CAPÍTULO 5

ALGORITMOS

5.1. RESPUESTA DINÁMICA

5.1.1. Respuesta de osciladores de 1GL

Para obtener la respuesta dinámica de un oscilador de 1GL es necesario resolver la ecuación de equilibrio dinámico (2.10). No es posible obtener una solución analítica de la ecuación de movimiento si la carga $p(t)$ o la aceleración del suelo $\ddot{x}_S(t)$ varían arbitrariamente con el tiempo y dado que el objetivo del programa es calcular y graficar la respuesta de los osciladores ante cualquier tipo de excitación que el usuario defina, es necesario utilizar un método numérico para resolver la ecuación de movimiento (2.10).

Existen varios métodos que resuelven la ecuación de equilibrio:

- Métodos basados en la interpolación lineal de la función de la excitación.
- Métodos basados en expresiones de diferencia finita de velocidad y aceleración
- Métodos que asumen variación de aceleración

No obstante, todos deben cumplir con tres características importantes: ser precisos, estables y deben converger a la solución exacta del problema.

Para utilizar un método numérico es necesario que la aplicación de la señal de la excitación $p(t)$ sea dada por valores discretizados:

$p_i = p(t_i),\quad Desde\ i = 0\ \ hasta\ i = N$

Donde :

i = instante de tiempo *discretizados*

N = número total de puntos de la señal de exitación

Para determinar la respuesta dinámica se utilizó una aproximación numérica de la integral de Duhamel, conocido también como el método de las *ocho constantes*, que es un método numérico altamente eficiente para sistemas lineales. Consiste en determinar la respuesta de desplazamiento y velocidad para cada intervalo de tiempo por medio de la ayuda de constantes (4 para desplazamiento y 4 para velocidad) que se calculan una sola vez en el procedimiento (Ordaz, 2006).

El método es especialmente eficiente (Chopra, 2001) cuando el registro de aceleraciones tiene un intervalo de tiempo (Δt) constante entre cada dato del registro, y además, supone una relación lineal entre ellos. Si los intervalos (Δt) son pequeños la interpolación lineal se considera satisfactoria.

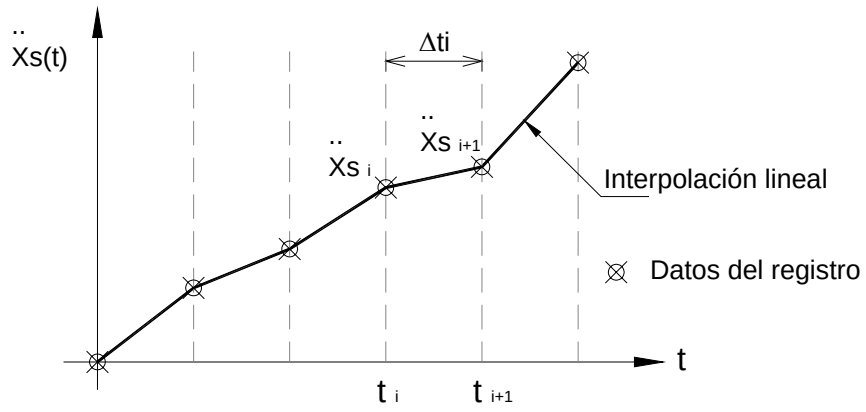

Figura 5.1 Función de carga arbitraria

La figura 5.1 muestra la representación de un registro de aceleraciones arbitrarias con intervalo Δti, donde *i* es un instante de tiempo que puede valer desde 1 hasta N. El intervalo de tiempo se calcula como:

$$\Delta t_i = t_{i+1} - t_i \quad (5.1)$$

El método de las *ocho constantes* resuelve la ecuación 2.10 y obtiene dos expresiones: una para calcular el desplazamiento (5.2) y otra para la velocidad (5.3), ambas en cada instante de tiempo i.

$$x_{i+1} = \alpha_1 x_i + \alpha_2 \dot{x}_i + \alpha_3 \ddot{x}s_i + \alpha_4 \ddot{x}s_{i+1} \quad (5.2)$$

$$\dot{x}_{i+1} = \beta_1 x_i + \beta_2 \dot{x}_i + \beta_3 \ddot{x}s_i + \beta_4 \ddot{x}s_{i+1} \quad (5.3)$$

Los coeficientes $\alpha 1$, $\alpha 2$, $\alpha 3$ y $\alpha 4$ son las constantes para desplazamiento y se obtienen con las ecuaciones 5.10 a 5.13 respectivamente. Asimismo, los coeficientes $\beta 1$, $\beta 2$, $\beta 3$ y $\beta 4$ son las constantes para calcular la velocidad y se obtienen con las ecuaciones 5.14 a 5.17.

Para calcular la aceleración relativa simplemente se despeja $\ddot{x}(t)$ de la ecuación 2.10 obteniendo la siguiente ecuación para cada instante i.

$$\ddot{x}_i = -\ddot{x}_{S\,i} - 2\xi\Omega\dot{x}_i - \Omega^2 x_i \quad (5.4)$$

La aceleración absoluta para cada intervalo de tiempo se calcula como:

$$\ddot{x}_{Ai} = \ddot{x}_i + \ddot{x}_{Si} \quad (5.5)$$

Para simplificar un poco las ecuaciones 5.10 a 5.17, se hicieron las siguientes sustituciones:

$$E = e^{-\xi\Omega\Delta t} \quad (5.6)$$

$$S = \operatorname{sen}(\Omega_D \Delta t) \quad (5.7)$$

$$C = \cos(\Omega_D \Delta t) \quad (5.8)$$

$$\Omega_D = \Omega\sqrt{1-\xi^2} \quad (5.9)$$

Las siguientes ecuaciones son las conocidas como las ocho constantes:

$$\alpha_1 = E\left(\frac{\xi \cdot \Omega \cdot S}{\Omega_D} + C\right) \quad (5.10)$$

$$\alpha_2 = \frac{E \cdot S}{\Omega_D} \quad (5.11)$$

$$\alpha_3 = \frac{\left[\frac{\left(\xi\Omega + \frac{2\xi^2 - 1}{\Delta t}\right)S}{\Omega_D} + \left(1 + \frac{2\xi}{\Omega\Delta t}\right)C\right]E - \frac{2\xi}{\Omega\Delta t}}{\Omega^2} \quad (5.12)$$

$$\alpha_4 = \frac{\left[-\frac{\left(2\xi^2 - 1\right)S}{\Omega_D \Delta t} - \frac{2\xi C}{\Omega\Delta t}\right]E - 1 + \frac{2\xi}{\Omega\Delta t}}{\Omega^2} \quad (5.13)$$

$$\beta_1 = -\frac{E\Omega S}{\sqrt{1-\xi^2}} \quad (5.14)$$

$$\beta_2 = \left(-\frac{E\Omega S}{\Omega_D} + C\right)E \quad (5.15)$$

$$\beta_3 = \frac{1 + \left(-\frac{(\Omega\Delta t + \xi)S}{\sqrt{1-\xi^2}} - C\right)E}{\Omega^2 \Delta t} \quad (5.16)$$

$$\beta_4 = \frac{\left(\frac{\xi S}{\sqrt{1-\xi^2}} + C\right)E - 1}{\Omega^2 \Delta t} \quad \text{.....(5.17)}$$

Para cada valor discretizado de la señal de excitación se genera un valor de la respuesta dinámica; el número total de puntos (N) de las señales de excitación puede variar, pero cuando este valor comienza a ser muy grande, calcular la respuesta para cada punto puede ser muy tardado; no obstante, para la computadora es muy rápido calcular las respuestas.

El método resulta ser muy práctico, pues para calcular la respuesta dinámica de un oscilador sólo es necesario calcular una sola vez las ocho constantes y sustituirlas en las ecuaciones 5.2 y 5.3 para cada instante de tiempo i.

Para implementar la respuesta dinámica al programa, se creó la clase *Respuesta1GL.dll*, que genera el cálculo de la respuesta de un oscilador de 1GL sujeto a cualquier tipo de excitación en su base.

La clase a través de su único constructor *Sub New* (código 5.1) recibe los siguientes parámetros para calcular la respuesta:

Tabla 5.1 Parámetros que necesita el constructor de la clase *Respuesta1GL*

T	Periodo en segundos
Xi	Amortiguamiento
Xo	Desplazamiento inicial
Vo	Velocidad inicial
DatosSeñal	Registro completo de la señal de excitación
MaxP	Número total de puntos de la señal
dt	Intervalo de tiempo

```
    Sub New(ByVal T As Double, ByVal Xi As Double, ByVal x0 As Double, ByVal v0 As Double,
ByVal DatosSeñal() As Double, ByVal MaxP As Integer, ByVal dt As Double)

        Me.Omega = (2 / T) * PI       'Calcula la frecuencia apartir del periodo
        Me.amort = Xi                 'Pasa el valor del amortiguamiento
        Me.Xo = x0                    'Indica el desplazamiento inicial
        Me.Vo = v0                    'Indica la velocidad inicial
        Me.OmegaD = Me.Omega * Math.Sqrt(1 - Me.amort ^ 2)  'cálculo de la omega D

        Me.MaxP = MaxP                'Número total de puntos de la señal de exitación
        Me.deltaT = dt                'Intervalo de tiempo

        Call variablesESC()                         'Invoca a la función
        Call ochoconstantesA()                      'Invoca a la función
        Call metodoJaramilloA(DatosSeñal, Me.MaxP)  'Invoca a la función
    End Sub
```

Código fuente 5.1 Constructor de la clase Respuesta1GL.dll

En el cuerpo del constructor (código 5.1), se calculan las variables *Omega* y *OmegaD*; además, se pasan los valores de los parámetros mencionados a las variables globales de la clase.

Posteriormente se invocan las funciones por medio de la instrucción *Call*, dichas funciones se encuentran en los códigos 5.2, 5.3 y 5.4.

El código fuente 5.2 calcula las ecuaciones 5.6 a 5.8, que serán utilizadas para calcular las ocho constantes.

```
    Private Sub variablesESC()
        Me.E = Exp(-Me.amort * Me.Omega * Me.deltaT)
        Me.S = Sin(Me.OmegaD * Me.deltaT)
        Me.C = Cos(Me.OmegaD * Me.deltaT)
    End Sub
```

Código fuente 5.2 Función para calcular las variables E, S y C

El código fuente 5.3 muestra la función que calcula las ocho constantes; ésta no recibe ningún parámetro pues los valores que necesita para calcular las constantes se declararon como variables globales dentro de la clase y dichas variables se les asignó valor en el código fuente 5.1.

Para la función se declararon algunas variables auxiliares (ax1, ax2, ax4, bx1, bx2, bx3, bx4 y bx5) que simplifican un poco el cálculo de algunas constantes. Dentro del código, las ocho constantes son: para desplazamiento ctes.a1, ctes.a2, ctes.a3 y ctes.a4 que corresponden a las ecuaciones 5.10 a 5.13 respectivamente y para velocidad ctes.b1, ctes.b2, ctes.b3 y ctes.b4 que corresponden a su vez a las ecuaciones 5.14 a 5.17.

```
Private Sub ochoconstantesA()
    Dim ax1, ax2, ax4, bx1, bx2, bx3, bx4, bx5 As Double

    '<<<<<<<<<< Desplazamiento  >>>>>>>>>>>>>>>>>

    ctes.a1 = E * ((amort * Omega * S / OmegaD) + C)
    ctes.a2 = E * S / OmegaD

    ax1 = ((amort * Omega) + ((2 * amort ^ 2) - 1) / deltaT) * S / OmegaD
    ax2 = (2 * amort) / (deltaT * Omega)
    ctes.a3 = ((ax1 + (1 + ax2) * C) * E - ax2) / Omega ^ 2

    ax4 = ((2 * amort ^ 2) - 1) * S / (OmegaD * deltaT)
    ctes.a4 = ((-ax4 - ax2 * C) * E - 1 + ax2) / Omega ^ 2

    '<<<<<<<<<< Velocidad  >>>>>>>>>>>>>>>>>

    ctes.b1 = (-1 * E * S * Omega) / Sqrt(1 - amort ^ 2)
    ctes.b2 = (((-amort * Omega * S) / OmegaD) + C) * E

    bx1 = amort * (-1 + amort * deltaT * Omega + 2 * amort ^ 2) / (Omega * OmegaD * deltaT)
    bx2 = (deltaT * Omega + 2 * amort) * OmegaD / (deltaT * Omega ^ 3)
    bx3 = amort * (deltaT * Omega + 2 * amort) / (deltaT * Omega ^ 2)
    bx4 = (-1 + amort * deltaT * Omega + 2 * amort ^ 2) / (deltaT * Omega ^ 2)
    bx5 = 1 / (Omega ^ 2 * deltaT)

    ctes.b3 = ((-bx1 - bx2) * S + (-bx3 + bx4) * C) * E + bx5
    ctes.b4 = (-1 + E * (amort * S / (Sqrt(1 - amort ^ 2)) + C)) / (deltaT * Omega ^ 2)
End Sub
```

Código fuente 5.3 Función para calcular las ocho constantes

Las ocho constantes también fueron declaradas como globales; por lo tanto, se pueden usar en cualquier parte del código de la clase.

El código fuente 5.4 contiene las ecuaciones para calcular la respuesta dinámica para cada instante de tiempo *i*. Esta función del tipo *Private sub* con nombre "metodoJaramilloA", recibe un sólo parámetro para calcular la respuesta: el vector completo de la excitación "acelerograma ( )".

```
Private Sub metodoJaramilloA(ByVal acelerograma() As Double)

    ReDim Me.X(Me.MaxP)
    ReDim Me.V(Me.MaxP)
    ReDim Me.Ar(Me.MaxP)
    ReDim Me.Aa(Me.MaxP)

    Me.X(1) = Me.Xo : Me.V(1) = Me.Vo
    Me.Ar(1) = -acelerograma(1) - 2 * amort * Omega * V(1) - Omega ^ 2 * X(1)
    Me.Aa(1) = Ar(1) + acelerograma(1)

    For Me.i = 2 To Me.MaxP
        Me.X(i) = ctes.a1 * X(i - 1) + ctes.a2 * V(i - 1) + ctes.a3 * acelerograma(i - 1) +
ctes.a4 * acelerograma(i)
        Me.V(i) = ctes.b1 * X(i - 1) + ctes.b2 * V(i - 1) + ctes.b3 * acelerograma(i - 1) +
ctes.b4 * acelerograma(i)
        Me.Ar(i) = -acelerograma(i) - 2 * amort * Omega * V(i) - Omega ^ 2 * X(i)
        Me.Aa(i) = Ar(i) + acelerograma(i)
    Next i
End Sub
```

Código fuente 5.4 Función para calcular la respuesta dinámica

Los 4 renglones que comienzan con la leyenda *Redim* indican que el programa dimensiona a los vectores de desplazamiento X, velocidad V, aceleración relativa Ar y absoluta Aa, que contendrán a la respuesta dinámica, con el mismo número de puntos que tiene la señal (MaxP).

Posteriormente se indican los valores para la respuesta en el instante i=1, en el caso de X (1) y V (1) corresponden a las condiciones iniciales (desplazamiento y velocidad respectivamente). Para las aceleraciones (relativa y absoluta) se calculan los valores en el mismo instante de tiempo i=1, utilizando las ecuaciones 5.4 y 5.5 respectivamente.

Para el cálculo de la respuesta dinámica a partir del segundo punto (i=2) hasta el número total de puntos N, se hace a con la ayuda de un bucle *for* que alberga las ecuaciones 5.2 a 5.5 para cada instante *i*; los valores que se calculan en cada instante i se van almacenando en los vectores que se definieron al inicio de la función.

Como se puede observar, el método es bastante práctico pues permite calcular la respuesta dinámica con pocas ecuaciones y a su vez el código de programación es pequeño.

Para obtener los vectores con la respuesta dinámica, se creó una función (código fuente 5.5) que recibe como parámetros 4 vectores vacíos del tipo *double* donde se guardará la respuesta dinámica. Lo que hace la función es pasar por referencia los vectores completos de la respuesta dinámica que se calculó en el código fuente 5.4.

```
Sub Resulatdos(ByRef respX() As Double, ByRef respV() As Double, ByRef respAr() As Double,
ByRef respAa() As Double)
        respX = Me.X
        respV = Me.V
        respAr = Me.Ar
        respAa = Me.Aa
End Sub
```

Código fuente 5.5 Función que regresa los vectores de respuesta

La clase cuenta con la función “CalculaMaximos” (código fuente 5.6) para obtener el valor máximo absoluto de cada tipo de respuesta (X, V, Ar y Aa); para esto el programa barre los vectores de cada respuesta y va calculando el máximo para cada una a través de un ciclo *for* y una sentencia *if* para cada tipo respuesta.

Los valores máximos se van almacenando en el vector “Max” de tipo *double* con 4 dimensiones, que corresponden a los tipo de respuesta: X, V, Ar y Aa.

A través de la función “ValoresMaximos” del tipo *ReadOnly Property* (mostrada en la parte baja del código fuente 5.6), la clase exporta los valores máximos calculados en la función “CalculaMaximos”.

```
    Private Sub CalculaMaximos()
        Max(1) = 0 : Max(2) = 0 : Max(3) = 0 : Max(4) = 0

        For i As Integer = 1 To Me.MaxP
            If Abs(Max(1)) < Abs(Me.X(i)) Then
                Max(1) = Me.X(i)
            End If

            If Abs(Max(2)) < Abs(Me.V(i)) Then
                Max(2) = Me.V(i)
            End If

            If Abs(Max(3)) < Abs(Me.Ar(i)) Then
                Max(3) = Me.Ar(i)
            End If

            If Abs(Max(4)) < Abs(Me.Aa(i)) Then
                Max(4) = Me.Aa(i)
            End If
        Next
    End Sub

'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
    Public ReadOnly Property ValoresMaximos()
        Get
            Call CalculaMaximos()
            Return Max
        End Get
    End Property
```

Código fuente 5.6 Función para obtener las respuestas máximas

Ahora bien, para llamar a la clase *Respuesta1GL* desde el programa principal, se hace con en el código fuente 5.7. Para esto se declara un objeto llamado “ObjCalculo” seguido de los parámetros (tabla 5.1) que necesita la clase para calcular la respuesta.

La declaración del objeto “ObjCalculo” se hizo dentro de un bucle *for*; esto es porque el programa calcula las respuestas de 1, 2 o 3 osciladores, dependiendo de cuantos osciladores defina el usuario.

Posteriormente, la variable “ObjCalculo” hace la referencia a la función *Resultados* (código fuente 5.5) y pasa como parámetros cuatro vectores (RX, RV, RAr y RAa) para recibir la respuesta dinámica. Cada vector es seguido del número de oscilador al que corresponde; por ejemplo RX1: respuesta de desplazamiento del oscilador 1.

La referencia a la función *Resultados* se hace dentro de una sentencia *Select Case* porque para cada oscilador se almacena la respuesta en vectores diferentes.

```
Private Sub CalculoRespuesta(ByVal Oscil As Integer)

  For n As Integer = 1 To Oscil
     Dim ObjCalculo As New CalculoRes1GL.Respuesta1GL(Propiedades(n, 1), Propiedades(n, 2),
Propiedades(n, 3), Propiedades(n, 4), DatosGráficaSeñal, maxpuntos, deltaT)

            Select Case n
                Case 1
                    ObjCalculo.Resulatdos(RX1, RV1, RAr1, RAa1)
                Case 2
                    ObjCalculo.Resulatdos(RX2, RV2, RAr2, RAa2)
                Case 3
                    ObjCalculo.Resulatdos(RX3, RV3, RAr3, RAa3)
            End Select

            Dim ValMax() As Double = ObjCalculo.ValoresMaximos
            For Me.i = 1 To 4
                RespMax(n, i) = ValMax(i)
            Next

     Call CalcularVbasal(n)
  Next n
End Sub
```

Código fuente 5.7 Uso de la clase en el programa

En el código fuente 5.7 también se puede observar que se hace referencia a la función *ValoresMaximos* para obtener la máxima amplitud de cada respuesta. Para esto, los valores se almacenan primero en un vector de tipo *double* (ValMax).

Posteriormente con la ayuda de un ciclo *for* se pasan los valores a una matriz bidimensional llamada *RespMax*; su tamaño fue determinado previamente con 5 columnas y tantas filas como osciladores haya definido el usuario.

5.1.2. Espectros de respuesta

Es un gráfico (figura 5.2) que muestra la respuesta máxima (expresada en términos de desplazamiento, velocidad o aceleraciones) que produce una acción dinámica determinada en varios osciladores de 1GL.

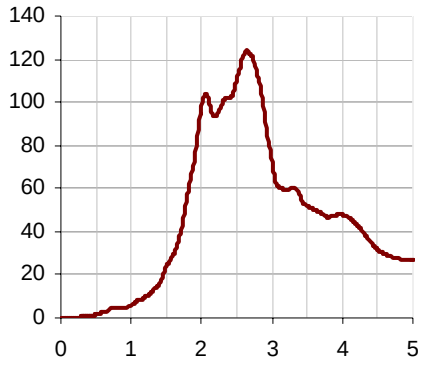

Figura 5.2 Espectro de respuesta

En el eje de las absisas (X) se encuentra el periodo propio de la estructura (o la frecuencia) y en las ordenadas (Y) la respuesta máxima calculada para cada periodo T del oscilador, con el mismo factor de amortiguamiento.

Los espectros se utilizan fundamentalmente para estudiar las características del terremoto y su efecto sobre las estructuras. Las curvas de los espectros presentan variaciones bruscas, con numerosos picos y valles, que resultan de la complejidad del registro de aceleraciones del terremoto. El concepto de espectro de respuesta es una importante herramienta de la dinámica estructural, de gran utilidad en el área de diseño sismorresistente. (Crisafulli y Villafañe *et. al* 2002)

El programa calcula sólo espectros de respuesta lineales, debido a que la respuesta que se obtiene es de osciladores elásticos lineales.

Para construir un espectro de respuesta es necesario calcular la respuesta dinámica de varios osciladores de 1GL con diferentes periodos de vibrar T y con igual factor de amortiguamiento. Para todos y cada uno de ellos se exhibirá una respuesta diferente.

Una vez calculada la respuesta de los osciladores, se determina el máximo (en valor absoluto, dado que el sigo no tiene importancia) de cada uno de ellos y se coloca en un gráfico en función del periodo de vibración, para obtener así un espectro de respuesta. Es decir, que la respuesta máxima de cada oscilador con periodo T representa un punto del espectro.

El programa calcula los espectros de respuesta con la ayuda de un control de tipo *Timer* llamado "TmEspectros"; el cual llama a la función "DatosPreliminaresRespuesta" para cada punto del espectro a intervalos de tiempo de 1 milisegundo.

El código fuente 5.8 muestra la función "DatosPreliminaresRespuesta", que entre otras instrucciones se encuentran el llamado a las funciones *CalculodeRespuesta* (código fuente 5.7) para determinar la respuesta dinámica, e inmediatamente después llama a la función *CrearMatrizEspectros* para construir el vector del espectro con los valores máximos de cada tipo respuesta.

```
Private Sub DatosPreliminaresRespuesta(ByVal gráficas As Integer, ByVal CalculaBloque As
Boolean)
……
…….
Call CalculoRespuesta(NumOscil)             'Calcula la respuesta del sistema
Call CrearMatrizEspectros(p, NumEspectros)  'Crea la matriz de espectros
……
…….
End Sub
```

Código fuente 5.8 Función DatosPrelimaresRespuesta

Obsérvese que el valor máximo de cada respuesta, para un valor de T, se calculó en el código fuente 5.7 y se guardó en la matriz "Respmax"; no obstante, esta matriz sólo puede almacenar las respuestas máximas de tres osciladores, por lo que no es suficiente para almacenar cada punto del espectro que puede tener hasta 500 datos por respuesta.

En este caso, se utiliza la función "CrearMatrizEspectros" (código fuente 5.9) que va guardando las respuestas máximas para cada cambio de valor del periodo T (punto del espectro) en la matriz "PuntoMax".

Se declaró a la matriz "PuntosMax" como una matriz de matrices de tres dimensiones. Este tipo de matrices tiene una ventaja importante con respecto a las matrices comunes, y es que sus elementos pueden ser también matrices o vectores, de ahí el nombre de matriz de matrices.

El programa puede calcular de 1 a 3 bloques de espectros; cada bloque contiene un grupo de 4 respuestas (D, V, Ar y Aa) y cada respuesta es un vector de datos que corresponden a los valores máximos de cada respuesta. Recuperar los datos almacenados en la matriz "PuntosMax" sería mucho más laborioso de no ser una matriz de matrices, ya en que de esta manera sólo se llama al vector completo que contiene a un espectro en específico.

```
    Private Sub CrearMatrizEspectros(ByVal n As Integer, ByVal BloqueEsp As Integer)
        For j = 1 To BloqueEsp
            PuntosMax(j)(1)(n) = Math.Abs(RespMax(j, 1)) 'Desplazamientos
            PuntosMax(j)(2)(n) = Math.Abs(RespMax(j, 2)) 'Velocidad
            PuntosMax(j)(3)(n) = Math.Abs(RespMax(j, 3)) 'Aceleración relativa
            PuntosMax(j)(4)(n) = Math.Abs(RespMax(j, 4)) 'Aceleración absoluta
        ……..
        …….
        Next
    End Sub
```

Código fuente 5.9 Creación de la matriz para espectros de respuesta

5.1.3. Fuerza cortante

Es la fuerza interna del oscilador que se genera por ejercer un desplazamiento horizontal en él. Una vez calculada la respuesta de desplazamiento $x(t)$ por el análisis dinámico del oscilador, las fuerzas internas pueden determinarse mediante un análisis estático para cada instante de tiempo *i*, a través de la ecuación 2.1 basada en el concepto de la fuerza estática equivalente.

$$f_S(t) = k \cdot x(t)$$

Para un oscilador simple la fuerza cortante basal (V_B) se puede determinar a partir de la ecuación 5.18. La figura 5.3 muestra la representación gráfica de las fuerzas Fs y V_B.

$$V_B(t) = f_S(t) \quad \text{.....(5.18)}$$

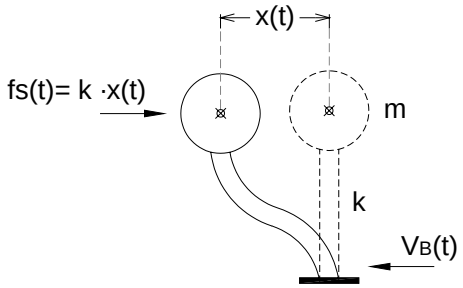

Figura 5.3 Fuerza cortante

En el código 5.7 hay una línea con la sentencia *Call CalcularVbasal(n)*, la cual invoca a la función para determinar la fuerza cortante; ésta se muestra en el código fuente 5.10.

```
    Private Sub CalcularVbasal(ByVal Osc As Integer)
        Select Case Osc
            Case 1
                ReDim Fs1(maxpuntos)       'Fuerza cortante del 1er oscilador
                For j = 1 To maxpuntos
                    Fs1(j) = K1GL(1) * RX1(j)
                Next j
            Case 2
                ReDim Fs2(maxpuntos)       'Fuerza cortante del 2do oscilador
                For j = 1 To maxpuntos
                    Fs2(j) = K1GL(2) * RX2(j)
                Next j
            Case 3
                ReDim Fs3(maxpuntos)       'Fuerza cortante del 3er oscilador
                For j = 1 To maxpuntos
                    Fs3(j) = K1GL(3) * RX3(j)
                Next j
        End Select
    End Sub
```

Código fuente 5.10 Calculo del cortante basal

La función, dentro del código fuente 5.10, requiere de un parámetro (número del oscilador) para calcular la fuerza cortante; dependiendo de este valor del parámetro (1, 2 o 3) el programa calcula la fuerza fs, con la ecuación 2.10, para cada instante de tiempo i.

5.1.4. Respuesta de osciladores de VGL

Al igual que los osciladores de 1GL, para encontrar la respuesta de un oscilador de VGL es necesario resolver la ecuación de equilibrio dinámico; para este caso se trata de la ecuación 2.20. Las características que tiene la señal excitación son las mismas: variar arbitrariamente con el tiempo y su registro debe estar discretizado en intervalos de tiempo pequeños.

Para el oscilador de VGL, descrito en la sección 2.3 del capítulo 2, las matrices de masas y rigidez se formulan de la siguiente manera:

$$m = \begin{bmatrix} m_1 & & & \\ & m_2 & & \\ & & \ddots & \\ & & & m_N \end{bmatrix} \qquad k = \begin{bmatrix} k_1 + k_2 & -k_2 & & \\ -k_2 & k_N + k_{N+1} & -k_{N+1} & \\ & -k_{N+1} & \ddots & -k_{N+1} \\ & & -k_{N+1} & k_N \end{bmatrix}$$

Debe notarse que la matriz de masa es diagonal para este sistema, sin considerar acoplamiento entre las masas; no obstante, en la matriz de rigidez se presentan acoplamientos entre los valores, lo que complica la solución de la ecuación de equilibrio dinámico.

La obtención de la respuesta dinámica de este oscilador se hizo a través del método de la superposición modal. Que consiste en modificar el sistema de referencia de tal forma que se pueda determinar la respuesta para cada grado de libertad como un sistema de 1GL

En el método de la superposición modal, la determinación de los eigenvectores es la fase más importante y que consume, relativamente, más tiempo dentro de la solución. (Pérez, 1990)

Uno de los métodos para obtener los eigen valores es el de la sub estructuración dinámica, que reduce el sistema de n grados de libertad a otro de menor tamaño, pero de tal manera que los vectores y valores característicos de baja frecuencia se conservan. (Pérez, 1990)

El oscilador de VGL (considerado en este trabajo) se razona como un sistema reducido de grados de libertad. En general, las estructuras tienen un gran número de grados de libertad, pero estos se pueden reducir de manera significativa de tal manera que el sistema se simplifica.

Una de las maneras de reducir el sistema de n grados de libertad, es por medio del método de Guyan, que reduce simultáneamente a las matrices de rigidez y masas del problema, en base a la selección de grados de libertad maestros; estos últimos son aquellos en los que se supone que está concentrada la masa de la estructura. Después de la reducción, sólo los grados de libertad maestros quedan representados en el oscilador (Pérez, 1990).

Este sistema parece razonable en estructuras como lo son los edificios regulares, pues a menudo se suponen las siguientes condiciones (Paz, 1992):

- Toda la masa de la estructura está concentrada al nivel de los pisos. Esto transforma el sistema, con un número infinito de grados de libertad a un sistema que tiene solamente tantos grados de libertad como número de masas concentradas.

- Las vigas en los pisos son infinitamente rígidas, con relación a la rigidez de las columnas. Esto introduce el requisito de que las uniones entre las vigas y las columnas estén fijas sin rotación.

- La deformación de la estructura es independiente de las fuerzas axiales presentes en las columnas. Establece que las vigas rígidas en los pisos permanezcan horizontales durante el movimiento de la estructura.

En este caso no se requieren más que número muy reducido de coordenadas para determinar, con mucha precisión, la respuesta de la estructura.

Por otro lado, el método de reducción de grados de libertad se vuelve incapaz de representar adecuadamente la respuesta de la estructura cuando se trata de una excitación que contenga componentes de alta frecuencia porque en general la configuración de las estructuras ante tal tipo de excitación es compleja y requiere en gran número de coordenadas para ser representada adecuadamente. (Pérez, 1990)

En el presente trabajo no se trata el tema de cómo reducir los grados de libertad de una estructura, sino que, para usar el programa, el usuario previamente debe haber reducido los grados de libertad de manera que la estructura se asemeje al oscilador de VGL descrito en esta tesis.

Para ello es necesario calcular las frecuencias y modos de vibrar (formas modales) del oscilador; el sistema de VGL tendrá tantas frecuencias como grados libertad.

La siguiente ecuación se obtiene de analizar el oscilador de VGL cuando está sujeto a vibración libre; en este caso no existen fuerzas externas y su amortiguamiento es considerado cero.

$$\left(\left[k\right]-\Omega^2\cdot\left[m\right]\right)\cdot\phi=0 \quad (5.19)$$

La ecuación 5.19 es un problema de valores característicos que tiene una solución no trivial sólo si el determinante de los coeficientes es igual a cero, es decir, las frecuencias naturales Ω y los modos de vibrar deben satisfacer la ecuación 5.20.

$$Det\left(\left[k\right]-\Omega^2\cdot\left[m\right]\right)=0 \quad (5.20)$$

El desarrollo del determinante (ecuación 5.20) conduce a un polinomio de grado n, las raíces del cual son los valores de las frecuencias del oscilador al cuadrado (Ω^2). Sustituyendo estos valores en la ecuación 5.19 se obtienen los valores para cada modo de vibrar (ϕ); estos pueden ser acomodados en forma matricial de la siguiente manera:

$$\Omega^2=\begin{Bmatrix}\Omega_1\\ \Omega_2\\ \vdots\\ \Omega_N\end{Bmatrix} \qquad \phi=\begin{bmatrix}\phi_{11} & \phi_{21} & \cdots & \phi_{N1}\\ \phi_{12} & \phi_{22} & \cdots & \phi_{N2}\\ \vdots & \vdots & \ddots & \vdots\\ \phi_{1N} & \phi_{2N} & \cdots & \phi_{NN}\end{bmatrix}$$

Con la matriz ϕ es posible hacer el cambio de coordenadas mediante la ecuación de la superposición modal:

$$x=\phi\cdot y \quad (5.21)$$

Por lo tanto las matrices de masas, rigidez y amortiguamiento quedan de la siguiente manera:

$$m_n=\left[\phi^T\right]\cdot\left[m\right]\cdot\left[\phi\right] \quad (5.22)$$
$$k_n=\left[\phi^T\right]\cdot\left[k\right]\cdot\left[\phi\right] \quad (5.23)$$
$$C_n=\left[\phi^T\right]\cdot\left[c\right]\cdot\left[\phi\right] \quad (5.24)$$

Sustituyendo estas ecuaciones por sus correspondientes en la ecuación 2.20, se obtiene:

$$\left[\phi^T\right]\cdot\left[m\right]\cdot\left[\phi\right]\cdot\ddot{y}+\left[\phi^T\right]\cdot\left[c\right]\cdot\left[\phi\right]\cdot\dot{y}+\left[\phi^T\right]\cdot\left[k\right]\cdot\left[\phi\right]\cdot y=-x_S\cdot\left[\phi^T\right]\cdot\left[m\right]\cdot\left\{j\right\} \quad (5.25)$$

Dividiendo la ecuación entre m_n y sustituyendo los factores de participación (Fp) se obtiene:

$$Fp=\frac{\left[\phi^T\right]\cdot\left[m\right]\cdot\left\{j\right\}}{\left[\phi^T\right]\cdot\left[m\right]\cdot\left[\phi\right]} \quad (5.26)$$

$$\ddot{y}(t)+2\xi\cdot\Omega\cdot\dot{y}(t)+\Omega^2\cdot y=-Fp\cdot\ddot{x}_S(t) \quad (5.27)$$

Con la ecuación 5.27 es posible resolver cada modo como si fuera un oscilador de 1GL; para resolver la ecuación de movimiento para cada modo, se aplica el método de las ocho constantes explicado en la sección 5.13.

Finalmente, para obtener la respuesta en las coordenadas originales del sistema se aplica de nuevo la ecuación de la superposición modal para cada tipo de respuesta (desplazamiento, velocidad y aceleración relativa respectivamente):

$$\{x_i(t)\} = \sum_{i=1}^{N} \{y_i(t)\} \cdot \phi_i \quad (5.28)$$

$$\{\dot{x}_i(t)\} = \sum_{i=1}^{N} \{\dot{y}_i(t)\} \cdot \phi_i \quad (5.29)$$

$$\{\ddot{x}_i(t)\} = \sum_{i=1}^{N} \{\ddot{y}_i(t)\} \cdot \phi_i \quad (5.30)$$

Para las condiciones iniciales también es necesario hacer el cambio de coordenadas, para ello se utilizan las siguientes ecuaciones:

$$\{y_0\} = [\phi]^{-1} \cdot \{x_0\} \quad (5.31)$$

$$\{\dot{y}_0\} = [\phi]^{-1} \cdot \{\dot{x}_0\} \quad (5.32)$$

El algoritmo se implementó al programa a través de un *Módulo* llamado "Cálculos" (código fuente 5.11); el cual contiene dos funciones declaradas como públicas y dos del tipo *private* (propias del módulo) para llevar a cabo el procedimiento descrito anteriormente.

```
Module Cálculos
    Public Sub ObtenerPropiedades(ByVal Ordenar As String)

    Public Sub ObtenerRespuesta(ByVal DatosGráficaSeñal() As Double, ByVal DeltaT As Single,
ByVal MaxPuntos As Integer)

    Private Sub CalculaVbasal(ByVal maxpuntos As Integer)

    Private Sub EnsambladoK(ByVal Numeracion As String)

End Module
```

Código fuente 5.11 Módulo para obtener la respuesta del oscilador de VGL

La función *Public Sub ObtenerPropiedades* (código fuente 5.12) calcula las propiedades del oscilador de VGL que se muestran en la tabla 5.2.

Tabla 5.2 Propiedades que se calculan para el oscilador de VGL

Ω^2	Frecuencias del sistema al cuadrado (ordenadas de menor a mayor)
ϕ	Matriz de valores característicos (formas modales)
Mn	Matriz de masa modal
Ln	Matriz de carga modal
Kn	Matriz de rigidez modal
qXo	Desplazamientos iniciales en forma modal
qYo	Velocidades iniciales en forma modal
Fp	Factores de participación
Ω	Frecuencias del sistema
T	Periodos del sistema

En las primera líneas del código fuente 5.12 se encuentra la declaración de las matrices involucradas, todas ellas comienzan con la leyenda *Dim* y en algunas casos *ReDim* que significa que esas matrices ya habían sido declaradas como tipo global y que ahí sólo se dimensionaron.

Después de las declaraciones de matrices el programa invoca a la función "EnsambladorK" (código fuente 5.13) pasando un parámetro de tipo *string* (para indicar la forma de numeración), esta función es la que se encarga de crear la matriz de rigidez dependiendo de la forma de numerar los grados de libertad en el oscilador (Arriba hacia abajo o viceversa).

```
Public Sub ObtenerPropiedades(ByVal Ordenar As String)
        Dim Mn(NGL, NGL) As Double    'Matriz de masa modal
        Dim Kn(NGL, NGL) As Double    'Matriz de rigidez modal
        Dim Ln(NGL, 1) As Double      'Matriz de carga modal
        Dim FiT(NGL, NGL) As Double   'Matriz Transpuesta de Fi
        ReDim qXo(NGL, 1)             'Matriz de desplazamientos iniciales Modal
        ReDim qVo(NGL, 1)             'Matriz de velocidades iniciales Modal
        Dim MatA(NGL, NGL) As Double  'Matriz auxiliar
        Dim FiInv(NGL, NGL) As Double 'Matriz inversa de Fi

        ReDim Tvgl(NGL)
        ReDim OmegasVGL(NGL)

        Call EnsambladoK(Ordenar)           'Ensambla las matrices de Rigidez
        Omegas2VGL = EigenVal(KVGL, MasaVGL) 'obtiene las frecuencias al cuadrado, ordenadas

        'Obtiene los eigenvectores normalizados
        Fi = EigenValores(NGL, KVGL, MasaVGL, Omegas2VGL, Ordenar)

        'Ahora pasamos a las coordenadas modales
        FiT = Transpuesta(Fi)
        MatA = MultMatriz(MasaVGL, Fi)
        Mn = MultMatriz(FiT, MatA)

        MatA = MultMatriz(KVGL, Fi)
        Kn = MultMatriz(FiT, MatA)

        MatA = MultMatriz(MasaVGL, Vj)
        Ln = MultMatriz(FiT, MatA)

        FiInv = Inversa(Fi, NGL)
        qXo = MultMatriz(FiInv, XoVGL)
        qVo = MultMatriz(FiInv, VoVGL)

        ReDim MatA(NGL, NGL)
        For i = 1 To NGL
            MatA(i, i) = 1 / (Mn(i, i)) 'inversa de la matriz de masa "Solo Para Masa"
        Next
        FP = MultMatriz(MatA, Ln)       'Factores de particiapción

        For i = 1 To NGL
            OmegasVGL(i) = Sqrt(Omegas2VGL(i))   'Frecuencias del sistema
            Tvgl(i) = 2 * pi / OmegasVGL(i)      'Periodos del sistema
        Next i
    End Sub
```

Código fuente 5.12 Función para calcular las propiedades del oscilador

Con la matriz de rigidez ensamblada y la matriz de masas (creada al introducir los datos) se llama a la función "EigenVal" que calcula las frecuencias de vibrar al cuadrado y guarda el vector en la matriz "Omegas2VGL".

Teniendo las matrices de masas, rigidez y frecuencias al cuadrado se procede a calcular la matriz de valores característicos ϕ, para esto se llama a la función "Eigenvalores ( )" pasando como parámetros el

número de grados de libertad, la matriz de masas, rigidez, y el vector de frecuencias al cuadrado, así como el parámetro de tipo *string* para indicar la forma de numeración de los grados de libertad.

Lo que hace la función es sustituir las Ω^2 en la ecuación 5.19 y despejar cada término para encontrar los valores de ϕ.

```
Private Sub EnsambladoK(ByVal Numeracion As String)
  ReDim KVGL(NGL, NGL)              'Se dimensiona la matriz de rigidez

  If Numeracion = "Arriba" Then 'Crea la matriz k con la numeración de ARRIBA-ABAJO
            KVGL(1, 1) = Rigidez(1)
            For i = 2 To NGL
                KVGL(i, i) = Rigidez(i) + Rigidez(i - 1)
                KVGL(i - 1, i) = -1 * Rigidez(i - 1)
                KVGL(i, i - 1) = -1 * Rigidez(i - 1)
            Next i
  ElseIf Numeracion = "Abajo" Then'Crea la matriz k con la numeración de ABAJO-ARRIBA
            For i = 1 To NGL - 1
                KVGL(i, i) = Rigidez(i) + Rigidez(i + 1)
                KVGL(i + 1, i) = -1 * Rigidez(i + 1)
                KVGL(i, i + 1) = -1 * Rigidez(i + 1)
            Next i
            KVGL(NGL, NGL) = Rigidez(NGL)
        End If
End Sub
```

Código fuente 5.13 Función para ensamblar la matriz de rigidez

Después de calcular la matriz ϕ, el programa calcula las matrices Mn y Kn correspondientes a las ecuaciones 5.22 y 5.23, Ln que corresponde a $\left[\phi^T\right]\cdot\left[m\right]\cdot\left\{j\right\}$ que posteriormente contribuye a calcular los factores de participación y también calcula las condiciones iniciales (ecuaciones 5.31 y 5.32).

Finalmente la función calcula frecuencias y periodos de vibrar del oscilador con las siguientes expresiones:

$$\Omega = \sqrt{\Omega^2}$$

$$T = \frac{2\pi}{\Omega}$$

Para calcular la respuesta del oscilador se invoca a la función pública “ObtenerRespuesta” (código fuente 5.14) pasando como parámetros: el vector de la excitación “DatosGráficaSeñal ()”, el intervalo de tiempo “DeltaT” y el número total de puntos de la señal “NP”. Las demás variables que necesita la función (Ω^2, ξ, y condiciones iniciales) fueron declaradas como globales, así que se pueden usar en cualquier parte del módulo.

Lo primero que hace la función (código fuente 5.14) es dimensionar las matrices (*Redim*) que almacenaran la respuesta de cada grado de libertad. Estas matrices fueron declaradas como matriz de matrices de dos dimensiones, es decir, que cada dato de la matriz es un vector de datos que corresponde a la respuesta de un grado de libertad en específico.

```
Public Sub ObtenerRespuesta(ByVal DatosGráficaSeñal() As Double, ByVal DeltaT As Single,
ByVal NP As Integer)
        ReDim auxD(NGL)
        ReDim auxV(NGL)
        ReDim auxAr(NGL)
        ReDim auxAa(NGL)

        'se llama a la clase respuesta, se le pasan los arguemtos incluyendo el del numero
        'de grado de libertad para indicar que se trata del sistema de VGL
        For i = 1 To NGL
            Dim calculo As New Respuesta(i, OmegasVGL(i), XhiVGL(i), qXo(i, 1), qVo(i, 1),
DatosGráficaSeñal, NP, DeltaT)
        Next i

        'Ahora las respuestas de cada grado de libertad se deben sumar alterandose por
        'la matriz Fi (vectores caracteristicos) y los Factores de participación
        'El número de nodos a superponer afectara solo en cuantos grados de libertad se
        'suman para obtener la respuesta final

        For i = 0 To NGL
            For j = 1 To NP
                For k = 1 To NMS
                    Dvgl(i)(j) += auxD(k)(j) * Fi(i, k) * FP(k, 1)
                    Vvgl(i)(j) += auxV(k)(j) * Fi(i, k) * FP(k, 1)
                    Arvgl(i)(j) += auxAr(k)(j) * Fi(i, k) * FP(k, 1)
                    Aavgl(i)(j) += auxAa(k)(j) * Fi(i, k) * FP(k, 1)

                Next k
            Next j
        Next i

        Call CalculaVbasal(MaxPuntos) 'Calcula el cortante Basal en cada grado
    End Sub
```

Código fuente 5.14 Función para calcular las propiedades del oscilador VGL

Después, el programa calcula la respuesta dinámica para cada grado de libertad con la ecuación 5.27. En este caso, la señal de excitación no se multiplicó por los factores de participación (FP) como se muestra en la ecuación, ya que por motivos de programación se decidió introducirlos cuando se calcula la respuesta completa. Esto no afecta el resultado de los cálculos ya que los factores de participación siguen siendo introducidos en la ecuación.

Para calcular la respuesta se creó un módulo de clase llamado “Respuesta” (código fuente 5.15), el cual calcula la respuesta dinámica de la misma manera que la clase *Respuesta1GL.dll*, la diferencia es que la clase “Respuesta” recibe un parámetro más (número de grado de libertad) y va guardando la respuesta en matrices especialmente creadas para ello (AuxD (), AuxV (), AuxAr () y AuxAa ()).

Observe que el código fuente 5.15 es similar al código del constructor de la clase “Respuesta1GL” (código fuente 5.1). Las funciones que son invocadas en el constructor de la clase “Respuesta” (código 5.15) son las mismas que se utilizaron anteriormente y se muestran en los códigos fuente 5.2, 5.3 y 5.4 respectivamente.

Al tener calculada la respuesta para cada grado de libertad lo que prosigue en el programa es superponer la respuesta por modos para encontrar la respuesta real en cada grado de libertad. Esto se hace con las ecuaciones 5.28, 5.29 y 5.30, multiplicando además por los FP.

En el código fuente 5.14 se observa la superpoción modal en la parte que muestra tres ciclos *for* anidados, indicando que la respuesta de cada grado de libertad se obtiene multiplicando la matriz auxiliar (Aux ()) correspondiente por la matriz ϕ, y por los factores de participación (FP).

```
Module Respuesta
     Sub New(ByVal grado As Integer, ByVal Frec As Double, ByVal Xi As Double, ByVal x0 As
Double, ByVal v0 As Double, ByVal DatosSeñal1() As Double, ByVal MaxP As Integer, ByVal dt
As Double)

        Me.Omega = Frec
        Me.amort = Xi
        Me.Xo = x0
        Me.Vo = v0
        Me.deltaT = dt
        Me.OmegaD = Me.Omega * Math.Sqrt(1 - Me.amort ^ 2)  'calculo de la omega D

        Call variablesESC()
        Call ochoconstantesA()
        Call metodoJaramilloA(DatosSeñal1, MaxP)

        auxD(grado) = X
        auxV(grado) = V
        auxAr(grado) = Ar
        auxAa(grado) = Aa
    End Sub
End Module
```

Código fuente 5.15 Módulo "Respuesta" para osciladores de VGL

Al mismo tiempo que se van calculando las respuestas modales para cada grado de libertad, estas se van adicionando dependiendo del número de modos a superponer (NMS) que el usuario haya definido.

De esta manera el programa calcula la respuesta dinámica de cada grado.

5.1.5. Espectros de piso

Un espectro de piso es un espectro de respuesta que se obtiene de analizar un oscilador de 1GL ubicado en un nivel de otro oscilador de VGL.

Para visualizar mejor el concepto de espectro de piso, en la figura 5.4 se muestra un oscilador de 1GL ubicado en el segundo nivel de la estructura; el oscilador puede ser un equipo o componente con su propio periodo de vibración.

Para dicho oscilador es posible determinar su espectro de respuesta tomando como excitación ya no el registro de aceleración en la base de la estructura, sino la historia de aceleraciones absolutas en el segundo nivel; al espectro obtenido se le conoce como espectro de piso.

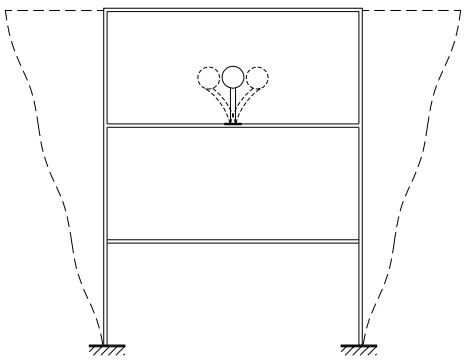

Figura 5.4 Esquema conceptual para espectros de piso

Obviamente, se debe de calcular previamente la respuesta dinámica de la estructura principal (como un oscilador de VGL) considerando como excitación el registro de aceleraciones en su base y posteriormente calcular el espectro de piso.

El análisis de espectro de piso es útil para equipos, componentes o alguna otra estructura pequeña que se ubique dentro del oscilador de VGL y que su masa sea menor en relación a las masas del oscilador de VGL ya que de lo contrario su masa podría interactuar con las demás masas de la estructura principal afectando la ecuación de equilibrio 2.13.

En la figura 5.5 se muestra una gráfica de espectros de piso superpuestos a la misma escala. Se puede observar que las aceleraciones en el equipo o componente a diseñar pueden ser significativamente mayores, dependiendo de su periodo de vibrar, especialmente cuando se acerca al periodo fundamental de la estructura.

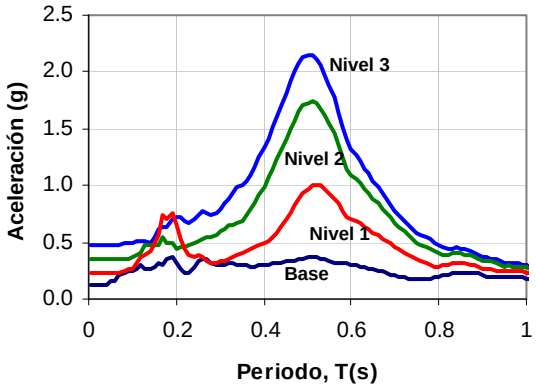

Figura 5.5 Espectros de piso para distintos niveles de la estructura

Los espectros de piso tienen las mismas características que los espectros de respuesta descritos en la sección 5.1.2.

La generación de los espectros de piso en el programa se hizo de manera similar a los espectros de respuesta; sólo que en este caso se requiere de más cálculos, pues el programa calcula cada espectro de piso para cada grado de libertad que tenga el oscilador de VGL.

El programa a través del control *Timer3* (código fuente 5.16), llama a la clase "Respuesta1GL" (código fuente 5.2) que calcula la respuesta para cada periodo T del espectro de respuesta de cada nivel del oscilador de VGL.

El código fuente 5.16 muestra al objeto "ObjCalculo" que hace referencia a la clase "Respuesta1GL.dll" y pasa los parámetros mostrados en la tabla 5.1; a diferencia de los espectros de respuesta aquí la señal de la excitación es el registro de la aceleración absoluta "Aavgl"

El programa, a través de los contadores "p" y "q1", controla los puntos del espectro y los grados de libertad del oscilador respectivamente. El contador p, tiene un rango de valores de 10 a 200 puntos, que son los puntos del espectro, y el contador q1 tiene rango de valores de 2 a 25, que es el rango de los grados de libertad que puede tener el oscilador de VGL.

Cuando los contadores llegan a sus valores límite, el programa manda un mensaje (con la sintaxis *MsgBox*) que indica que el cálculo de los espectros ha terminado.

```
    Private Sub Timer3_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Timer3.Tick
        If p < Me.ListaPuntosEsp.Length Then

            Dim ObjCalculo As New CalculoRes1GL.Respuesta1GL(Me.ListaPuntosEsp(p),
Me.P_Espectros(1, 1), Me.P_Espectros(1, 2), Me.P_Espectros(1, 3), Aavgl(q1), maxpuntos,
deltaT)
            ObjCalculo.ValoresMaximos(Me.PuntosEspPiso(q1)(1)(p),
Me.PuntosEspPiso(q1)(2)(p), Me.PuntosEspPiso(q1)(3)(p), Me.PuntosEspPiso(q1)(4)(p))
            p += 1
        Else
            If q1 < Me.NUDGL.Value Then
                q1 += 1
                p = 1
            Else
                Me.Timer3.Stop()
                q1 = 1
                p = 1
                MsgBox("Terminaron los cálculos de los Espectros de Piso",
MsgBoxStyle.Information, version)
            End If
        End If
    End Sub
```

Código fuente 5.16 Función para calcular los espectros de piso

5.1.6. Fuerza cortante

La fuerza cortante en el oscilador de VGL se calcula con la misma ecuación 2.1 que se usa en los osciladores de 1GL, sólo que para aplicarla se introduce el desplazamiento relativo entre las masas del oscilador.

La figura 5.6 muestra un esquema de cómo se toman los desplazamientos (X_1, X_2 y X_3) para calcular la fuerza cortante en cada grado de libertad. La fuerza Vbasal es el resultado de sumar todas las fuerzas cortantes (*fs*) que actúan al mismo tiempo en el oscilador de VGL.

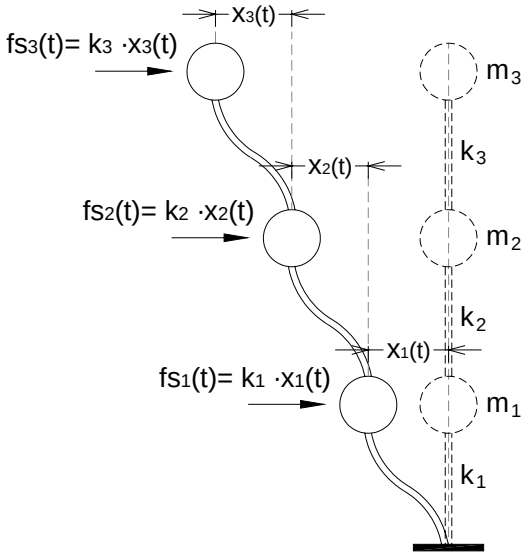

Figura 5.6 Fuerzas sísmicas en el oscilador de VGL

El programa calcula las fuerzas cortantes de cada instante de tiempo *i*, para cada grado de libertad, mediante la función "CalculaVbasal" (código fuente 5.17). Éste muestra dos ciclos *for* para calcular los desplazamientos relativos entre las masas (X_1, X_2 y X_3 mostrados en la figura 5.6) que se multiplican por el valor de la rigidez correspondiente de cada grado de libertad.

El resultado de las operaciones se almacena en la matriz "FuerzasV".

```
    Private Sub CalculaVbasal(ByVal maxpuntos As Integer)
        ReDim FuerzasV(NGL)

        For i = 1 To NGL - 1
            ReDim FuerzasV(i)(maxpuntos)
            For j = 1 To maxpuntos
                FuerzasV(i)(j) = (Dvgl(i)(j) - Dvgl(i + 1)(j)) * Rigidez(i)
            Next
        Next

        ReDim FuerzasV(NGL)(maxpuntos)
        For j = 1 To maxpuntos
            FuerzasV(NGL)(j) = Dvgl(NGL)(j) * Rigidez(NGL)
        Next
    End Sub
```

Código fuente 5.17 Función para calcular la fuerza cortante

5.2. GRÁFICAS

Son diagramas que representan datos numéricos por medio de una serie de puntos unidos mediante líneas para demostrar una conexión entre ellos.

El objetivo de las gráficas dentro de la interfaz gráfica de usuario es mostrar la respuesta dinámica, así como los dibujos de los osciladores para que el usuario vea el movimiento de éste en cada instante de tiempo debido a ejercer una señal excitación al sistema.

Para generar las gráficas en el programa fue necesario usar controles del tipo *PictureBox*, ubicarlos dentro del formulario y posteriormente dibujar sobre éste cada parte necesaria para formar las gráficas.

El programa muestra 3 tipos de gráficas: los dibujos de las señales (respuesta dinámica y excitación del sistema), el dibujo de los osciladores (1GL y VGL) y los dibujos de los espectros respuesta y de piso.

5.2.1. Señales

Para este tipo de gráficas se declaró una matriz de controles de tipo *PictureBox* de la siguiente manera:

```
Dim AreaGráfica(3) As New PictureBox
```

La matriz de controles permite optimizar el código, pues en vez de declarar una variable para cada gráfica podemos agrupar todas en una sola matriz. Se llama matriz de controles porque sus elementos son precisamente controles, en este caso controles del tipo *PictureBox*.

Para ambas ventana (osciladores de 1GL y VGL) se declaró una matriz de controles con el mismo nombre "AreaGráfica"; lo único que cambia entre ellas es el tamaño 3 y 4 elementos respectivamente.

Una vez declarados los *PictureBox*, se establecen las dimensiones mínimas (tabla 5.3) para cada control y posteriormente se ubican dentro del área de trabajo de cada ventana.

Tabla 5.3 Dimensiones mínimas (píxeles) de las gráficas en las ventanas principales

Ventana	Gráfica de carga		Gráfica de respuesta	
	Ancho	Alto	Ancho	Alto
Osciladores 1GL	670	120	670	95
Osciladores VGL	520	120	520	120

El área de trabajo tiene un sistema de coordenadas (x, y) ubicado en la parte superior izquierda de la ventana (figura 5.7); donde la dirección "y" positiva es hacia abajo y la dirección "x" positiva es hacia la derecha.

Para ubicar los controles dentro del formulario sólo se necesita especificar un solo punto (superior izquierdo) de cada gráfica. La ubicación del *PictureBox* para la gráfica de carga, en ambas ventanas, se encuentra en la coordenada: *x=10, y=10* (figura 5.7).

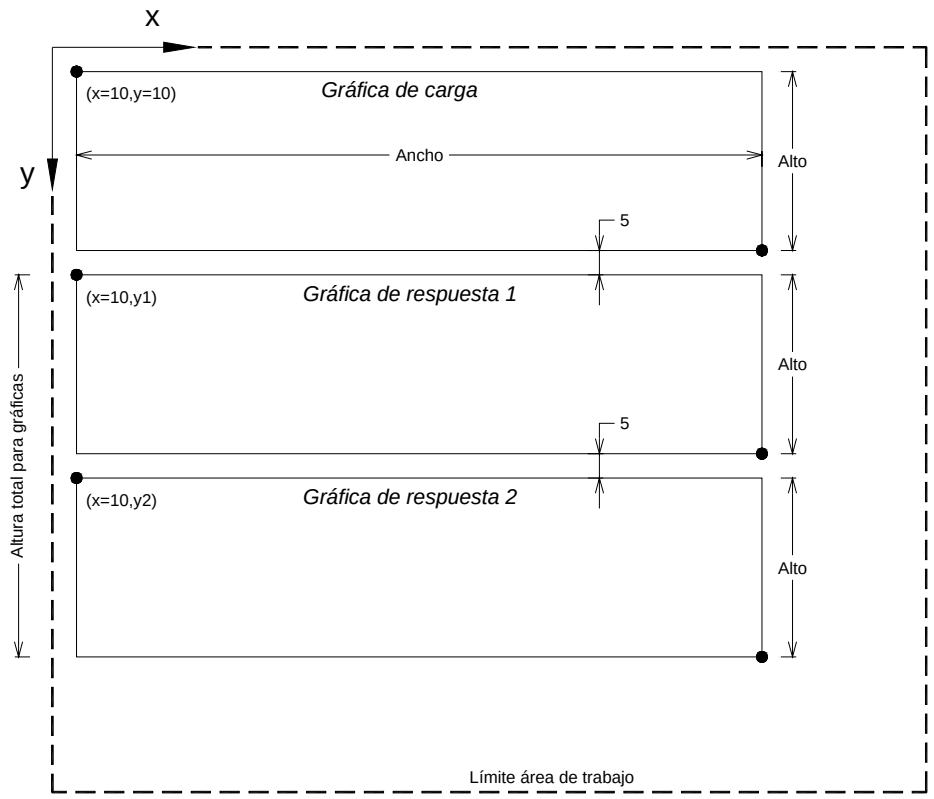

Figura 5.7 Ubicación de los *PictureBox* en el área de trabajo

Los *PictureBox* para la respuesta dinámica se ubican debajo de la gráfica de carga y tienen una separación entre ellos de 5 píxeles, su altura mínima se especifica en la tabla 5.7. Cuando el usuario cambia el número de gráficas que desea ver en pantalla, la altura de las gráficas se modifica ajustándose de manera que se distribuyen en todo lo alto de la "altura total para las gráficas" (figura 5.7) manteniendo la misma separación.

Las gráficas contenidas en el programa están formadas por los siguientes elementos:

- ✓ Escala numérica
- ✓ Fondo cuadriculado
- ✓ Pestaña
- ✓ Dibujo de la señal

Cada *PictureBox* se dividió, como se muestra en la figura 5.8, para albergar a cada elemento de la gráfica. La parte 1 se destinó a la escala numérica; la parte 2 corresponde (en su caso) a la pestaña de la gráfica de carga y por último la parte 3 contiene al dibujo de la señal correspondiente en un fondo cuadriculado.

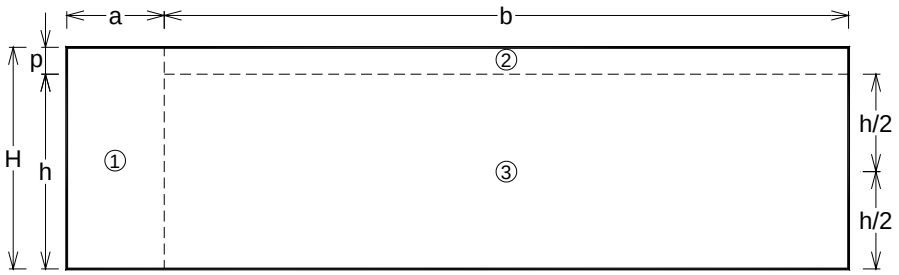

Figura 5.8 División de los *PictureBox* para las gráficas

Para dibujar cada parte de las gráficas dentro del *PictureBox*, se creó la clase *GráficarSeñal* (código fuente 5.18). Esta clase contiene 7 constructores definidos con la palabra clave *Sub New* y una función del tipo *Private*. Los constructores 1, 2, 3 y 7 dibujan la gráfica en forma estática y los constructores 4, 5 y 6 son para dibujar las señales cuando se encuentren en animación. Estos últimos se explican en la sección 5.3.

Los cuatro constructores mencionados (1, 2, 3 y 7) son similares entre si. Cada uno dibuja la gráfica con diferentes elementos; por ejemplo, el constructor 2 dibuja la gráfica con todos sus elementos, mientras que el constructor 1 omite el dibujo de la señal.

Cada constructor necesita de parámetros para generar la gráfica. La tabla 5.4 muestra el significado de cada parámetro que se ocupan en los constructores.

Tabla 5.4 Parámetros en los constructores

Variable	Descripción
pic	Nombre del *PictureBox* donde se va a dibujar la gráfica
texto	Título que se le va a colocar a la gráfica
Sx	Separación (píxeles) de la cuadrícula en dirección X
Sy	Separación (píxeles) de la cuadrícula en dirección Y
DatosSeñal() Puntos()	Vector de coordenadas (x,y) de la señal a dibujar
FE / FEV	Factor de escala vertical
Tiempo	Etiqueta para mostrar la duración de la señal

ColorLapiz	Color de la línea que representa a la señal
Grosor	Espesor (píxeles) de la línea que representa a la señal
Btmp	*Bitmap* que va unido a cada gráfica
X1, Y1	Coordenadas (x,y) del punto inicial para dibujar una línea
X2, Y2	Coordenadas (x,y) del punto final para dibujar una línea

```
Public Class GráficarSeñal

'Constructor 1 Dibuja la gráfica con el fondo, sin la señal y con la pestaña
Sub New(ByVal pic As PictureBox, ByVal texto As String, ByVal Sx As Single, ByVal Sy As
Single)

'Constructor 2 Dibuja la gráfica con el fondo, la señal y la pestaña
Sub New(ByVal pic As PictureBox, ByVal texto As String, ByVal Sx As Single, ByVal Sy As
Single, ByVal DatosSeñal() As PointF, ByVal FEV As Double, ByVal Tiempo As String)

'Constructor 3 Dibuja la gráfica con el fondo, la señal y sin la pestaña
Sub New(ByVal pic As PictureBox, ByVal texto As String, ByVal Sx As Single, ByVal Sy As
Single, ByVal FE As Double, ByVal puntos() As PointF, ByVal ColorLapiz As Color, ByVal
Grueso As Integer, ByRef Btmp As Bitmap)

'Constructor 4 Dibuja la gráfica unicamente con la señal, sin fondo, sin pestaña, para la
animación en la parte de los espectros
Sub New(ByVal pic As PictureBox, ByVal puntos() As PointF, ByVal ColorLapiz As Color, ByVal
Grueso As Integer)

'Constructor 5 Dibuja la gráfica con el fondo, sin la señal, sin la pestaña antes de cada
animación de la señal
Sub New(ByVal pic As PictureBox, ByVal texto As String, ByVal Sx As Single, ByVal Sy As
Single, ByVal FE As Double, ByRef Btmp As Bitmap)

'Constructor 6 Dibuja la gráfica con la señal por pasos, sin la pestaña y sin el fondo
Sub New(ByVal pic As PictureBox, ByRef Btmp As Bitmap, ByVal X1 As Single, ByVal Y1 As
Single, ByVal X2 As Single, ByVal Y2 As Single, ByVal ColorLapiz As Color, ByVal Grueso As
Integer)

'Constructor 7 Dibuja la gráfica con el fondo, sin la señal y sin la pestaña
 Sub New(ByVal pic As PictureBox, ByVal Sx As Single, ByVal Sy As Single)

'Función del tipo Privada para dibujar el fondo cuadriculado en las gráficas
Private Sub dibujafondo(ByVal Pic As PictureBox, ByVal mapa As Graphics, ByVal Sx As Single,
ByVal Sy As Single, ByVal CoordY As Integer)

End Class
```

Código fuente 5.18 Clase *GráficarSeñal*

Para que el dibujo de la señal se vea adecuadamente dentro de los *PictureBox*, se definieron factores de escala vertical y horizontal, que se calculan de la siguiente manera:

$$Fev = \frac{h/2}{A\max} = \frac{h}{2A\max}$$

$$Feh = \frac{b}{Duración}$$

Donde:

Fev = Factor de escala vertical
Feh = Factor de escala horizontal
*A*max = Amplitud máxima de la señal en valor absoluto

Los factores de escala multiplican a cada uno de los valores de la señal (x, y) para que se ajusten al tamaño que les corresponde en los *PictureBox*. Para calcularlos es necesario tener primero los datos de cada señal que se va a dibujar.

El código fuente 5.19 muestra el contenido del constructor 2, que dibuja la gráfica con todos sus elementos. El código se encuentra definido dentro de las palabras clave *Sub new* y *End Sub.*

Para dibujar sobre un *PictureBox*, es necesario primero crear un área de dibujo relacionada a él; esto se hace con la ayuda de un mapa de bits (*Bitmap*) y se conectan entre si como se muestra en las primeras 4 líneas dentro del constructor en el código 5.19.

Después, se calculan las variables *Posicionmedia, MitadAltura y Numlineas* que sirven para conocer la distancia vertical donde se va a dibujar la línea de referencia de color rojo que se muestra en las gráficas y a partir de ésta dibujar la cuadrícula de fondo y el dibujo de la señal.

La siguiente línea en el código invoca a la función *dibujafondo* (código 5.20) que es la encargada de dibujar el fondo cuadriculado de color gris en la gráfica; la cual al ser del tipo *Private* sólo puede ser usada dentro de la clase *GráficarSeñal*.

Posteriormente se tienen las instrucciones (*mapa.Drawlines*) para dibujar por completo la señal correspondiente a la gráfica y la línea horizontal de referencia color rojo.

```
'Constructor 2 Dibuja la gráfica con el fondo, la señal y la pestaña
Sub New(ByVal pic As PictureBox, ByVal texto As String, ByVal Sx As Single, ByVal Sy As
Single, ByVal DatosSeñal() As PointF, ByVal FEV As Double, ByVal Tiempo As String)

Dim btmp1 As New Bitmap(pic.Width, pic.Height) 'Crea un Bitmap del tamaño del pictureBox
Dim mapa As Graphics = pic.CreateGraphics()    'Crea un área de trabajo a partir del Pb
pic.Image = btmp1                              'La imagen del PictureBox será el Bitmap
mapa = Graphics.FromImage(btmp1)               'Permite un buen efecto de animación
                                               'en cada cambio de dibujo

Posicionmedia = pic.Height / 2 + 10   'Valor de la coordenada en Y de la línea de referencia
MitadAltura = (pic.Height - 20) / 2   'Valor de la mitad de la altura libre para dibujar
NumLineas = CInt(MitadAltura \ Sy)    'Número de líneas a dibujar en la cuadrícula

Call dibujafondo(pic, mapa, Sx, Sy, 20) 'Dibuja el fondo con los valores calculados

mapa.DrawLines(lapiz1, DatosSeñal)      'Dibuja la señal
mapa.DrawLine(lapiz2, CoordX, Posicionmedia, pic.Width, Posicionmedia) 'línea de y= 0

Dim objetoPESTAÑA As New Pestaña(mapa, pic, CoordX) 'Dibuja la pestaña sobre el PictureBox

Dim objetoTEXTO2 As New TextoHorizontal(NumLineas, CInt(Posicionmedia), Sy, FEV, mapa, "t =
" & Tiempo, pic.Width - 70)            'Dibuja la escala númerica

mapa.DrawString(texto, fuenteN, Brushes.Black, CoordX + 5, 5)'Dibuja el título de la gráfica
End Sub
```

Código fuente 5.19 Constructor 2 de la clase *GráficarSeñal*

Para colocar la pestaña y la escala numérica en la gráfica se crearon dos clases auxiliares: la clase *Pestaña* (código fuente 5.21) y la clase *Textohorizontal* (código fuente 5.22) respectivamente. En el código fuente 5.19 se muestra cómo se utilizaron los objetos (*objetoPestaña y objetoTEXTO2*) para usar las clases *Pestaña y Textohorizontal*, respectivamente.

Por último, en el constructor 2 de la clase *GráficarSeñal* se indica mediante la instrucción *mapa.DrawingString* que se coloque el título de la gráfica.

```
Private Sub dibujafondo(ByVal Pic As PictureBox, ByVal mapa As Graphics, ByVal Sx As Single,
ByVal Sy As Single, ByVal CoordY As Integer)
        Dim respuesta As Boolean

        With mapa
            .FillRectangle(Brushes.Transparent, 0, 0, Pic.Width, Pic.Height)
            .FillRectangle(Brushes.White, 53, 0, Pic.Width - 53, Pic.Height)

            .FillRectangle(relleno, CoordX, CoordY, Pic.Width - CoordX, Pic.Height - CoordY)
            .DrawRectangle(Pens.Black, CoordX, CoordY, Pic.Width - (CoordX + 1), Pic.Height
- (CoordY + 1))

            'malla vertical
            For Me.i = 1 To Pic.Width
                .DrawLine(lapiz, CoordX + i * Sx, CoordY, CoordX + i * Sx, Pic.Height)
                If (CoordX + (i + 1) * Sx) > Pic.Width Then
                    Exit For
                End If
            Next

            'malla horizontal parte baja
            For Me.i = NumLineas To 1 Step -1
                .DrawLine(lapiz, CoordX, Posicionmedia + i * Sy, Pic.Width, Posicionmedia +
i * Sy)
            Next

            'malla horizontal parte alta
            For Me.i = NumLineas To 1 Step -1
                .DrawLine(lapiz, CoordX, Posicionmedia - i * Sy, Pic.Width, Posicionmedia -
i * Sy)
            Next
  End Sub
```

Código fuente 5.20 Función *dibujaFondo*

La clase *Pestaña* (código fuente 5.21) tiene un solo constructor y no cuenta con más funciones de ningún tipo, por lo tanto todas sus instrucciones se encuentra dentro de él. Primero, se definen cuatro variables (*relleno, relleno2, lápiz y lapiz2*) para almacenar los colores de relleno y el color de la línea de contorno.

Después, se definen dos vectores (*puntos y puntos2*) para contener las coordenadas de la poligonal de la pestaña; ésta se dibujará a todo lo ancho del *PictureBox* con una altura definida de 10 píxeles. Por último, las instrucciones (*FillPolygon* y *DrawPolygon*) indican que se dibuje el polígono con un color de relleno y su contorno de otro color.

La clase *TextoHorizontal* (código fuente 5.22) también cuenta con un solo constructor. Primero, se especifica la distancia vertical de la línea que marca la amplitud cero de cada gráfica; en seguida establece el tipo de letra, formato y color que tendrá la escala numérica. Después con la ayuda de un bucle *for* calcula y coloca los textos de la escala numérica correspondientes a la cuadrícula de la gráfica creada anteriormente. Por último, coloca un texto que indica la duración de la señal dentro de la pestaña en la parte superior derecha.

```
Public Class Pestaña
 Sub New(ByVal mapa As Graphics, ByVal pic As PictureBox, ByVal X1 As Integer)

  Dim relleno As New SolidBrush(Color.FromArgb(50, Color.Yellow)) 'Color de relleno
  Dim lapiz As New Pen(Color.FromArgb(250, Color.DarkRed), 1)     'Color de línea
  Dim relleno2 As New SolidBrush((SystemColors.Control))          'Color de relleno
  Dim lapiz2 As New Pen(SystemColors.Control, 1)                  'Color de línea

  Dim puntos() As Point = {New Point(X1, 10), New Point(X1 + 10, 0), New Point(pic.Width -
1, 0), New Point(pic.Width - 1, 20), New Point(X1, 20)}  'Coordenadas del poligono

  'Coordenadas del triangulo en la esquina superior izqueirda
  Dim puntos2() As Point = {New Point(X1, 0), New Point(X1, 10), New Point(X1 + 10, 0)}

  mapa.FillPolygon(relleno2, puntos2)  'Dibujo del poligono relleno
  mapa.DrawPolygon(lapiz2, puntos)     'Dibujo del contorno del poligono
  mapa.FillPolygon(relleno, puntos)    'Dibujo del triangulo relleno
  mapa.DrawPolygon(lapiz, puntos)      'Dibujo del contorno del triangulo

  End Sub
End Class
```

Código fuente 5.21 Clase *Pestaña*

```
Public Class TextoHorizontal
    Dim Yo As Integer

    Sub New(ByVal Numlineas As Integer, ByVal Yo As Integer, ByVal Sy As Single, ByVal
FactEscalaV As Double, ByVal mapa As Graphics, ByVal Titulo As String, ByVal PosX As
Integer)

        Me.Yo = Yo - 7     'Altura inicial para comenzar a colocar los textos

        Dim texto As String                     'Variable que recibe la etiqueta
        Dim fuente As New Font("Arial", 8)                  'Tipo de letra
        Dim fuenteN As New Font("Arial", 8, FontStyle.Bold) 'Tipo de letra en Negrita
        Dim brocha As New SolidBrush(Color.Black)           'Color del texto
        Dim x As Single = CoordX - 3                        'Posición en X
        Dim formato As New StringFormat                     'Variable para formato

        'Se define formato de escritura de derecha a izquierda
        formato.FormatFlags = StringFormatFlags.DirectionRightToLeft

        'Se escribe el texto "0.0" en la mitad de la gráfica
        mapa.DrawString(texto, fuenteN, brocha, x, Me.Yo, formato)

        For i = 1 To Numlineas                  'Ciclo para colocar los demás textos
            If FactEscalaV = 0 Then             'En caso de que el Factor de escala sea 0
                texto = "" & Format(0, "0.00E+00") 'Coloca texto "0"
            Else                                'En caso contrario
                texto = "" & Format(i * Sy / FactEscalaV, "0.00E+00")'Calcula las escalas
            End If

           'Coloca los textos en la escala númerica
            mapa.DrawString(texto, fuente, brocha, x, Me.Yo - i * Sy, formato) 'arriba
            mapa.DrawString(texto, fuente, brocha, x, Me.Yo + i * Sy, formato) 'abajo
        Next i

        mapa.DrawString(Titulo, fuenteN, Brushes.Black, PosX + 5, 5) 'Duración de la señal

    End Sub
```

Código fuente 5.22 Clase *TextoHorizontal*

5.2.2. Osciladores

El procedimiento para dibujar los osciladores es similar al descrito anteriormente para las gráficas de las señales. Cada oscilador fue dibujado en con la ayuda de dos *PictureBox* superpuestos uno encima del otro, es decir, un dibujo por capas. En un *PictureBox* se dibujó el fondo del oscilador mientras que en el otro se dibujó el sistema de masas y resortes. Esta manera de dibujar fue muy útil en el proceso de animación del oscilador, que se explica en la sección 5.3.2.

5.2.2.1. Osciladores de 1GL

Para la ventana de osciladores de 1GL se declaró un par de matrices de controles del tipo *PictureBox* con el nombre "AreaOscilador" y "CapaOscilador"; la primera para el dibujo del fondo y la segunda para el oscilador (masa y resorte). La dimensión de ambas matrices es de 3 elementos, ya que en esta ventana se muestran hasta 3 osciladores al mismo tiempo.

```
Dim AreaOscilador(3) As New PictureBox
Dim CapaOscilador(3) As New PictureBox
```

Las dimensiones de los osciladores son fijas: 200 x 200 píxeles; éstas no se modifican aun cuando el usuario seleccione la opción de ajustar las gráficas al tamaño de la pantalla.

La ubicación de los osciladores es diferente para cada ventana. En el caso de los osciladores de 1GL los *PictureBox* se encuentran en la parte baja de la ventana, 20 píxeles debajo de las gráficas de respuesta. La separación lateral máxima entre ellos es de 40 píxeles y la mínima de 8 píxeles. La figura 5.9 muestra su ubicación dentro de la ventana.

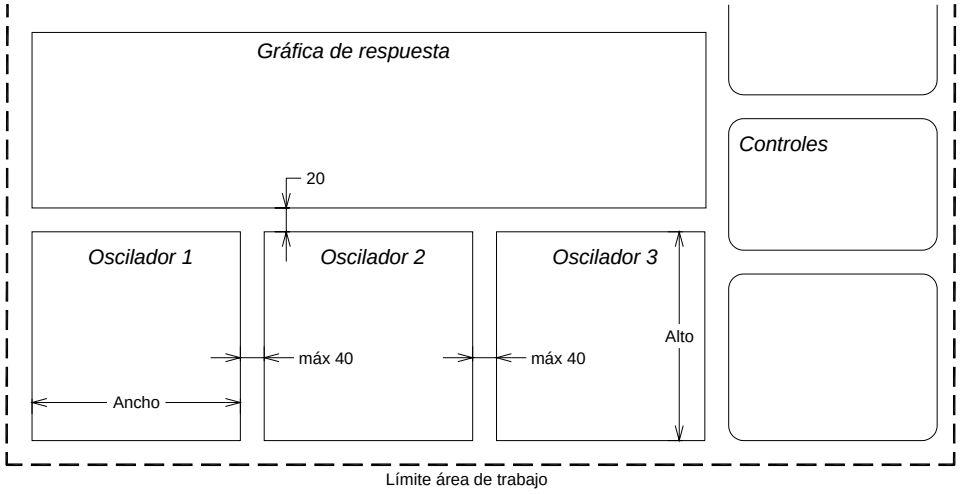

Figura 5.9 Ubicación de los osciladores de 1GL en la ventana

Cada gráfica de los osciladores consta de 3 elementos:

- ✓ Fondo cuadriculado
- ✓ Pestaña
- ✓ Dibujo del oscilador según corresponda

De igual forma que en las gráficas de las señales, los *PictureBox* (variables "AreaOscilador") se dividieron para dibujar por partes las gráficas de los osciladores (figura 5.10). La parte 1 corresponde,

para todos los osciladores, a la pestaña de la gráfica y la parte 2 contiene al dibujo del fondo cuadriculado, similar al que se encuentra en las gráficas de las señales.

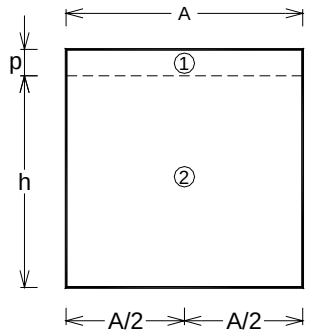

Figura 5.10 División de los *PictureBox* para las osciladores

Para dibujar los elementos de las gráficas fue necesario crear la clase "fondoOscilador" (código fuente 5.23).

Esta clase tiene 3 constructores y 6 funciones del tipo *Private*. El constructor 1 dibuja en el *PictureBox* dos líneas verticales que representan hasta dónde llegará el desplazamiento máximo de la masa. El constructor 2 dibuja sólo la masa y el resorte dependiendo del valor de la posición vertical que se indique y por último el constructor 3 dibuja el fondo cuadriculado de la gráfica; éste es muy parecido al mostrado en el código fuente 5.20.

```
Public Class FondoOscilador

'Constructor 1  Dibuja las líneas de escala horizontal del Oscilador
Sub New(ByVal pic As PictureBox, ByVal Sx As Integer, ByVal Sy As Integer, ByVal tamaño As
Integer, ByVal ColorCentro As Color, ByVal ColorExt As Color, ByVal ValorX As Integer)

'constructor 2 Dibuja SOLO los osciladores
Sub New(ByVal pic As PictureBox, ByVal tamaño As Integer, ByVal ColorCentro As Color, ByVal
ColorExt As Color, ByVal Posx As Double, ByVal N As Integer, ByVal despl As Double)

'constructor 3 Dibuja SOLO el fondo
Sub New(ByVal pic As PictureBox, ByVal texto As String, ByVal Sx As Integer, ByVal Sy As
Integer)


Private Function CreaAreaDibujo(ByVal Pic As PictureBox) As Graphics

Private Sub dibujafondo(ByVal mapa As Graphics)

Private Sub dibujaPestaña(ByVal mapa As Graphics, ByVal Texto As String)

Private Sub dibujamasa(ByVal mapa As Graphics, ByVal X As Double, ByVal Tamaño As Integer)

Private Sub dibujaColumna(ByVal mapa As Graphics, ByVal X As Double, ByVal Diam As Integer)

Private Sub dibujaLineasEscala(ByVal mapa As Graphics, ByVal tamaño As Integer)

End Class
```

Código fuente 5.23 Clase *fondoOscilador*

La tabla 5.5 muestra el significado de cada parámetro que se requieren introducir en cada constructor de la clase *fondoOscilador*.

Tabla 5.5 Parámetros en los constructores

Variable	Descripción
pic	Nombre del *PictureBox* donde se va a dibujar la gráfica
Sx	Separación (píxeles) de la cuadrícula en dirección X
Sy	Separación (píxeles) de la cuadrícula en dirección Y
Tamaño	diámetro de la masa en píxeles
ColorCentro	Color del centro de la masa
ColorExt	Color del relleno de la masa
ValorX	Distancia (píxeles) horizontal de las líneas verticales al centro del *PictureBox*
PosX	Posición (píxeles) de la masa dentro del *PictureBox*
N	Número de punto de la señal de desplazamiento
despl	Valor del desplazamiento de la masa que corresponde al punto N
Texto	Título de la gráfica

El código fuente 5.24 contiene al constructor 2, que dibuja solamente la masa y resorte de cada oscilador. El código se encuentra definido dentro de las palabras clave *Sub new* y *End Sub*.

En las primeras líneas del código, el constructor pasa los valores recibidos a las variables globales, después llama a la función "CrearAreaDibujo" (código fuente 5.25) para establecer un espacio sobre el cual se dibujan cada elemento de la gráfica y lo almacena en la variable "mapa". Posteriormente se indica que se coloque un texto con el valor del deplazamiento de la masa en un instante N, que corresponde a un punto de la señal. Por último, el constructor llama a dos funciones "dibujaColumna" y "dibujamasa" (código fuente 5.26 y 5.27) para dibujar la masa y columna respectivamente.

```
'constructor 2 Dibuja SOLO los osciladores
Sub New(ByVal pic As PictureBox, ByVal tamaño As Integer, ByVal ColorCentro As Color, ByVal
ColorExt As Color, ByVal Posx As Double, ByVal N As Integer, ByVal despl As Double)
        Me.pic = pic                    'Pasa el picturebox a la variable global de la clase
        Me.ColorCentro = ColorCentro                'Color del centro de la masa
        Me.ColorExt = ColorExt                      'Color del exterior de la masa
        Me.Posx = (pic.Width / 2) + Posx            'Posición en X del centro de la masa

        mapa = CreaAreaDibujo(pic)      'crea el área de dibujo a partir del PictureBox

        'Escribe el instante y el valor del desplazamiento en ese instante
        mapa.DrawString("D[" & N & "]= " & Format(despl, "0.0E+00"), fuenteN,
Brushes.DarkRed, pic.Width - 5, 4, formato)

        Call dibujaColumna(mapa, Me.Posx, tamaño)     'Dibuja la columna
        Call dibujamasa(mapa, Me.Posx, tamaño)        'Dibuja la masa
End Sub
```

Código fuente 5.24 Constructor 2 de la clase *fondoOscilador*

El cuerpo de la función para establecer el área gráfica (código fuente 5.25) es el mismo que se encuentra en las primeras cuatro instrucciones del constructor del código fuente 5.19.

Se define primero un mapa de bits (*Bitmap*) y se conecta con el *PictureBox*, por medio de la propiedad *Image* del control.

```
Private Function CreaAreaDibujo(ByVal Pic As PictureBox) As Graphics
   Dim Btmp1 As New Bitmap(Pic.Width, Pic.Height) 'Crea un Bitmap del tamaño del pictureBox
   Dim mapa As Graphics = Pic.CreateGraphics()    'Crea un área de trabajo a partir del
                                                   PictureBox
   Pic.Image = Btmp1                              'La imagen del PictureBox será el Bitmap
   mapa = Graphics.FromImage(Btmp1)               'Permite un buen efecto de animación en
                                                   cada cambio de dibujo
   Return mapa                                    'Regresa la variable mapa
End Function
```

Código fuente 5.25 Función para asignar el área gráfica

El código fuente 5.26 se encarga de dibujar únicamente la masa de cada oscilador; para ello se requiere establecer el diámetro de la masa en píxeles y su posición dentro del *PictureBox*. Cuando los osciladores permanecen estáticos el valor "X" de la posición es la mitad del ancho del *PictureBox*.

La función define un círculo estableciendo su ubicación y degradación de colores (del centro hacia el exterior) y agregando un contorno de color negro del mismo tamaño que el diámetro de la masa.

```
Private Sub dibujamasa(ByVal mapa As Graphics, ByVal X As Double, ByVal Tamaño As Integer)

        Dim trayecto As GraphicsPath = New GraphicsPath  'variable para dibujar el degradado
        Dim rectangulo As New RectangleF(X - (Tamaño / 2), 50, Tamaño, Tamaño)
        trayecto.AddEllipse(rectangulo)                  'Estable el límite del circulo

        Dim brocha As PathGradientBrush = New PathGradientBrush(trayecto) 'Inica gradiente

        brocha.CenterPoint = New PointF(X, (50 + Tamaño / 2)) 'Ubicación del centro de la m
        brocha.CenterColor = ColorCentro                      'Establece el color del centro
        brocha.SurroundColors = New Color() {ColorExt}        'Establece el color exterior

        mapa.FillPath(brocha, trayecto)          'Dibuja el relleno con degradación de color
        mapa.DrawEllipse(lapiz1, rectangulo)     'Dibuja el contorno de la masa
End Sub
```

Código fuente 5.26 Función para dibujar la masa

La columna se dibujó como una línea curva con tres puntos de control: inicio, centro y final. En la figura 5.11a se muestra la ubicación de estos tres puntos (p1, p2 y p3 respectivamente). La altura máxima de la columna es de 95 píxeles y depende del diámetro de la masa que especifique el usuario.

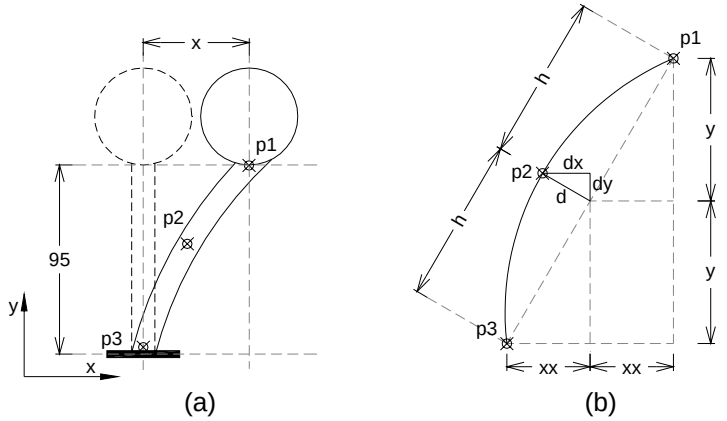

Figura 5.11 Puntos de control para dibujar la columna

La función que dibuja la columna (código fuente 5.27) establece primero la posición de los puntos 1 y 3 que se encuentran en la parte baja de la masa (p1) y la base de la columna (p3).

Después, para determinar la posición del punto intermedio de la curva (p2), se calculan las distancias "dx" y "dy" con la relación de triángulos semejantes a partir de las distancias "xx", "y" y "h" (mostradas en la figura 5.11b); la distancia "d" (perpendicular a la línea recta entre los puntos p1 y p3) se toma como el 20% de la distancia "xx".

Las coordenadas de los tres puntos cambian conforme el desplazamiento del oscilador va cambiando también; esto sucede cuando la animación del oscilador está en ejecución. Mientras el oscilador permanezca en posición estática (desplazamiento=0) la columna se dibujará de manera vertical, pues la coordenada "x" de los tres puntos es la misma.

La instrucción para dibujar la columna es "mapa.DrawCurve" la cual recibe como parámetros el formato de la línea (espesor de 8 píxeles y color igual al color exterior de la masa) y las coordenadas de los puntos mencionados.

```
Private Sub dibujaColumna(ByVal mapa As Graphics, ByVal X As Double, ByVal Diam As Integer)
        Dim LapizColumna As New System.Drawing.Pen(ColorExt, 8)
        Dim h, dx, dy, xx, y, d As Single

        Dim punto1 As New PointF(X, 45 + Diam)                   'Parte baja de la masa
        Dim punto3 As New PointF(pic.Width / 2, pic.Height - 10) 'Base de la columna

        xx = (punto1.X - punto3.X) / 2
        y = (punto3.Y - punto1.Y) / 2
        h = Sqrt(xx ^ 2 + y ^ 2)
        d = 0.2 * xx

        dx = d * y / h
        dy = d * xx / h

        Dim punto2 As New PointF(punto3.X + xx - dx, punto3.Y - y - dy) 'Punto de control 2
        Dim puntoscurva As PointF() = {punto1, punto2, punto3}          'Puntos de la curva
        mapa.DrawCurve(LapizColumna, puntoscurva)                        Dibuja la columna
End Sub
```

Código fuente 5.27 Función para dibujar la columna

El constructor 3 de la clase "fondoOscilador" (código fuente 5.28) dibuja sobre los *PictureBox* el fondo de las gráficas de los osciladores y la pestaña con el título de cada una.

```
'constructor 3 Dibuja SOLO el fondo
Sub New(ByVal pic As PictureBox, ByVal texto As String, ByVal Sx As Integer, ByVal Sy As
Integer)
        Me.Sx = Sx                      'Valor de la separación en X de la malla
        Me.Sy = Sy                      'Valor de la separación en Y de la malla
        Me.pic = pic                    'Pasa el picturebox a la Variable global de la clase

        mapa = CreaAreaDibujo(pic)      'crea el área de dibujo a partir del PictureBox

        MitadAncho = (pic.Width \ 2)          'Mitad de ancho del picturebox
        NumLineas = CInt(MitadAncho \ Sx)     'Numero de líneas verticales
        Call dibujafondo(mapa)                'Dibuja el fondo de los osciladores
        Call dibujaPestaña(mapa, texto)       'Dibuja la pestaña con el nombre del oscilador
End Sub
```

Código fuente 5.28 Constructor 3 de la clase *fondoOscilador*

En el constructor 3, al igual que en el código fuente 5.24, pasan los valores de los parámetros recibidos a las variables globales; después, se llama a la función "CrearAreaDibujo" (código fuente 5.25). Al final se invocan las funciones "dibujafondo" y "dibujaPestaña" (código fuente 5.29) para dibujar los elementos mencionados.

El código fuente de la función "dibujafondo" es similar al código fuente 5.20 de la gráfica de señales; éste se encarga de cuadricular el fondo de las gráficas tomando como parámetros el valor de las variables globales "Sx" y "Sy". En este caso los valores son constantes de 12 y 20 píxeles respectivamente, y no hay manera de que el usuario manipule estos valores.

Para dibujar la pestaña en las gráficas de los osciladores, se creó la función "dibujaPestaña" (código fuente 5.29), en la cual se define el objeto "objetoPESTAÑA" que hace referencia a la clase "Pestaña" (código fuente 5.21); esta clase se encarga de dibujar sobre el *PictureBox* la pestaña en lo alto del control con un ancho de 10 píxeles y establece el título del oscilador.

```
    Private Sub dibujaPestaña(ByVal mapa As Graphics, ByVal Texto As String)
        Dim objetoPESTAÑA As New Pestaña(mapa, pic, 0)
        mapa.DrawString(Texto, fuenteN, Brushes.Black, 5, 5)
    End Sub
```

Código fuente 5.29 Función para dibujar la pestaña

5.2.2.2. Oscilador VGL

En la ventana de osciladores de VGL se declaró un par de variables ("AreaOscilador" y "CapaOscilador"), sin necesidad de ser matrices, pues en esta ventana sólo hay un oscilador.

```
Dim AreaOscilador As New PictureBox
Dim CapaOscilador As New PictureBox
```

Las dimensiones mínimas para la gráfica del oscilador varían dependiendo del número de grados de libertad; estas se muestran en la tabla 5.6.

Tabla 5.6 Dimensiones mínimas (píxeles) del oscilador de VGL

Grados de libertad	Alto	Ancho
2	230	180
3	320	
4	410	
5	500	
6 ≤	590	

El *PictureBox* se ubica entre las gráficas de respuesta y el panel de controles. El oscilador se halla siempre centrado en lo alto de la ventana como se muestra en la figura 5.12, y dependerá de su altura mostrada en la tabla 5.6.

El *PictureBox* fue dividido en partes de la misma forma que los osciladores de 1GL descritos anteriormente. De igual manera se elaboró una clase "fondoOsciladorVGL" para dibujar la gráfica del oscilador.

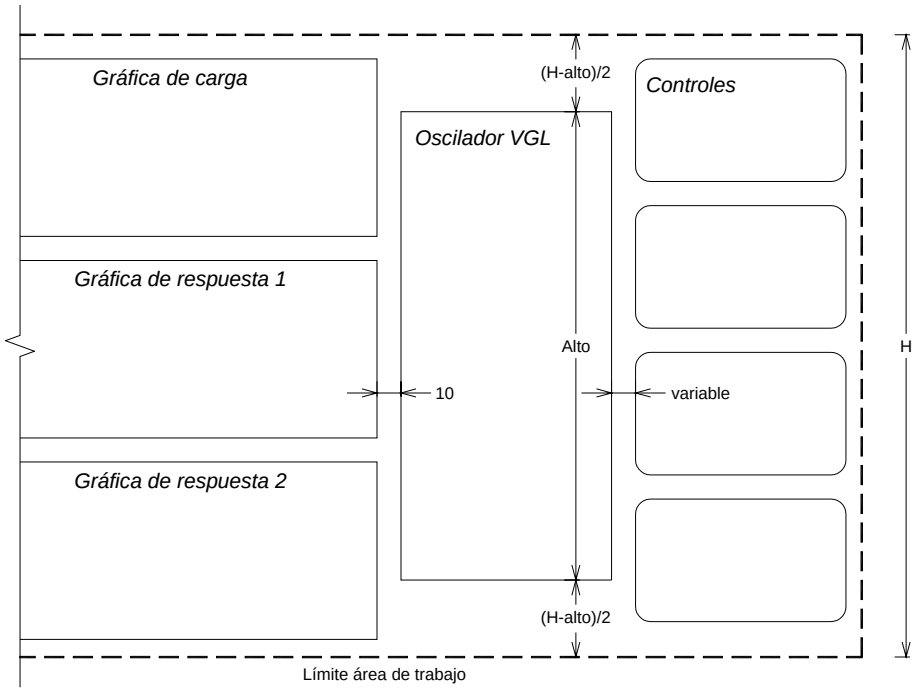

Figura 5.12 Ubicación del oscilador de VGL en la ventana

La clase "fondoOsciladorVGL" (código fuente 5.30) tiene 2 constructores y 4 funciones del tipo *Private*. Los constructores son similares a los de la clase "fondoOscilador": uno dibuja el fondo cuadriculado del oscilador (constructor 1) y el otro (constructor 2) dibuja sistema de masas y resortes para N grados de libertad que se hayan definido.

La tabla 5.7 muestra el significado de cada uno de los parámetros que requieren los constructores de la clase.

```
Imports System.Drawing.Drawing2D
Imports System.Math

Public Class FondoOsciladorVGL

'Constructor 1 Dibuja EL FONDO del oscilador
Sub New(ByVal pic As PictureBox, ByVal Sx As Integer, ByVal Sy As Integer, ByVal nGL As
Integer)

'constructor 2 Dibuja SOLO los osciladores (masas y resortes) en cada instante
Sub New(ByVal pic As PictureBox, ByVal Texto As String, ByVal tamaño As Single, ByVal
Puntos() As Double, ByVal Valores() As Double, ByVal nGL As Integer, ByVal MostrarVals As
Boolean, ByVal N As Integer)

Private Sub dibujafondo(ByVal mapa As Graphics)

Private Sub dibujaPestaña(ByVal mapa As Graphics, ByVal n As Integer)

Private Sub dibujamasa(ByVal mapa As Graphics, ByVal X As Double, ByVal Y As Double)

Private Sub dibujaColumna(ByVal mapa As Graphics, ByVal X1 As Double, ByVal Y1 As Double,
ByVal X2 As Double, ByVal Y2 As Double)

End Class
```

Código fuente 5.30 Clase *fondoOsciladorVGL*

En el constructor 2 (código fuente 5.31), se asignan los valores de los parámetros recibidos a las variables globales de la clase y declara una matriz "PosXx ( )" para almacenar los valores de las coordenadas de las masas en cada instante de tiempo *i*, así como las variables para almacenar las coordenadas (x, y) de los textos que muestran los valores de los desplazamientos de cada masa

Después se establece un área de dibujo ligado al *PictureBox* a través de un *Bitmap*, como se mencionó anteriormente, esto se encuentra en el código fuente 5.31 a partir de la sexta línea de código.

```
'constructor 2 Dibuja SOLO los osciladores en cada instante
Sub New(ByVal pic As PictureBox, ByVal Texto As String, ByVal tamaño As Single, ByVal
Puntos() As Double, ByVal Valores() As Double, ByVal nGL As Integer, ByVal MostrarVals As
Boolean, ByVal N As Integer)

Me.pic = pic                              'Pasa el picturebox a la Variable global de la clase
Me.Tamaño = tamaño                        'Tamaño de la masa
Me.nGL = nGL                              'Número de grados de libertad

Dim PosXx() As Double      'Coordenadas en pixeles de las masas dentro del PictureBox
Dim x, y As Single         'Coordenadas del texto que muestra el valor desplazamiento

Dim Btmp As New Bitmap(pic.Width, pic.Height)   'Se crea un Bitmap de las dimensiones del Pb
Dim mapa As Graphics = pic.CreateGraphics()     'Crea un área de trabajo a partir del Pb
pic.Image = Btmp                                'La imagen del PictureBox será el Bitmap
mapa = Graphics.FromImage(Btmp)                 'Permite un buen efecto de animación en cada
                                                 cambio de dibujo

ReDim PosXx(nGL)                   'Valores de los desplazamientos de cada grado de libertad

For Me.i = 0 To (Me.nGL - 1)                     'Recorre todos los espacios del vector
   PosXx(i) = (pic.Width / 2) + Puntos(i + 1)    'Establece la posición a partir de la mitad
Next i                                            de ancho del PictureBox

Dim Py1, Py2, Largo As Double                    'Variables auxiliares
Py1 = 40 + Me.Tamaño * 0.99
Largo = 1.5 * Me.Tamaño + Me.Tamaño * 0.02
Py2 = Py1 + Largo

For Me.i = 0 To (Me.nGL - 1)                     'Ciclo que recorre cada GL
   If i = Me.nGL - 1 Then
      Call dibujaColumna(mapa, PosXx(i), Py1, Me.pic.Width / 2, Me.pic.Height - 10)
   Else
      Call dibujaColumna(mapa, PosXx(i), Py1, PosXx(i + 1), Py2)
      Py1 = Py2 + Me.Tamaño * 0.98
      Py2 = Py1 + Largo
   End If

Call dibujamasa(mapa, PosXx(i), 40 + i * Me.Tamaño * 2.5) 'Dibuja la masa

   If MostrarVals = True Then                            'Muestra los valores de Despl
      y = (32 + Me.Tamaño / 2) + i * Me.Tamaño * 2.5     'Coordenada Y de cada texto
      If Valores(i + 1) <= 0 Then                        'Coordenada X
          x = PosXx(i) + Me.Tamaño / 2 + 5
          mapa.DrawString(Format(Valores(i + 1), "0.000"), fuenteN, Brushes.Black, x, y)
      Else
          x = PosXx(i) - Me.Tamaño / 2 - 5
          mapa.DrawString(Format(Valores(i + 1), "0.000"), fuenteN, Brushes.Black, x, y,
formato)
      End If
   End If
Next i
'Título de la gráfica del oscilador
mapa.DrawString(Texto, fuenteN, Brushes.DarkRed, pic.Width - 10, 4, formato)
End Sub
```

Código fuente 5.31 Constructor 2 de la clase *fondoOsciladorVGL*

El segundo bucle *for* dentro del código fuente 5.31 controla el llamado a las funciones que dibujan las masas y las columnas para cada grado de libertad del oscilador. Además, el segundo *if* evalúa si se colocan los letreros de texto que muestran los valores de desplazamientos en cada masa del oscilador.

Los parámetros que se necesitan para cada constructor de la clase "fondoOsciladorVGL" se encuentran en la tabla 5.7.

Tabla 5.7 Parámetros en los constructores

Variable	Descripción
pic	Nombre del *PictureBox* donde se va a dibujar la gráfica
Sx	Separación (píxeles) de la cuadrícula en dirección X
Sy	Separación (píxeles) de la cuadrícula en dirección Y
nGL	Número de grados de libertad
Texto	Título de la gráfica
Tamaño	diámetro de las masas en píxeles
Puntos()	Posición (píxeles) de las masas dentro del *PictureBox*
Valores()	Valor del desplazamiento horizontal de cada masa
MostrarVals	Condicional (falso o verdadero) para mostrar los valores de los desplazamientos
N	Número de punto de la señal de desplazamiento

Las funciones "dibujafondo", "dibujaPestaña" y "dibujamasa"son las mismas que se utilizaron para el oscilador de 1GL. La función para dibujar la columna es diferente para este tipo de oscilador ya que estas columnas se deforman con una doble curvatura, como se muestra en la figura 5.13.

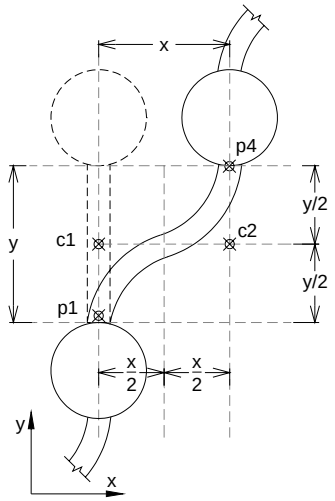

Figura 5.13 Puntos de control para dibujar la columna

El código fuente 5.32 muestra la función "dibujaColumna"; ésta declara primero una variable (grueso) para controlar el espesor de la columna dependiendo del número de GL seleccionado.

Después, se calculan cuatro puntos de control para la columna (p1, c1, c2 y p4 mostrados en la figura 5.13). Para dibujar la columna con doble curvatura se utilizó la función *DrawBezier* la cual recibe como parámetros el formato de la línea y los puntos de control mencionados.

La función *DrawBezier* dibuja una línea curva tomando el primer y último punto (p1 y p4) como el inicio y final de la curva respectivamente. Los dos puntos intermedios (c1 y c2) establecen el radio de giro para dibujar la curvatura de la columna.

Como el llamado de la función "dibujaColumna" en el código fuente 5.31, que se encuentra dentro del segundo bucle *for*, se ejecuta tantas veces como grados de libertad tenga el oscilador, y para cada uno van cambiando las coordenadas de los puntos de control de la curva.

```
Private Sub dibujaColumna(ByVal mapa As Graphics, ByVal X1 As Double, ByVal Y1 As Double,
ByVal X2 As Double, ByVal Y2 As Double)

Dim grueso As Integer            'Espesor de la columna por default es igual a cero

'Las siguientes condiciones evaluan a los GL, es importante el orden en que están acomodadas
las condiciones pues de esta manera la evaluación se hace por rango de valores

If Me.nGL < 15 Then : grueso = 2 : End If   'Si los GL < 15 el espesor de la columna es de 2
If Me.nGL < 12 Then : grueso = 4 : End If   'Si los GL < 12 el espesor de la columna es de 4
If Me.nGL < 9 Then : grueso = 6 : End If    'Si los GL < 9 el espesor de la columna es de 6

'Se crea el tipo de línea definiendo Color y Grosor
Dim LapizColumna As New System.Drawing.Pen(ColorExt, grueso)

Dim Y As Single = (Y1 - Y2) / 2 + Y2  'Punto medio de la columa
Dim punto1 As New PointF(X2, Y2)      'Base de la columna
Dim Control1 As New PointF(X2, Y)     'Punto de control 1
Dim Control2 As New PointF(X1, Y)     'Punto de control 2
Dim punto4 As New PointF(X1, Y1)      'Punto final de la columna

'Se dibuja la curva Bezier de 4 puntos
mapa.DrawBezier(LapizColumna, punto1, Control1, Control2, punto4)

End Sub
```

Código fuente 5.32 Clase *fondoOsciladorVGL*

5.2.3. Espectros

5.2.3.1. Espectros de respuesta

Los espectros se dibujan sobre los *PictureBox* "AreaOscilador", mismos que se utilizaron para dibujar los osciladores, tomando las mismas dimensiones y ubicación dentro de la ventana de los osciladores de 1GL. Así la declaración de matrices que se usó es la misma que se mostró en la sección 5.2.2.1.

Para dibujar los espectros de repuesta, los *PictureBox* se dividieron de la misma forma que se muestra en la figura 5.10. Además, se estableció la clase "fondoEspectro" (código fuente 5.33) para dibujar cada elemento de la gráfica; la clase contiene 3 constructores y 2 funciones del tipo *Private*.

El constructor 1 dibuja sólo el fondo y la pestaña de la gráfica; el constructor 2 dibuja la gráfica con todos sus elementos y por último el constructor 3 sirve para dibujar la gráfica cuando ésta se encuentra en animación. Las funciones del tipo *Private* se utilizan para dibujar sobre el *PictureBox* la pestaña correspondiente a cada gráfica y el fondo cuadriculado en la misma, este último es muy parecido al fondo de que se utilizó en las gráficas de los osciladores, sólo que ocupan un color rojizo en lugar de un color gris.

```
Imports System.Drawing.Drawing2D
Imports System.Math

Public Class FondoEspectro

'Constructor 1 Dibuja la gráfica vacia
Sub New(ByVal pic As PictureBox, ByVal texto As String, ByVal Sx As Integer, ByVal Sy As
Integer, ByRef Btmp As Bitmap)

'Constructor 2 Dibuja la gráfica CON el espectro en forma ESTATICA
Sub New(ByVal pic As PictureBox, ByVal texto As String, ByVal Sx As Integer, ByVal Sy As
Integer, ByRef puntos() As PointF)

'Constructor 3 Dibuja la gráfica del espectro POR PASOS
Sub New(ByVal pic As PictureBox, ByRef Btmp As Bitmap, ByVal X1 As Single, ByVal Y1 As
Single, ByVal X2 As Single, ByVal Y2 As Single)

Private Sub dibujafondo(ByVal mapa As Graphics)

Private Sub dibujaPestaña(ByVal mapa As Graphics)

End Class
```

Código fuente 5.33 Clase *fondoEspectro*

El código fuente 5.34 muestra el cuerpo del constructor 2 que dibuja la gráfica de los espectros con todos sus elementos. Primero pasa los valores de los parámetros que recibe a las variables globales, después establece un área de dibujo como ya se ha mencionado en repetidas ocasiones, posteriormente llama a las funciones “dibujafondo” y “dibujaPestaña”. La primera es similar a la que se encuentra en el código fuente 5.20, para dibujar el fondo de las gráficas de señales y de los osciladores. La segunda se muestra en el código fuente 5.29.

Por último, con la sentencia “mapa.DrawLines” dibuja la gráfica correspondiente al espectro de respuesta; ésta sólo requiere de dos parámetros, el formato de la línea (“lapiz3”) y el vector con las coordenadas de cada punto que conforman al espectro (“puntos”).

```
'Constructor 2 Dibuja la gráfica CON el espectro en forma ESTATICA
Sub New(ByVal pic As PictureBox, ByVal texto As String, ByVal Sx As Integer, ByVal Sy As
Integer, ByRef puntos() As PointF)

Me.Sx = Sx                          'Separación en dirección X de la malla
Me.Sy = Sy                          'Separación en dirección Y de la malla
Me.pic = pic                        'PictureBox sobre el que se dibuja
Me.Texto = texto                    'Nombre de la gráfica

Dim Btmp1 As New Bitmap(pic.Width, pic.Height) 'Crea un Bitmap del tamaño del pictureBox
Dim mapa As Graphics = pic.CreateGraphics()    'Crea un área de trabajo a partir del
                                                PictureBox
pic.Image = Btmp1                              'La imagen del PictureBox será el Bitmap
mapa = Graphics.FromImage(Btmp1)               'Permite un buen efecto de animación en cada
                                                cambio de dibujo

Call dibujafondo(mapa)                         'Dibuja fondo
Call dibujaPestaña(mapa)                       'Dibuja Pestaña

mapa.DrawLines(lapiz3, puntos)                 'Dibuja la señal

End Sub
```

Código fuente 5.34 Constructor 2 de la clase *fondoEspectro*

5.2.3.2. Espectros de piso

A diferencia de las demás gráficas mencionadas, (señales, osciladores y espectros de respuesta) los espectros de piso no tienen una ubicación fija dentro de la ventana, debido a que aparecen cuando el usuario hace clic sobre el área de una masa en específico y desaparecen cuando se hace clic en cualquier otra parte.

Debido a este efecto, para dibujar los espectros de piso fue necesario hacerlo sobre un formulario nuevo, el cual posee el tamaño justo del espectro de piso (180 x 180 píxeles) y el único control que contiene es del tipo *PictureBox* en el cual se dibuja la gráfica del espectro.

Para dibujar el espectro se utilizó la clase "fondoEspectro" (código fuente 5.33), que se explicó en la sección anterior.

El código fuente 5.35 muestra la clase que corresponde al formulario "frmEspecrosPiso". La clase contiene un constructor y además tres funciones delimitadas por las palabras clave *Private Sub* y *End Sub*.

El constructor recibe los parámetros para poder ubicar el formulario y dibujar el espectro de piso. El único objetivo del constructor es pasar los valores de los parámetros a las variables globales de la clase. La tabla 5.8 muestra el significado de cada uno.

Tabla 5.8 Parámetros del constructor

Variable	Descripción
X, Y	Coordenadas para ubicar al formulario
Texto	Título de la gráfica
Puntos()	Vector de las coordenadas de cada punto que conforman al espectro
Ti	Valor del periodo inicial del espectro
Intervalo	Intervalo de tiempo entre los puntos del espectro
FEV	Factor de escala vertical
FEH	Factor de escala horizontal

La primera función en el código fuente 5.35 corresponde al evento *Load*; éste se ejecuta cada vez que el formulario aparece en pantalla. Esta función ubica a el formulario en las coordenadas (X, Y) y después invoca a la función "CambiarForma".

La segunda función, corresponde al evento *LostFocus*, que se ejecuta cuando el formulario pierde el enfoque, es decir cada vez que el usuario haga clic en cualquier otra parte que no sea el formulario. La única sentencia dentro de esta función es *Close*, que cierra el formulario.

Por último, la función "CambiarForma" establece una nueva forma geométrica al formulario, para ello se definen las coordenadas de los puntos mediante las sentencias *ogPath.AddLine* y con la instrucción *Me.Region=New Region (ogPath)* se indica la forma del formulario. En la última línea de la función se declaró la variable "ObjetoEspectro" para hacer referencia a la clase "FondoEspectro" que dibuja la gráfica del espectro de piso sobre el *PictureBox*.

```
Imports System.Windows.Forms
Imports System.Drawing.Drawing2D
Imports System.Drawing

Public Class FrmEspectroPiso

Sub New(ByVal X As Integer, ByVal Y As Integer, ByVal Texto As String, ByVal Puntos() As
PointF, ByVal Ti As Single, ByVal Intervalo As Single, ByVal FEV As Single, ByVal FEH As
Single)

        ' This call is required by the Windows Form Designer.
        InitializeComponent()
        Me.X = X
        Me.Y = Y
        Me.PuntosGráfica = Puntos
        Me.Texto = Texto
        Me.Intervalo = Intervalo
        Me.Ti = Ti

        If FEV = 0 Then
            Me.FEV = 1
        Else
            Me.FEV = FEV
        End If

        Me.FEH = FEH
End Sub

'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Private Sub FrmEspectroPiso_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load                         'Evento cargar formulario

       Me.Location = New Point(Me.X, Me.Y) 'Ubicación del formulario
       Call CambiarForma()                  'Cambiar la forma del formulario
End Sub

'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Private Sub Dialog1_LostFocus(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.LostFocus                                'Evento cambiar de enfoque
       Me.Close()                           'Cierra el formulario
End Sub

'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Private Sub CambiarForma()        'Función cambiar forma del formulario
Dim ogPath As New GraphicsPath    'variable para almaenar las coordenadas de la nueva
                                   forma del formulario

ogPath.AddLine(9, 0, 180, 0)      'Coordenadas de cada punto de la nueva forma
ogPath.AddLine(180, 0, 180, 180)  'que adoptara el formulario
ogPath.AddLine(180, 180, 0, 180)
ogPath.AddLine(0, 180, 0, 9)

Me.Region = New Region(ogPath)    'Se pasa el poligono como region para el form

Dim ObjetoEspectro As New FondoEspectro(Me.PictureBox1, Me.Texto, 10, 10, Me.PuntosGráfica)
End Sub

End Class
```

Código fuente 5.35 Formulario para espectros de Piso

5.3. ANIMACIÓN

En general, los procesos de animación consisten en colocar de forma consecutiva una serie de imágenes que al proyectarse en un mismo espacio dan una ilusión de movimiento.

Por ejemplo, para crear la animación de movimiento del oscilador (mostrado en la figura 5.14) de la posición A la posición B en un tiempo de 0.4 s, es necesario dibujar las imágenes intermedias entre estas dos posiciones y colocarlas consecutivamente una tras otra a cada intervalo de tiempo.

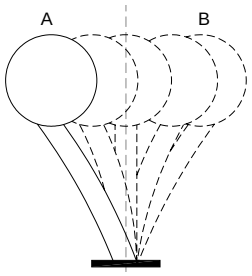

Figura 5.14 Movimiento de un oscilador de 1GL

Al ejecutar la animación, lo que el usuario verá serán las imágenes mostradas en la figura 5.15, una seguida de la otra en la misma posición pero no al mismo tiempo, lo que da la ilusión de que el oscilador se mueve desde el punto A hasta el punto B, manteniendo el eje y la base del oscilador en la misma ubicación. En este ejemplo se usaron 5 imágenes que se muestran en un intervalo de tiempo de 0.1 s, es decir, el usuario observará a cada 0.1 s una imagen distinta del oscilador proyectándose en la misma posición.

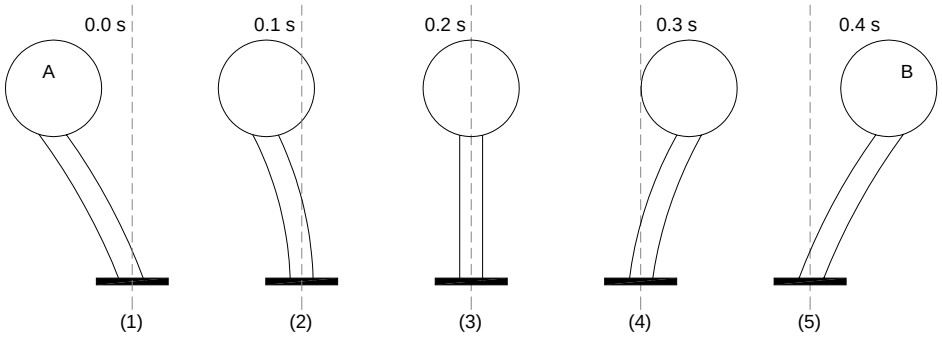

Figura 5.15 Movimiento por escenas de un oscilador de 1GL

Para tener un buen efecto de animación, las imágenes que se proyecten deben tener una estrecha relación con la consecutiva siguiente. Por ejemplo, en la figura 5.15 se observa cómo el oscilador se va moviendo paulatinamente del punto A al punto B, debido a que las imágenes varían sólo un poco una con respecto a su consecutiva siguiente. Por el otro el lado, si para mostrar la animación sólo se seleccionaran las imágenes 1, 3 y 5 el efecto de animación sería menos notorio y no se apreciaría bien.

Entre más imágenes haya para mostrar una animación mejor efecto se tendrá. Además, es necesario contar con un intervalo de tiempo pequeño entre las imágenes, ya que entre más corto sea mejor será el efecto. No obstante, existe un límite de imágenes, ya que el ojo no percibe más allá de 24 cuadros por segundo.

El programa a través del control *Timer* controla y ejecuta la proyección consecutiva de imágenes (señales de respuesta, osciladores y espectros de respuesta) a intervalos de tiempo definidos. Dichas imágenes son dibujadas dependiendo de los cálculos obtenidos para la respuesta dinámica.

5.3.1. Señales

Sólo las señales de la respuesta dinámica muestran un efecto de animación dentro del programa mientras que la señal de carga permanece estática en todo el proceso.

Las señales presentan dos tipos de animación: por adición y por sustitución.

- ✓ Por adición

Ocurre cuando las imágenes proyectadas cambian una con respecto a otra por que se le agrega una parte a la imagen anterior, manteniendo todo lo demás igual. Por ejemplo, en la figura 5.16 se muestran 6 imágenes que pertenecen a una animación de una señal; se puede observar que cada imagen es igual a la anterior pero con la diferencia de que se le agregó una línea más, dicha línea une al punto siguiente en la gráfica.

Si se colocan estas imágenes en una proyección, se vería a la respuesta dinámica recorriendo su historia en el tiempo, uniendo cada punto de la señal con una línea recta hasta completar su duración.

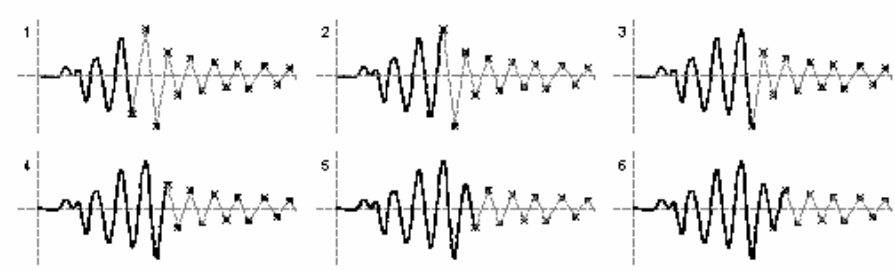

Figura 5.16 Secuencia de imágenes para la animación de una señal

Este tipo de animación se muestra en el programa cuando se presiona el botón *Play* en el panel de controles, al hacerlo se puede ver en pantalla la animación de las señales sincronizada con el movimiento del oscilador, este último se describe en la sección 5.3.2.

Para ejecutar la animación primero se calculan y luego se dibujan las señales que conforman a la respuesta dinámica; esto se describió en las secciones 5.1 y 5.2 respectivamente.

Cuando el usuario presiona el botón *Play* de la animación de los osciladores en el panel de controles, el programa llama al constructor 5 (código fuente 5.36) de la clase "GráficarSeñal" e inmediatamente después se ejecuta la función (evento *tick*) del control *Timer* mostrado en el código fuente 5.37.

El constructor 5 crea la primera imagen de la animación, la cual contiene cada parte de la gráfica (fondo, título, escala numérica, línea de referencia horizontal) con excepción de la señal. Dicha imagen funciona como base para las demás imágenes requeridas en el proceso de animación.

La explicación del contenido del constructor 5 es similar a la del constructor 2 mostrado en el código fuente 5.19 en la sección 5.2. La diferencia entre ellos es que el constructor 5 omite las líneas de código para dibujar la señal y la pestaña de la gráfica.

La instrucción que se encuentra en la penúltima línea del código fuente 5.36, copia el dibujo realizado sobre el *PictureBox* (imagen base) a la variable *Btmp*.

```
'Constructor 5 Dibuja la cuadrícula SIN la señal, SIN LA PESTAÑA antes de cada animación de
la señal
Sub New(ByVal pic As PictureBox, ByVal texto As String, ByVal Sx As Single, ByVal Sy As
Single, ByVal FE As Double, ByRef Btmp As Bitmap)

Dim Btmp1 As New Bitmap(pic.Width, pic.Height) 'Crea un Bitmap del tamaño del pictureBox
Dim mapa As Graphics = pic.CreateGraphics()    'Crea un área de trabajo a partir del Pb
pic.Image = Btmp1                              'La imagen del PictureBox será el Bitmap
mapa = Graphics.FromImage(Btmp1)               'Permite un buen efecto de animación en cada
                                                cambio de dibujo
Posicionmedia = pic.Height / 2       'Valor de la coordenada en Y de la línea de referencia
MitadAltura = (pic.Height) / 2       'Valor de la mitad de la altura libre para dibujar
NumLineas = CInt(MitadAltura \ Sy)             'Número de líneas a dibujar en la malla
Call dibujafondo(pic, mapa, Sx, Sy, 0)         'Dibuja el fondo con los valores calculados

mapa.DrawLine(lapiz2, CoordX, Posicionmedia, pic.Width, Posicionmedia) 'línea de referencia

Dim objetoTEXTO As New TextoHorizontal(NumLineas, CInt(pic.Height / 2), Sy, FE, mapa, texto,
CoordX)          'Coloca la escala númerica de la gráfica

Btmp = Btmp1     'Pasa el dibujo que hay en el Bitmap para guardarlo como fondo
End Sub
```

Código fuente 5.36 Constructor 5 de la clase *GráficaSeñal*

El código fuente 5.37 es el que hace posible la animación de las señales y los osciladores; el código muestra la función del control *Timer1*. Cuando la función es llamada por el programa, las instrucciones que se encuentran dentro de ella se ejecutan una y otra vez hasta que se indique que se detenga. Cada ciclo del *Timer* significa que dibuja una imagen de la animación.

El control tiene un comportamiento parecido a un bucle (*for o while*) pero con un intervalo de tiempo entre cada ciclo. El intervalo de tiempo del control *Timer1* es de 1 milisegundo, pero la velocidad de ejecución de las instrucciones contenidas en él, es relativa a las propiedades de la computadora, por ello la animación de la señal no puede tener un tiempo de reproducción igual al de la duración real de la respuesta dinámica.

Para saber en qué momento detener el control *Timer*, se colocó una sentencia *if* al principio de la función; ésta evalúa que el contador de puntos de la señal (n) no exceda al número total de puntos. El contador (n) se incrementa cada vez que el control realiza un ciclo.

Dentro del *if* se encuentran dos bucles *for*, el primero sirve para crear las animaciones de los osciladores, y el segundo para las animaciones de las señales. Los bucles controlan la cantidad de gráficas correspondientes a dibujar (osciladores y señales).

En el segundo bucle *for*, se encuentra la sentencia para llamar a la clase “GráficarSeñal” a través de su constructor 6 (código fuente 5.38). Cada vez que el control *Timer* ejecuta un ciclo, el bucle *for* ejecuta la llamada a la clase tantas veces como gráficas se hayan definido.

El constructor 6 (código fuente 5.38) es el encargado de dibujar sobre la imagen base (contenida en la variable *Btmp*) cada línea siguiente en la gráfica, para ello recibe entre sus parámetros un par de coordenadas (X1, Y1) y (X2, Y2) que indican el punto de inicio y final de la línea a dibujar, así como la imagen contenida en la variable *Btmp*.

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer1.Tick

If n < maxpuntos Then         'Verifica que n sea menor al número de puntos de la señal
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  For j = 1 To Me.NUD.Value   'Ciclo para dibujar el movimiento de los osciladores
    Dim ObjetoOsc As New FondoOscilador(CapaOscilador(j), Me.NUDTamañoOscil.Value,
CCenOsc(j), CExtOsc(j), RD(j)(n) * FactorEscalaHorizOscil, n, RD(j)(n))
  Next j

'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  For j = 1 To Me.NUDGráficasAct.Value 'Ciclo para dibujar el movimiento de las señales
    Dim objetoGráfica1 As New GráficarSeñal(AreaGráfica(j), BtmpGraf(j), Rr(j)(n).X,
Rr(j)(n).Y, Rr(j)(n + 1).X, Rr(j)(n + 1).Y, LapizGráfica(j), Grueso(j))
  Next j

  n += Me.NUDincremento.Value  'Contador de puntos de la señal
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Else                           'En caso de llegar al numero de puntos de la señal
  Call DetenerAnimacion(True)  'Llama a la función para detener la animación
End If                         'Fin de la condición

End Sub                        'Fin de la función Timer
```

Código fuente 5.37 Evento *Tick* del control *Timer1*

Para que el constructor 6 dibuje sobre el *PictureBox* es necesario establecer un área de dibujo, (como se ha mencionado anteriormente). En este caso se observa que el *Bitmap* pasa como parámetro de referencia del constructor; esto quiere decir que la variable *Btmp* almacena una imagen previa, y además al pasar por valor de referencia la variable guarda cada cambio que sufra la misma.

Es decir, el constructor 6 dibuja una línea sobre una imagen definida (en este caso la gráfica de la señal hasta un determinado punto), después la misma variable *Btmp* guarda la imagen modificada y se repite el proceso en cada ciclo hasta que el dibujo de la señal termina.

Al ver este proceso ejecutarse de manera secuencial en intervalos de tiempo pequeños entre las imágenes, se obtiene el efecto de movimiento de la señal como si recorriera su historia en el tiempo.

```
'Constructor 6 Dibuja la señal por PASOS SIN LA PESTAÑA, SIN fondo
Sub New(ByVal pic As PictureBox, ByRef Btmp As Bitmap, ByVal X1 As Single, ByVal Y1 As
Single, ByVal X2 As Single, ByVal Y2 As Single, ByVal ColorLapiz As Color, ByVal Grueso As
Integer)

Dim PuntosAux(1) As System.Drawing.PointF   'Se crea un par de puntos(X,Y) para el inicio y
                                             final de la línea a dibujar
Dim mapa As Graphics = pic.CreateGraphics() 'Crea un área de trabajo a partir del PictureBox
pic.Image = Btmp                            'La imagen del PictureBox será el Bitmap que
                                             guardo el fondo
mapa = Graphics.FromImage(Btmp)             'Permite un buen efecto de animación en cada
                                             cambio de dibujo
PuntosAux(0) = New PointF(X1, Y1)           'Se definen las coordenadas del punto inicial
PuntosAux(1) = New PointF(X2, Y2)           'Se definen las coordenadas del punto final

Dim lapiz1 As New System.Drawing.Pen(ColorLapiz, Grueso) 'Formato de la línea de la gráfica
mapa.DrawLines(lapiz1, PuntosAux)                        'Se dibuja la línea
End Sub
```

Código fuente 5.38 Constructor 6 de la clase *GráficarSeñal*

- ✓ Por sustitución

Éste ocurre cuando en las imágenes a proyectar es necesario sustituir alguna parte del dibujo por otra nueva, manteniendo los demás elementos del dibujo igual.

La animación por sustitución se utiliza en la parte correspondiente a los espectros de respuesta; al momento de calcular los espectros y cuando se hace clic en el botón *Play* en el panel de controles, se observa como la respuesta dinámica de los osciladores va cambiando conforme su periodo cambia.

Para lograr esta animación es necesario dibujar las gráfica de respuesta del oscilador para cada periodo T. Las imágenes mantienen ciertos elementos de las gráfica constantes (fondo, título, escala numérica, línea de referencia mostrados en la figura 5.17B) y sólo cambia de una imagen a otra el dibujo de la señal (figura 5.17A).

En este caso se trata de una animación por sustitución pues en cada imagen sólo hay que sustituir la línea que representa a la respuesta dinámica por la que corresponda en cada valor del periodo.

A diferencia de la animación por adición, los cálculos correspondientes se van realizando al mismo tiempo que se ejecuta la animación, esto con el fin de evitar saturar la memoria almacenando los datos de las respuestas de los osciladores para cada espectro.

Las imágenes de la animación son dibujadas con la ayuda de dos *PictureBox* superpuestos uno encima del otro, es decir, un dibujo por capas. En la figura 5.17_A se muestra el dibujo de la señal sobre un *PictureBox* con fondo transparente; la figura 5.17_B muestra el dibujo de los elementos de las gráficas que permanecen constantes en cada imagen. Al superponer las dos imágenes se obtiene la figura 5.17_C.

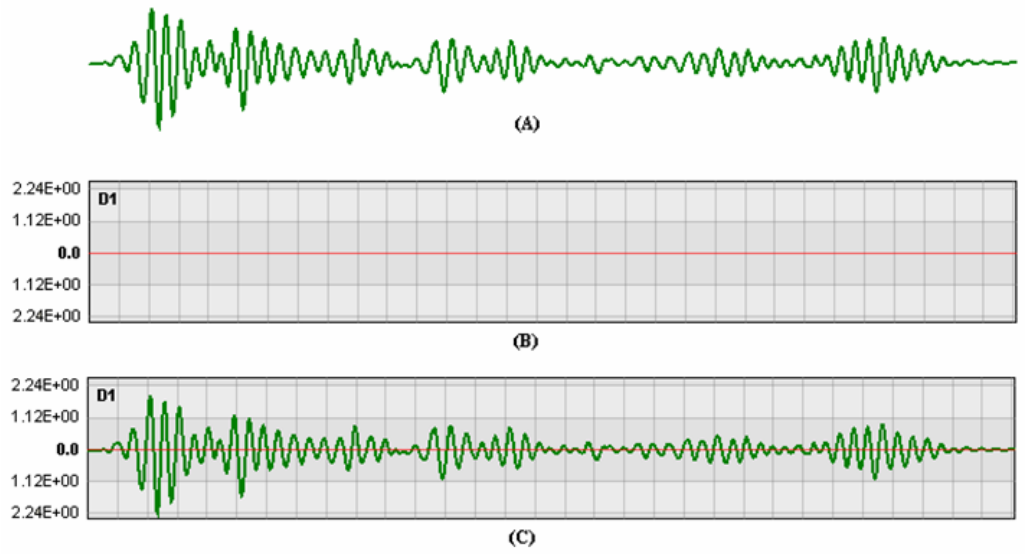

Figura 5.17 Dibujo de una señal por capas

En el proceso de las animaciones, el *PictureBox* declarado anteriormente en la sección 5.2.1 para dibujar las gráficas funcionará ahora para crear la imagen que permanece constante (figura 5.17_B).

Para dibujar las señales (5.17_A) se declaró una nueva matriz de controles *PictureBox* de nombre "CapaAnimaciónSeñal" en ambas ventanas (osciladores de 1GL y VGL), dichas matrices tienen el mismo tamaño que la matriz "AreaGráfica" correspondiente en cada ventana.

```
Dim CapaAnimacionSeñal(3) As New PictureBox
```

Cada vez que la animación se ejecuta, el programa llama a la función "PosiciónCapasAnimacionSeñal" (código fuente 5.39); ésta contiene un bucle *for* que establece las características para todas las gráficas activadas en la ventana.

Primero, se crea la imagen base para las gráficas (figura 5.17_B) llamando a la clase "GráficarSeñal" a través de su constructor 5 (código fuente 5.36); después se vinculan las dos matrices de controles definidas anteriormente mediante la sentencia *Controls.Add*, esto con la finalidad de establecer el fondo transparente en la matriz de controles "CapaAnimaciónSeñal" por medio de la propiedad *BackColor*. Por último se especifican las dimensiones (*Width y Height*) de los *PictureBox*.

```
Private Sub PosiciónCapasAnimacionSeñal(ByVal visibles As Boolean, ByVal gráficas As
Integer)

'Establece las propiedades de las capas para las animaciones de las señales
For Me.i = 1 To 3

Dim ObjetoGráfica As New GráficarSeñal(AreaGráfica(i), título(i), SeparacionX, SeparacionY,
Factor(i), BtmpGraf(i))

AreaGráfica(i).Controls.Add(CapaAnimacionSeñal(i))  'Agrega la capa como control del Pb
CapaAnimacionSeñal(i).BackColor = Color.Transparent 'Asigna el color transparente a la capa

CapaAnimacionSeñal(i).Visible = visibles
CapaAnimacionSeñal(i).Width = AreaGráfica(i).Width
CapaAnimacionSeñal(i).Height = AreaGráfica(i).Height
Next
End Sub
```

Código fuente 5.39 Llamado de la clase *GráficaSeñal*

Una vez dibujada la imagen base, se ejecuta la función del control *Timer* (código fuente 5.40) para generar la animación. La función tiene las mismas características que la descrita en el código 5.37. En este caso el control tiene el nombre de *TmEspectros* para diferenciarlo de la función que controla la animación de los osciladores.

```
Private Sub TmEspectros_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles
TmEspectros.Tick

If p <= PuntosEsp Then         'Verifica que n sea menor al número de puntos del espectro
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
For Me.i = 1 To NUDGráficasAct.Value 'Ciclo para dibujar el movimiento de los espectros
  Dim ObjetoFondo As New FondoEspectro(Me.AreaOscilador(i), BtmpEsp(i), Er(i)(m).X,
Er(i)(m).Y, Er(i)(m + 1).X, Er(i)(m + 1).Y)
Next

'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
For Me.i = 1 To gráficas         'Ciclo para dibujar el movimiento de las señales
  Dim ObjetoSeñal As New GráficarSeñal(CapaAnimacionSeñal(i), Raux(i), LapizGráfica(i),
Grueso(i))
Next
p += 1                          'Contador de puntos del espectro
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Else                                    'En caso de llegar al numero de puntos de la señal
  Call DetenerAnimacionEspectros(True) 'Llama a la función para detener la animación
End If                                  'Fin de la condición
End Sub                                 'Fin de la función Timer
```

Código fuente 5.40 Evento *Tick* del control *TmEspectros*

El código fuente 5.40 es el encargado de mostrar la animación de las señales en sincronía con los espectros de respuesta.

Al igual que la función del control *Timer* (código fuente 5.37), se colocó una sentencia *if* al principio de la función; la cual evalúa que el contador de puntos del espectro (p) no exceda al número total de puntos del mismo. El contador (p) se incrementa cada vez que el control realiza un ciclo.

Dentro de la sentencia *if* se encuentran dos bucles *for*, el primero controla las animaciones de los espectros de respuesta (descritos en la sección 5.3.3) y el segundo hace lo propio para las animaciones de las gráficas de respuesta.

En el segundo bucle *for* se encuentra la sentencia para llamar a la clase "GráficarSeñal" a través de su constructor 4 (código fuente 5.41). Cada vez que el control *Timer* ejecuta un ciclo, el bucle *for* ejecuta la llamada a la clase tantas veces como gráficas se hayan definido.

Los parámetros que requiere el constructor 4 se enlistan en la siguiente tabla.

Tabla 5.9 Parámetros del constructor

Variable	Descripción
Pic	Nombre del *PictureBox* donde se va a dibujar la gráfica
Puntos()	Vector de las coordenadas de cada punto que conforman a la señal
ColorLapiz	Color de la línea de la señal
Grueso	Espesor (píxeles) de la línea de la señal

Obsérvese que en el código fuente 5.39 el parámetro que pasa como *PictureBox* es la matriz de controles "CapaAnimaciónSeñal" destinada para dibujar sólo la señal de la gráfica, tal como se mencionó anteriormente.

```
'Constructor 4 Dibuja la señal completa, SIN fondo, SIN pestaña, SIN nada, para la animación
en la parte de los espectros
Sub New(ByVal pic As PictureBox, ByVal puntos() As PointF, ByVal ColorLapiz As Color, ByVal
Grueso As Integer)

Dim lapiz1 As New System.Drawing.Pen(ColorLapiz, Grueso) 'Establece formato de la línea

Dim Btmp As New Bitmap(pic.Width, pic.Height)   'Crea un Bitmap del tamaño del PictureBox
Dim mapa As Graphics = pic.CreateGraphics()     'Crea un área de trabajo a partir del Pb
pic.Image = Btmp                                'La imagen del PictureBox será el Bitmap
mapa = Graphics.FromImage(Btmp)                 'Permite un buen efecto de animación en cada
                                                 cambio de dibujo
mapa.DrawLines(lapiz1, puntos)                  'Dibuja la señal
End Sub
```

Código fuente 5.41 Constructor 4 de la clase *GráficaSeñal*

El constructor 4 de la clase "Gráficarseñal" (código fuente 5.41), define primero un formato para la línea de la señal a partir de los valores de los parámetros que recibe (color y espesor de línea), después establece un área de dibujo (descrita anteriormente), y por último la única instrucción de dibujo *DrawLines* dentro del constructor indica que se dibuje la señal a partir del vector de puntos que también paso como parámetro.

5.3.2. Osciladores

La animación de ambos tipos de osciladores (1GL y VGL) es por sustitución. Se dibujan en un *PictureBox* las partes de la señal que permanecen estáticas en todo el proceso y en un segundo control se dibuja el oscilador (masas y resortes) que es el elemento de las gráficas que cambia en cada instante de tiempo.

5.3.2.1. Osciladores de 1GL

Debido a la forma de dibujar las gráficas de los osciladores (descrita en la sección 5.2.2.1), sobre el control *PictureBox* "AreaOscilador" se dibuja el fondo de los osciladores (figura 5.18b) y en el *PictureBox* "CapaOscilador" se dibuja la masa y el resorte junto con el texto que indica el desplazamiento en cada instante de tiempo (figura 5.18a). Al superponer las dos imágenes se obtiene la figura 5.18c.

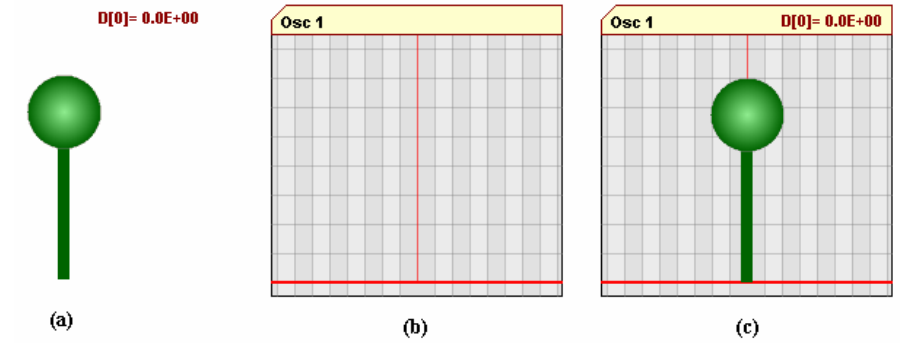

Figura 5.18 Dibujo del oscilador por capas

Debido a que el movimiento de los osciladores está sincronizado con la animación de las señales, el llamado a las clases para dibujar ambas gráficas (oscilador y señales) se encuentra dentro del mismo control *Timer*, (código fuente 5.37).

El llamado a la clase "FondoOscilador" se encuentra dentro del primer bucle *for* de la función del control *Timer*, y se hace por medio del constructor 2 de dicha clase (código fuente 5.24).Los parámetros que necesita el constructor se encuentran en la tabla 5.4.

Se puede observar que al llamar a la clase "FondoOscilador", en el código fuente 5.37, pasa como parámetro la matriz "CapaOscilador", que es la que contiene a los *PictureBox* sobre los cuales se dibujará cada imagen del oscilador (figura 5.18a) desplazado de su eje según corresponda la respuesta dinámica.

5.3.2.2. Oscilador de VGL

Para crear el efecto de animación de este oscilador se sigue el mismo procedimiento descrito para los osciladores de 1GL. La forma de dibujarlo es por medio de dos *PictureBox*, el proceso se describió en la sección 5.2.2.2.

Una vez que el usuario hace clic en el botón *Play* en el panel de controles, el programa llama a la función *Timer* (código fuente 5.42); esta función y la mostrada en el código fuete 5.37 se declaran en diferentes ventanas (osciladores 1GL y VGL) por lo tanto no tienen relación entre ellos.

El código fuente 5.42 es el encargado de crear la animación de las señales y el oscilador en la ventana de Osciladores de VGL. La función tiene las mismas características y comportamiento que la función mostrada en el código fuente 5.37.

La primera instrucción dentro de la sentencia *if*, es para llamar a la clase "FondoOsciladorVGL", por medio de su constructor 2 (código fuente 5.31); obsérvese que pasa como parámetro la matriz "CapaOscilador" que es la que contiene a los *PictureBox* sobre los cuales se dibujará cada imagen del oscilador (masas y resortes).

En el bucle *for*, se encuentra la sentencia para llamar a la clase "GráficarSeñal" a través de su constructor 6 (código fuente 5.38). Esta parte de la función es exactamente igual a la descrita en el código fuente 5.37 y su función es también crear la animación de las señales de respuesta.

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer1.Tick

If n < maxpuntos Then    'Verifica que n sea menor al número de puntos de la señal

'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
   Dim ObjetoFondo1 As New FondoOsciladorVGL(CapaOscilador, "[" & paso & "] Punto", tamañoM,
PuntosAux, Valores, NGL, Me.ChBMostrarDesp.Checked, paso)

'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
   For Me.i = 1 To Me.NUDGráficasAct.Value
       Dim ObjetoGráfica As New GráficarSeñal(AreaGráfica(i), BtmpGraf(i), Rr(i)(n).X,
Rr(i)(n).Y, Rr(i)(n + 1).X, Rr(i)(n + 1).Y, LapizGráfica(i), Grueso(i))
   Next

   n += Me.NUDincremento.Value  'Contador de puntos de la señal
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Else                            'En caso de llegar al numero de puntos de la señal
   Call DetenerAnimacion(True)  'Llama a la función para detener la animación
End If                          'Fin de la condición

End Sub                         'Fin de la función Timer
```

Código fuente 5.42 Evento *Tick* del control *Timer1*

5.3.3. Espectros

Sólo los espectros de respuesta tienen efecto de animación, mientras que los espectros de piso se muestran estáticos a lo largo del programa y aparecen cuando el usuario hace clic sobre alguna de las masas del oscilador de VGL.

Los espectros de respuesta presentan el tipo de animación por adición. Se dibuja en un *PictureBox* la imagen base (figura 5.19a) y para cada imagen consecutiva se va agregando la línea que une el punto siguiente en el espectro de respuesta a cada intervalo de tiempo (figura 5.19b, c y d).

Debido a que el movimiento de los espectros está sincronizado con la animación de las señales, el llamado a las clases para dibujar ambas gráficas (señales y espectros) se encuentra dentro del control *TmEspectros*, (código fuente 5.40).

El llamado a la clase "FondoEspectro" se encuentra dentro del primer bucle *for* de la función, y se hace por medio del constructor 3 de dicha clase (código fuente 5.43).Los parámetros que necesita el constructor se encuentran en la tabla 5.10.

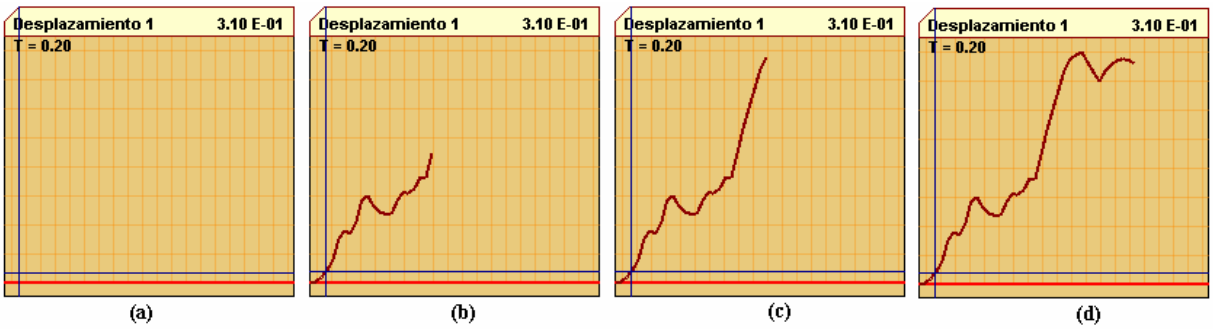

Figura 5.19 Dibujo por adición del espectro

Para que el constructor 3 dibuje sobre el *PictureBox* es necesario establecer un área de dibujo, (como se ha mencionado anteriormente). En este caso se observa que el *Bitmap* pasa como parámetro de referencia del constructor; esto quiere decir que la variable *Btmp* almacena una imagen previa, y además al pasar por valor de referencia la variable guarda cada cambio que sufra la misma.

Es decir, el constructor 3 dibuja una línea sobre una imagen definida (figura 5.19a), después la misma variable *Btmp* guarda la imagen modificada y se repite el proceso en cada ciclo hasta que el dibujo del espectro esté terminado.

Tabla 5.10 Parámetros del constructor 3 de la clase "FondoEspectro"

Variable	Descripción
Pic	Nombre del *PictureBox* donde se va a dibujar la gráfica
Btmp	Imagen contenida en un mapa de bits (*Bitmap*)
(X1, Y1)	Coordenada del punto inicial de la línea que se va dibujar
(X2, Y2)	Coordenada del punto final de la línea que se va dibujar

Al ver este proceso ejecutarse de manera secuencial en intervalos de tiempo pequeños entre las imágenes, se obtiene el efecto de movimiento de la señal del espectro y pareciera como si recorriera su trayectoria en el espectro.

```
'Constructor 3 Dibuja la gráfica del espectro POR PASOS
Sub New(ByVal pic As PictureBox, ByRef Btmp As Bitmap, ByVal X1 As Single, ByVal Y1 As
Single, ByVal X2 As Single, ByVal Y2 As Single)

Dim PuntosAux(1) As System.Drawing.PointF    'Se crea un par de puntos(X,Y) para el inicio y
                                              final de la línea a dibujar
Dim mapa As Graphics = pic.CreateGraphics()  'Crea un área de trabajo a partir del PictureBox
pic.Image = Btmp                             'La imagen del PictureBox será el Bitmap que
                                              guardo la imagen base
mapa = Graphics.FromImage(Btmp)              'Permite un buen efecto de animación en cada
                                              cambio de dibujo
PuntosAux(0) = New PointF(X1, Y1)            'Se definen las coordenadas del punto inicial
PuntosAux(1) = New PointF(X2, Y2)            'Se definen las coordenadas del punto final

mapa.DrawLines(lapiz3, PuntosAux)            'Se dibuja la línea
End Sub                                      'Fin del constructor
```

Código fuente 5.43 Constructor 3 de la clase *FondoEspectro*

5.4. MANEJO DE ERRORES

Se contemplaron dos tipos errores: lógico y de ejecución. Estos últimos son los más notorios para el usuario pues el programa muestra un mensaje de error cuando se producen.

5.4.1. Errores de ejecución

Los errores de ejecución se controlaron mediante la restricción de datos de entrada en los campos de texto. El programa está diseñado para detectar en qué momento se introducen valores erróneos en los controles *Textbox*, *DataGridView* y *RichTexbox*.

El software realiza el proceso de recibir los datos, evaluarlos y determinar si son aceptables o no; en el caso de ser datos incorrectos se envía un mensaje de advertencia y se suspende el proceso (que se esté llevando a cabo en ese momento) hasta que se corrija el dato erróneo. Para efectuar el proceso se usaron los mensajes de advertencia junto con la validación de los campos de texto; estos puntos se describen en las secciones 5.4.3 y 5.4.4 de este capítulo.

5.4.2. Errores lógicos

Estos fueron resueltos en la depuración del programa, a través del proceso de "prueba y error", es decir, durante su elaboración el software se ponía a prueba para cerciorarse de que funcionara adecuadamente cada vez que se agregaba un control a la interfaz gráfica y/o modifica el código fuente.

El programa es confiable en cuanto al procesamiento de datos (respuesta dinámica) ya que ésta fue verificada con otros programas y cálculos en libros; esto se describe en el capítulo siguiente.

Con respecto al uso de los controles, la depuración continua ayudó a establecer perfectamente el uso adecuado de cada uno en el programa; además, el software fue proporcionado a los alumnos de la clase de Dinámica Estructural para su valoración, esto se describe también en el siguiente capítulo.

El programa puede sufrir errores debido a un mal manejo, en cuyo caso el software terminará su ejecución debido a que dichas acciones no están contempladas dentro de su uso. No obstante, la GUI está diseñada de manera tal que es poco probable que se presenten errores debido a estas situaciones.

5.4.3. Mensajes de advertencia

Los Message Box (*MsgBox*) son controles predeterminados de Visual Studio para mostrar un mensaje al usuario. Su uso es muy variado y depende propiamente de cómo los utilice el programador.

Su función (en el presente trabajo) es mostrar un mensaje de texto al usuario, que indica que ha habido un error en la introducción de datos y advierte al usuario de que es o no posible llevar a cabo la operación.

La figura 5.20 muestra un *MsgBox* que aparece en el programa cuando se intenta introducir datos erróneos en campos de texto numéricos. El formato de este control es estándar y las partes que lo componen se indican debajo de la figura.

Cada elemento del MsgBox señalado en la figura 5.20 lo puede editar el programador. Los mensajes que se muestran a lo largo del programa dependen del tipo de error en el que se incurra, el icono expuesto se utilizó siempre para los mensajes de advertencia o aviso, y los botones en la parte baja del formulario dependen de las opciones para resolver dicho problema.

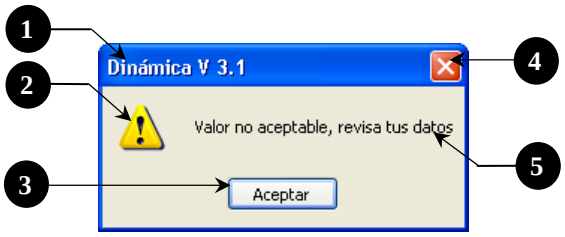

Figura 5.20 Ventana de MsgBox

#	Control	Descripción
1	Título	Muestra el nombre del programa y su versión
2	Icono	Figura predeterminada de Visual Studio para indicar una advertencia
3	*Button*	Botón de opción del mensaje, en algunos casos mostrará más de uno con opciones como: Aceptar, Cancelar y/o Reintentar
4	*Button*	Botón para cerrar la ventana
5	*Label*	Mensaje que muestra cuál es el error en el programa

Para mostrar el *MsgBox* de la figura 5.20 en el programa, se requiere del código fuente 5.44; esta instrucción es general para cualquier Msgbox que se desee mostrar:

```
MsgBox("Valor no aceptable, revisa tus datos", MsgBoxStyle.Exclamation, "Dinámica V 3.1")
```

Código fuente 5.44 Llamado de un MsgBox

La sentencia *MsgBox* ( ) muestra al control; dentro del paréntesis se indican los parámetros, separados por comas, para cada parte del mismo, de izquierda a derecha son:

- ✓ Mensaje de texto a mostrar en el control
- ✓ Tipo de icono
- ✓ Título de la ventana

Se puede observar que en ninguna parte del código fuente 5.44 se especificaron los botones que se deseaba ver; Visual Studio coloca por default el botón aceptar (parte baja de la ventana) y el botón para cerrar la ventana (parte superior derecha).

Para definir los botones a mostrar en la ventana se hace en conjunto con la indicación del icono agregando el signo "+" y el grupo de botones a mostrar. El código fuente 5.45 expone cómo se agregan los botones "Ok" y "Cancel" a un *MsgBox*.

```
MsgBox("La señal tiene más de 12000 datos " & vbCrLf & "¿Desea cargar los datos?",
MsgBoxStyle.Exclamation + MsgBoxStyle.OkCancel, version)
```

Código fuente 5.45 Llamado de un MsgBox con especificación de botones

Específicamente los *Msgbox* se programaron para aparecer cuando se presenta cualquiera de los siguientes casos:

- ✓ Por la mala escritura de un número, por ejemplo: en vez de escribir *12.34* escribe *12.3.4*.
- ✓ Cuando los valores que introduce están fuera de un rango, tal es el caso de los valores de amortiguamiento ($0 < \xi \leq 1$) y el valor de los periodos ($T \geq 0$) por mencionar algunos.
- ✓ Cuando los resultados de las operaciones con dichos datos es errónea. Por ejemplo, al calcular el número de puntos (Np) de la señal tipo Senoide esté fuera de rango ($50 \leq Np \leq 12000$).

Los mensajes de advertencia aparecen sólo en los controles que admiten valor, es decir, cuando contienen un campo de texto (*Textbox*, *DataGridView* o *RichTexbox*). Debido al diseño de la GUI son pocas las ventanas que tienen esta característica, a lo largo del programa el usuario encontrará diversos mensajes que le indicaran cuando se presente un error.

Como ejemplo de los mensajes de advertencia, la tabla 5.11 muestra los que la ventana "Senoide" contiene.

Tabla 5.11 Mensajes de error en la ventana "Senoide"

Ventana	Control	Causa de error	Mensaje
Senoide	*TextBox*	(Amplitud, Duración, Periodo y Δt) < 0	Ninguno de los datos puede ser menor o igual a CERO
		Número de puntos: Np ($50 < Np < 12000$)	La señal debe tener como mínimo 50 puntos y como máximo 12000 puntos
		Escribir mal algún valor	Valor no aceptable, revisa tus datos

Como se mencionó anteriormente, el programa puede contener ciertos errores debido a un mal uso. Cuando esto suceda, Visual Studio abrirá una ventana de mensaje de error predeterminada (figura 5.21), la cual muestra el motivo de dicho error (explicado en términos de lenguaje de programación) y da opciones de continuar o salir del programa en ese momento.

Figura 5.21 Ventana de error no controlado

5.4.4. Validación de los campos de texto

Validar significa (en términos de programación) aprobar los datos que se introducen en un control, a partir de ciertos parámetros establecidos.

Para validar los datos que se introducen en los campos de texto de los controles *TextBox* y *DataGridView* de cualquier ventana dentro del programa se creó la clase "Validación" (código fuente 5.46).

La clase "Validación" presenta 4 constructores, uno para ingresar los datos provenientes del *TextBox* y los otros tres para el control *DataGridView*. Además, contiene cuatro funciones (eventos de validación) para el control *TextBox* y tres funciones (un evento de validación, una función del tipo *Public* y otra *Private*) para el control *DataGridView*.

```
Public Class ClaseValidacion
    WithEvents TextBox As New TextBox       'Recibe todos los argumentos del Textbox
    WithEvents DataGrid As New DataGridView 'Recibe todos los argumentos del DatagGridView

    Dim TextoValido As String               'Caracteres validos para el textBox
    Dim RevisionEntero As Boolean           'Indica si revisa que el dato sea de tipo entero
    Dim ValorMinimo, ValorMaximo As Single  'Valores que delímitan al rango
    Dim RepetirValor As Boolean             'Indica si buscan valores repetidos en la fila
    Dim LimitarValorMaximo As Boolean
    Dim LimitarValorMinimo As Boolean

    '1er Constructor: TEXTBOX
    Sub New(ByVal TextBox As TextBox, ByVal TextoValido As String)

    '2do Constructor: DATAGRIDVIEW
    Sub New(ByVal DataGrid As DataGridView, ByVal Entero As Boolean, ByVal ValorMinimo As
Single, ByVal ValorMaximo As Single, ByVal RepetirValorFila As Boolean)

    '3ro Constructor: DATAGRIDVIEW
    Sub New(ByVal DataGrid As DataGridView, ByVal Entero As Boolean, ByVal ValorMinimo As
Single, ByVal RepetirValorFila As Boolean)

    '4to Constructor: DATAGRIDVIEW
    Sub New(ByVal DataGrid As DataGridView, ByVal Entero As Boolean, ByVal RepetirValorFila
As Boolean)

    '************************** VALIDACION DEL TEXTBOX ************************

    Private Sub TextBox_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles TextBox.KeyPress

    Private Sub TextBox_Validating(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles TextBox.Validating

    Private Sub TextBox_GotFocus(ByVal sender As Object, ByVal e As System.EventArgs)
Handles TextBox.GotFocus

    Private Sub TextBox_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
TextBox.Click

    '************************** VALIDACION DEL DATAGRID ************************

   Private Sub DataGridView_CellValidating(ByVal sender As Object, ByVal e As
System.Windows.Forms.DataGridViewCellValidatingEventArgs) Handles DataGrid.CellValidating

   Public Sub DetenerValidacionDataGrid()

   Private Sub MensajeError(ByVal Mensaje As String, ByVal fila As Integer)
```

Código fuente 5.46 Clase Validación

Al principio de la clase "Validación" se declararon dos variables del mismo tipo que los controles en cuestión (*TextBox y DataGrid*); esto con la finalidad de pasar todos los valores y reconocer los eventos de dichos controles en la clase.

Los cuatro constructores sólo tienen la finalidad de pasar los valores de los parámetros que reciben a las variables globales definidas en la parte superior de la clase con la palabra clave *Dim*.

El código fuente 5.47 muestra el contenido de los cuatro constructores. El primero es para la validación de datos del *TextBox;* los parámetros que recibe son: el nombre del control y el texto valido o permitido para ese control.

En los constructores para los DataGridView se requieren más parámetros, ya que se revisan las siguientes condiciones para los valores introducidos en las celdas:

- ✓ Si es número entero o decimal
- ✓ Si requiere cumplir con un valor mínimo
- ✓ Si requiere cumplir con un valor máximo
- ✓ Si se permite repetir el mismo valor en la fila

```
     '1er Constructor: TEXTBOX
    Sub New(ByVal TextBox As TextBox, ByVal TextoValido As String)
        Me.TextBox = TextBox
        Me.TextoValido = TextoValido
    End Sub

    '2do Constructor: DATAGRIDVIEW Revisar valores máximos y mínimos y repeteción de datos
    Sub New(ByVal DataGrid As DataGridView, ByVal Entero As Boolean, ByVal ValorMinimo As
Single, ByVal ValorMaximo As Single, ByVal RepetirValorFila As Boolean)
        Me.DataGrid = DataGrid
        Me.RevisionEntero = Entero
        Me.ValorMaximo = ValorMaximo
        Me.ValorMinimo = ValorMinimo
        Me.RepetirValor = RepetirValorFila
        Me.LimitarValorMaximo = True
        Me.LimitarValorMinimo = True
    End Sub

    '3er Constructor: DATAGRIDVIEW Revisar valores mínimos y repeteción de datos
    Sub New(ByVal DataGrid As DataGridView, ByVal Entero As Boolean, ByVal ValorMinimo As
Single, ByVal RepetirValorFila As Boolean)
        Me.DataGrid = DataGrid
        Me.RevisionEntero = Entero
        Me.ValorMinimo = ValorMinimo
        Me.RepetirValor = RepetirValorFila
        Me.LimitarValorMaximo = False
        Me.LimitarValorMinimo = True
    End Sub

    '4to Constructor: DATAGRIDVIEW Revisar repeteción de datos
    Sub New(ByVal DataGrid As DataGridView, ByVal Entero As Boolean, ByVal RepetirValorFila
As Boolean)
        Me.DataGrid = DataGrid
        Me.RevisionEntero = Entero

        Me.RepetirValor = RepetirValorFila
        Me.LimitarValorMinimo = False
        Me.LimitarValorMaximo = False
    End Sub
```

Código fuente 5.47 Constructores de la clase Validación

Los parámetros que requieren los constructores de la clase "Validación" (para el control *Data Grid View*) se muestran en la tabla 5.12. Obsérvese que cada constructor sólo pasa los valores de sus parámetros a las variables globales.

Los tres constructores para el control *DataGridView* requieren como parámetros, el nombre del control, la variable condicional para revisar si el valor es número entero y también la variable condicional para permitir la repetición del valor en la misma fila. La diferencia entre los constructores es que el primero requiere de un valor máximo y un mínimo, el segundo sólo requiere el valor mínimo y el tercero no requiere valores.

Tabla 5.12 Parámetros de los constructores (2, 3 y 4) de la clase Validación

Variable	Descripción
DataGrid	Nombre del DataGridView que recibe el valor
Entero	Condición (True o False) que indica si revisa que el número sea entero o no
ValorMínimo	Valor numérico mínimo que debe cumplir el valor introducido
ValorMaximo	Valor numérico máximo que debe cumplir el valor introducido
RepetirValorFila	Condición (True o False) que indica si permite la repetición del valor introducido en la misma fila.

La revisión de los datos se hace a través de las funciones (eventos de validación) de cada control. Para el *TextBox* se muestran en el código fuente 5.48 el cuerpo de dichas funciones y en la tabla 5.13 el momento cuando se activa cada evento.

```
'****************************** VALIDACION DEL TEXTBOX *********************************

    Private Sub TextBox_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles TextBox.KeyPress
        If InStr(Me.TextoValido & Chr(8), e.KeyChar) = 0 Then
            e.Handled = True
            Beep()
        End If
    End Sub

    Private Sub TextBox_Validating(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles TextBox.Validating
        If Not IsNumeric(Me.TextBox.Text) Then
            MsgBox("Valor no aceptable, rebice sus datos", MsgBoxStyle.Exclamation, version)
            e.Cancel = True
        End If
    End Sub

    Private Sub TextBox_GotFocus(ByVal sender As Object, ByVal e As System.EventArgs)
Handles TextBox.GotFocus
        Me.TextBox.SelectionStart = 0
        Me.TextBox.SelectionLength = Len(Me.TextBox.Text)
    End Sub

    Private Sub TextBox_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
TextBox.Click
        Me.TextBox.SelectionStart = 0
        Me.TextBox.SelectionLength = Len(Me.TextBox.Text)
    End Sub
```

Código fuente 5.48 Eventos de validación del control *TextBox*

El evento *KeyPress* (código fuente 5.48) del control *TextBox* permite sólo escribir en el control los caracteres permitidos, contenidos en el parámetro “TextoValido” más la tecla “Delete”, bloqueando todos los demás.

El evento *Validating* del control *TextBox* verifica que el texto introducido sea un valor numérico; en caso de no serlo el programa muestra un *MsgBox* con el mensaje "Valor no aceptable, revisa tus datos", y mediante la instrucción ***e.cancel***, se especifica que el control no pierda el enfoque hasta corregir el valor erróneo.

Los eventos *Got Focus* y *Click* seleccionan el texto en el control *TextBox* cuando se hace clic en él.

Tabla 5.13 Eventos para el control *TextBox*

Evento	Se activa cuando …
KeyPress	se introduce un dato cualquiera
Validating	se hace clic fuera del control
Got Focus	el control obtiene el enfoque
Click	se hace clic en el control

El *DataGridView* sólo requiere de un evento de validación llamado *CellValidating*, (código fuente 5.49); éste se activa cuando el usuario introduce un valor en cualquiera de las celdas del control. En el cuerpo de la función se revisan cinco condiciones con la ayuda de la sentencia *if*.

La primera condición verifica que el valor introducido sea numérico; de no serlo el programa llama a la función “MensajeError” para mostrar un MsgBox con el mensaje " El dato no es un valor numérico"; después con la instrucción *Me.DataGrid.CancelEdit* ( ) se borra el valor erróneo en la celda y se recupera el valor correcto anterior, al ejecutar esto el programa sale de la función.

En caso de pasar la primera verificación (que el dato sea numérico), el programa revisa cuatro condiciones más, dependiendo de los valores capturados en las variables globales.

El segundo *if* sirve para comprobar si el valor numérico es del tipo entero (en caso de que el parámetro del constructor haya especificado “Entero” = *True*). El *if* muestra las mismas líneas de código que la primera condición, en caso de no ser un entero el programa llama a la función “MensajeError” especificando el mensaje como: "El valor debe ser un valor ENTERO”.

El tercer *if* evalúa (si “Me.LimitarValorMinimo” = *True*) que el valor introducido en la celda sea mayor que el mínimo especificado. Al igual que los dos *if* anteriores, de no cumplir con la condición el programa llama a la función “MensajeError” con el texto "El valor MÍNIMO es: " seguido del valor que se especificó como mínimo en los parámetros de constructor.

El cuarto *if* hace lo propio con el valor máximo (en el caso de que “Me.LimitarValorMaximo” = *True*) llamando a la función “MensajeError” con el texto "El valor MAXIMO es: " seguido del valor que se especificó en los parámetros de constructor.

El último *if* restringe la repetición del valor introducido en la misma fila (en caso de que “Me.RepetirValor” = *True*) y de igual forma que los anteriores llama a la función “MensajeError” con el texto "No se pueden repetir los valores en la tabla".

```
    Private Sub DataGridView_CellValidating(ByVal sender As Object, ByVal e As
System.Windows.Forms.DataGridViewCellValidatingEventArgs) Handles DataGrid.CellValidating
        Dim Valor As String = e.FormattedValue.ToString() 'Dato que se introdujo
        Me.DataGrid.Rows(e.RowIndex).ErrorText = ""       'Mensaje de error para la fila

        '1) Primera revisión {DATO NUMERICO}
        If Not IsNumeric(Valor) Then
            e.Cancel = True
            MensajeError("El dato no es un valor númerico", e.RowIndex)
            Me.DataGrid.CancelEdit() 'Deshace el cambio de valor en la celda
        Else
            '2) Segunda revisión {DATO NUMERICO DEL TIPO ENTERO}
            If Me.RevisionEntero = True Then
                Dim ValorRes As Integer
                If Not Integer.TryParse(Valor, ValorRes) Then 'Solo numeros enteros
                    e.Cancel = True
                    MensajeError("El valor debe ser un valor ENTERO", e.RowIndex)
                    Me.DataGrid.CancelEdit()'Deshace el cambio de valor en la celda
                    Exit Sub
                End If
            End If

            '3) Tercera revisión {VALOR MINIMO}
            If Me.LimitarValorMinimo = True Then
                If Valor < ValorMinimo Then
                    e.Cancel = True
                    MensajeError("El valor MÍNIMO es: " & ValorMinimo, e.RowIndex)
                    Me.DataGrid.CancelEdit()'Deshace el cambio de valor en la celda
                    Exit Sub
                End If
            End If

            '4) Cuarta revisión {VALOR MAXIMO}
            If Me.LimitarValorMaximo = True Then
                If Valor > ValorMaximo Then
                    e.Cancel = True
                    MensajeError("El valor MÁXIMO es: " & ValorMaximo, e.RowIndex)
                    Me.DataGrid.CancelEdit()'Deshace el cambio de valor en la celda
                    Exit Sub
                End If
            End If

            '5) Quinta revisión {REPETICIÓN DE VALOR EN LA MISMA FILA}
            If Me.RepetirValor = True Then
                For i = 0 To Me.DataGrid.ColumnCount - 1
                    If CInt(Valor) = Me.DataGrid.Item(i, e.RowIndex).Value Then
                        If i <> e.ColumnIndex Then
                            e.Cancel = True
                            MensajeError("No se pueden repetir los valores en la tabla",
e.RowIndex)
                            Me.DataGrid.CancelEdit()'Deshace el cambio de valor en la celda
                            Exit Sub
                        End If
                    End If
                Next
            End If
        End If
    End Sub

    Public Sub DetenerValidacionDataGrid()
        Me.DataGrid.CancelEdit()'Deshace el cambio de valor en la celda
    End Sub

    Private Sub MensajeError(ByVal Mensaje As String, ByVal fila As Integer)
        Beep()
        Me.DataGrid.Rows(fila).ErrorText = Mensaje
        MsgBox(Mensaje, MsgBoxStyle.Exclamation, version)
    End Sub
```

Código fuente 5.49 Funciones del control DataGridView

El código fuente 5.50 muestra como se implementa la clase "Validación" sobre los controles *TextBox* y *DataGridView* de la ventana Espectros de respuesta. Se puede observar en la sección 4.2.5.7 del capítulo anterior que la ventana presenta los dos tipos de controles.

Primero se declara un objeto de la clase "Validación" al principio del código. Después en el evento *Load* del formulario (ventana), dentro de una sentencia *if*, se hace la referencia completa a la clase. La variable "ObjetoData" invoca a la clase validación mediante su constructor 4 (mostrado en el código fuente 5.47) indicando en sus parámetros el nombre del control *DataGridView* (Me.DGEspectros), que no haga la revisión de numero entero (Entero = *False*) y tampoco que haga la revisión para repetir el mismo valor en la fila (RepetirValor = *False*).

Después, se declaran tres objetos de la clase "Validación" (ObjetoText1, ObjetoText 2 y ObjetoText 3) para cada control *TextBox* que se encuentre en la ventana, pasando como parámetros: el nombre del control *Textbox* (según corresponda) y entre comillas los caracteres validos para ese control (0.123456789); en este caso se trata sólo de caracteres numéricos.

De esta manera quedan relacionados los controles con la clase "Validación". Cada vez que se introduzca valores en los controles mencionados los eventos de dichos controles se activarán y ejecutarán como se describió anteriormente.

```
Dim ObjetoData As ClaseValidacion 'Objeto para la validación del DataGrid

Private Sub FrmEspectros_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
……
……….

If ObjetoData Is Nothing Then
    ObjetoData = New ClaseValidacion(Me.DGEspectros, False, False)

   Dim ObjetoText1 As New ClaseValidacion(Me.TbFinal, "0.123456789")
   Dim ObjetoText2 As New ClaseValidacion(Me.TbInicio, "0.123456789")
   Dim ObjetoText3 As New ClaseValidacion(Me.TbIntervalo, "0.123456789")
End If
……
……
……….
End Sub
```

Código fuente 5.50 Llamado de la Clase Validación desde la ventana Espectros

Los puntos (…….) dentro del código fuente 5.50 indican que hay más instrucciones propias de cada ventana pero no es necesario mostrarlas para efectos de este ejemplo.

5.4.5. Controles Try Cath

Para la validación de los datos en el control *RichTextBox* de la ventana "AbrirArchivo" se implementó otro tipo de revisión. Ésta se hizo con la ayuda del bloque de código *Try Cath*, conocido también como controlador estructurado de errores y se utiliza (en el programa) para detectar errores al momento de leer el archivo de datos.

El código fuente 5.51 muestra las instrucciones para leer un archivo de texto; dentro del bucle *for* se encuentra el bloque de código *Try Cath*. Primero, el programa lee cada valor del archivo de texto con la instrucción *Input* ( ), en el caso de que la lectura de datos llegue a su máximo valor (12000 datos por columna) el programa por medio de la instrucción *Catch When Err.Number = 9* muestra un *MsgBox* para advertir al usuario de que el archivo contiene más datos de los permitidos.

La instrucción *Err.Number = 9* corresponde a un error de desbordamiento de datos en el vector donde se almacenan los valores, pero antes de que se produzca el error la instrucción *Cath When* captura el error y ejecuta las instrucciones que se muestran debajo de esa línea de código.

En este caso el programa muestra al usuario un *MsgBox* con dos botones (*Ok y Cancel*). Si la elección del usuario es *Ok*, el programa acepta la lectura de datos capturada hasta ese momento y finaliza de inmediato la lectura del archivo; en el caso de seleccionar *Cance*l, el programa rechaza los datos capturados y detiene también la lectura del archivo.

```
Do Until EOF(1)                          'Lee hasta el final del archivo
For i = 1 To NUDColumnas.Value           'Lee valores por columnas
    Me.Longitudes(i) = j                 'Determina la longitud de los vectores

    Try                                  'Prueba si hay datos disponibles
        Input(1, LecturaValores(i)(j)) 'Los guarda en un vector
    Catch When Err.Number = 9            'Cuando el indice está fuera del intervalo de puntos
                                         'Envia mensaje de texto
        If MsgBox("La señal tiene más de 12000 datos " & vbCrLf & "Desea cargar los datos",
MsgBoxStyle.Exclamation + MsgBoxStyle.OkCancel, version) = MsgBoxResult.Cancel Then
           Me.ArchivoCorrecto = False  ' El usuario selecciona el Boton Cancelar
        Else
           Me.ArchivoCorrecto = True   ' El usuario selecciona el Boton Aceptar
        End If

        Me.Longitudes(i) = j - 1         'Se obtiene el valor la longitud de cada vector
        FileClose(1)                     'Cierra el archivo de texto
        Exit Sub                         'Sale completamente de la función
……
…….
    End Try                              'Termina la función Try Catch
Next                                     'Siguiente columna de datos
j += 1                                   'Siguiente Línea del archivo de texto
Loop                                     'Finaliza la lectura del archivo
FileClose(1)                             'Cierra el archivo
```

Código fuente 5.51 Bloques de código Try Catch

El uso del controlador de errores es muy útil para descubrir y reconocer errores durante la ejecución del programa, justo en el momento en que ocurren; además suprime mensajes de error no deseado, como el mostrado en la figura 5.21 y ajusta las condiciones del programa de manera que la aplicación pueda retomar el control y seguir en ejecución.