                    Select Case tipoestrato(i)

                    Case Is = "Rellenos"
                        Qp_lp = 0
                    Case Is = "Granular"
                        'Resistencia por fuste a corto plazo en T/m2
                        tf = 0.1 * c(i) + ko(i) * Tan(angroz2(i)) * ten_efec(longitud)
                        Qp_lp = 0.15 * tf * areal(pil)

                        If Módulo1.Punta_suelos = False Then Qp_lp = 0

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_lp = 0
                        If Módulo1.Punta_suelos = False Then Qp_lp = 0
```

```
                    Case Is = "Cohesivo"

                        'Resistencia por fuste a corto plazo en T/m2
                         tf = 0.1 * c(i) + ko(i) * Tan(angroz2(i)) * ten_efec(longitud)

                        Qp_lp = 0.15 * tf * areal(pil)

                        If Módulo1.Punta_suelos = False Then Qp_lp = 0

                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_lp = 0
                        If Módulo1.Punta_suelos = False Then Qp_lp = 0

                    Case Is = "Roca"
                        Qp_lp = 0.07 * qu(i) * areal(pil) * Módulo1.coef_seguridad
                        If (longitud - hini(i)) < 6 * diam(pil) Or (hfinal(i) - longitud)
< 3 * diam(pil) Then Qp_lp = 0

                    End Select
                End Select
            Exit For
            End If
        Next
        End Function
```

3.3. RESISTENCIA UNITARIA POR FUSTE A CORTO PLAZO

```
        Public Function Qf_cp(longitud As Single, pil As Integer)
        'Determina la resistencia por fuste a corto plazo
        Dim i As Integer

        Qf_cp = 0
            For i = 1 To Módulo1.n_estratos
                If longitud >= hfinal(i) Then

                    Select Case Módulo1.normativa
                    Case Is = "GCOC"
                        Qf_cp = Qf_cp + qf_cp_GCOC(i, hini(i), hfinal(i), pil)
                    Case Is = "CTE-SE-C"
                        Qf_cp = Qf_cp + qf_cp_CTE(i, hini(i), hfinal(i), pil)
                    Case Is = "ROM 05 05"
                        Qf_cp = Qf_cp + qf_cp_ROM(i, hini(i), hfinal(i), pil)
```

```
                    Case Is = "GMOC (Micropilotes)"
                        Qf_cp = Qf_cp + qf_cp_GMOC(i, hini(i), hfinal(i), pil)
                    End Select

            ElseIf longitud < hfinal(i) And longitud > hini(i) Then

                If Módulo1.n_freatico > hini(i) And Módulo1.n_freatico < hfinal(i) Then

                    If longitud <= Módulo1.n_freatico Then
                        Select Case Módulo1.normativa
                        Case Is = "GCOC"
                            Qf_cp = Qf_cp + qf_cp_GCOC(i, hini(i), longitud, pil)
                        Case Is = "CTE-SE-C"
                            Qf_cp = Qf_cp + qf_cp_CTE(i, hini(i), longitud, pil)
                        Case Is = "ROM 05 05"
                            Qf_cp = Qf_cp + qf_cp_ROM(i, hini(i), longitud, pil)
                            Case Is = "GMOC (Micropilotes)"
                            Qf_cp = Qf_cp + qf_cp_GMOC(i, hini(i), longitud, pil)
                        End Select

                    Else
                        Select Case Módulo1.normativa
                        Case Is = "GCOC"
                             Qf_cp = Qf_cp + qf_cp_GCOC(i, hini(i), Módulo1.n_freatico,
pil) + qf_cp_GCOC(i, Módulo1.n_freatico, longitud, pil)
                        Case Is = "CTE-SE-C"
                             Qf_cp = Qf_cp + qf_cp_CTE(i, hini(i), Módulo1.n_freatico,
pil) + qf_cp_CTE(i, Módulo1.n_freatico, longitud, pil)
                        Case Is = "ROM 05 05"
                             Qf_cp = Qf_cp + qf_cp_ROM(i, hini(i), Módulo1.n_freatico,
pil) + qf_cp_ROM(i, Módulo1.n_freatico, longitud, pil)
                        Case Is = "GMOC (Micropilotes)"
                             Qf_cp = Qf_cp + qf_cp_GMOC(i, hini(i), Módulo1.n_freatico,
pil) + qf_cp_GMOC(i, Módulo1.n_freatico, longitud, pil)
                        End Select

                    End If

                Else
                     Select Case Módulo1.normativa
                     Case Is = "GCOC"
                         Qf_cp = Qf_cp + qf_cp_GCOC(i, hini(i), longitud, pil)
                     Case Is = "CTE-SE-C"
                         Qf_cp = Qf_cp + qf_cp_CTE(i, hini(i), longitud, pil)
                     Case Is = "ROM 05 05"
                         Qf_cp = Qf_cp + qf_cp_ROM(i, hini(i), longitud, pil)
```

```
                    Case Is = "GMOC (Micropilotes)"
                        Qf_cp = Qf_cp + qf_cp_GMOC(i, hini(i), longitud, pil)
                    End Select
                End If

            Exit For
            End If
        Next

    End Function
```

3.3.1. Resistencia unitaria por fuste a corto plazo-CTE

```
    Public Function qf_cp_CTE(estrato As Integer, h1 As Single, h2 As Single, pil As Integer) As Single

        Select Case tipoestrato(estrato)
        Case Is = "Rellenos"
            qf_cp_CTE = 0
        Case Is = "Granular"
            'Parámetros
            If Módulo1.ejecucion = "Pilotes perforados" Then
                kf = 0.75
                f = 1
            Else
                kf = 1
                If Módulo1.material = "Hormigón armado" Or Módulo1.material = "Hormigón pretensado" Then
                    f = 0.9
                Else
                    f = 0.8
                End If
            End If

            'Resistencia por fuste

            tf = kf * f * Tan(angroz(estrato))
            If tf > 12 Then tf = 12

            qf_cp_CTE = tf * perimetro(pil) * (h2 - h1) * ten_efec((h1 + h2) / 2)

             If Módulo1.Fuste_suelos = False Then qf_cp_CTE = 0
```

```
        Case Is = "Cohesivo"
            'Resistencias a corto plazo

            tf = 100 * 0.1 * cu(estrato) / (100 + cu(estrato))
            If Módulo1.material = "Metálicos" Then tf = 0.8 * tf

            qf_cp_CTE = tf * perimetro(pil) * (h2 - h1)

            If Módulo1.Fuste_suelos = False Then qf_cp_CTE = 0

        Case Is = "Roca"
            tf = 20 * (qu(estrato) / 100) ^ 0.5
            qf_cp_CTE = tf * perimetro(pil) * (h2 - h1) * Módulo1.coef_seguridad
            If (h2 - h1) < 6 * diam(pil) Then tf = 0
            If Módulo1.Fuste_roca = False Then qf_cp_CTE = 0

        End Select

    End Function
```

3.3.2. Resistencia unitaria por fuste a corto plazo-GCOC

```
    Public Function qf_cp_GCOC(estrato As Integer, h1 As Single, h2 As Single, pil As In-
teger) As Single

        Select Case tipoestrato(estrato)
        Case Is = "Rellenos"
            qf_cp_GCOC = 0
        Case Is = "Granular"
            If Módulo1.ejecucion = "Pilotes perforados" Then
                m = 1
            ElseIf Módulo1.ejecucion = "Pilotes hincados" Then
                If Módulo1.material = "Hormigón armado" Or Módulo1.material = "Hormigón
pretensado" Then
                    m = 1.3
                ElseIf Módulo1.material = "Metálicos" Then
                    m = 0.9
                ElseIf Módulo1.material = "Madera" Then
                    m = 1.4
                End If
            End If

            'Resistencia por fuste a largo plazo en T/m2
```

```
            tf = 0.1 * c(estrato) + ko(estrato) * Tan(angroz2(estrato)) * ten_efec((h1 +
h2) / 2)
            If tf > 9 Then tf = 9
            qf_cp_GCOC = m * tf * perimetro(pil) * (h2 - h1)

            If Módulo1.Fuste_suelos = False Then qf_cp_GCOC = 0

        Case Is = "Cohesivo"
            If Módulo1.ejecucion = "Pilotes perforados" Then
                m = 1
            ElseIf Módulo1.ejecucion = "Pilotes hincados" Then
                If Módulo1.material = "Hormigón armado" Or Módulo1.material = "Hormigón
pretensado" Then
                    m = 0.9
                ElseIf Módulo1.material = "Metálicos" Then
                    m = 0.6
                ElseIf Módulo1.material = "Madera" Then
                    m = 1
                End If
            End If

            'Resistencia por fuste a corto plazo en T/m2
            tf = 0.1 * 100 * cu(estrato) / (cu(estrato) + 100)
            If tf > 7 Then tf = 7
            qf_cp_GCOC = m * tf * perimetro(pil) * (h2 - h1)

            If Módulo1.Fuste_suelos = False Then qf_cp_GCOC = 0


        Case Is = "Roca"
           'Factor de empotramiento
           df = 2

           'Resistencia por punta

           Qp = 2 * 100 * df * factordiaclas(estrato) * (qu(estrato) / 100) ^ 0.5
           If Qp > 2000 Then Qp = 2000

           'Resistencia por fuste

           tf = 0.1 * Qp

           If (h2 - h1) < 6 * diam(pil) Then tf = 0

           qf_cp_GCOC = tf * perimetro(pil) * (h2 - h1)
```

```
            If Módulo1.Fuste_roca = False Then qf_cp_GCOC = 0


        End Select

    End Function
```

3.3.3. Resistencia unitaria por fuste a corto plazo-ROM

```
    Public Function qf_cp_ROM(estrato As Integer, h1 As Single, h2 As Single, pil As In-
teger) As Single

        Select Case tipoestrato(estrato)
        Case Is = "Rellenos"
            qf_cp_ROM = 0
        Case Is = "Granular"
            'Factores de corrección

            If Módulo1.ejecucion = "Pilotes hincados" Then
                k = 0.75
            Else
                k = 0.5
            End If

            tf = k * Tan(angroz(estrato)) * ten_efec((h1 + h2) / 2)
            If Módulo1.material = "Metálicos" Then tf = 0.9 * tf

            If tf > 12.5 And Módulo1.ejecucion = "Pilotes hincados" Then tf = 12.5
            If tf > 7 And Módulo1.ejecucion = "Pilotes excavados" Then tf = 7

            qf_cp_ROM = tf * perimetro(pil) * (h2 - h1)
            If Módulo1.Fuste_suelos = False Then qf_cp_ROM = 0
        Case Is = "Cohesivo"

            'Resistencia por fuste a corto plazo
            tf = 100 * 0.1 * cu(estrato) / (100 + cu(estrato))
            qf_cp_ROM = tf * perimetro(pil) * (h2 - h1)
            If Módulo1.Fuste_suelos = False Then qf_cp_ROM = 0
        Case Is = "Roca"
            If 2 * (s(estrato) * 0.001 / diam(pil)) ^ 0.5 < 0.2 * (RQD(estrato) / di-
am(pil)) ^ 0.5 Then
               fd = 2 * (s(estrato) * 0.001 / diam(pil)) ^ 0.5
           Else
```

```
                fd = 0.2 * (RQD(estrato) / diam(pil)) ^ 0.5
            End If

            If fd > 1 Then fd = 1
            'Resistencia por fuste

            tf = 0.3 * 100 * ((0.01 * qu(estrato)) ^ 0.5) * fd * factordiaclas(estrato)
            If tf > 200 Then tf = 200

            qf_cp_ROM = tf * perimetro(pil) * (h2 - h1)

            If (h2 - h1) < 6 * diam(pil) Then qf_cp_ROM = 0
            If Módulo1.Fuste_roca = False Then qf_cp_ROM = 0

        End Select

    End Function
```

3.3.4. Resistencia unitaria por fuste a corto plazo-GMOC

```
    Public Function qf_cp_GMOC(estrato As Integer, h1 As Single, h2 As Single, pil As In-
teger) As Single

        Select Case tipoestrato(estrato)
        Case Is = "Rellenos"
            qf_cp_GMOC = 0
        Case Is = "Granular"

            'Resistencia por fuste a corto plazo en T/m2
            tf = 0.1 * c(estrato) + ko(estrato) * Tan(angroz2(estrato)) * ten_efec((h1 +
h2) / 2)
            qf_cp_GMOC = tf * perimetro(pil) * (h2 - h1)

            If Módulo1.Fuste_suelos = False Then qf_cp_GMOC = 0

        Case Is = "Cohesivo"

            'Resistencia por fuste a corto plazo en T/m2
            tf = 0.1 * cu(estrato)

            qf_cp_GMOC = tf * perimetro(pil) * (h2 - h1)

            If Módulo1.Fuste_suelos = False Then qf_cp_GMOC = 0
```

```
    Case Is = "Roca"

       'Se opta por 30 por ser un valor intermedio de los especificados en la guía

       qf_cp_GMOC = 30 * perimetro(i) * (h2 - h1)

       If Módulo1.Fuste_roca = False Then qf_cp_GMOC = 0


    End Select

End Function
```

3.4. RESISTENCIA UNITARIA POR FUSTE A LARGO PLAZO

```
Public Function Qf_lp(longitud As Single, pil As Integer) As Single
'Determina la resistencia por fuste a largo plazo
Qf_lp = 0
Dim i As Integer
For i = 1 To Módulo1.n_estratos
        If longitud >= hfinal(i) Then

                Select Case Módulo1.normativa
                Case Is = "GCOC"
                    Qf_lp = Qf_lp + qf_lp_GCOC(CInt(i), hini(i), hfinal(i), pil)
                Case Is = "CTE-SE-C"
                    Qf_lp = Qf_lp + qf_lp_CTE(CInt(i), hini(i), hfinal(i), pil)
                Case Is = "ROM 05 05"
                    Qf_lp = Qf_lp + qf_lp_ROM(i, hini(i), hfinal(i), pil)
                Case Is = "GMOC (Micropilotes)"
                    Qf_lp = Qf_lp + qf_lp_GMOC(i, hini(i), hfinal(i), pil)
                End Select

        ElseIf longitud < hfinal(i) And longitud > hini(i) Then

            If Módulo1.n_freatico > hini(i) And Módulo1.n_freatico < hfinal(i) Then

                If longitud <= Módulo1.n_freatico Then
                    Select Case Módulo1.normativa
                    Case Is = "GCOC"
```

```
                            Qf_lp = Qf_lp + qf_lp_GCOC(i, hini(i), longitud, pil)
                        Case Is = "CTE-SE-C"
                            Qf_lp = Qf_lp + qf_lp_CTE(i, hini(i), longitud, pil)
                        Case Is = "ROM 05 05"
                            Qf_lp = Qf_lp + qf_lp_ROM(i, hini(i), longitud, pil)
                        Case Is = "GMOC (Micropilotes)"
                            Qf_lp = Qf_lp + qf_lp_GMOC(i, hini(i), longitud, pil)
                        End Select

                    Else
                        Select Case Módulo1.normativa
                        Case Is = "GCOC"
                             Qf_lp = Qf_lp + qf_lp_GCOC(i, hini(i), Módulo1.n_freatico,
pil) + qf_lp_GCOC(i, Módulo1.n_freatico, longitud, pil)
                        Case Is = "CTE-SE-C"
                             Qf_lp = Qf_lp + qf_lp_CTE(i, hini(i), Módulo1.n_freatico,
pil) + qf_lp_CTE(i, Módulo1.n_freatico, longitud, pil)
                        Case Is = "ROM 05 05"
                             Qf_lp = Qf_lp + qf_lp_ROM(i, hini(i), Módulo1.n_freatico,
pil) + qf_lp_ROM(i, Módulo1.n_freatico, longitud, pil)
                        Case Is = "GMOC (Micropilotes)"
                            Qf_lp = Qf_lp + qf_lp_GMOC(i, hini(i), Módulo1.n_freatico,
pil) + qf_lp_GMOC(i, Módulo1.n_freatico, longitud, pil)
                        End Select

                    End If

                Else
                    Select Case Módulo1.normativa
                    Case Is = "GCOC"
                        Qf_lp = Qf_lp + qf_lp_GCOC(i, hini(i), longitud, pil)
                    Case Is = "CTE-SE-C"
                        Qf_lp = Qf_lp + qf_lp_CTE(i, hini(i), longitud, pil)
                    Case Is = "ROM 05 05"
                        Qf_lp = Qf_lp + qf_lp_ROM(i, hini(i), longitud, pil)
                    Case Is = "GMOC (Micropilotes)"
                        Qf_lp = Qf_lp + qf_lp_GMOC(i, hini(i), longitud, pil)
                    End Select
                End If

            Exit For
            End If
        Next

    End Function
```

3.4.1. Resistencia unitaria por fuste a largo plazo-CTE

```
Public Function qf_lp_CTE(estrato As Integer, h1 As Single, h2 As Single, pil As Integer) As Single

    Select Case tipoestrato(estrato)
    Case Is = "Rellenos"
        qf_lp_CTE = 0
    Case Is = "Granular"
        'Parámetros
        If Módulo1.ejecucion = "Pilotes perforados" Then
            kf = 0.75
            f = 1
        Else
            kf = 1
            If Módulo1.material = "Hormigón armado" Or Módulo1.material = "Hormigón pretensado" Then
                f = 0.9
            Else
                f = 0.8
            End If
        End If

        'Resistencia por fuste

        tf = kf * f * Tan(angroz(estrato))
        If tf > 12 Then tf = 12

        qf_lp_CTE = tf * perimetro(pil) * (h2 - h1) * ten_efec((h1 + h2) / 2)
        If Módulo1.Fuste_suelos = False Then qf_lp_CTE = 0

    Case Is = "Cohesivo"
       'Parámetros
        If Módulo1.ejecucion = "Pilotes perforados" Then
            kf = 0.75
            f = 1
        Else
            kf = 1
            If Módulo1.material = "Hormigón armado" Or Módulo1.material = "Hormigón pretensado" Then
                f = 0.9
            Else
                f = 0.8
```

```
                End If
            End If

            'Resistencia por fuste

            tf = kf * f * Tan(angroz(estrato))
            If tf > 12 Then tf = 12

            qf_lp_CTE = tf * perimetro(pil) * (h2 - h1) * ten_efec((h1 + h2) / 2)
            If Módulo1.Fuste_suelos = False Then qf_lp_CTE = 0

        Case Is = "Roca"
            tf = 20 * (qu(estrato) / 100) ^ 0.5
            If (h2 - h1) < 6 * diam(pil) Then tf = 0
            qf_lp_CTE = tf * perimetro(pil) * (h2 - h1) * Módulo1.coef_seguridad
            If Módulo1.Fuste_roca = False Then qf_lp_CTE = 0

        End Select

    End Function
```

3.4.2. Resistencia unitaria por fuste a largo plazo-GCOC

```
    Public Function qf_lp_GCOC(estrato As Integer, h1 As Single, h2 As Single, pil As In-
teger) As Single

        Select Case tipoestrato(estrato)
        Case Is = "Rellenos"
            qf_lp_GCOC = 0
        Case Is = "Granular"
            If Módulo1.ejecucion = "Pilotes perforados" Then
                m = 1
            ElseIf Módulo1.ejecucion = "Pilotes hincados" Then
                If Módulo1.material = "Hormigón armado" Or Módulo1.material = "Hormigón
pretensado" Then
                    m = 1.3
                ElseIf Módulo1.material = "Metálicos" Then
                    m = 0.9
                ElseIf Módulo1.material = "Madera" Then
                    m = 1.4
                End If
            End If
```

```
            'Resistencia por fuste a largo plazo en T/m2
            tf = 0.1 * c(estrato) + ko(estrato) * Tan(angroz2(estrato)) * ten_efec((h1 +
h2) / 2)
            If tf > 9 Then tf = 9
            qf_lp_GCOC = m * tf * perimetro(pil) * (h2 - h1)

            If Módulo1.Fuste_suelos = False Then qf_lp_GCOC = 0


        Case Is = "Cohesivo"
            If Módulo1.ejecucion = "Pilotes perforados" Then
                m = 1
            ElseIf Módulo1.ejecucion = "Pilotes hincados" Then
                If Módulo1.material = "Hormigón armado" Or Módulo1.material = "Hormigón
pretensado" Then
                    m = 0.9
                ElseIf Módulo1.material = "Metálicos" Then
                    m = 0.6
                ElseIf Módulo1.material = "Madera" Then
                    m = 1
                End If
            End If

            'Resistencia por fuste a largo plazo en T/m2
            tf = 0.1 * c(estrato) + ko(estrato) * Tan(angroz2(estrato)) * ten_efec((h1 +
h2) / 2)
            If tf > 9 Then tf = 9

            qf_lp_GCOC = m * tf * perimetro(pil) * (h2 - h1)

            If Módulo1.Fuste_suelos = False Then qf_lp_GCOC = 0

        Case Is = "Roca"
            'Factor de empotramiento
           df = 2

           'Resistencia por punta

           Qp = 2 * 100 * df * factordiaclas(estrato) * (qu(estrato) / 100) ^ 0.5
           If Qp > 2000 Then Qp = 2000

           'Resistencia por fuste

           tf = 0.1 * Qp

           If (h2 - h1) < 6 * diam(pil) Then tf = 0
```

```
            qf_lp_GCOC = qf_lp_GCOC + tf * perimetro(pil) * (h2 - h1)

            If Módulo1.Fuste_roca = False Then qf_lp_GCOC = 0

        End Select

    End Function
```

3.4.3. Resistencia unitaria por fuste a largo plazo-ROM

```
    Public Function qf_lp_ROM(estrato As Integer, h1 As Single, h2 As Single, pil As In-
teger) As Single
        Select Case tipoestrato(estrato)
        Case Is = "Rellenos"
            qf_lp_ROM = 0
        Case Is = "Granular"
            'Factores de corrección

            If Módulo1.ejecucion = "Pilotes hincados" Then
                k = 0.75
            Else
                k = 0.5
            End If

            tf = k * Tan(angroz(estrato)) * ten_efec((h1 + h2) / 2)
            If Módulo1.material = "Metálicos" Then tf = 0.9 * tf

            If tf > 12.5 And Módulo1.ejecucion = "Pilotes hincados" Then tf = 12.5
            If tf > 7 And Módulo1.ejecucion = "Pilotes excavados" Then tf = 7

            qf_lp_ROM = tf * perimetro(pil) * (h2 - h1)
            If Módulo1.Fuste_suelos = False Then qf_lp_ROM = 0

        Case Is = "Cohesivo"

            If Módulo1.ejecucion = "Pilotes hincados" Then
                k = 0.75
            Else
                k = 0.5
            End If

            tf = k * Tan(angroz(estrato)) * ten_efec((h1 + h2) / 2) + c(estrato) * 0.1
```

```
            qf_lp_ROM = tf * perimetro(pil) * (h2 - h1)
            If Módulo1.Fuste_suelos = False Then qf_lp_ROM = 0

        Case Is = "Roca"
            If 2 * (s(estrato) * 0.001 / diam(pil)) ^ 0.5 < 0.2 * (RQD(estrato) / di-
am(pil)) ^ 0.5 Then
                fd = 2 * (s(estrato) * 0.001 / diam(pil)) ^ 0.5
            Else
                fd = 0.2 * (RQD(estrato) / diam(pil)) ^ 0.5
            End If

            If fd > 1 Then fd = 1
            'Resistencia por fuste

            tf = 0.3 * 100 * ((0.01 * qu(estrato)) ^ 0.5) * fd * factordiaclas(estrato)
            If tf > 200 Then tf = 200

            qf_lp_ROM = tf * perimetro(pil) * (h2 - h1)

            If (h2 - h1) < 6 * diam(pil) Then qf_lp_ROM = 0
            If Módulo1.Fuste_roca = False Then qf_lp_ROM = 0

        End Select

    End Function
```

3.4.4. Resistencia unitaria por fuste a largo plazo-GMOC

```
    Public Function qf_lp_GMOC(estrato As Integer, h1 As Single, h2 As Single, pil As In-
teger) As Single

        Select Case tipoestrato(estrato)
        Case Is = "Rellenos"
            qf_lp_GMOC = 0
        Case Is = "Granular"

            'Resistencia por fuste a corto plazo en T/m2
            tf = 0.1 * c(estrato) + ko(estrato) * Tan(angroz2(estrato)) * ten_efec((h1 +
h2) / 2)
            qf_lp_GMOC = tf * perimetro(pil) * (h2 - h1)

            If Módulo1.Fuste_suelos = False Then qf_lp_GMOC = 0
```

```
        Case Is = "Cohesivo"

            'Resistencia por fuste a corto plazo en T/m2
            tf = 0.1 * c(estrato) + ko(estrato) * Tan(angroz2(estrato)) * ten_efec((h1 +
h2) / 2)
            qf_lp_GMOC = tf * perimetro(pil) * (h2 - h1)

            If Módulo1.Fuste_suelos = False Then qf_lp_GMOC = 0

        Case Is = "Roca"

           'Se opta por 30 por ser un valor intermedio de los especificados en la guía

           qf_lp_GMOC = 30 * perimetro(i) * (h2 - h1)

           If Módulo1.Fuste_roca = False Then qf_lp_GMOC = 0


        End Select

    End Function
```

3.5. CARGA MÁXIMA SOBRE PILOTE

```
    Public Function Nmax(N, Mx, My, encep, k, pil) As Single

    'Esta función permite calcular el axil máximo sobre los pilotes según las cargas y la
disposición de encepado escogida
    Nimax = 0
    'Nn es el axil debido al axil sobre el arranque
    Nn = N / disposiciones(encep, 0)

    'Se comprueba el axil para todos los pilotes del encepado, seleccionando el máximo

    For i = 1 To (CSng(disposiciones(encep, 0)) * 2 - 1) Step 2
        Ni = Nn + Mx * k * diam(pil) * disposiciones(encep, i) / (disposiciones2(encep,
6) * (k ^ 2) * (diam(pil) ^ 2)) + My * k * diam(pil) * disposiciones(encep, i + 1) / (dis-
posiciones2(encep, 7) * (k ^ 2) * (diam(pil) ^ 2))
            If Ni > Nimax Then Nimax = Ni
    Next

    Nmax = Nimax

    End Function
```

3.6. CARGA MÍNIMA SOBRE PILOTE

```
'Esta función permite calcular el axil mínimo sobre los pilotes según las cargas y la disposición de encepado escogida
'Servirá para determinar
Nimin = Nmax(N, Mx, My, encep, k, pil)
'Nn es el axil debido al axil sobre el arranque
Nn = N / disposiciones(encep, 0)

'Se comprueba el axil para todos los pilotes del encepado, seleccionando el máximo

For i = 1 To (CSng(disposiciones(encep, 0)) * 2 - 1) Step 2
    Ni = Nn + Mx * k * diam(pil) * disposiciones(encep, i) / (disposiciones2(encep, 6) * (k ^ 2) * (diam(pil) ^ 2)) + My * k * diam(pil) * disposiciones(encep, i + 1) / (disposiciones2(encep, 7) * (k ^ 2) * (diam(pil) ^ 2))
        If Ni < Nimin Then Nimin = Ni
Next

Nmin = Nimin

End Function
```

3.7. CORTANTE MÁXIMO SOBRE PILOTE

```
Public Function Qmax(encep, Qx, Qy, Mt, k, pil) As Single

Qmax = 0

For i = 1 To (CSng(disposiciones(encep, 0)) * 2 - 1) Step 2
    Hx = Qx / Módulo1.disposiciones(encep, 0) - Mt * k * diam(pil) * disposiciones(encep, i + 1) / (Módulo1.disposiciones2(encep, 8) * (k ^ 2) * (diam(pil) ^ 2))
    Hy = Qy / Módulo1.disposiciones(encep, 0) + Mt * k * diam(pil) * disposiciones(encep, i) / (Módulo1.disposiciones2(encep, 8) * (k ^ 2) * (diam(pil) ^ 2))
    Qi = (Hx ^ 2 + Hy ^ 2) ^ 0.5
    If Qi > Qmax Then Qmax = Qi
Next
```

3.8. PESO DEL ENCEPADO

```
Public Function Nencep(encep, k, pil As Integer) As Single
    'Calcula la carga correspondiente al peso del encepado
```

```
        If Módulo1.tipoencepados_carga <> 3 Then
            If Módulo1.tipoencepados_carga = 1 Then
                h = disposiciones2(encep, 5) * k * diam(pil)
            Else
                If diam(pil) > 0.4 Then
                    h = diam(pil)
                Else
                    h = 0.4
                End If
            End If
            'A1 es la aproximación circular y A2 la rectangular,ambas siguen los crite-
rios de la EHE-08

            A1 = 3.1416 * 0.25 * (disposiciones2(encep, 5) * 2 * k * diam(pil) +
diam(pil) + 0.5) ^ 2
            A2 = ((disposiciones2(encep, 1) - disposiciones2(encep, 2)) * k * diam(pil) +
diam(pil) + 0.5) * ((disposiciones2(encep, 3) - disposiciones2(encep, 4)) * k * diam(pil) +
diam(pil) + 0.5)

            If A1 > A2 Then
                Nencep = A2 * h * Módulo1.densidad_hormigon / disposiciones(encep, 0)
            Else
                Nencep = A1 * h * Módulo1.densidad_hormigon / disposiciones(encep, 0)
            End If
        Else
            Nencep = 0
        End If

    End Function
```

3.9. PESO PROPIO DEL PILOTE

```
    Public Function Npilote(pil As Integer, longitud As Single) As Single
    'Calcula el peso propio del pilote. No se considera la reducción de este al estar por
debajo del nivel freático
    If Módulo1.material = "Hormigón armado" Or Módulo1.material = "Hormigón pretensado"
Then
        densidad = Módulo1.densidad_hormigon
        Npilote = longitud * areal(pil) * densidad + (aencerr(pil) - areal(pil)) *
ten_efec(longitud)
    ElseIf Módulo1.material = "Metálicos" Then
        densidad = Módulo1.densidad_acero
        Npilote = longitud * areal(pil) * densidad + (aencerr(pil) - areal(pil)) *
ten_efec(longitud)
    Else
```

```
        densidad = Módulo1.densidad_madera
        Npilote = longitud * areal(pil) * densidad + (aencerr(pil) - areal(pil)) *
ten_efec(longitud)
    End If
```

3.10. FRICCIÓN NEGATIVA

```
    Public Function Nrozneg(pil As Integer) As Single

    Nrozneg = 0

    If Módulo1.rozamiento_negativo = False Then Exit Function

        For i = 1 To Módulo1.n_estratos
            If tipoestrato(i) = "Relleno" Then
                Nrozneg = Nrozneg + (ko(i) * Tan(angroz2(i)) * tenmed(i)) * perimet-
ro(pil) * Esp(i)
            Else
                Nrozneg = Nrozneg + 0
            End If
        Next

    End Function
```

3.11. TOPE ESTRUCTURAL

```
    Public Function comprobacion_tope_estructural(N As Single, pil As Integer) As Boolean

    If N > 100 * Módulo1.capacidad_portante * areal(pil) Then
        comprobacion_tope_estructural = False
    Else
        comprobacion_tope_estructural = True
    End If

    End Function
```

3.11.1. Tope estructural-micropilotes

```
Public Function tope_estructural_micropilote(pil)

'diametro en mm
diametro = CSng(Mid(Módulo1.listapilotes(pil, 0), 3, 3))
```

```
dext = CSng(Mid(Módulo1.listapilotes(pil, 0), 10, 5))
espesor = CSng(Mid(Módulo1.listapilotes(pil, 0), 16, 4))
fyk = CSng(Mid(Módulo1.listapilotes(pil, 0), 21, 3))
nrefuerzos = CSng(Mid(Módulo1.listapilotes(pil, 0), 25, 1))
fsk = CSng(Mid(Módulo1.listapilotes(pil, 0), 30, 3))
Re = Módulo1.Re
fck = CSng(Mid(Módulo1.listapilotes(pil, 0), 7, 2))

Aabrut = 3.1416 * 0.25 * (dext ^ 2 - (dext - 2 * espesor) ^ 2)
Aanet = 3.1416 * 0.25 * ((dext - 2 * Re) ^ 2 - (dext - 2 * espesor) ^ 2)

A_s = 3.1416 * 0.25 * nrefuerzos * 25 ^ 2

Ac = 3.1416 * 0.25 * diametro ^ 2 - A_s - Aabrut
If fsk = 400 Then
    fsd = 400 / 1.15
ElseIf fsk = 500 Then
    fsd = 400
End If

If fyk / 1.1 > 400 Then
    fyd = 400
ElseIf fsk = 400 Then
    fyd = fyk / 1.1
End If
fcd = fck / 1.5
Ncd = (0.85 * Ac * fcd + A_s * fsd + Aanet * fyd) * R / (1.2 * fe)

'al final se pasa a MPa

tope_estructural_micropilote = Ncd / (3.1416 * 0.25 * diametro ^ 2)

End Function
```

3.12. DIMENSIONAMIENTO DE LONGITUD

```
    Public Function dimensionamientoL(N As Single, Mx As Single, My As Single, encep As
Integer, k As Single, pil As Integer)
    'Determina la longitud de pilote necesaria para soportar las cargas
    If Módulo1.rozamiento_negativo = True Then
        n1 = Nmax(N, Mx, My, encep, k, pil) + Nencep(encep, k, CInt(pil)) + Nroz-
neg(CInt(pil))
    Else
        n1 = Nmax(N, Mx, My, encep, k, pil) + Nencep(encep, k, CInt(pil))
```

```
    End If

    Dim L As Single

    For L = 0.5 To hfinal(Módulo1.n_estratos) Step 0.5
        Ntot = n1 + Npilote(CInt(pil), CSng(L))
        'Aquí se comprueba que cumple con la capacidad portante, de no ser así no dejaría
seguir
        If Módulo1.normativa = "GMOC (Micropilotes)" Then Módulo1.capacidad_portante =
tope_estructural_micropilote(pil)


        If Ntot - Qf_cp(L, pil) > 100 * Módulo1.capacidad_portante * areal(pil) Or Ntot -
Qf_lp(L, pil) > 100 * Módulo1.capacidad_portante * areal(pil) Then
            dimensionamientoL = -1
            Exit For
        End If



        If (Qp_cp(L, pil) + Qf_cp(L, pil)) > Módulo1.coef_seguridad * Ntot And (Qp_lp(L,
pil) + Qf_lp(L, pil)) > Módulo1.coef_seguridad * Ntot Then
            If comprobacion_efecto_grupo(N, encep, k, pil, L) = True Then
                    If comprobacion_arranque(N, Mx, My, encep, k, pil, L) = True Then
                        dimensionamientoL = L
                        Exit For
                    End If
            End If
        End If
    Next

    End Function
```

3.13. COMPROBACIÓN DEL EFECTO GRUPO

```
    Public Function comprobacion_efecto_grupo(N As Single, encep As Integer, k As Single,
pil As Integer, L As Single) As Boolean


    If Módulo1.efecto_grupo_coeficiente_eficiencia = True Then

        If k >= 3 Then

            comprobacion_efecto_grupo = True
```

```
        Else

        mu = 0.7 + 0.15 * (k - 1)

            If mu * Módulo1.disposiciones(encep, 0) * (Qp_cp(L, pil) + Qf_cp(L, pil)) /
Módulo1.coef_seguridad > N + Módulo1.disposiciones(encep, 0) * (Nencep(encep, k, CInt(pil))
+ Npilote(CInt(pil), CSng(L))) And mu * Módulo1.disposiciones(encep, 0) * (Qp_lp(L, pil) +
Qf_lp(L, pil)) / Módulo1.coef_seguridad > N + Módulo1.disposiciones(encep, 0) * (Nen-
cep(encep, k, CInt(pil)) + Npilote(CInt(pil), CSng(L))) Then
                comprobacion_efecto_grupo = True
            Else
                comprobacion_efecto_grupo = False
            End If
        End If

    Else

        Módulo1.listapilotes(Módulo1.npilotes + 1, 0) = "Grupo1"
            A1 = 3.1416 * 0.25 * (disposiciones2(encep, 5) * 2 * k * diam(pil) +
diam(pil)) ^ 2
            A2 = ((disposiciones2(encep, 1) - disposiciones2(encep, 2)) * k * diam(pil) +
diam(pil)) * ((disposiciones2(encep, 3) - disposiciones2(encep, 4)) * k * diam(pil) +
diam(pil))

        Módulo1.listapilotes(Módulo1.npilotes + 1, 1) = 10000 * disposiciones(encep, 0) *
areal(pil)

        If A1 < A2 Then
            Módulo1.listapilotes(Módulo1.npilotes + 1, 2) = 10000 * A1
            Módulo1.listapilotes(Módulo1.npilotes + 1, 3) = 100 * 3.1416 * (disposicio-
nes2(encep, 4) * 2 * k * diam(pil) + diam(pil))
        Else
            Módulo1.listapilotes(Módulo1.npilotes + 1, 2) = 10000 * A2
            Módulo1.listapilotes(Módulo1.npilotes + 1, 3) = 100 * 2 * (((disposicio-
nes2(encep, 1) - disposiciones2(encep, 2)) * k * diam(pil) + diam(pil)) + ((disposicio-
nes2(encep, 3) - disposiciones2(encep, 4)) * k * diam(pil) + diam(pil)))
        End If

            'Aquí se decide si se emplea el diámetro del grupo o el individual para la
determinación de las zonas activa y pasiva
            If Módulo1.diametro_efecto_grupo_individual = True Then
                Módulo1.listapilotes(Módulo1.npilotes + 1, 4) = diam(pil) * 100
            Else
                Módulo1.listapilotes(Módulo1.npilotes + 1, 4) = (4 * Módu-
lo1.listapilotes(Módulo1.npilotes + 1, 2) / 3.1416) ^ 0.5
            End If
```

```
            Ntot = N + Nencep(encep, k, CInt(pil)) * disposiciones(encep, 0) + Npilo-
te(Módulo1.npilotes + 1, CSng(L))

            If (Qp_cp(L, Módulo1.npilotes + 1) + Qf_cp(L, Módulo1.npilotes + 1)) > Módu-
lo1.coef_seguridad * Ntot And (Qp_lp(L, Módulo1.npilotes + 1) + Qf_lp(L, Módulo1.npilotes +
1)) > Módulo1.coef_seguridad * Ntot Then
                comprobacion_efecto_grupo = True
            Else
                comprobacion_efecto_grupo = False
            End If
    End If

    Módulo1.listapilotes(Módulo1.npilotes + 1, 0) = ""
    Módulo1.listapilotes(Módulo1.npilotes + 1, 1) = ""
    Módulo1.listapilotes(Módulo1.npilotes + 1, 2) = ""
    Módulo1.listapilotes(Módulo1.npilotes + 1, 3) = ""
    Módulo1.listapilotes(Módulo1.npilotes + 1, 4) = ""
    End Function
```

3.14. COMPROBACIÓN ARRANQUE

```
    Public Function comprobacion_arranque(N As Single, Mx As Single, My As Single, encep
As Integer, k As Single, pil As Integer, L As Single) As Boolean

        If (-Nmin(N, Mx, My, encep, k, pil) - Nencep(encep, k, CInt(pil))) > 0 Then
            If Módulo1.normativa = "CTE-SE-C" Then
                If     Módulo1.Situacion_proyecto     =     "Permanente"     Or     Módu-
lo1.Situacion_proyecto = "Transitoria" Then
                    coef = 3.5
                ElseIf Módulo1.Situacion_proyecto = "Accidental" Then
                    coef = 2.3
                End If

                For i = 1 To Módulo1.n_estratos
                    If L > hini(i) And L < hfinal(i) Then estrato = i
                Next

                If tipoestrato(estrato) = "Roca" Then
                    If 0.7 * Qf_cp(L, pil) / (Módulo1.coef_seguridad * coef) > (-Nmin(N,
Mx, My, encep, k, pil) - Nencep(encep, k, pil)) And 0.7 * Qf_lp(L, pil) / (Módu-
lo1.coef_seguridad * coef) > (-Nmin(N, Mx, My, encep, k, pil) - Nencep(encep, k, pil)) Then
                        comprobacion_arranque = True
                    Else
                        comprobacion_arranque = False
```

```
                    End If
                Else

                    If 0.7 * Qf_cp(L, pil) / coef > (-Nmin(N, Mx, My, encep, k, pil) -
Nencep(encep, k, pil)) And 0.7 * Qf_lp(L, pil) / coef > (-Nmin(N, Mx, My, encep, k, pil) -
Nencep(encep, k, pil)) Then
                        comprobacion_arranque = True
                    Else
                        comprobacion_arranque = False
                    End If

                End If

            ElseIf Módulo1.normativa = "GCOC" Or Módulo1.normativa = "GMOC (Micropi-
lotes)" Then

                    If 0.7 * Qf_cp(L, pil) / Módulo1.coef_seguridad > (-Nmin(N, Mx, My,
encep, k, pil) - Nencep(encep, k, pil)) And 0.7 * Qf_lp(L, pil) / Módulo1.coef_seguridad >
(-Nmin(N, Mx, My, encep, k, pil) - Nencep(encep, k, pil)) Then
                        comprobacion_arranque = True
                    Else
                        comprobacion_arranque = False
                    End If


            ElseIf Módulo1.normativa = "ROM 05 05" Then

                If 0.5 * Qf_cp(L, pil) / Módulo1.coef_seguridad + Npilote(pil, L) > (-
Nmin(N, Mx, My, encep, k, pil) - Nencep(encep, k, pil)) And 0.5 * Qf_lp(L, pil) / Módu-
lo1.coef_seguridad + Npilote(pil, L) > (-Nmin(N, Mx, My, encep, k, pil) - Nencep(encep, k,
pil)) Then
                    comprobacion_arranque = True
                Else
                    comprobacion_arranque = False
                End If

            End If
        Else
            comprobacion_arranque = True
        End If

    End Function
```

3.15. ASIENTO DE PILOTES-ASIENTO INDIVIDUAL

```
    Public Function asiento_individual_lp(P As Single, pil As Integer, L As Sin-
gle) As Single

    'Faltaría revisar las condiciones de la GCOC y la ROM
    'Se mete aqui tambien el efecto grupo

    alfa = (0.5 * Qf_lp(L, pil) + Qp_lp(L, pil)) / (Qf_lp(L, pil) + Qp_lp(L,
pil))

    If Módulo1.material = "Hormigón armado" Or Módulo1.material = "Hormigón pre-
tensado" Then
        modulo_elastico_material = Módulo1.modulo_hormigon
    ElseIf Módulo1.material = "Metálicos" Then
        modulo_elastico_material = Módulo1.modulo_acero
    ElseIf Módulo1.material = "Madera" Then
        modulo_elastico_material = Módulo1.modulo_madera
    End If

    If Módulo1.normativa = "GMOC (Micropilotes)" Then

        diametro = CSng(Mid(Módulo1.listapilotes(pil, 0), 3, 3))
        dext = CSng(Mid(Módulo1.listapilotes(pil, 0), 10, 5))
        espesor = CSng(Mid(Módulo1.listapilotes(pil, 0), 16, 4))
        nrefuerzos = CSng(Mid(Módulo1.listapilotes(pil, 0), 25, 1))

        Aabrut = 3.1416 * 0.25 * (dext ^ 2 - (dext - 2 * espesor) ^ 2)

        A_s = 3.1416 * 0.25 * nrefuerzos * 25 ^ 2

        Ac = 100 * 3.1416 * 0.25 * diam(Int(pil)) ^ 2 - A_s - Aabrut
        modulo_elastico_material = (Módulo1.modulo_acero * (A_s + Aabrut) + Mó-
dulo1.modulo_hormigon * Ac) / (A_s + Aabrut + Ac)
    End If

    asiento_individual_lp = P * (diam(pil) / (40 * (Qp_lp(L, pil) + Qf_lp(L,
pil))) + alfa * L / (areal(pil) * modulo_elastico_material * 100000))
```

3.16. ASIENTO-DE PILOTES-EFECTO GRUPO

```
Public Function asiento_grupo(N As Single, encep As Integer, k As Single, pil As Integer, L As Single) As Single

For i = 1 To Módulo1.n_estratos
    If L > hini(i) And L <= hfinal(i) Then
        estratopunta = i
    End If
Next

If Módulo1.normativa = "GCOC" Then

    If tipoestrato(estratopunta) = "roca" Then
        h1 = L - hini(i)
    Else
        h1 = L / 3
    End If
Else

    alfa = (0.5 * Qf_lp(L, pil) + Qp_lp(L, pil)) / (Qf_lp(L, pil) + Qp_lp(L, pil))
    h1 = (1 - alfa) * L

End If

    A1 = 3.1416 * 0.25 * (disposiciones2(encep, 5) * 2 * k * diam(pil) + diam(pil) + h1) ^ 2
    A2 = ((disposiciones2(encep, 1) - disposiciones2(encep, 2)) * k * diam(pil) + diam(pil) + h1) * ((disposiciones2(encep, 3) - disposiciones2(encep, 4)) * k * diam(pil) + diam(pil) + h1)

    If A1 < A2 Then
        area = A1
    Else
        area = A2
    End If

    asiento_grupo = 0.8 * (N + Nencep(encep, k, pil)) * (1 - v(CInt(estratopunta)) ^ 2) / (E(CInt(estratopunta)) * 10 ^ 5 * A1 ^ 0.5)

End Function
```

3.17. INTERPOLACIONES DE SONDEOS

```
    Public Sub calcular_profundidades_estratos_arranques()

        If Módulo1.tipo_interpolacion = 0 Then

            For arranque = 1 To Módulo1.n_arranques
                For estrato = 1 To Módulo1.n_estratos
                    Módulo1.profundidades_estratos_arranques(estrato, arranque) = Módu-
lo1.listaestratos2(estrato, 0)
                Next
            Next

        ElseIf Módulo1.tipo_interpolacion = 1 Then

            For arranque = 1 To Módulo1.n_arranques
                dist = 10 ^ 5
                For sondeo = 1 To Módulo1.n_sondeos
                    'seleccion del más cercano
                    If distancia_sondeo_arranque(sondeo, arranque) < dist Then
                        distancia_sondeo_arranque(sondeo, arranque) = dist
                        sondeoselecc = sondeo
                    End If
                Next
                    'traslado de valores
                    For estrato = 1 To Módulo1.n_estratos
                        Módulo1.profundidades_estratos_arranques(estrato, arranque) = Mó-
dulo1.profundidades_estratos_sondeos(estrato, sondeoselecc)
                    Next
            Next

        ElseIf Módulo1.tipo_interpolacion = 2 Then
            For arranque = 1 To Módulo1.n_arranques
                sondeo1 = 1
                sondeo2 = 2

                If distancia_sondeo_arranque(sondeo1, arranque) > distan-
cia_sondeo_arranque(sondeo2, arranque) Then
                    sondeo_dist_max = sondeo1
                Else
                    sondeo_dist_max = sondeo2
                End If
```

```
                For sondeo = 3 To Módulo1.n_sondeos

                    If  distancia_sondeo_arranque(sondeo_dist_max,  arranque)  >  distan-
cia_sondeo_arranque(sondeo, arranque) Then

                        If sondeo_dist_max = sondeo1 Then sondeo1 = sondeo
                        If sondeo_dist_max = sondeo2 Then sondeo2 = sondeo

                        If   distancia_sondeo_arranque(sondeo1,   arranque)   >   distan-
cia_sondeo_arranque(sondeo2, arranque) Then
                            sondeo_dist_max = sondeo1
                        Else
                            sondeo_dist_max = sondeo2
                        End If

                    End If
                Next
                'una vez seleccionadoslos máscercanos se ejecuta la interpolación
                Call interpolacion_2_sondeos(arranque, sondeo1, sondeo2)

            Next

        ElseIf Módulo1.tipo_interpolacion = 3 Then

            For arranque = 1 To Módulo1.n_arranques
                sondeo1 = 1
                sondeo2 = 2
                sondeo3 = 3

                If     distancia_sondeo_arranque(sondeo1,     arranque)     >     distan-
cia_sondeo_arranque(sondeo2,  arranque)  &  distancia_sondeo_arranque(sondeo1,  arranque)  >
distancia_sondeo_arranque(sondeo3, arranque) Then
                    sondeo_dist_max = sondeo1
                ElseIf    distancia_sondeo_arranque(sondeo2,    arranque)    >    distan-
cia_sondeo_arranque(sondeo1,  arranque)  &  distancia_sondeo_arranque(sondeo2,  arranque)  >
distancia_sondeo_arranque(sondeo3, arranque) Then
                    sondeo_dist_max = sondeo2
                Else
                    sondeo_dist_max = sondeo3
                End If

                For sondeo = 4 To Módulo1.n_sondeos

                    If  distancia_sondeo_arranque(sondeo_dist_max,  arranque)  >  distan-
cia_sondeo_arranque(sondeo, arranque) Then
```

```
                        If sondeo_dist_max = sondeo1 Then sondeo1 = sondeo
                        If sondeo_dist_max = sondeo2 Then sondeo2 = sondeo
                        If sondeo_dist_max = sondeo3 Then sondeo3 = sondeo

                            If distancia_sondeo_arranque(sondeo1, arranque) > distancia_sondeo_arranque(sondeo2, arranque) & distancia_sondeo_arranque(sondeo1, arranque) > distancia_sondeo_arranque(sondeo3, arranque) Then
                                sondeo_dist_max = sondeo1
                            ElseIf distancia_sondeo_arranque(sondeo2, arranque) > distancia_sondeo_arranque(sondeo1, arranque) & distancia_sondeo_arranque(sondeo2, arranque) > distancia_sondeo_arranque(sondeo3, arranque) Then
                                sondeo_dist_max = sondeo2
                            Else
                                sondeo_dist_max = sondeo3
                            End If
                    End If
                Next
                'una vez seleccionadoslos máscercanos se ejecuta la interpolación
                Call interpolacion_3_sondeos(arranque, sondeo1, sondeo2, sondeo3)

            Next
        End If
    End Sub
```

3.17.1. Dos sondeos

```
    Public Sub interpolacion_2_sondeos(arranque, sondeo1, sondeo2)
    'Se realiza como unainterpolación de 3 añadiendo un punto 3 comoel punto 1
    'al final esmas fácil de loque parece

        xa = CSng(Módulo1.arranques(arranque, 1))
        ya = CSng(Módulo1.arranques(arranque, 2))
        x1 = CSng(Módulo1.lista_sondeos(sondeo1, 1))
        y1 = CSng(Módulo1.lista_sondeos(sondeo1, 2))
        x2 = CSng(Módulo1.lista_sondeos(sondeo2, 1))
        y2 = CSng(Módulo1.lista_sondeos(sondeo2, 2))
        x3 = x1 + y2 - y1
        y3 = y1 + x1 - x2

        For estrato = 1 To Módulo1.n_estratos
            z1 = Módulo1.profundidades_estratos_sondeos(estrato, sondeo1)
            z2 = Módulo1.profundidades_estratos_sondeos(estrato, sondeo2)
            z3 = Módulo1.profundidades_estratos_sondeos(estrato, sondeo1)
```

```
            'matriz de resolución
            M11 = xa - x1
            M12 = ya - y1
            M21 = x2 - x1
            M22 = y2 - y1
            M23 = z2 - z1
            M31 = x3 - x1
            M32 = y3 - y1
            M33 = z3 - z1

            'Cálculo matricial

            Módulo1.profundidades_estratos_arranques(estrato, arranque) = (M12 * M21 *
M33 + M11 * M23 * M32 - M11 * M22 * M33 - M12 * M23 * M31) / (M21 * M32 - M22 * M31) + z1
                'En caso de que por lo que sea salga una menor que la del estrato ante-
rior se corrige y se considera que ese estrato no tiene espesor en el sondeo
            If  Módulo1.profundidades_estratos_arranques(estrato,  arranque)  <  Módu-
lo1.profundidades_estratos_arranques(estrato     -     1,     arranque)     Then     Módu-
lo1.profundidades_estratos_arranques(estrato,          arranque)          =          Módu-
lo1.profundidades_estratos_arranques(estrato - 1, arranque)
        Next
    End Sub
```

3.17.2. Tres sondeos

```
    Public Sub interpolacion_3_sondeos(arranque, sondeo1, sondeo2, sondeo3)

        xa = CSng(Módulo1.arranques(arranque, 1))
        ya = CSng(Módulo1.arranques(arranque, 2))
        x1 = CSng(Módulo1.lista_sondeos(sondeo1, 1))
        y1 = CSng(Módulo1.lista_sondeos(sondeo1, 2))
        x2 = CSng(Módulo1.lista_sondeos(sondeo2, 1))
        y2 = CSng(Módulo1.lista_sondeos(sondeo2, 2))
        x3 = CSng(Módulo1.lista_sondeos(sondeo3, 1))
        y3 = CSng(Módulo1.lista_sondeos(sondeo3, 2))


        For estrato = 1 To Módulo1.n_estratos
            z1 = Módulo1.profundidades_estratos_sondeos(estrato, sondeo1)
            z2 = Módulo1.profundidades_estratos_sondeos(estrato, sondeo2)
            z3 = Módulo1.profundidades_estratos_sondeos(estrato, sondeo3)
            'matriz de resolución

            M11 = xa - x1
            M12 = ya - y1
```

```
            M21 = x2 - x1
            M22 = y2 - y1
            M23 = z2 - z1
            M31 = x3 - x1
            M32 = y3 - y1
            M33 = z3 - z1

            'Cálculo matricial

            Módulo1.profundidades_estratos_arranques(estrato, arranque) = (M12 * M21 *
M33 + M11 * M23 * M32 - M11 * M22 * M33 - M12 * M23 * M31) / (M21 * M32 - M22 * M31) + z1
                'En caso de que por lo que sea salga una menor que la del estrato ante-
rior se corrige y se considera que ese estrato no tiene espesor en el sondeo
            If  Módulo1.profundidades_estratos_arranques(estrato,  arranque)  <  Módu-
lo1.profundidades_estratos_arranques(estrato     -     1,     arranque)     Then     Módu-
lo1.profundidades_estratos_arranques(estrato,          arranque)          =          Módu-
lo1.profundidades_estratos_arranques(estrato - 1, arranque)
         Next

      End Sub
```

4. FUNCIÓN DE COSTE

```
      Public Function coste_ejecucion(encep As Integer, pil As Integer, k As Single, L As
Single) As Single
      'Función objetivo de la optimización de cada encepado

      If Módulo1.material = "Hormigón armado" Or Módulo1.material = "Hormigón pretensado"
Then

          If Módulo1.ejecucion = "Pilotes perforados" Then
              If Módulo1.armado_continuo = True Then
                  coste_pilotes = (Módulo1.precio_hormigon + Módulo1.precio_armaduras * Mó-
dulo1.cuantia_pilotes) * areal(pil) * L * Módulo1.disposiciones(encep, 0)
              Else
                  If Módulo1.longitud_armado_en_diametros = True Then
                      L_armada = Módulo1.longitud_armado_d * diam(pil)
                  Else
                      L_armada = Módulo1.longitud_armado_l
                  End If

                  If L_armada > L Then L_armada = L

                  coste_pilotes = (Módulo1.precio_hormigon * L + Módulo1.precio_armaduras *
Módulo1.cuantia_pilotes * L_armada) * areal(pil) * Módulo1.disposiciones(encep, 0)


              End If
          Else
              coste_pilotes = (Módulo1.precio_hormigon + Módulo1.precio_armaduras * Módu-
lo1.cuantia_pilotes) * areal(pil) * L * Módulo1.disposiciones(encep, 0)
          End If

          If Módulo1.normativa = "GMOC (Micropilotes)" Then coste_pilotes = Módu-
lo1.disposiciones(encep, 0) * coste_micropilote(pil, L)

      ElseIf Módulo1.material = "Metálicos" Then
          precio = Módulo1.precio_acero * 1000 * Módulo1.densidad_acero
          coste_pilotes = precio * areal(pil) * L * Módulo1.disposiciones(encep, 0)
      Else
          precio = Módulo1.precio_madera
          coste_pilotes = precio * areal(pil) * L * Módulo1.disposiciones(encep, 0)
      End If
```

Función de coste

```
      'Para micropilotes se supone que son barrenados CPI-7

      If Módulo1.ejecucion = "Pilotes perforados" Then
          Select Case Módulo1.tipo_perforacion
          'Los costes de perforación se han obtenido de la base de precios de la construc-
ción del gobierno de extremadura 2012
              Case Is = "CPI-2"
                  coste_perforacion = (60.4 * diam(pil) + 8.65) * L * Módu-
lo1.disposiciones(encep, 0)
                  If diam(pil) <= 0.3 Then coste_perforacion = 100000
              Case Is = "CPI-3"
                  coste_perforacion = (53.8 * diam(pil) + 13.12) * L * Módu-
lo1.disposiciones(encep, 0)
                  If diam(pil) <= 0.3 Then coste_perforacion = 100000
              Case Is = "CPI-4"
                  If Módulo1.normativa = "GCOC" Then
                      coste_perforacion = (61.2 * diam(pil) + 28) * L * Módu-
lo1.disposiciones(encep, 0)
                      If L > 10 Then coste_perforacion = (66.6 * diam(pil) + 48) * L * Mó-
dulo1.disposiciones(encep, 0)
                  End If
                  If diam(pil) < 0.45 Then coste_perforacion = 100000
              Case Is = "CPI-5"
                  coste_perforacion = (52.4 * diam(pil) + 22.7) * L * Módu-
lo1.disposiciones(encep, 0)
                  If diam(pil) < 0.5 Then coste_perforacion = 100000
              Case Is = "CPI-6"
                   If Módulo1.normativa = "GCOC" Then
                      coste_perforacion = (82.3 * diam(pil) + 11.472) * L * Módu-
lo1.disposiciones(encep, 0)
                  Else
                      coste_perforacion = (70 * diam(pil) + 0.08) * L * Módu-
lo1.disposiciones(encep, 0)
                  End If
                  If diam(pil) < 0.45 Then coste_perforacion = 100000
              Case Is = "CPI-7"
                   If Módulo1.normativa = "GCOC" Then
                      coste_perforacion = (55.7 * diam(pil) + 22) * L * Módu-
lo1.disposiciones(encep, 0)
                  If diam(pil) < 0.4 Then coste_perforacion = 100000
                  Else
                      coste_perforacion = (68.6 * diam(pil) - 0.45) * L * Módu-
lo1.disposiciones(encep, 0)
                  End If
              Case Is = "CPI-8"
```

```
                    coste_perforacion = (54.9 * diam(pil) - 4.473) * L * Módu-
lo1.disposiciones(encep, 0)
                If diam(pil) < 0.35 Then coste_perforacion = 100000
            End Select
        ElseIf Módulo1.ejecucion = "Pilotes hincados" Then

            coste_perforacion = 14.6 * diam(pil) + 3.44
            'Los datos de precio de hinca se han obtenido del Cuadro de Precios de la Direc-
ción General de Carreteras

        End If

        If Módulo1.normativa = "GMOC (Micropilotes)" Then
            coste_perforacion = (250.4 * diam(pil) + 13.11) * L * Módu-
lo1.disposiciones(encep, 0)
        End If



        coste_encepado = (Módulo1.precio_hormigon_encepado + Módulo1.cuantia_encepado * Módu-
lo1.precio_armaduras) * Nencep(encep, k, CInt(pil)) * Módulo1.disposiciones(encep, 0) /
Módulo1.densidad_hormigon

        coste_ejecucion = coste_pilotes + coste_encepado + coste_perforacion

        End Function
```

5. FUNCIONES DE OPTIMIZACIÓN

5.1. ARRANQUE AISLADO

```
    For pil = 1 To Módulo1.npilotes
        For encep = 1 To Módulo1.n_encepados
            For k = Módulo1.separacion_ejes_minima To Módulo1.separacion_ejes_maxima Step
Módulo1.separacion_ejes_incremento

                L  =  dimensionamientoL(CSng(N.Value),  CSng(Mx.Value),  CSng(My.Value),
CInt(encep), CSng(k), CInt(pil))

                If L = -1 Or L > hfinal(Módulo1.n_estratos) Then

                Else
                    If L <> Vacío Then
                        If coste_ejecucion(encep, CInt(pil), k, CSng(L)) < coste Then
                            coste = coste_ejecucion(encep, CInt(pil), k, CSng(L))
                            pildef = pil
                            encepdef = encep
                            kdef = k
                            Ldef = L
                        End If
                    End If
                End If

            Next
        Next
    Next
```

5.2. PLANTA-UN PILOTE

```
    Public Sub optimizacion_planta_1_diametro()

    'Optimización económica de la planta con un solo tipo de pilote

    ReDim Módulo1.arranques_optimizacion(50, 10, 100)

    coste_planta = 10 ^ 9
    For i = 1 To Módulo1.npilotes
        coste_planta_i = 0
```

```
        'Primero se optimizan todos los arranques
        For i2 = 1 To Módulo1.n_arranques

            'Aquí se cambiarán los estratos obtenidos según la interpolación
            Call cambiar_estratos("arranque", i2)

            coste = 10 ^ 6
            For encep = 1 To Módulo1.n_encepados
                For k = Módulo1.separacion_ejes_minima To Módulo1.separacion_ejes_maxima
Step Módulo1.separacion_ejes_incremento

                    L = dimensionamientoL(CSng(Módulo1.arranques(i2, 3)),
CSng(Módulo1.arranques(i2, 4)), CSng(Módulo1.arranques(i2, 5)), CInt(encep), CSng(k),
CInt(i))

                    If L = -1 Or L > hfinal(Módulo1.n_estratos) Then

                    Else
                        If coste_ejecucion(CInt(encep), CInt(i), CSng(k), CSng(L)) <
coste Then
                            coste = coste_ejecucion(CInt(encep), CInt(i), CSng(k),
CSng(L))
                            Módulo1.arranques_optimizacion(i2, 0, i) = i
                            Módulo1.arranques_optimizacion(i2, 2, i) = encep
                            Módulo1.arranques_optimizacion(i2, 1, i) = k
                            Módulo1.arranques_optimizacion(i2, 3, i) = L
                            Módulo1.arranques_optimizacion(i2, 4, i) = coste
                            Módulo1.arranques_optimizacion(i2, 5, i) = asien-
to_individual_lp(CSng(Módulo1.arranques(i2, 3)) / Módulo1.disposiciones(encep, 0) + Nen-
cep(encep, k, CInt(i)), CInt(i), CSng(L)) + asiento_grupo(CSng(Módulo1.arranques(i2, 3)) +
Módulo1.disposiciones(encep, 0) * Nencep(encep, k, CInt(i)), CInt(encep), CSng(k), CInt(i),
CSng(L))
                        End If
                    End If
                Next
            Next

            'Aquí se calcula el coste total de la planta
            coste_planta_i = coste_planta_i + coste
        Next
        'Aquí selecciona la más económica
        If coste_planta_i < coste_planta Then
        pilote = i
        coste_planta = coste_planta_i
        End If
    Next
```

```
'Una vez seleccionado el diámetro se va a proceder a detectar y corregir fallos por distorsión angular

For A1 = 1 To Módulo1.n_arranques

    For b = 0 To 5
        Módulo1.arranques(A1, 9) = Módulo1.arranques_optimizacion(A1, 2, pilote)
        Módulo1.arranques(A1, 10) = Módulo1.arranques_optimizacion(A1, 1, pilote)
        Módulo1.arranques(A1, 11) = pilote
        Módulo1.arranques(A1, 12) = Módulo1.arranques_optimizacion(A1, 3, pilote)
    Next
Next



    ReDim Módulo1.distorsiones_angulares(1000, 10)
    Módulo1.distorsiones_angulares(0, 0) = "Arranque"
    Módulo1.distorsiones_angulares(0, 1) = "Asiento (mm)"
    Módulo1.distorsiones_angulares(0, 2) = "Arranque"
    Módulo1.distorsiones_angulares(0, 3) = "Asiento(mm)"
    Módulo1.distorsiones_angulares(0, 4) = "Distancia (m)"
    Módulo1.distorsiones_angulares(0, 5) = "Distorsión"
    Módulo1.distorsiones_angulares(0, 6) = "1/"
    Módulo1.distorsiones_angulares(0, 7) = "Estado"

    P = 1
    For arranque1 = 1 To Módulo1.n_arranques - 1
        For arranque2 = arranque1 + 1 To Módulo1.n_arranques
        'Los datos a emplear son de la matriz arranques (¡A SECAS¡
        'Si no la solución no tendría que ser válida
            Módulo1.distorsiones_angulares(P, 0) = arranque1
            Módulo1.distorsiones_angulares(P, 1) = Round(1000 * (asiento_individual_lp(CSng(Módulo1.arranques(arranque1, 3)) / Módulo1.disposiciones(Módulo1.arranques(arranque1, 9), 0) + Nencep(Módulo1.arranques(arranque1, 9), Módulo1.arranques(arranque1, 10), CInt(Módulo1.arranques(arranque1, 11))), CInt(Módulo1.arranques(arranque1, 11)), CSng(Módulo1.arranques(arranque1, 12))) + asiento_grupo(CSng(Módulo1.arranques(arranque1, 3)) + Módulo1.disposiciones(Módulo1.arranques(arranque1, 9), 0) * Nencep(Módulo1.arranques(arranque1, 9), Módulo1.arranques(arranque1, 10), CInt(Módulo1.arranques(arranque1, 11))), CInt(Módulo1.arranques(arranque1, 9)), CSng(Módulo1.arranques(arranque1, 10)), CInt(Módulo1.arranques(arranque1, 11)), CSng(Módulo1.arranques(arranque1, 12)))), 1)
            Módulo1.distorsiones_angulares(P, 2) = arranque2
            Módulo1.distorsiones_angulares(P, 3) = Round(1000 * (asiento_individual_lp(CSng(Módulo1.arranques(arranque2, 3)) / Módu-
```

```
lo1.disposiciones(Módulo1.arranques(arranque2, 9), 0) + Nencep(Módulo1.arranques(arranque2,
9),     Módulo1.arranques(arranque2,     10),     CInt(Módulo1.arranques(arranque2,     11))),
CInt(Módulo1.arranques(arranque2, 11)), CSng(Módulo1.arranques(arranque2, 12))) + asien-
to_grupo(CSng(Módulo1.arranques(arranque2,              3))              +              Módu-
lo1.disposiciones(Módulo1.arranques(arranque2, 9), 0) * Nencep(Módulo1.arranques(arranque2,
9),     Módulo1.arranques(arranque2,     10),     CInt(Módulo1.arranques(arranque2,     11))),
CInt(Módulo1.arranques(arranque2,      9)),      CSng(Módulo1.arranques(arranque2,      10)),
CInt(Módulo1.arranques(arranque2, 11)), CSng(Módulo1.arranques(arranque2, 12)))), 1)
                Módulo1.distorsiones_angulares(P,                   4)                  =
Round(distancia_arranques(arranque1, arranque2), 2)
                Módulo1.distorsiones_angulares(P,                   5)                  =
Round(Abs(Módulo1.arranques_optimizacion(arranque1,      5,      pilote)      -      Módu-
lo1.arranques_optimizacion(arranque2, 5, pilote)) / distancia_arranques(arranque1, arran-
que2), 6)
                Módulo1.distorsiones_angulares(P,       6)       =       Round(1       /
(Abs(Módulo1.arranques_optimizacion(arranque1,       5,       pilote)       -       Módu-
lo1.arranques_optimizacion(arranque2, 5, pilote)) / distancia_arranques(arranque1, arran-
que2)), 0)

                If      CSng(Módulo1.distorsiones_angulares(P,      5))      >      Módu-
lo1.distorsion_angular Then
                    Módulo1.distorsiones_angulares(P, 7) = "NO CUMPLE"
                Else
                    Módulo1.distorsiones_angulares(P, 7) = "CUMPLE"
                End If

                P = P + 1
            Next
        Nex
        'Aquí se selecciona la distorsión pésima
        dist_max = Módulo1.distorsion_angular
        For i = 0 To 1000
            If Módulo1.distorsiones_angulares(i, 0) = "" Then Exit For
            If Módulo1.distorsiones_angulares(i, 5) > dist_max Then
                dist_max = Módulo1.distorsiones_angulares(i, 5)
                arranque1 = Módulo1.distorsiones_angulares(i, 0)
                arranque2 = Módulo1.distorsiones_angulares(i, 2)
            End If
        Next



        If dist_max = Módulo1.distorsion_angular Then Exit Sub
        'En caso de no  haber ningún problema de distorsión la planta propuesta será la
óptima
    End Sub
```

5.3. PLANTA- DOS PILOTES

```
Public Sub optimizacion_planta_2_diametros()

'Primero se crea la (hiper)matriz con las optimizaciones de todos los arranques para todos los diámetros

ReDim Módulo1.arranques_optimizacion(50, 10, 100)

For pil = 1 To Módulo1.npilotes
    For arranque = 1 To Módulo1.n_arranques

        'Una vez seleccionado el arranque se cambian los estratos
        Call cambiar_estratos("arranque", arranque)

        coste_arranque = 10 ^ 6
        For encep = 1 To Módulo1.n_encepados

            For k = Módulo1.separacion_ejes_minima To Módulo1.separacion_ejes_maxima Step Módulo1.separacion_ejes_incremento



                L = dimensionamientoL(CSng(Módulo1.arranques(arranque, 3)), CSng(Módulo1.arranques(arranque, 4)), CSng(Módulo1.arranques(arranque, 5)), CInt(encep), CSng(k), CInt(pil))

                If L = -1 Or L > hfinal(Módulo1.n_estratos) Then
                    Módulo1.arranques_optimizacion(arranque, 4, pil) = 9999
                Else
                    If coste_ejecucion(CInt(encep), CInt(pil), CSng(k), CSng(L)) < coste_arranque Then
                        coste_arranque = coste_ejecucion(CInt(encep), CInt(pil), CSng(k), CSng(L))
                        Módulo1.arranques_optimizacion(arranque, 0, pil) = pil
                        Módulo1.arranques_optimizacion(arranque, 1, pil) = k
                        Módulo1.arranques_optimizacion(arranque, 2, pil) = encep
                        Módulo1.arranques_optimizacion(arranque, 3, pil) = L
                        Módulo1.arranques_optimizacion(arranque, 4, pil) = coste_arranque
                        Módulo1.arranques_optimizacion(arranque, 5, pil) = asiento_individual_lp(CSng(Módulo1.arranques(arranque, 3)), CInt(pil), CSng(L))
                        End If
                    End If
                Next
            Next
```

```
        Next
    Next

    'A partir de ella se pueden evaluar las combinaciones y elegir la más económica

    coste_planta_minimo = 10 ^ 9
    For pilote1 = 1 To Módulo1.npilotes - 1
        For pilote2 = pilote1 + 1 To Módulo1.npilotes
            coste_planta = 0
            For arranque = 1 To Módulo1.n_arranques
                If Módulo1.arranques_optimizacion(arranque, 4, pilote1) > Módu-
lo1.arranques_optimizacion(arranque, 4, pilote2) Then
                    coste_planta = coste_planta + Módu-
lo1.arranques_optimizacion(arranque, 4, pilote2)
                Else
                    coste_planta = coste_planta + Módu-
lo1.arranques_optimizacion(arranque, 4, pilote1)
                End If
            Next

            If coste_planta < coste_planta_minimo Then
                coste_planta_minimo = coste_planta
                pilotedefinitivo1 = pilote1
                pilotedefinitivo2 = pilote2
            End If
        Next
    Next

    'Después solo quedaría trasladar el resultado a a la tabla

    For arranque = 1 To Módulo1.n_arranques
        If Módulo1.arranques_optimizacion(arranque, 4, pilotedefinitivo1) > Módu-
lo1.arranques_optimizacion(arranque, 4, pilotedefinitivo2) Then
            Módulo1.arranques(arranque, 9) = Módulo1.arranques_optimizacion(arranque, 2,
pilotedefinitivo2)
            Módulo1.arranques(arranque, 10) = Módulo1.arranques_optimizacion(arranque, 1,
pilotedefinitivo2)
            Módulo1.arranques(arranque, 11) = pilotedefinitivo2
            Módulo1.arranques(arranque, 12) = Módulo1.arranques_optimizacion(arranque, 3,
pilotedefinitivo2)
        Else
            Módulo1.arranques(arranque, 9) = Módulo1.arranques_optimizacion(arranque, 2,
pilotedefinitivo1)
            Módulo1.arranques(arranque, 10) = Módulo1.arranques_optimizacion(arranque, 1,
pilotedefinitivo1)
            Módulo1.arranques(arranque, 11) = pilotedefinitivo1
```

```
            Módulo1.arranques(arranque, 12) = Módulo1.arranques_optimizacion(arranque, 3,
pilotedefinitivo1)
        End If
    Next

    'Ahora se procedería a clacular las distorsiones angulares
        ReDim Módulo1.distorsiones_angulares(1000, 10)
        Módulo1.distorsiones_angulares(0, 0) = "Arranque"
        Módulo1.distorsiones_angulares(0, 1) = "Asiento (mm)"
        Módulo1.distorsiones_angulares(0, 2) = "Arranque"
        Módulo1.distorsiones_angulares(0, 3) = "Asiento(mm)"
        Módulo1.distorsiones_angulares(0, 4) = "Distancia (m)"
        Módulo1.distorsiones_angulares(0, 5) = "Distorsión"
        Módulo1.distorsiones_angulares(0, 6) = "1/"
        Módulo1.distorsiones_angulares(0, 7) = "Estado"

        P = 1
        For arranque1 = 1 To Módulo1.n_arranques - 1
            For arranque2 = arranque1 + 1 To Módulo1.n_arranques
                Módulo1.distorsiones_angulares(P, 0) = arranque1
                Módulo1.distorsiones_angulares(P, 1) = Round(1000 * (asien-
to_individual_lp(CSng(Módulo1.arranques(arranque1, 3)) / Módu-
lo1.disposiciones(Módulo1.arranques(arranque1, 9), 0) + Nencep(Módulo1.arranques(arranque1,
9), Módulo1.arranques(arranque1, 10), CInt(Módulo1.arranques(arranque1, 11))),
CInt(Módulo1.arranques(arranque1, 11)), CSng(Módulo1.arranques(arranque1, 12))) + asien-
to_grupo(CSng(Módulo1.arranques(arranque1, 3)) + Módu-
lo1.disposiciones(Módulo1.arranques(arranque1, 9), 0) * Nencep(Módulo1.arranques(arranque1,
9), Módulo1.arranques(arranque1, 10), CInt(Módulo1.arranques(arranque1, 11))),
CInt(Módulo1.arranques(arranque1, 9)), CSng(Módulo1.arranques(arranque1, 10)),
CInt(Módulo1.arranques(arranque1, 11)), CSng(Módulo1.arranques(arranque1, 12)))), 1)
                Módulo1.distorsiones_angulares(P, 2) = arranque2
                Módulo1.distorsiones_angulares(P, 3) = Round(1000 * (asien-
to_individual_lp(CSng(Módulo1.arranques(arranque2, 3)) / Módu-
lo1.disposiciones(Módulo1.arranques(arranque2, 9), 0) + Nencep(Módulo1.arranques(arranque2,
9), Módulo1.arranques(arranque2, 10), CInt(Módulo1.arranques(arranque2, 11))),
CInt(Módulo1.arranques(arranque2, 11)), CSng(Módulo1.arranques(arranque2, 12))) + asien-
to_grupo(CSng(Módulo1.arranques(arranque2, 3)) + Módu-
lo1.disposiciones(Módulo1.arranques(arranque2, 9), 0) * Nencep(Módulo1.arranques(arranque2,
9), Módulo1.arranques(arranque2, 10), CInt(Módulo1.arranques(arranque2, 11))),
CInt(Módulo1.arranques(arranque2, 9)), CSng(Módulo1.arranques(arranque2, 10)),
CInt(Módulo1.arranques(arranque2, 11)), CSng(Módulo1.arranques(arranque2, 12)))), 1)
                Módulo1.distorsiones_angulares(P, 4) =
Round(distancia_arranques(arranque1, arranque2), 2)
                Módulo1.distorsiones_angulares(P, 5) = Round(0.001 *
Abs(Módulo1.distorsiones_angulares(P, 1) - Módulo1.distorsiones_angulares(P, 3)) / distan-
cia_arranques(arranque1, arranque2), 6)
```

```
                Módulo1.distorsiones_angulares(P,       6)       =       Round(1000       /
(Abs(Módulo1.distorsiones_angulares(P, 1) - Módulo1.distorsiones_angulares(P, 3)) / distan-
cia_arranques(arranque1, arranque2)), 0)

                If      CSng(Módulo1.distorsiones_angulares(P,      5))      >      Módu-
lo1.distorsion_angular Then
                    Módulo1.distorsiones_angulares(P, 7) = "NO CUMPLE"
                Else
                    Módulo1.distorsiones_angulares(P, 7) = "CUMPLE"
                End If

                P = P + 1
            Next
        Next



        'Aquí se selecciona la distorsión pésima
        dist_max = Módulo1.distorsion_angular
        For i = 0 To 1000
            If Módulo1.distorsiones_angulares(i, 0) = "" Then Exit For
            If Módulo1.distorsiones_angulares(i, 5) > dist_max Then
                dist_max = Módulo1.distorsiones_angulares(i, 5)
                arranque1 = Módulo1.distorsiones_angulares(i, 0)
                arranque2 = Módulo1.distorsiones_angulares(i, 2)
            End If
        Next



        If dist_max = Módulo1.distorsion_angular Then Exit Sub
        'En caso de no  haber ningún problema de distorsión la planta propuesta será la
óptima


    End Sub
```

5.4. CORRECCION DISTORSIONES

5.4.1. Corrección- un pilote

```
Public Sub correccion_distorsiones_1_pilote()
'Será igual que la otra pero eliminando las variaciones de los pilotes


'Aquí se selecciona la distorsión pésima y cuenta el número totalde distorsiones que
no cumplen
          dist_max = Módulo1.distorsion_angular
          numero_distorsiones_no_cumplen = 0
          For i = 1 To 1000
              If Módulo1.distorsiones_angulares(i, 0) = "" Then Exit For
                  If Módulo1.distorsiones_angulares(i, 5) > Módulo1.distorsion_angular Then
                      numero_distorsiones_no_cumplen = numero_distorsiones_no_cumplen + 1
                      If Módulo1.distorsiones_angulares(i, 5) > dist_max Then
                          dist_max = Módulo1.distorsiones_angulares(i, 5)
                          arranque1 = Módulo1.distorsiones_angulares(i, 0)
                          arranque2 = Módulo1.distorsiones_angulares(i, 2)
                      End If
                  End If
          Next



          If dist_max = Módulo1.distorsion_angular Then Exit Sub
          'En caso de no  haber ningún problema de distorsión la planta propuesta será la
óptima

          'aquí se toman los pilotes empleados
          pil1 = Módulo1.arranques(1, 11)
          pil2 = Módulo1.arranques(1, 11)

      Do While dist_max < Módulo1.distorsion_angular

          'ahora se calculan los asientos originales de cada arranque
          asientooriginal1  =  asiento_individual_lp(CSng(Módulo1.arranques(arranque1,  3)),
CInt(Módulo1.arranques(arranque1,  11)),  CSng(Módulo1.arranques(arranque1,  12)))  +  asien-
to_grupo(CSng(Módulo1.arranques(arranque1,   1)),   CInt(Módulo1.arranques(arranque1,   9)),
CSng(Módulo1.arranques(arranque1,     10)),     CInt(Módulo1.arranques(arranque1,     11)),
CSng(Módulo1.arranques(arranque1, 12)))
```

```
        asientooriginal2 = asiento_individual_lp(CSng(Módulo1.arranques(arranque2, 3)),
CInt(Módulo1.arranques(arranque2, 11)), CSng(Módulo1.arranques(arranque2, 12))) + asien-
to_grupo(CSng(Módulo1.arranques(arranque2, 1)), CInt(Módulo1.arranques(arranque2, 9)),
CSng(Módulo1.arranques(arranque2, 10)), CInt(Módulo1.arranques(arranque2, 11)),
CSng(Módulo1.arranques(arranque2, 12)))


        'y los costes
        costeoriginal1 = coste_ejecucion(CInt(Módulo1.arranques(arranque1, 9)),
CInt(Módulo1.arranques(arranque1, 11)), CSng(Módulo1.arranques(arranque1, 10)),
CSng(Módulo1.arranques(arranque1, 12)))
        costeoriginal2 = coste_ejecucion(CInt(Módulo1.arranques(arranque2, 9)),
CInt(Módulo1.arranques(arranque2, 11)), CSng(Módulo1.arranques(arranque2, 10)),
CSng(Módulo1.arranques(arranque2, 12)))


        'Y las distorsiones excesivas en que está envuelto cada uno
        distorsiones_arranque_1 = 0
        distorsiones_arranque_2 = 0
         For i = 0 To 1000
            If Módulo1.distorsiones_angulares(i, 0) = "" Then Exit For
                If Módulo1.distorsiones_angulares(i, 5) > Módulo1.distorsion_angular Then
                    If Módulo1.distorsiones_angulares(i, 0) = arranque1 Or Módu-
lo1.distorsiones_angulares(i, 2) = arranque1 Then distorsiones_arranque_1 = distor-
siones_arranque_1 + 1
                    If Módulo1.distorsiones_angulares(i, 0) = arranque2 Or Módu-
lo1.distorsiones_angulares(i, 2) = arranque2 Then distorsiones_arranque_2 = distor-
siones_arranque_2 + 1
                End If
        Next



        'Hijo 1: modificación del arranque 1
        For pilote1 = pil1 To pil2 Step (pil2 - pil1)
            For k1 = Módulo1.separacion_ejes_minima To Módulo1.separacion_ejes_maxima
Step Módulo1.separacion_ejes_incremento
                For encepado1 = 1 To Módulo1.n_encepados
                    For incrementoL1 = 0 To 2 Step 0.5
                        numero_distorsiones_angulares_cambio = 0

                        L1 = dimensionamientoL(CSng(Módulo1.arranques(arranque1, 3)),
CSng(Módulo1.arranques(arranque1, 4)), CSng(Módulo1.arranques(arranque1, 5)),
CInt(encepado1), CSng(k1), CInt(pilote1))
```

```
                                asiento1                        =                       asien-
to_individual_lp(CSng(Módulo1.arranques(arranque1, 3)), CInt(pilote1), CSng(L1 + incremen-
toL1)) + asiento_grupo(CSng(Módulo1.arranques(arranque1, 3)), CInt(encepado1), CSng(k1),
CInt(pilote1), CSng(L1 + incrementoL1))

                            If     Abs((asiento1     -     asientooriginal2)     /     distan-
cia_arranques(arranque1, arranque2)) < Módulo1.distorsion_angular Then
                                    'comprobación de que se anula la distorsión angular
                                For i = 0 To 1000
                                    If  Módulo1.distorsiones_angulares(i,  0)  =  ""  Then  Exit
For
                                            If  Módulo1.distorsiones_angulares(i,  0)  =  ar-
ranque1 Then
                                                If        Abs((asiento1        -        Módu-
lo1.distorsiones_angulares(i, 3) / 1000) / Módulo1.distorsiones_angulares(i, 4)) > Módu-
lo1.distorsion_angular Then
                                                    numero_distorsiones_angulares_cambio     =
numero_distorsiones_angulares_cambio + 1
                                                End If
                                            ElseIf  Módulo1.distorsiones_angulares(i,  2)  =
arranque1 Then
                                                If        Abs((asiento1        -        Módu-
lo1.distorsiones_angulares(i, 1) / 1000) / Módulo1.distorsiones_angulares(i, 4)) > Módu-
lo1.distorsion_angular Then
                                                    numero_distorsiones_angulares_cambio     =
numero_distorsiones_angulares_cambio + 1
                                                End If
                                            End If
                                Next

                                If     numero_distorsiones_angulares_cambio_1     <     distor-
siones_arranque_1 Then
                                   distorsiones_arranque_1              =              nume-
ro_distorsiones_angulares_cambio_1

                                    nuevopilote1 = pilote1
                                    nuevok1 = k1
                                    nuevoencepado1 = encepado1
                                    nuevoL1 = L1 + incrementoL1
                                    nuevocoste1       =       coste_ejecucion(CInt(encepado1),
CInt(pilote1), CSng(k1), CSng(nuevoL1))


                                    'comprobación de que reduce el número de distorsiones an-
gulares(el máximo posible)
                                    'aquí se sustituiría directamente
```

```
                            ElseIf  numero_distorsiones_angulares_cambio_1  =  distorsio-
nes_arranque_1 Then
                                If    coste_ejecucion(CInt(encepado1),    CInt(pilote1),
CSng(k1), CSng(nuevoL1)) < nuevocoste1 Then
                                    nuevopilote1 = pilote1
                                    nuevok1 = k1
                                    nuevoencepado1 = encepado1
                                    nuevoL1 = L1 + incrementoL1
                                    nuevocoste1    =    coste_ejecucion(CInt(encepado1),
CInt(pilote1), CSng(k1), CSng(nuevoL1))
                                End If
                            End If

                        'si reduce respecto al anterior entra directamente aunque sea más
caro
                        'comprobación de coste mínimo
                        End If
                    Next
                Next
            Next
        Next



        'Hijo 2:modificación del arranque 2
        For pilote2 = pil1 To pil2 Step (pil2 - pil1)
            For  k2  =  Módulo1.separacion_ejes_minima  To  Módulo1.separacion_ejes_maxima
Step Módulo1.separacion_ejes_incremento
                For encepado2 = 1 To Módulo1.n_encepados
                    For incrementoL2 = 0 To 2 Step 0.5
                        numero_distorsiones_angulares_cambio = 0

                        L2  =  dimensionamientoL(CSng(Módulo1.arranques(arranque2,  3)),
CSng(Módulo1.arranques(arranque2,     4)),     CSng(Módulo1.arranques(arranque2,     5)),
CInt(encepado2), CSng(k2), CInt(pilote2))

                            asiento2                       =                       asien-
to_individual_lp(CSng(Módulo1.arranques(arranque2, 3)), CInt(pilote2), CSng(L2 + incremen-
toL2))  +  asiento_grupo(CSng(Módulo1.arranques(arranque2,  3)),  CInt(encepado2),  CSng(k2),
CInt(pilote2), CSng(L2 + incrementoL2))

                        If    Abs((asiento2    -    asientooriginal1)    /    distan-
cia_arranques(arranque1, arranque2)) < Módulo1.distorsion_angular Then
                            'comprobación de que se anula la distorsión angular
                            For i = 0 To 1000
```

```
                                If Módulo1.distorsiones_angulares(i, 0) = "" Then Exit
For
                                    If Módulo1.distorsiones_angulares(i, 0) = ar-
ranque2 Then
                                        If Abs((asiento2 - Módu-
lo1.distorsiones_angulares(i, 3) / 1000) / Módulo1.distorsiones_angulares(i, 4)) > Módu-
lo1.distorsion_angular Then
                                            numero_distorsiones_angulares_cambio =
numero_distorsiones_angulares_cambio + 1
                                        End If
                                    ElseIf Módulo1.distorsiones_angulares(i, 2) =
arranque2 Then
                                        If Abs((asiento2 - Módu-
lo1.distorsiones_angulares(i, 1) / 1000) / Módulo1.distorsiones_angulares(i, 4)) > Módu-
lo1.distorsion_angular Then
                                            numero_distorsiones_angulares_cambio =
numero_distorsiones_angulares_cambio + 1
                                        End If
                                    End If
                            Next

                            If numero_distorsiones_angulares_cambio_2 < distor-
siones_arranque_2 Then
                               distorsiones_arranque_2 = nume-
ro_distorsiones_angulares_cambio_2

                                nuevopilote2 = pilote2
                                nuevok2 = k2
                                nuevoencepado2 = encepado2
                                nuevoL2 = L2 + incrementoL2
                                nuevocoste2 = coste_ejecucion(CInt(encepado2),
CInt(pilote2), CSng(k2), CSng(nuevoL2))


                                'comprobación de que reduce el número de distorsiones an-
gulares(el máximo posible)
                                'aquí se sustituiría directamente
                            ElseIf numero_distorsiones_angulares_cambio_2 = distorsio-
nes_arranque_2 Then
                                If coste_ejecucion(CInt(encepado2), CInt(pilote2),
CSng(k2), CSng(nuevoL2)) < nuevocoste2 Then
                                    nuevopilote2 = pilote2
                                    nuevok2 = k2
                                    nuevoencepado2 = encepado2
                                    nuevoL2 = L2 + incrementoL2
```

```
                                    nuevocoste2     =     coste_ejecucion(CInt(encepado2),
CInt(pilote2), CSng(k2), CSng(nuevoL2))
                                End If
                            End If

                        'si reduce respecto al anterior entra directamente aunque sea más
caro
                        'comprobación de coste mínimo
                        End If
                    Next
                Next
            Next
        Next


        'Aquí se decide cual de los dos es la mejor opción, según los criterios
        '1)La opción que reduzca más el número de distorsiones angulares será mejor
        '2)En caso de igual reducción, el de menor coste

        If  numero_distorsiones_angulares_cambio1  <  numero_distorsiones_angulares_cambio2
Then

            Módulo1.arranques(arranque1, 9) = nuevoencepado1
            Módulo1.arranques(arranque1, 10) = nuevok1
            Módulo1.arranques(arranque1, 11) = nuevopilote1
            Módulo1.arranques(arranque1, 12) = nuevoL1

        ElseIf           numero_distorsiones_angulares_cambio1           >           nume-
ro_distorsiones_angulares_cambio2 Then
            Módulo1.arranques(arranque2, 9) = nuevoencepado2
            Módulo1.arranques(arranque2, 10) = nuevok2
            Módulo1.arranques(arranque2, 11) = nuevopilote2
            Módulo1.arranques(arranque2, 12) = nuevoL2
        Else

            If nuevocoste1 + costeoriginal2 < nuevocoste2 + costeoriginal1 Then

                Módulo1.arranques(arranque1, 9) = nuevoencepado1
                Módulo1.arranques(arranque1, 10) = nuevok1
                Módulo1.arranques(arranque1, 11) = nuevopilote1
                Módulo1.arranques(arranque1, 12) = nuevoL1

            ElseIf nuevocoste1 + costeoriginal2 > nuevocoste2 + costeoriginal1 Then
                Módulo1.arranques(arranque2, 9) = nuevoencepado2
                Módulo1.arranques(arranque2, 10) = nuevok2
                Módulo1.arranques(arranque2, 11) = nuevopilote2
```

```
                Módulo1.arranques(arranque2, 12) = nuevoL2
            End If
        End If

        'Ahora hay que recalcular las que no cumplen

        ReDim Módulo1.distorsiones_angulares(1000, 10)
        Módulo1.distorsiones_angulares(0, 0) = "Arranque"
        Módulo1.distorsiones_angulares(0, 1) = "Asiento (mm)"
        Módulo1.distorsiones_angulares(0, 2) = "Arranque"
        Módulo1.distorsiones_angulares(0, 3) = "Asiento(mm)"
        Módulo1.distorsiones_angulares(0, 4) = "Distancia (m)"
        Módulo1.distorsiones_angulares(0, 5) = "Distorsión"
        Módulo1.distorsiones_angulares(0, 6) = "1/"
        Módulo1.distorsiones_angulares(0, 7) = "Estado"

        P = 1
        For arranque1 = 1 To Módulo1.n_arranques - 1
            For arranque2 = arranque1 + 1 To Módulo1.n_arranques
            'Los datos a emplear son de la matriz arranques (¡A SECAS¡
            'Si no la solución no tendría que ser válida
                Módulo1.distorsiones_angulares(P, 0) = arranque1
                Módulo1.distorsiones_angulares(P, 1) = Round(1000 * (asien-
to_individual_lp(CSng(Módulo1.arranques(arranque1, 3)) / Módu-
lo1.disposiciones(Módulo1.arranques(arranque1, 9), 0) + Nencep(Módulo1.arranques(arranque1,
9), Módulo1.arranques(arranque1, 10), CInt(Módulo1.arranques(arranque1, 11))),
CInt(Módulo1.arranques(arranque1, 11)), CSng(Módulo1.arranques(arranque1, 12))) + asien-
to_grupo(CSng(Módulo1.arranques(arranque1, 3)) + Módu-
lo1.disposiciones(Módulo1.arranques(arranque1, 9), 0) * Nencep(Módulo1.arranques(arranque1,
9), Módulo1.arranques(arranque1, 10), CInt(Módulo1.arranques(arranque1, 11))),
CInt(Módulo1.arranques(arranque1, 9)), CSng(Módulo1.arranques(arranque1, 10)),
CInt(Módulo1.arranques(arranque1, 11)), CSng(Módulo1.arranques(arranque1, 12)))), 1)
                Módulo1.distorsiones_angulares(P, 2) = arranque2
                Módulo1.distorsiones_angulares(P, 3) = Round(1000 * (asien-
to_individual_lp(CSng(Módulo1.arranques(arranque2, 3)) / Módu-
lo1.disposiciones(Módulo1.arranques(arranque2, 9), 0) + Nencep(Módulo1.arranques(arranque2,
9), Módulo1.arranques(arranque2, 10), CInt(Módulo1.arranques(arranque2, 11))),
CInt(Módulo1.arranques(arranque2, 11)), CSng(Módulo1.arranques(arranque2, 12))) + asien-
to_grupo(CSng(Módulo1.arranques(arranque2, 3)) + Módu-
lo1.disposiciones(Módulo1.arranques(arranque2, 9), 0) * Nencep(Módulo1.arranques(arranque2,
9), Módulo1.arranques(arranque2, 10), CInt(Módulo1.arranques(arranque2, 11))),
CInt(Módulo1.arranques(arranque2, 9)), CSng(Módulo1.arranques(arranque2, 10)),
CInt(Módulo1.arranques(arranque2, 11)), CSng(Módulo1.arranques(arranque2, 12)))), 1)
                Módulo1.distorsiones_angulares(P, 4) =
Round(distancia_arranques(arranque1, arranque2), 2)
```

```
                Módulo1.distorsiones_angulares(P,                5)                =
Round(Abs(Módulo1.arranques_optimizacion(arranque1,      5,      pilote)      -      Módu-
lo1.arranques_optimizacion(arranque2, 5, pilote)) / distancia_arranques(arranque1, arran-
que2), 6)
                Módulo1.distorsiones_angulares(P,       6)       =       Round(1       /
(Abs(Módulo1.arranques_optimizacion(arranque1,       5,       pilote)       -       Módu-
lo1.arranques_optimizacion(arranque2, 5, pilote)) / distancia_arranques(arranque1, arran-
que2)), 0)

                If      CSng(Módulo1.distorsiones_angulares(P,      5))      >      Módu-
lo1.distorsion_angular Then
                    Módulo1.distorsiones_angulares(P, 7) = "NO CUMPLE"
                Else
                    Módulo1.distorsiones_angulares(P, 7) = "CUMPLE"
                End If

                P = P + 1
            Next
        Next

        dist_max = Módulo1.distorsion_angular
        numero_distorsiones_no_cumplen = 0
        For i = 0 To 1000
            If Módulo1.distorsiones_angulares(i, 0) = "" Then Exit For
                If Módulo1.distorsiones_angulares(i, 5) > Módulo1.distorsion_angular Then
                    numero_distorsiones_no_cumplen = numero_distorsiones_no_cumplen + 1
                    If Módulo1.distorsiones_angulares(i, 5) > dist_max Then
                        dist_max = Módulo1.distorsiones_angulares(i, 5)
                        arranque1 = Módulo1.distorsiones_angulares(i, 0)
                        arranque2 = Módulo1.distorsiones_angulares(i, 2)
                    End If
                End If
        Next

    Loop


    End Sub
```

5.4.2. Corrección- dos pilotes

```
Public Sub correccion_distorsiones_2_pilotes()

'Aquí se selecciona la distorsión pésima y cuenta el número totalde distorsiones que no cumplen
          dist_max = Módulo1.distorsion_angular
          numero_distorsiones_no_cumplen = 0
          For i = 1 To 1000
              If Módulo1.distorsiones_angulares(i, 0) = "" Then Exit For
                  If Módulo1.distorsiones_angulares(i, 5) > Módulo1.distorsion_angular Then
                      numero_distorsiones_no_cumplen = numero_distorsiones_no_cumplen + 1
                      If Módulo1.distorsiones_angulares(i, 5) > dist_max Then
                          dist_max = Módulo1.distorsiones_angulares(i, 5)
                          arranque1 = Módulo1.distorsiones_angulares(i, 0)
                          arranque2 = Módulo1.distorsiones_angulares(i, 2)
                      End If
                  End If
          Next


          If dist_max = Módulo1.distorsion_angular Then Exit Sub
          'En caso de no  haber ningún problema de distorsión la planta propuesta será la óptima

          'aquí se toman los pilotes empleados
          pil1 = Módulo1.arranques(1, 11)
          For i = 1 To Módulo1.n_arranques
              If Módulo1.arranques(i, 11) <> pil1 Then
                  pil2 = Módulo1.arranques(i, 11)
                  Exit For
              End If
          Next


      Do While dist_max < Módulo1.distorsion_angular

          'ahora se calculan los asientos originales de cada arranque
          asientooriginal1 = asiento_individual_lp(CSng(Módulo1.arranques(arranque1, 3)), CInt(Módulo1.arranques(arranque1, 11)), CSng(Módulo1.arranques(arranque1, 12))) + asiento_grupo(CSng(Módulo1.arranques(arranque1, 3)), CInt(Módulo1.arranques(arranque1, 9)), CSng(Módulo1.arranques(arranque1, 10)), CInt(Módulo1.arranques(arranque1, 11)), CSng(Módulo1.arranques(arranque1, 12)))
```

```
        asientooriginal2 = asiento_individual_lp(CSng(Módulo1.arranques(arranque2, 3)),
CInt(Módulo1.arranques(arranque2, 11)), CSng(Módulo1.arranques(arranque2, 12))) + asien-
to_grupo(CSng(Módulo1.arranques(arranque2, 3)), CInt(Módulo1.arranques(arranque2, 9)),
CSng(Módulo1.arranques(arranque2, 10)), CInt(Módulo1.arranques(arranque2, 11)),
CSng(Módulo1.arranques(arranque2, 12)))


        'y los costes
        costeoriginal1 = coste_ejecucion(CInt(Módulo1.arranques(arranque1, 9)),
CInt(Módulo1.arranques(arranque1, 11)), CSng(Módulo1.arranques(arranque1, 10)),
CSng(Módulo1.arranques(arranque1, 12)))
        costeoriginal2 = coste_ejecucion(CInt(Módulo1.arranques(arranque2, 9)),
CInt(Módulo1.arranques(arranque2, 11)), CSng(Módulo1.arranques(arranque2, 10)),
CSng(Módulo1.arranques(arranque2, 12)))


        'Y las distorsiones excesivas en que está envuelto cada uno
        distorsiones_arranque_1 = 0
        distorsiones_arranque_2 = 0
         For i = 0 To 1000
            If Módulo1.distorsiones_angulares(i, 0) = "" Then Exit For
                If Módulo1.distorsiones_angulares(i, 5) > Módulo1.distorsion_angular Then
                    If Módulo1.distorsiones_angulares(i, 0) = arranque1 Or Módu-
lo1.distorsiones_angulares(i, 2) = arranque1 Then distorsiones_arranque_1 = distor-
siones_arranque_1 + 1
                    If Módulo1.distorsiones_angulares(i, 0) = arranque2 Or Módu-
lo1.distorsiones_angulares(i, 2) = arranque2 Then distorsiones_arranque_2 = distor-
siones_arranque_2 + 1
                End If
        Next



        'Hijo 1: modificación del arranque 1
        For pilote1 = pil1 To pil2 Step (pil2 - pil1)
            For k1 = Módulo1.separacion_ejes_minima To Módulo1.separacion_ejes_maxima
Step Módulo1.separacion_ejes_incremento
                For encepado1 = 1 To Módulo1.n_encepados
                    For incrementoL1 = 0 To 2 Step 0.5
                        numero_distorsiones_angulares_cambio = 0

                        L1 = dimensionamientoL(CSng(Módulo1.arranques(arranque1, 3)),
CSng(Módulo1.arranques(arranque1, 4)), CSng(Módulo1.arranques(arranque1, 5)),
CInt(encepado1), CSng(k1), CInt(pilote1))
```

```
                                asiento1                        =                       asien-
to_individual_lp(CSng(Módulo1.arranques(arranque1, 3)), CInt(pilote1), CSng(L1 + incremen-
toL1)) + asiento_grupo(CSng(Módulo1.arranques(arranque1, 3)), CInt(encepado1), CSng(k1),
CInt(pilote1), CSng(L1 + incrementoL1))

                            If     Abs((asiento1     -     asientooriginal2)     /     distan-
cia_arranques(arranque1, arranque2)) < Módulo1.distorsion_angular Then
                                    'comprobación de que se anula la distorsión angular
                                For i = 0 To 1000
                                    If  Módulo1.distorsiones_angulares(i,  0)  =  ""  Then  Exit
For
                                            If  Módulo1.distorsiones_angulares(i,  0)  =  ar-
ranque1 Then
                                                If        Abs((asiento1        -        Módu-
lo1.distorsiones_angulares(i, 3) / 1000) / Módulo1.distorsiones_angulares(i, 4)) > Módu-
lo1.distorsion_angular Then
                                                    numero_distorsiones_angulares_cambio     =
numero_distorsiones_angulares_cambio + 1
                                                End If
                                            ElseIf Módulo1.distorsiones_angulares(i, 2) = ar-
ranque1 Then
                                                If        Abs((asiento1        -        Módu-
lo1.distorsiones_angulares(i, 1) / 1000) / Módulo1.distorsiones_angulares(i, 4)) > Módu-
lo1.distorsion_angular Then
                                                    numero_distorsiones_angulares_cambio     =
numero_distorsiones_angulares_cambio + 1
                                                End If
                                            End If
                                Next

                                If     numero_distorsiones_angulares_cambio_1     <     distor-
siones_arranque_1 Then
                                   distorsiones_arranque_1               =               nume-
ro_distorsiones_angulares_cambio_1

                                    nuevopilote1 = pilote1
                                    nuevok1 = k1
                                    nuevoencepado1 = encepado1
                                    nuevoL1 = L1 + incrementoL1
                                    nuevocoste1       =       coste_ejecucion(CInt(encepado1),
CInt(pilote1), CSng(k1), CSng(nuevoL1))


                                    'comprobación de que reduce el número de distorsiones an-
gulares(el máximo posible)
                                    'aquí se sustituiría directamente
```

```
                            ElseIf  numero_distorsiones_angulares_cambio_1  =  distorsio-
nes_arranque_1 Then
                                If    coste_ejecucion(CInt(encepado1),    CInt(pilote1),
CSng(k1), CSng(nuevoL1)) < nuevocoste1 Then
                                    nuevopilote1 = pilote1
                                    nuevok1 = k1
                                    nuevoencepado1 = encepado1
                                    nuevoL1 = L1 + incrementoL1
                                    nuevocoste1    =    coste_ejecucion(CInt(encepado1),
CInt(pilote1), CSng(k1), CSng(nuevoL1))
                                End If
                            End If

                        'si reduce respecto al anterior entra directamente aunque sea más
caro
                        'comprobación de coste mínimo
                        End If
                    Next
                Next
            Next
        Next



        'Hijo 2:modificación del arranque 2
        For pilote2 = pil1 To pil2 Step (pil2 - pil1)
            For  k2  =  Módulo1.separacion_ejes_minima  To  Módulo1.separacion_ejes_maxima
Step Módulo1.separacion_ejes_incremento
                For encepado2 = 1 To Módulo1.n_encepados
                    For incrementoL2 = 0 To 2 Step 0.5
                        numero_distorsiones_angulares_cambio = 0

                        L2  =  dimensionamientoL(CSng(Módulo1.arranques(arranque2,  3)),
CSng(Módulo1.arranques(arranque2,     4)),     CSng(Módulo1.arranques(arranque2,     5)),
CInt(encepado2), CSng(k2), CInt(pilote2))

                            asiento2                       =                       asien-
to_individual_lp(CSng(Módulo1.arranques(arranque2, 3)), CInt(pilote2), CSng(L2 + incremen-
toL2))  +  asiento_grupo(CSng(Módulo1.arranques(arranque2,  3)),  CInt(encepado2),  CSng(k2),
CInt(pilote2), CSng(L2 + incrementoL2))

                        If    Abs((asiento2    -    asientooriginal1)    /    distan-
cia_arranques(arranque1, arranque2)) < Módulo1.distorsion_angular Then
                            'comprobación de que se anula la distorsión angular
                            For i = 0 To 1000
```

```
                                If Módulo1.distorsiones_angulares(i, 0) = "" Then Exit
For
                                    If Módulo1.distorsiones_angulares(i, 0) = ar-
ranque2 Then
                                        If Abs((asiento2 - Módu-
lo1.distorsiones_angulares(i, 3) / 1000) / Módulo1.distorsiones_angulares(i, 4)) > Módu-
lo1.distorsion_angular Then
                                            numero_distorsiones_angulares_cambio =
numero_distorsiones_angulares_cambio + 1
                                        End If
                                    ElseIf Módulo1.distorsiones_angulares(i, 2) = ar-
ranque2 Then
                                        If Abs((asiento2 - Módu-
lo1.distorsiones_angulares(i, 1) / 1000) / Módulo1.distorsiones_angulares(i, 4)) > Módu-
lo1.distorsion_angular Then
                                            numero_distorsiones_angulares_cambio =
numero_distorsiones_angulares_cambio + 1
                                        End If
                                    End If
                            Next

                            If numero_distorsiones_angulares_cambio_2 < distor-
siones_arranque_2 Then
                               distorsiones_arranque_2 = nume-
ro_distorsiones_angulares_cambio_2

                                nuevopilote2 = pilote2
                                nuevok2 = k2
                                nuevoencepado2 = encepado2
                                nuevoL2 = L2 + incrementoL2
                                nuevocoste2 = coste_ejecucion(CInt(encepado2),
CInt(pilote2), CSng(k2), CSng(nuevoL2))


                                'comprobación de que reduce el número de distorsiones an-
gulares(el máximo posible)
                                'aquí se sustituiría directamente
                            ElseIf numero_distorsiones_angulares_cambio_2 = distorsio-
nes_arranque_2 Then
                                If coste_ejecucion(CInt(encepado2), CInt(pilote2),
CSng(k2), CSng(nuevoL2)) < nuevocoste2 Then
                                    nuevopilote2 = pilote2
                                    nuevok2 = k2
                                    nuevoencepado2 = encepado2
                                    nuevoL2 = L2 + incrementoL2
```

```
                                    nuevocoste2     =     coste_ejecucion(CInt(encepado2),
CInt(pilote2), CSng(k2), CSng(nuevoL2))
                                End If
                            End If

                        'si reduce respecto al anterior entra directamente aunque sea más
caro
                        'comprobación de coste mínimo
                        End If
                    Next
                Next
            Next
        Next


        'Aquí se decide cual de los dos es la mejor opción, según los criterios
        '1)La opción que reduzca más el número de distorsiones angulares será mejor
        '2)En caso de igual reducción, el de menor coste

        If  numero_distorsiones_angulares_cambio1  <  numero_distorsiones_angulares_cambio2
Then

            Módulo1.arranques(arranque1, 9) = nuevoencepado1
            Módulo1.arranques(arranque1, 10) = nuevok1
            Módulo1.arranques(arranque1, 11) = nuevopilote1
            Módulo1.arranques(arranque1, 12) = nuevoL1

        ElseIf           numero_distorsiones_angulares_cambio1           >           nume-
ro_distorsiones_angulares_cambio2 Then
            Módulo1.arranques(arranque2, 9) = nuevoencepado2
            Módulo1.arranques(arranque2, 10) = nuevok2
            Módulo1.arranques(arranque2, 11) = nuevopilote2
            Módulo1.arranques(arranque2, 12) = nuevoL2
        Else

            If nuevocoste1 + costeoriginal2 < nuevocoste2 + costeoriginal1 Then

                Módulo1.arranques(arranque1, 9) = nuevoencepado1
                Módulo1.arranques(arranque1, 10) = nuevok1
                Módulo1.arranques(arranque1, 11) = nuevopilote1
                Módulo1.arranques(arranque1, 12) = nuevoL1

            ElseIf nuevocoste1 + costeoriginal2 > nuevocoste2 + costeoriginal1 Then
                Módulo1.arranques(arranque2, 9) = nuevoencepado2
                Módulo1.arranques(arranque2, 10) = nuevok2
                Módulo1.arranques(arranque2, 11) = nuevopilote2
```

```
                Módulo1.arranques(arranque2, 12) = nuevoL2
            End If
        End If

        'Ahora hay que recalcular las que no cumplen

        ReDim Módulo1.distorsiones_angulares(1000, 10)
        Módulo1.distorsiones_angulares(0, 0) = "Arranque"
        Módulo1.distorsiones_angulares(0, 1) = "Asiento (mm)"
        Módulo1.distorsiones_angulares(0, 2) = "Arranque"
        Módulo1.distorsiones_angulares(0, 3) = "Asiento(mm)"
        Módulo1.distorsiones_angulares(0, 4) = "Distancia (m)"
        Módulo1.distorsiones_angulares(0, 5) = "Distorsión"
        Módulo1.distorsiones_angulares(0, 6) = "1/"
        Módulo1.distorsiones_angulares(0, 7) = "Estado"

        P = 1
        For arranque1 = 1 To Módulo1.n_arranques - 1
            For arranque2 = arranque1 + 1 To Módulo1.n_arranques
            'Los datos a emplear son de la matriz arranques (¡A SECAS¡
            'Si no la solución no tendría que ser válida
                Módulo1.distorsiones_angulares(P, 0) = arranque1
                Módulo1.distorsiones_angulares(P, 1) = Round(1000 * (asien-
to_individual_lp(CSng(Módulo1.arranques(arranque1, 3)) / Módu-
lo1.disposiciones(Módulo1.arranques(arranque1, 9), 0) + Nencep(Módulo1.arranques(arranque1,
9), Módulo1.arranques(arranque1, 10), CInt(Módulo1.arranques(arranque1, 11))),
CInt(Módulo1.arranques(arranque1, 11)), CSng(Módulo1.arranques(arranque1, 12))) + asien-
to_grupo(CSng(Módulo1.arranques(arranque1, 3)) + Módu-
lo1.disposiciones(Módulo1.arranques(arranque1, 9), 0) * Nencep(Módulo1.arranques(arranque1,
9), Módulo1.arranques(arranque1, 10), CInt(Módulo1.arranques(arranque1, 11))),
CInt(Módulo1.arranques(arranque1, 9)), CSng(Módulo1.arranques(arranque1, 10)),
CInt(Módulo1.arranques(arranque1, 11)), CSng(Módulo1.arranques(arranque1, 12)))), 1)
                Módulo1.distorsiones_angulares(P, 2) = arranque2
                Módulo1.distorsiones_angulares(P, 3) = Round(1000 * (asien-
to_individual_lp(CSng(Módulo1.arranques(arranque2, 3)) / Módu-
lo1.disposiciones(Módulo1.arranques(arranque2, 9), 0) + Nencep(Módulo1.arranques(arranque2,
9), Módulo1.arranques(arranque2, 10), CInt(Módulo1.arranques(arranque2, 11))),
CInt(Módulo1.arranques(arranque2, 11)), CSng(Módulo1.arranques(arranque2, 12))) + asien-
to_grupo(CSng(Módulo1.arranques(arranque2, 3)) + Módu-
lo1.disposiciones(Módulo1.arranques(arranque2, 9), 0) * Nencep(Módulo1.arranques(arranque2,
9), Módulo1.arranques(arranque2, 10), CInt(Módulo1.arranques(arranque2, 11))),
CInt(Módulo1.arranques(arranque2, 9)), CSng(Módulo1.arranques(arranque2, 10)),
CInt(Módulo1.arranques(arranque2, 11)), CSng(Módulo1.arranques(arranque2, 12)))), 1)
                Módulo1.distorsiones_angulares(P, 4) =
Round(distancia_arranques(arranque1, arranque2), 2)
```

```
                Módulo1.distorsiones_angulares(P, 5) =
Round(Abs(Módulo1.arranques_optimizacion(arranque1, 5, pilote) - Módu-
lo1.arranques_optimizacion(arranque2, 5, pilote)) / distancia_arranques(arranque1, arran-
que2), 6)
                Módulo1.distorsiones_angulares(P, 6) = Round(1 /
(Abs(Módulo1.arranques_optimizacion(arranque1, 5, pilote) - Módu-
lo1.arranques_optimizacion(arranque2, 5, pilote)) / distancia_arranques(arranque1, arran-
que2)), 0)

                If CSng(Módulo1.distorsiones_angulares(P, 5)) > Módu-
lo1.distorsion_angular Then
                    Módulo1.distorsiones_angulares(P, 7) = "NO CUMPLE"
                Else
                    Módulo1.distorsiones_angulares(P, 7) = "CUMPLE"
                End If

                P = P + 1
            Next
        Next

        dist_max = Módulo1.distorsion_angular
        numero_distorsiones_no_cumplen = 0
        For i = 0 To 1000
            If Módulo1.distorsiones_angulares(i, 0) = "" Then Exit For
                If Módulo1.distorsiones_angulares(i, 5) > Módulo1.distorsion_angular Then
                    numero_distorsiones_no_cumplen = numero_distorsiones_no_cumplen + 1
                    If Módulo1.distorsiones_angulares(i, 5) > dist_max Then
                        dist_max = Módulo1.distorsiones_angulares(i, 5)
                        arranque1 = Módulo1.distorsiones_angulares(i, 0)
                        arranque2 = Módulo1.distorsiones_angulares(i, 2)
                    End If
                End If
        Next

    Loop


    End Sub
```

Funciones de optimización

ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

ANEXOS

SOFTWARE DE CÁLCULO Y OPTIMIZACIÓN DE CIMENTACIONES POR PILOTES

ANEXO II

Nº TFG: 423.15.5

Autor: David Ostáriz Falo

Director: José Ángel Pérez Benedicto

Fecha: 28-6-2016

INDICE DE CONTENIDO

1. INTRODUCCIÓN 3

2. INICIO DEL PROGRAMA 5

2.1. PROBLEMAS FRECUENTES 6

3. MENÚ INICIO 7

4. DATOS GENERALES 9

5. DATOS DEL TERRENO 11

6. ENCEPADOS 13

7. PILOTES 16

7.1. MICROPILOTES 17

8. ENCEPADO AISLADO 19

9. PLANTA DE CIMENTACIÓN 22

10. AJUSTES DE CÁLCULO 25

1. INTRODUCCIÓN

OPTIPIL es un programa creado para poder realizar los cálculos relativos al dimensionamiento de pilotes, aspecto no observado por la mayoría delos programas existentes en el mercado. Se centra exclusivamente en este aspecto, obviando cualquier otro que ya se contemple en lamayoría de software comercial, como por ejemplo la combinación de acciones, el dimensionado de elementos superficiales como encepados o vigas de cimentación y el cálculo de los estados límite a flexión y a cortante de los elementos proyectados.

El cálculo se centra en los esfuerzos axiles sobre el pilote, dado que son el aspecto más relevante para su dimensionamiento. En consecuencia no será de utilidad en casos en que se vayan a producir esfuerzos cortantes y flectores de magnitud tal que sea necesarias comprobaciones estructurales.

Ha sido desarrollado en Visual Basic para Aplicaciones con la versión 2010 de Microsoft Excel, no habiendo presentado problemas durante la fase de prueba con versiones posteriores del mismo. Aún así, no puede garantizarse que vaya a funcionar correctamiente en una versión que no sea la mencionada. Para su correcta utilización

el sistema operativo debe encontrarse configurado con configuración numérica española.

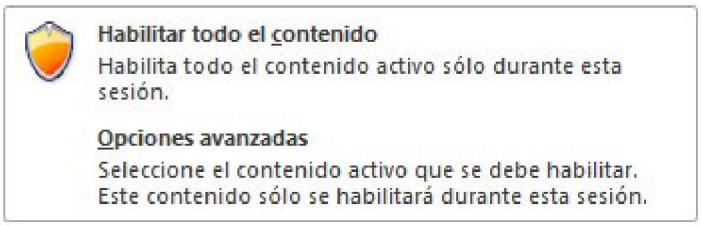
2. INICIO DEL PROGRAMA

Cuando abre un archivo con macros, aparece la barra de mensajes amarilla con un icono de escudo y el botón Habilitar contenido. Si sabe que las macros proceden de un origen confiable, use las siguientes instrucciones:

En la Barra de mensajes, haga clic en Habilitar contenido.

Otro método para habilitar macros en un archivo es mediante la vista Backstage de Microsoft Office, la que aparece al hacer clic en la pestaña Archivo, cuando aparece la barra de mensajes amarilla.

1. Haga clic en la pestaña Archivo.
2. En el área Advertencia de seguridad, haga clic en Habilitar contenido.
3. En Habilitar todo el contenido, haga clic en Habilitar siempre el contenido activo del documento.

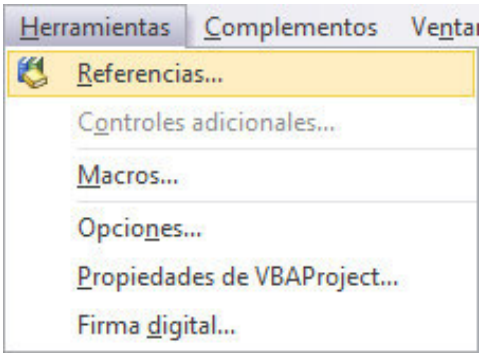


Utilice las siguientes instrucciones para habilitar macros mientras el archivo esté abierto. Cuando cierre el archivo y, a continuación, vuelva a abrirlo, la advertencia aparecerá de nuevo.

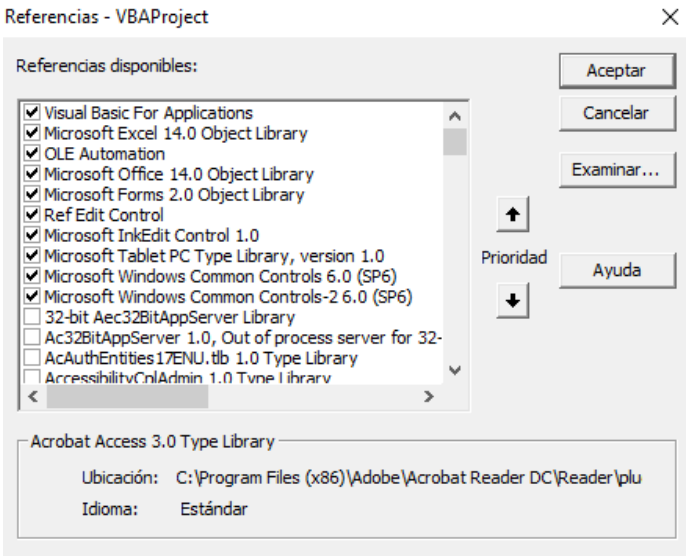
1. Haga clic en la pestaña Archivo.
2. En el área Advertencia de seguridad, haga clic en Habilitar contenido.
3. Seleccione Opciones avanzadas.
4. En el cuadro de diálogo Opciones de seguridad de Microsoft Office, haga clic en Habilitar contenido para esta sesión para cada macro.
5. Haga clic en Aceptar.

2.1. PROBLEMAS FRECUENTES

En caso de que se produzca un error de compilación al intentar iniciar por primera vez el programa en el que se lea "no se puede encontrar el proyecto o la biblioteca", debe accederse al menú referencias dentro de la barra de herramientas, desde la ventana de programador, que debería abrirse automáticamente al saltar el error.



En la pestaña de referencias se muestran todas aquellas bibliotecas a las que tiene acceso el programa. En caso de que el nombre de alguna de las seleccionadas comience con la palabra FALTA o MISSING deberá desactivarse y aceptar. De esta forma el error no se repetirá en usos posteriores.

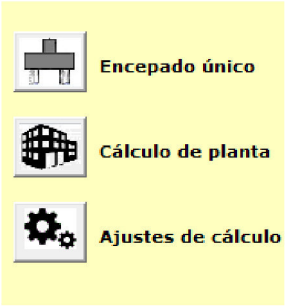


3. MENÚ INICIO

El menú de inicio sirve de acceso a las distintas funciones del programa:

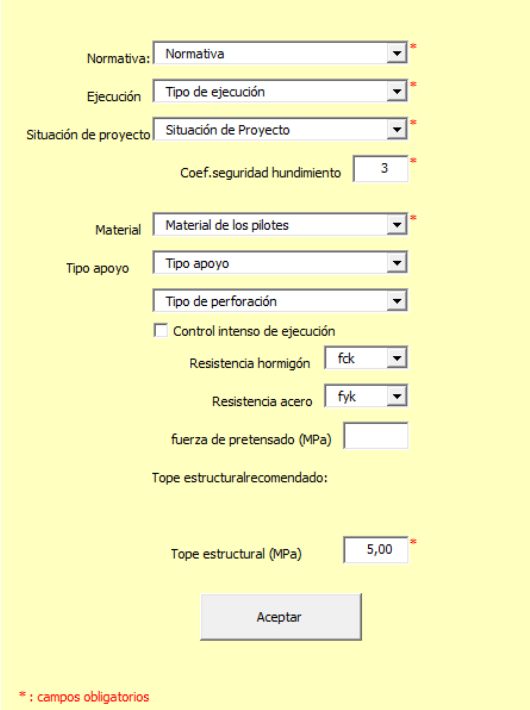
- Introducción de datos
  - **Datos generales**: en este apartado se eligen los aspectos globales del proyecto, tales como la normativa que regirá los cálculos, los procedimientos de ejecución, los materiales, etc.
  - **Datos del terreno**: en este apartado se especifican los datos relativos al terreno donde se proyecta la cimentación. Como cada normativa requiere distintos parámetros no se podrá acceder hasta haberla determinado.
  - **Encepados**: en este apartado se escogen o generan los encepados que se emplearán para los cálculos.
  - **Pilotes**: en este apartado se escogen o generan los pilotes que se emplearán en lños cálculos. Dado que la geometría dependerá en parte del material y del tipo de ejecución para acceder deberán determinarse para acceder a él.

- Cálculo
  - **Encepado único**: permite realizar los cálculos relativos a un arranque aislado.
  - **Cálculo de planta**: permite realizar los cálculos relativos a una planta de cimentación.
  - **Ajustes de cálculo**: permite seleccionar los parámetros de cálculos que mejor se ajusten al proyecto.



- Otros
  - **Guardar**: guarda los datos introducidos: normativa, materiales, ejecución, situación de proyecto, terreno, arranques, sondeos, pilotes, encepados, etc.
  - **Cargar**: carga los datos que han sido guardados con anterioridad.
  - **Salir**: salir del menú.

4. DATOS GENERALES



En este menú han de escogerse los parámetros generales del proyecto, que no deberán ser modificados posteriormente.

Normativa*: Permite escoger la normativa que rige el proyecto de entre las siguientes:

- Guía de Cimentaciones en Obras de Carretera
- Código Técnico de la Edificación: Documento Básico de Seguridad Estructural – Cimientos
- Recomendaciones Geotécnicas para Obras Marítimas y Portuarias
- Guía para el Proyecto y la Ejecución de Micropilotes en Obras de Carretera

Ejecución: Permite escoger el procedimiento de ejecución de los pilotes. En caso de haber escogido en la pestaña normativa la ejecución de micropilotes esta pestaña estará bloqueada.

- Pilotes perforados
- Pilotes hincados

Situación de proyecto: Permite escoger la situación de proyecto con el fin de determinar el coeficiente de seguridad mínimo frente a hundimiento. En caso de haber escogido en la pestaña normativa la ejecución de micropilotes esta pestaña estará bloqueada.

- Permanente
- Transitoria
- Accidental

Coeficiente de seguridad: permite introducir manualmente el valor del coeficiente de seguridad a hundimiento, siempre que éste no sea inferior al mínimo definido por la instrucción de referencia.

Material: Permite escoger el material de los pilotes. Tanto si se ha escogido la opción de pilotes perforados o la opción de micropilotes esta pestaña se encontrará bloqueada. Se podrá escoger entre:

- Hormigón armado
- Hormigón pretensado
- Acero
- Madera

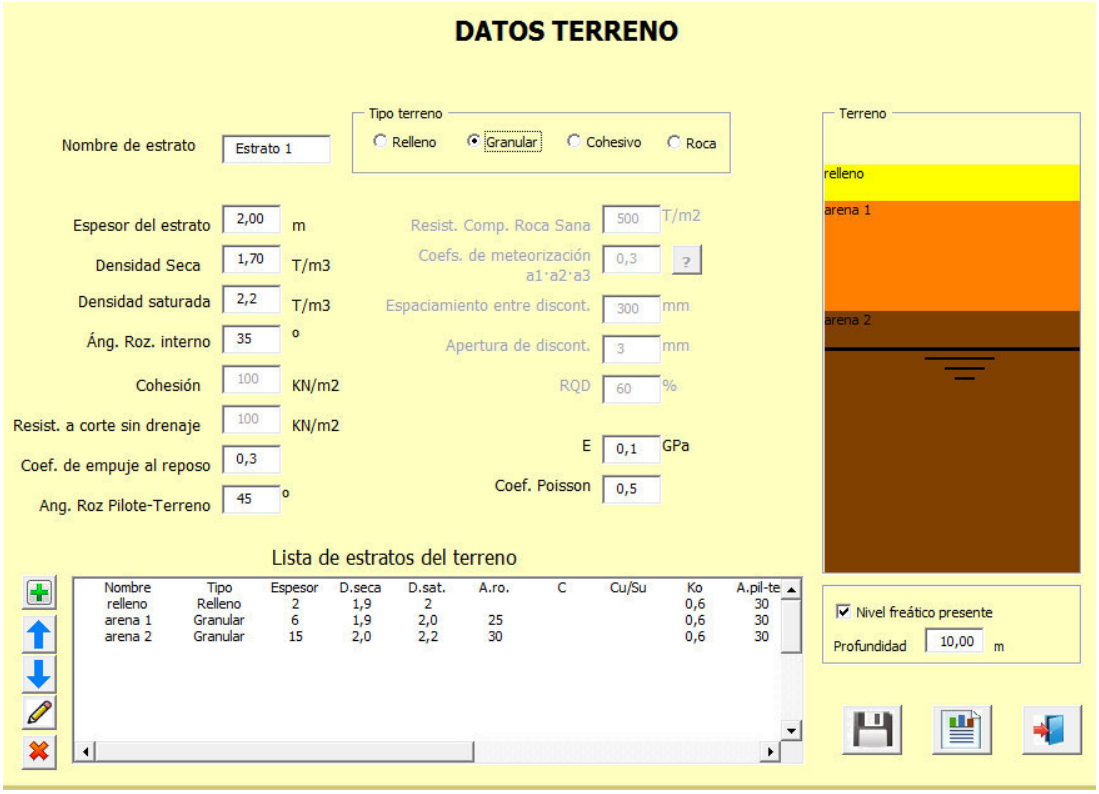
Las siguientes opciones no son aspectos fundamentales del proyecto, sino que permiten al usuario obtener el valor recomendado por la normativa para el tope estructural de los pilotes.

- Tipo de apoyo
- Tipo de perforación
- Control intenso de ejecución
- Resistencia acero
- Resistencia hormigón
- Fuerza de pretensado

Tope estructural: será la máxima tensión a la que podrán verse sometidos los pilotes, salvo en el caso de micropilotes, en cuya instrucción no está contemplado este concepto, y cuya resistencia estructural se calcula para cada caso individualmente.

*La normativa escogida es el parámetro fundamental que determinará la mayoría de opciones disponibles del programa. En caso de cambiarse una vez introducidos otros datos estos podrían dejar de ser válidos y llevar a errores. Por tanto se recomienda elegir la normativa inicialmente y no modificarse.

5. DATOS DEL TERRENO



En el menú de datos de terreno se introducen todos los datos relativos a los diferentes estratos existentes, definiendo para cada uno de ellos un nombre y un tipo*.

Según la normativa escogida y el tipo de estrato se permite la introducción de los datos que sean relevantes para cada uno de ellos, que se guardarán en la tabla inferior al pulsar el botón añadir.

Se recomienda no dejar ninguno de los campos que se indican sin determinar, dado que podría resultar en posteriores errores o imprecisiones de cálculo.

En caso de haber introducido valores erróneos puede optarse por seleccionar el estrato en la tabla inferior y editarlo o borrarlo.

Para borrarlo bastará con pulsar el botón eliminar.

Para editar alguno de los campos del estrato bastará con introducir el nuevo valor en la casilla correspondiente y pulsar el botón editar.

En caso de que se quiera alterar el orden de los estratos**, por ejemplo si se quiere añadir una capa de rellenos tras haber creado el resto de estratos o si simplemente se ha olvidado de introducir alguno, se podrán emplear los  botones con los iconos de flecha al lado de la lista con los estratos. El estrato de la lista seleccionado será el que se desplace mediante esta acción.

Además, en caso de que se haya detectado la presencia del nivel freático en la profundidad sondeada podrá incluirse este dato en el panel derecho inferior. En caso de que el terreno esté sumergido se deberá adoptar un valor de profundidad del nivel freático nulo.

En todo momento podrán visualizarse los estratos en la pantalla situada a la derecha. Esta es una pantalla gráfica dinámica no interactiva que cambia cada vez que se modifican o introducen nuevas características geométricas de los estratos o del nivel freático. De esta forma podrá comprobarse además si el programa lee los valores que se le introducen correctamente.

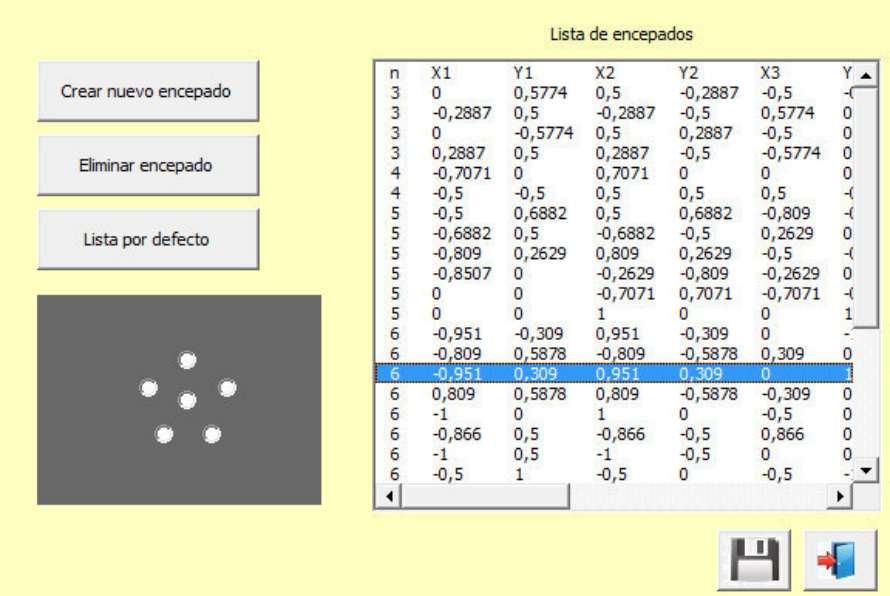
En la esquina inferior derecha se encuentran los botones para guardar la tabla de estratos, generar un listado de los estratos y sus características o salir del menú.

*Siempre que se estime que un terreno vaya a asentar, causando el fenómeno de rozamiento negativo sobre el pilote, deberá definirse el estrato como relleno.

**En ningún caso el programa determinará si las diferentes capas del terreno definido corresponden a una situación válida, y tampoco determinará los ajustes de cálculo necesarios para dicha situación.

Corresponde al usuario determinar una situación válida para el cálculo, dado que el programa realizará los cálculos independientemente de cómo se distribuyan los estratos.

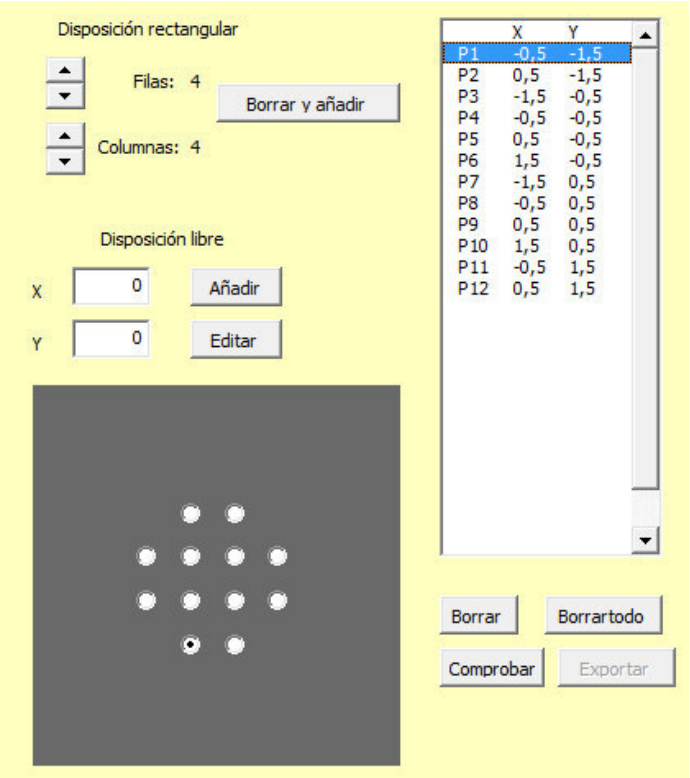
6. ENCEPADOS



En el menú de encepados se define la lista de disposicones de pilotes que se empleará posteriormente en los cálculos. En la pantalla principal simplemente existen las opciones de generar la lista por defecto, de abrir la ventana de creación de encepados.

Además existe la posibilidad de visualizar las disposiciones de la lista, para así poder identificar cada uno con facilidad.

Una vez definida la lista existe la opción de guardarla directamente desde la misma ventana.



La ventana de creación de encepado permite la creación de nuevas disposiciones. Las disposiciones se definen mediante las posiciones X e Y respecto del arranque en unidades de separación de diámetros.

La nueva disposición se puede crear definiendo la posición de cada pilote uno a uno o bien con la opción de crear disposiciones rectangulares, definiendo el número de pilotes a lo largo y a lo ancho*.

Las posiciones de los pilotes pueden editarse, y pueden eliminarse aquellos que no procedan.

Para facilitarse la creación y edición se ha dispuesto una pequeña ventana de visualización en la que además se marca el pilote seleccionado en la tabla, y así identificarlo fácilmente.

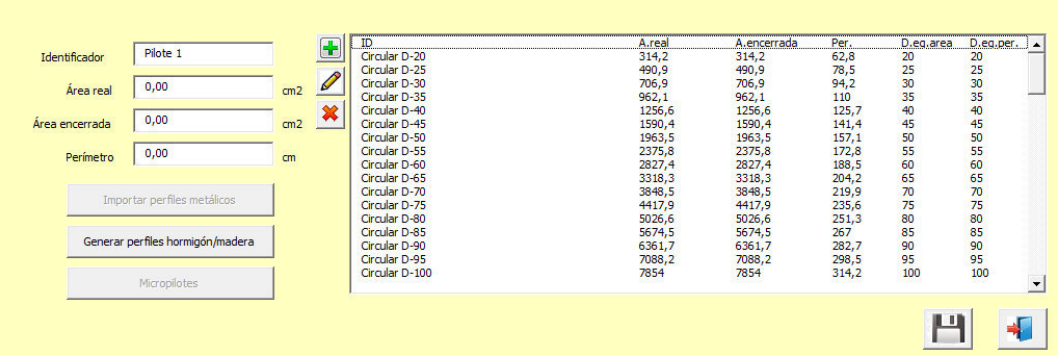
Una vez definida la disposición deberá comprobarse que cumple con las condiciones adecuadas para que sea apta para la metodología de cálculo empleada**.

* Las disposiciones de este tipo serán de un máximo de 25 pilotes, con máximo de 8 por dirección y sin poder definirse disposiciones tipo fila o columna.

**Estas condiciones son:

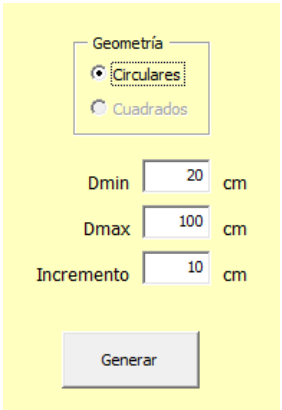
- La separación mínima entre pilotes dentro de la disposición no debe ser inferior a 1 (esta comprobación es realizada por el programa)
- La disposición no deberá nunca ser similar a una línea de pilotes (esta comprobación queda a criterio del usuario. En caso de no respetarse podría dar lugar a errores o imprecisiones)
- La disposición no tendrá menos de 3 pilotes (esta comprobación es realizada por el programa, y es una extensión de la anterior, dado que de haber menos pilotes el encepado será siempre asimilable a una línea)

7. PILOTES



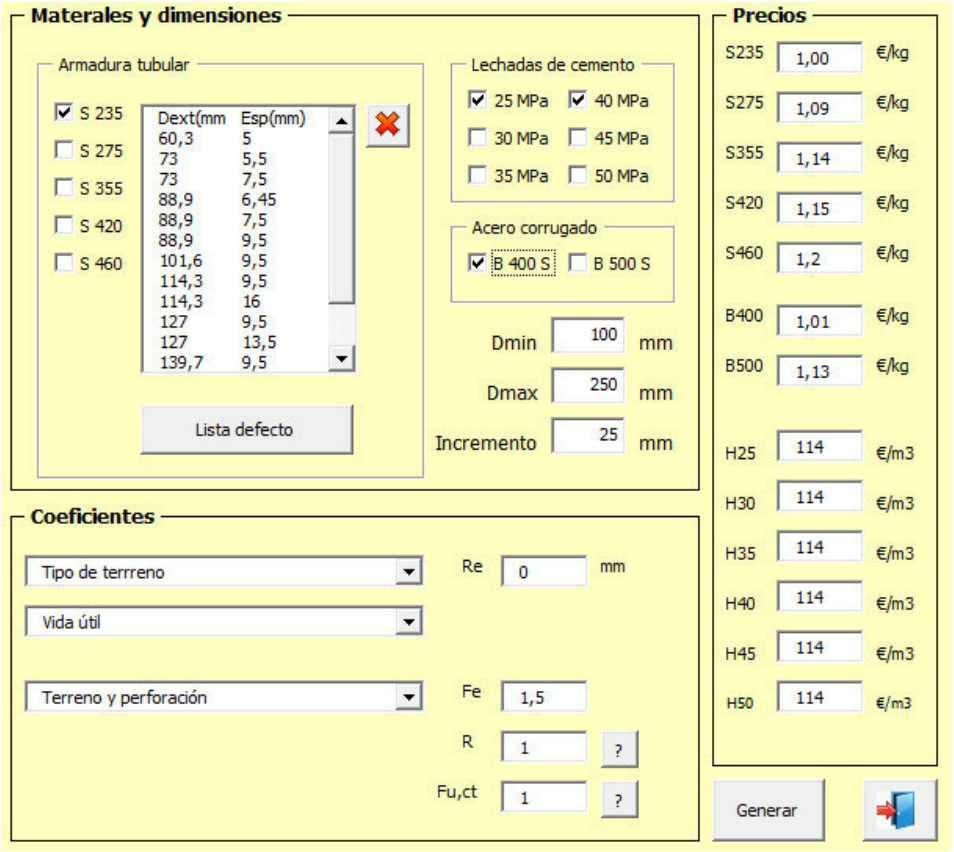
En esta pantalla se genera el conjunto de pilotes distintos que se emplearán en los cálculos. Éstos pueden crearse individualmente introduciendo sus características en los campos de la esquina superior izquierda, sin embargo existe la opción de generar o importar las tablas de pilotes automáticamente.

En caso de pilotes metálicos puede importarse una tabla con los perfiles comerciales, en otros casos podrán generarse perfiles macizos tanto circulares como cuadrados, como corresponda según el material a emplear y el procedimiento de ejecución escogido.



En dicho caso se podrá generar una tabla partiendo de límites geométricos y un incremento.

7.1. MICROPILOTES



En el caso de los micropilotes deberán determinarse una variedad de parámetros más amplia para su definición.

- Materiales y dimensiones
  - Armadura tubular

    Deben seleccionarse tanto los tipos de acero que se baraja emplear en el proyecto como los tubos a emplear. Se recomienda reducir al máximo posible la lista de armaduras tubulares. Pueden eliminarse las que se consideren inadecuadas seleccionándolas en la lista y pulsando el botón eliminar. En caso de querer recuperar la lista original habría que pulsar el botón de "Lista por defecto".
  - Lechadas de cemento

    Deben seleccionarse los tipos de lechada que se baraja emplear en el proyecto.
  - Acero corrugado

Debe seleccionarse el/los tipos de acero que se baraja emplear en el proyecto.

- Geometría exterior

  Al igual que con los pilotes circulares, se definen las diferentes geometrías a emplear mediante los valores extremos y un incremento.

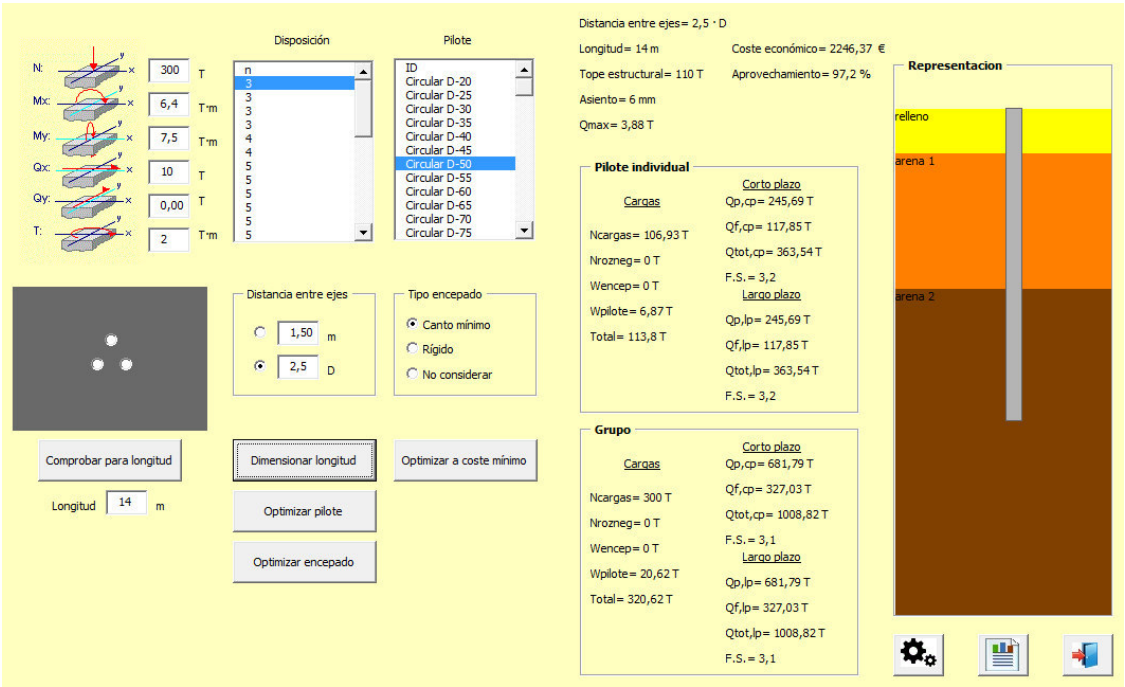
- Coeficientes

  Han de definirse los coeficientes que regirán los cálculos relativos a los micropilotes, escogiendo las opciones de las pestañas o consultando las ventanas de ayuda.

- Precios

Han de definirse los precios de todos los materiales que hayan de emplearse para una mayor fiabilidad de los resultados.

8. ENCEPADO AISLADO



La ventana de encepado aislado permite realizar los cálculos relativos y visualizar los resultados sobre un único arranque. Puede dividirse en dos secciones principales.

En la mitad izquierda se sitúa la entrada y selección de datos.

- Cargas sobre arranque: deben definirse las cargas sobre el arranque. En caso de no haber axil positivo no se permitirán realizar los cálculos
- Tabla de disposiciones definida en el apartado "Encepados".
- Tabla de pilotes definida en el apartado "Pilotes".
- Distancia entre ejes, definida en metros o en función del diámetro.
- Tipo de encepado, como se define en el apartado "Ajustes de Cálculo"
- Longitud: longitud de los pilotes planteadas.

Una vez introducidos y seleccionados los datos se presentan varias opciones de cálculo.

- **Comprobar para longitud dada***
  Para esta opción deben seleccionarse pilote, encepado y longitud, y proporcionará los resultados de resistencia del terreno y de factor de seguridad para dicha combinación.
- **Dimensionar longitud***
  Deben seleccionarse pilote y encepado, y el programa devolverá la longitud óptima del pilote, es decir, la mínima con la que se cumplen todas las condiciones de seguridad definidas.
- **Optimizar pilote****
  Debe seleccionarse únicamente el encepado. El programa devolverá el pilote que mejor solución ofrezca, habiendo iterado manteniendo el encepado fijo.
- **Optimizar encepado****
  Debe seleccionarse únicamente el pilote a emplear. El programa devolverá el mejor encepado.
- **Optimizar a coste mínimo****
  No debe seleccionarse ningún parámetro. El programa estudiará todas las combinaciones posibles y devolverá aquella que presente un menor coste económico.

*Estas opciones solo se ejecutarán cuando no se supere el tope estructural del pilote

**Estas opciones emplean la separación entre diámetros definida en la pantalla de ajustes de cálculo, a la que se puede acceder mediante el botón de la parte inferior derecha que lleva su símbolo.

En la mitad derecha se representan los resultados obtenidos tanto numérica como gráficamente. En la parte superior se dan:

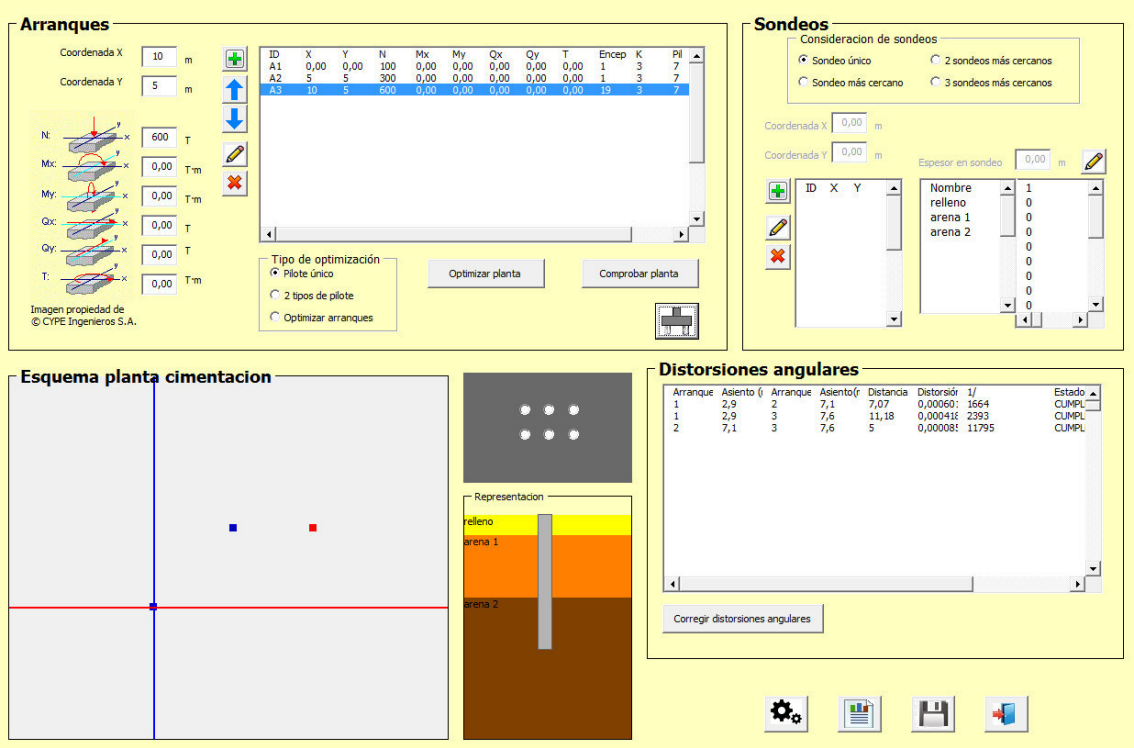
- Distancia entre ejes
- Longitud del pilote
- Tope estructural: carga máxima que puede soportar el pilote
- Asiento del arranque, considerando la suma del asiento del pilote individual con carga media y el efecto grupo
- Coste económico aproximado según los criterios definidos

- Aprovechamiento del pilote, en función de la carga máxima que soporta, que es igual a la suma de la carga en cabeza el peso del encepado y el rozamiento negativo.

Bajo estos valores se presenta un resumen de la comprobación a hundimiento del pilote más cargado dentro del grupo y, cuando corresponda, la comprobación considerando el efecto grupo. En ambos cuadros se presentan los distintos tipos de cargas, además de las resistencias del terreno a corto y a largo plazo, tanto por fuste como por punta. Por último presenta el factor de seguridad tanto a corto como a largo plazo.

Desde esta ventana puede accederse directamente a los ajustes de cálculo sin necesidad de salir de nuevo a la pantalla inicio, además de generarse los listados correspondientes a las comprobaciones del arranque.

9. PLANTA DE CIMENTACIÓN



La ventana de Planta de Cimentación permite el análisis tanto individual como global de un conjunto de arranques.

- Arranques

  En este apartado debe definirse la lista de arranques, indicando la posición de cada uno de ellos en el plano coordenado*. Los botones de añadir, eliminar, editar y mover funcionan igual que en los casos ya vistos en anteriores apartados.

  *Las coordenadas no se ven sujetas a ningún tipo de control, por lo que es el usuario el que debe asegurarse que los arranques se sitúan a una distancia adecuada.

  - Tipo de optimización
    - **Pilote único**: devolverá la solución más económica que emplee el mismo tipo de pilote en todos los arranques.
    - **Dos pilotes**: devolverá la solución más económica que emplee dos tipos de pilote en toda la planta.

        - **Optimizar arranques**: optimiza cada arranque por separado tal y como lo hacía la ventana de encepado único.
    - **Comprobar planta**: calcula las distorsiones angulares en una planta. Se emplea cuando se ha modificado alguno de los arranques.
    - **Edición de arranque**: este botón, que muestra en su icono un encepado, permite editar los resultados que devuelve el programa en un único arranque. La diferencia con el botón con el icono del lapicero es que este último solamente permite modificar las cargas y las coordenadas.
- Sondeos
    - En el caso de que se haya realizado más de un sondeo y se desee considerar los resultados obtenidos en el mismo podrán introducirse las coordenadas de los mismos junto a los espesores de los estratos hallados en cada posición. De esta manera podrá considerarse con mayor precisión el terreno que se halla bajo cada arranque.
        - Consideración de sondeos
            - **Sondeo único**: se tomarán para todos los arranques los espesores definidos en la ventana de datos del Terreno.
            - **Sondeo más cercano**: se tomarán para cada arranque los espesores hallados en el sondeo que se encuentre a menor distancia de su situación.
            - **2 sondeos más cercanos**: se interpola linealmente entre los dos sondeos más próximos al arranque, calculándose de esta forma los espesores.
            - **3 sondeos más cercanos**: se interpola linealmente entre los tres sondeos más próximos al arranque, calculándose de esta forma los espesores.

Para editar los espesores deberá seleccionarse el sondeo en la tabla izquierda, previa definición de sus coordenadas, y el estrato que se quiere modificar. Introduciendo el nuevo valor y pulsando el botón editar se sustituirá el valor por el introducido.

- Distorsiones angulares

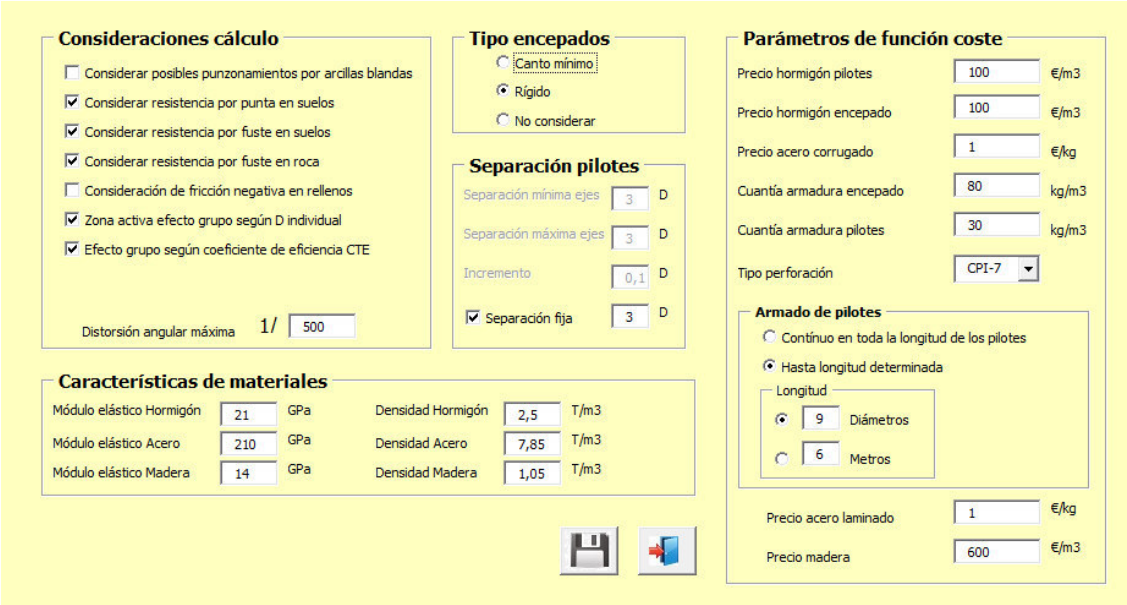
  En la tabla de distorsiones angulares podrán observarse las mismas entre los distintos arranques dos a dos, y comprobar si cumplen. En caso de que no cumplan podrá optarse por efectuar una corrección de las mismas, con el botón en la parte inferior.

Además, en esta pestaña podrá visualizarse en todo momento un esquema de la planta, en el cual podrá observarse la posición de los arranques respecto a los ejes coordenados. En caso de seleccionarse un arranque este se mostrará en rojo y, a la derecha, podrá visualizarse tanto el encepado que en él se encuentra como la disposi-

ción de estratos según los datos de terreno y de sondeos que se hayan introducido, además de la longitud del pilote para dicho arranque.

Las opciones de listados será aquí mucho más amplias que las hasta ahora vista, permitiendo la creación automática de las comprobaciones individuales de cada arranque, de las comprobaciones globales y de los datos obtenidos en los distintos sondeos.

10. AJUSTES DE CÁLCULO



La ventana de ajustes de cálculo permite establecer variaciones en los mismos según convenga al proyecto.

- Consideraciones de cálculo
  - **Punzonamiento por arcillas blandas**: Se trata de una reducción de capacidad por punta en suelos granulares debida a la presencia de estratos arcillosos flojos inferiores.
  - **Resistencia por punta en suelos**: en caso de no considerarla se estarían dimensionando los pilotes como estrictamente flotantes. Es lo habitual en el caso de micropilotes.
  - **Resistencia por fuste en suelos**: En caso de no considerarla se estaría teniendo en cuenta únicamente la resistencia por punta y fuste de estratos rocosos. Es lo habitual cuando el pilote se empotra en uno.
  - **Resistencia por fuste en roca**: en caso de no considerarla, solo se tendría en cuenta la resistencia por punta en estratos rocosos.
  - **Fricción negativa**: en caso de considerarla se añadiría a las cargas sobre el pilote la suma del rozamiento negativo producido por el asentamiento de todos los estratos de rellenos definidos.
  - **Zona activa del grupo**: En caso de que el efecto grupo se estudie mediante la técnica de la cimentación equivalente puede optarse por tomar como zona activa la relativa al diámetro del grupo o la relativa al pilote individual.

- **Efecto grupo según coeficiente de eficiencia**: Permite evaluar el efecto grupo mediante el procedimiento del coeficiente de eficiencia del CTE en lugar de por el de la cimentación equivalente.
- **Distorsión angular máxima**: Valor límite que se empleará para los cálculos de la planta.

- Tipo encepados

  Este apartado permite seleccionar el tipo de encepado que se considerará tanto para la determinación del peso equivalente sobre cabeza de pilote como para el cálculo de costes. Puede optarse por considerarlo rígido o imponer el canto mínimo, ambas según las especificaciones de la EHE, o simplemente no considerarlo.

  *Podría surgir la necesidad de querer tenerlo en cuenta para determinar las cargas máximas sobre pilote, pero sin que influya en el cálculo del coste. En tal caso se recomienda fijar el valor de precio de hormigón de encepado y de cuantía de armaduras en encepado a 0.

- Separación de pilotes:

  Determina el rango de separación de pilotes que se empleará en las optimizaciones. Puede definirse mediante los valores extremos y un incremento u optar por mantenerse constante, para así reducir el número de operaciones además de, en su caso, ahorrarse las comprobaciones relativas al efecto grupo.

- Características de materiales

  Aquí se definen las características de los materiales que se vayan a emplear en el proyecto. Aunque estos valores suelen ser estándar puede haber ocasiones en que sea necesario cambiarlos para obtener resultados más exactos.

- Parámetros de función coste

  En este apartado se pueden determinar los costes de los principales materiales que se emplearán en el proyecto, y las cuantías en caso del hormigón.

  - **Tipo de perforación**

    Permite seleccionar el procedimiento de perforación en caso de que los pilotes se realicen in situ, variando la función de costes de ejecución y, según el caso, la consideración del material en el fuste.

  - **Armado de pilotes**

    Permite determinar que longitud de los pilotes va a armarse, en metros o en función de los diámetros del pilote.

- **Precios de materiales**

  Permite la modificación de los valores empleados en las funciones de coste.

- **Cuantías de armaduras**

  Permite determinar el armado que llevarán tanto el encepado como los pilotes.

- Armado de pilotes

  Permite definir la longitud máxima a la que se armarán aquellos ejecutados in situ, tanto en unidad métrica como en función del diámetro del mismo.