Sección
Revista digital Matemática, Educación e Internet (www.cidse.itcr.ac.cr/revistamate/). Vol. 1, Nº 2. Agosto – Diciembre 2001.

Gráficos de funciones en Visual Basic

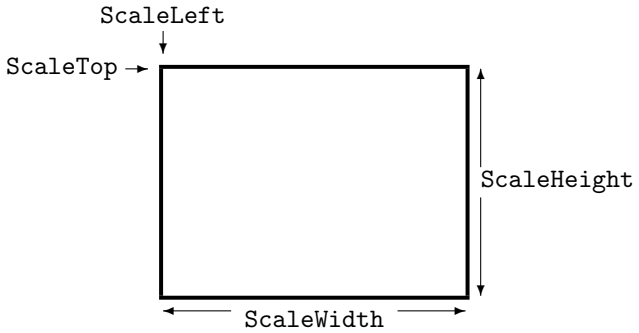
Luis Acuña P.
lacuna@itcr.ac.cr
Escuela de Matemática
Instituto Tecnológico de Costa Rica

Introducción

En esta columna desarrollamos un programa en Visual Basic que permitirá al usuario graficar una función cuadrática y observar el efecto de cada uno de los coeficientes sobre el gráfico.
Visual Basic es un ambiente de programación fundamentalmente gráfico. Todos los controles que se colocan en un formulario, y el texto que se escriba en ellos, se "dibuja" como gráficos en la pantalla. Muchos controles permiten que se muestren gráficos en ellos, como los botones de comando y los cuadros de imagen. Estos gráficos normalmente vienen de archivos cuyo contenido se muestra en el control.
Pero los formularios y los cuadros de dibujo van más allá: estos tienen propiedades y métodos diseñados específicamente para graficar en ellos. No solamente para mostrar archivos de imágenes, sino para dibujar puntos, segmentos, curvas y otras figuras geométricas.

Cuadros de dibujo

Los formularios y los cuadros de dibujo (`PictureBox`) tienen las propiedades `ScaleLeft`, `ScaleTop`, `ScaleWidth` y `ScaleHeight`, que determinan la escala de los gráficos que contendrán. Podemos pensar en los formularios y cuadros de dibujo como ventanas a un sector rectangular de un plano coordenado. Las propiedades `ScaleLeft` y `ScaleTop` dan las coordenadas de la esquina superior izquierda del sector, y las propiedades `ScaleWidth` y `ScaleHeight` dan el ancho y el alto del sector, como se ve en el siguiente diagrama:



Por omisión, los valores de `ScaleLeft` y `ScaleTop` son 0, y los de `ScaleWidth` y `ScaleHeight` son iguales a `Width` y `Height`, el tamaño del formulario o control en las unidades de medida activas (twips, pixels, cms, etc.). Pero usualmente `ScaleWidth` y `ScaleHeight` recibirán valores que dependen del sector del plano que queremos representar, independientemente del tamaño real. Por ejemplo, si en un rectángulo de 4 cm por 5 cm queremos representar el rectángulo $[3,9]\times[-1,8]$ del plano cartesiano, entonces `Width` y `Height` serán 4cm y 5cm (o su equivalente en twips, pixels, etc), pero `ScaleWidth` y `ScaleHeight` serán 6 y9.
La esquina superior izquierda tiene coordenadas (`ScaleLeft`,`ScaleTop`) y la inferior derecha tiene coordenadas (`ScaleLeft` + `ScaleWidth`,`ScaleTop` + `ScaleHeight`). Para representar un sector del plano

cartesiano con la orientación usual (x mayores a la derecha, y mayores arriba), `ScaleWidth` debe ser positivo y `ScaleHeight` negativo. Por ejemplo, haciendo `ScaleLeft` $= 5$, `ScaleTop` $= 20$, `ScaleHeight` $= -15$ y `ScaleWidth` $= 40$ se consigue:

Los valores de estas propiedades pueden asignarse durante el diseño, o durante la ejecución de dos formas: con asignaciones explícitas (`ScaleLeft = 5`, etc.) o con la instrucción

Objeto`.Scale (`*XIzq*`, `*YArriba*`) - (`*XDer*`, `*YAbajo*`)`

donde *XIzq* y *XDer* son los extremos izquierdo y derecho de la coordenada X, y *YArriba* y *YAbajo* son los extremos superior e inferior de la coordenadaY. *Objeto* es el control para el cual se define la escala; si se omite, se supone que es el formulario. Para el cuadro mostrado, la instrucción es `Scale (5, 20) - (45, 5)`.

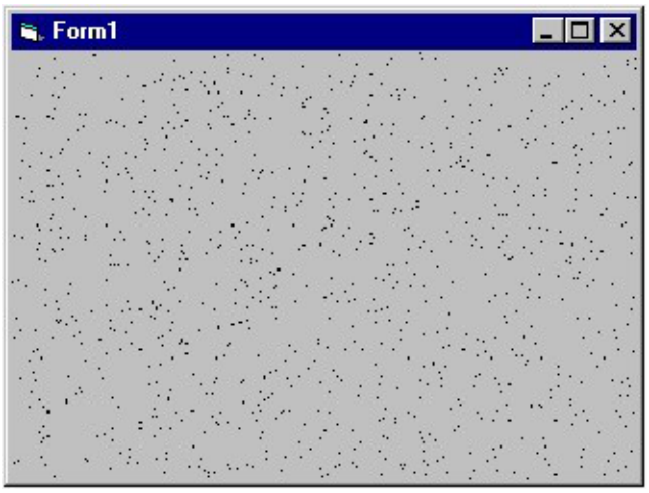
Puntos y segmentos

Para dibujar un punto en un formulario o cuadro de dibujo se usa el método `PSet`, con la sintaxis

Objeto`.PSet(`*x*`, `*y*`), `*color*

donde *x* y *y* son las coordenadas del punto, y *color* es el color deseado. El color es opcional; si no se indica se usa el `ForeColor` del objeto (como antes, si no se indica *Objeto* la instrucción se aplica al formulario). El siguiente ejemplo muestra cómo dibujar mil puntos aleatorios en el formulario cada vez que se hace click en él. En la Figura 1 vemos el resultado.

```
Private Sub Form_Click()
Dim i As Integer    ' contador
Dim x As Single, y As Single    ' coordenadas

For i = 1 To 1000   ' dibujar 1000 puntos
   x = Rnd * ScaleWidth
   y = Rnd * ScaleHeight
   PSet (x, y)
Next

End Sub
```

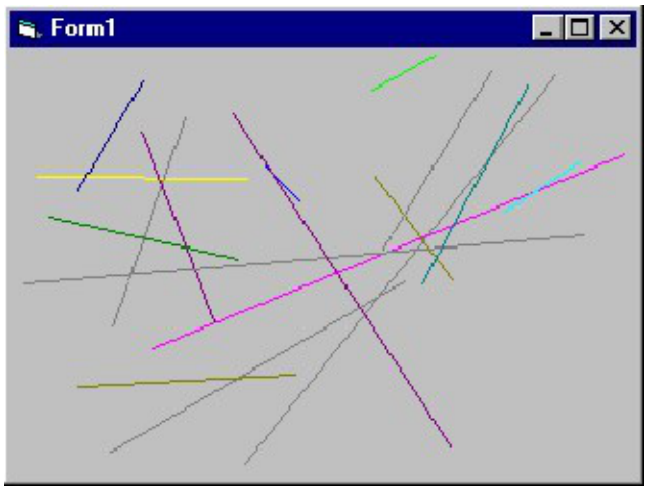


Para cada punto, sus coordenadas son números aleatorios entre 0 y el ancho o el alto del formulario. Esto distribuirá los puntos uniformemente si las coordenadas izquierda e inferior son ambas 0.
A menos que se defina otro valor para `ForeColor`, los puntos serán de color negro. Si la propiedad `AutoRedraw` del formulario se deja en su valor inicial (falso), puede notarse que al hacer más pequeño el formulario y luego volverlo a su tamaño anterior los puntos se pierden. Puede probarse lo siguiente: Correr el programa; hacer click sobre el formulario; maximizar el formulario (los puntos anteriores se mantienen); hacer click con el formulario maximizado; minimizar el formulario; maximizarlo de nuevo. En este momento se nota que los puntos que se habían dibujado la primera vez que se maximizó el formulario ya desaparecieron. En cambio, si la propiedad `AutoRedraw` se hace verdadera, todos los puntos se mantienen aunque el formulario cambie de tamaño.
Los segmentos de recta se dibujan con el método `Line`, cuya sintaxis es

Objeto`.Line (`*x1*`, `*y1*`)-(`*x2*`, `*y2*`), `*color*

donde $(x1, y1)$ y $(x2, y2)$ son las coordenadas de los extremos, y el parámetro *color* es opcional; si no se indica se usa `ForeColor`
Una forma de indicar un color aleatorio es con la expresión `RGB(255 * Rnd(), 255 * Rnd(), 255 * Rnd())`. La siguiente subrutina dibuja veinte segmentos aleatorios con color aleatorio en el formulario cada vez que se hace click en él. El resultado se muestra en la Figura 2.

```
Private Sub Form_Click()
Dim x1 As Single, y1 As Single  ' 1er punto
Dim x2 As Single, y2 As Single  ' 2do punto
Dim i As Byte                   ' contador

For i = 1 To 20     ' dibujar 20 segmentos
   x1 = Rnd * ScaleWidth
   y1 = Rnd * ScaleHeight
   x2 = Rnd * ScaleWidth
   y2 = Rnd * ScaleHeight
   Line (x1, y1)-(x2, y2), _
       RGB(255 * Rnd(), 255 * Rnd(), 255 * Rnd())
Next
End Sub
```



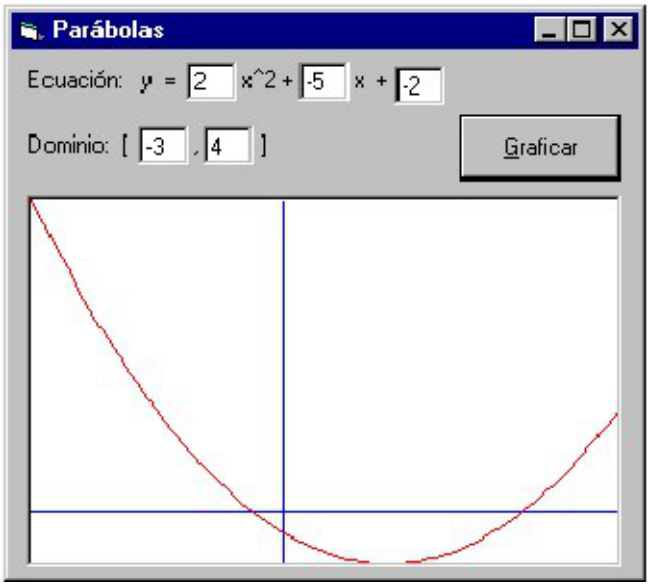
En el método `Line` es posible omitir el primer punto y escribir sólo

Objeto`.Line -(`*x2*`, `*y2*`)`

En ese caso, el punto inicial es el último punto que se había dibujado. Si en el ejemplo anterior quitamos la parte `(x1,y1)`, lo que obtenemos es una sucesión de segmentos conectados (el primer segmento empieza en $(0,0)$).

Curvas

Visual Basic no define ningún método para graficar curvas, pero la técnica usual es dibujar varios segmentos pequeños consecutivos. Vamos a desarrollar un programa que le permitirá al usuario definir la ecuación de una parábola, y le mostrará el gráfico en un cuadro de dibujo. El usuario escribe los coeficientes a, b y c en la ecuación $y = ax^2 + bx + c$ de la parábola, y también los extremos x_{min} y x_{max} del dominio. En la Figura3 vemos el resultado de este proyecto.



Los cuadros de texto con los coeficientes de x^2, de x y el constante, se llaman `txtA`, `txtB` y `txtC`. Los cuadros con los extremos del dominio se llaman `txtXmin` y `txtXmax`. El cuadro donde se graficará la función es `picGrfico`, y el botón de comando es `cmdGraficar`.

En la ventana de código definimos la función por graficar:

```
' Coeficientes de la funcin
Dim a As Single, b As Single, c As Single
' Dominio y rango del grfico
Dim Xmin As Single, Xmax As Single ' dominio
Dim Ymin As Single, Ymax As Single ' rango

Private Function f(x As Single) As Single
' La funcin por graficar
f = a * x ^ 2 + b * x + c
End Function
```

Las variables `a`, `b` y `c` son globales porque sus valores se les asignarán en `cmdGraficar_Click` y se usarán en `f`. También `Xmin`, `Xmax`, `Ymin` y `Ymax`, los extremos del dominio y del rango, son globales porque se definirán en `cmdGraficar_Click` y se usarán en `Graficar`, como veremos más tarde. A grandes rasgos, lo que `cmdGraficar_Click` debe hacer es lo siguiente:

1. Leer y validar los datos (coeficientes y dominio)
2. Calcular el rango (mínimo y máximo para y)
3. Definir la escala de `picGrfico`
4. Llamar `Graficar` para graficar los ejes y la parábola.

Como veremos, los preparativos tomarán mucho más trabajo que propiamente graficar la función.

Para validar los datos hay dos condiciones: El valor de a no puede ser cero, y x_{min} debe ser estrictamente menor que x_{max}. Si cualquiera de esas condiciones falla, la subrutina dará un mensaje de error y retornará.

Para calcular el rango necesitamos un poco de álgebra: Los puntos máximo y mínimo de una parábola restringida a un intervalo se encuentran en los extremos del intervalo (x_{min} o x_{max}) o en el vértice. El vértice tiene coordenada X igual a $x_v = -b/2a$, pero si x_v no pertenece al dominio, el máximo y el mínimo están en los extremos del intervalo. Entonces el valor máximo de y, que denotaremos y_{max}, es el mayor entre $f(x_{min})$, $f(x_{max})$ y $f(x_v)$ si $x_v \in [x_{min}, x_{max}]$, o solamente entre $f(x_{min})$ y $f(x_{max})$ si no. El cálculo de y_{min} es análogo.

Una vez determinados el máximo y el mínimo para x y para y, la escala de `picGrfico` se define con la instrucción

```
picGrfico.Scale (Xmin, Ymax)-(Xmax, Ymin)
```

(recordemos que el método `Scale` necesita la esquina superior izquierda y la inferior derecha).

Finalmente, `cmdGraficar_Click` llamará a un procedimiento `Graficar`. Éste usa las variables globales `Xmin`, `Xmax`, `Ymin` y `Ymax` y grafica la parábola en el rectángulo $[x_{min}, x_{max}] \times [y_{min}, y_{max}]$ del plano. Eso lo hace en dos pasos:

1. Graficar los ejes de coordenadas, para lo cual basta con dos instrucciones

   ```
   picGrfico.Line (0, Ymin)-(0, Ymax) picGrfico.Line (Xmin,0)-(Xmax, 0)
   ```

   Si alguno de los ejes no apareciera en el gráfico, la instrucción correspondiente no tiene ningún efecto visible.

2. Graficar la función con 100 segmentos de recta entre 101 puntos consecutivos sobre el gráfico (cien es un número razonable; se puede probar con otros números y comparar el tiempo de ejecución y la nitidez del gráfico). Para eso hace que x varíe de x_{min} a x_{max} en incrementos de $(x_{max} - x_{min})/100$, cada vez conectando el punto anterior con el nuevo punto $(x, f(x))$:

```
For x = Xmin To Xmax Step (Xmax - Xmin) / 100
picGrfico.Line -(x, f(x)) Next
```

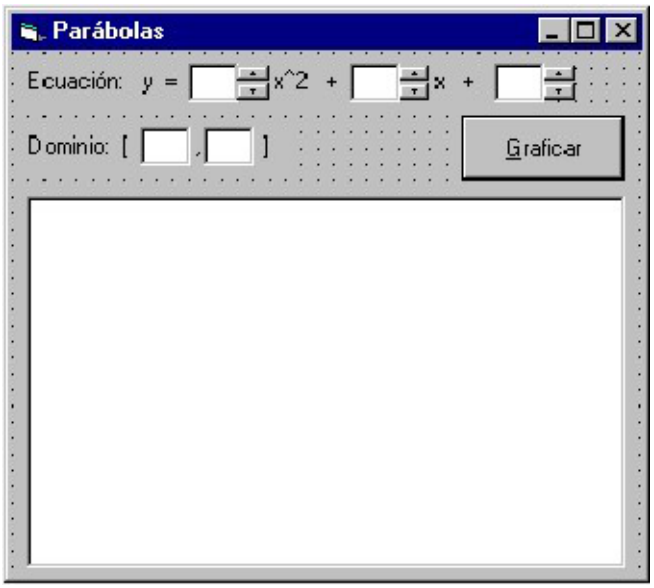
Y con eso casi terminamos, excepto por dos detalles: La pantalla debería borrarse antes de graficar. También, en el ciclo que acabamos de ver, la primera iteración dibuja un segmento hasta $(x_{min}, f(x_{min}))$, pero desde dónde? Desde el último punto anterior, que fue $(x_{max}, 0)$, al graficar el eje X. Eso no es correcto. Para evitarlo podemos dar la instrucción `PSet (Xmin, f(Xmin))` inmediatamente antes del ciclo.
Uniendo todas estas consideraciones llegamos a la siguiente forma de `cmdGraficar_Click` y `Graficar`:

```
Private Sub cmdGraficar_Click()
' Graficar la parbola

Dim Xv As Single        ' vrtice

' Leer los coeficientes y ver que a<>0
a = Val(txtA): b = Val(txtB): c = Val(txtC)
If a = 0 Then
   MsgBox "El Coeficiente de x^2 no puede"&"ser 0",vbCritical,"Error"
   txtA.SetFocus
   Exit Sub
End If

' Leer y validar los extremos del dominio
Xmin = Val(txtXmin): Xmax = Val(txtXmax)
If Xmin >= Xmax Then
   MsgBox "El dominio no es vlido", vbCritical, "Error"
   txtXmin.SetFocus
Exit Sub
End If

' Vrtice Xv = -b / (2 * a)
' Calcular Ymin = min{f(x),f(Xmin),f(Xmax)}
Ymin = f(Xmin)
If f(Xmax) < Ymin Then Ymin = f(Xmax)
If Xmin < Xv And Xv < Xmax And f(Xv) < Ymin  Then Ymin = f(Xv)

' Calcular Ymax = maxf(Xv),f(Xmin),f(Xmax)
Ymax = f(Xmin)
If f(Xmax) > Ymax Then Ymax = f(Xmax)
If Xmin < Xv And Xv < Xmax And f(Xv) > Ymax
   Then Ymax = f(Xv)

' Definir la escala del grfico
picGrfico.Scale (Xmin, Ymax)-(Xmax, Ymin)

' Graficar la parbola
Graficar
End Sub
```

```
Private Sub Graficar
Dim x As Single

' Graficar los ejes
picGrfico.Cls picGrfico.Line (0, Ymin)-(0, Ymax), vbBlue
picGrfico.Line (Xmin, 0)-(Xmax, 0), vbBlue

' Graficar la funcin
picGrfico.PSet (Xmin, f(Xmin)) ' inicio

For x = Xmin To Xmax Step (Xmax - Xmin) / 100
picGrfico.Line -(x, f(x))
Next

End Sub
```

Transformaciones dinámicas

Una posible mejora al proyecto recién completado consiste en añadir un mecanismo para que el usuario pueda cambiar los valores de a, b y c y ver inmediatamente el efecto del cambio en el gráfico. Para esto podemos poner un control `UpDown` al lado de cada coeficiente. El control `UpDown` es parte del componente "Microsoft Windows Common Controls 2". Para añadirlo a la caja de herramientas debe seleccionarse ese componente en la lista que aparece al escoger la opción **Componentes** del menú **Proyecto**.
Podemos programar el control `UpDown` para que aumente el valor del coeficiente en un 10% de su valor original. El 10% es una sugerencia; pueden probarse otros valores, y en especial debe resolverse el caso particular de que el valor inicial del coeficiente sea cero.
Vayamos aclarando las ideas. Tenemos tres coeficientes y tres cuadros de texto, y estamos a punto de añadir tres controles `UpDown`. Es mejor cambiar la nomenclatura y llamar los coeficientes `Coef(2)`, `Coef(1)` y `Coef(0)` en vez de a, b y c, respectivamente, y los cuadros de texto `txtCoef(2)`, `txtCoef(1)` y `txtCoef(0)` en vez de `txtA`, `txtB`, `txtC`. Finalmente, los controles `UpDown` se llamarán `udCoef(2)`, `udCoef(1)` y `udCoef(0)`. En la Figura4 vemos el formulario rediseñado.



Ahora tendremos un arreglo `Dim Coef(0 To 2) As Single` para los coeficientes. Pero también, para que cada incremento o decremento sea un 10% de los valores originales, necesitamos conservar copias de esos

valores en otro arreglo, que declararemos `Dim CoefOrig(0 To 2) As Single`. La sección `(General)` del formulario y la definición de la función f serán entonces así:

```
Option Explicit
' Coeficientes de la funcin
Dim Coef(0 To 2) As Single
Dim CoefOrig(0 To 2) As Single
' Dominio y rango del gr\'afico
Dim Xmin As Single, Xmax As Single
Dim Ymin As Single,
Ymax As Single

Private Function f(x As Single) As Single
' La funcin por graficar
f = Coef(2) * x^2 + Coef(1) * x +Coef(0)
End Function
```

El evento `cmdGraficar_Click()` cambia ligeramente, principalmente para acomodar los cambios en los nombres de los coeficientes:

```
Private Sub cmdGraficar\_Click()
' Graficar la par\'abola

Dim i As Byte ' contador
Dim Xv As Single ' v\'ertice

' Leer los coeficientes y ver que a<>0
For i = 0 To 2: Coef(i) = Val(txtCoef(i)): Next
If Coef(2) = 0 Then
  MsgBox "El coeficiente de x^2 no puede " & "ser 0",
    vbCritical,    "Error"
  txtCoef(2).SetFocus
  Exit Sub
End If

' Guardar los valores originales
For i = 0 To 2: CoefOrig(i) = Coef(i): Next

' Leer y validar los extremos del dominio
Xmin = Val(txtXmin): Xmax =Val(txtXmax)
If Xmin >= Xmax Then
MsgBox "El dominio no es v\'alido", vbCritical, "Error"
txtXmin.SetFocus
Exit Sub
End If

' V\'ertice: xv = -b / 2a
Xv = -Coef(1) / (2* Coef(2))

' Calcular Ymin = min{f(x),f(Xmin),f(Xmax)}
Ymin =f(Xmin)
If f(Xmax) < Ymin Then Ymin = f(Xmax) If Xmin < Xv And Xv < Xmax And f(Xv) < Ymin
Then Ymin = f(Xv)
```

```
' Calcular Ymax = max{f(Xv),f(Xmin),f(Xmax)}
Ymax = f(Xmin)
If f(Xmax) > Ymax Then Ymax = f(Xmax) If Xmin < Xv And Xv < Xmax And f(Xv) > Ymax
Then Ymax = f(Xv)

' Definir la escala del grfico
picGr\'afico.Scale (Xmin, Ymax)-(Xmax, Ymin)

' Graficar la par\'abola
Graficar
End Sub
```

La subrutina `Graficar` se mantiene, pero para responder a los cambios que el usuario quiera hacer en los coeficientes necesitamos programar los eventos `udCoef_UpClick` y `udCoef_DownClick`, que se dan cuando el usuario hace click sobre la flecha arriba o la flecha abajo de cada control `UpDown`. En cada caso, el trabajo es el siguiente:

1. Aumentar o disminuir el coeficiente respectivo en 10% de su valor original.
2. Reflejar el nuevo valor en el cuadro de texto.
3. Graficar la función con el nuevo coeficiente.

Vale la pena mencionar que el control `UpDown` tiene un mecanismo para que se le asigne un control "compañero" (con la propiedad `Buddy`) que cambiará automáticamente cuando el usuario hace click sobre el `UpDown`. Sin embargo, los cambios están restringidos a tamaños enteros, que para nuestro caso no son suficientes. Por eso estamos programando el cambio manualmente.
Lo hacemos de esta manera:

```
Private Sub udCoef_UpClick(Index As Integer)
Coef(Index) = Coef(Index) + Abs(CoefOrig(Index)) / 10
txtCoef(Index).Text = Coef(Index)
Graficar
End Sub

Private Sub udCoef_DownClick(Index As Integer)
Coef(Index) = Coef(Index) - Abs(CoefOrig(Index)) / 10
txtCoef(Index).Text = Coef(Index)
Graficar
End Sub
```

Con eso tenemos un programa sencillo (apenas 90 líneas de código) pero muy útil para ilustrar el efecto de cada coeficiente en el gráfico de una parábola. El usuario escribe los valores iniciales de los coeficientes y el dominio de la función; el programa grafica la parábola con un rango apropiado, y ahora el usuario puede modificar gradualmente los coeficientes para ver su efecto en el gráfico. Ocasionalmente, luego de varias modificaciones, el gráfico se saldrá de su cuadro, pero con sólo hacer click sobre el botón "Graficar" el programa ajustará el rango a los nuevos coeficientes.

Conclusión

Acabamos de ver un ejemplo de la facilidad con que se programan las aplicaciones gráficas en Visual Basic. El lector puede fácilmente modificar este ejemplo para aplicarlo, por ejemplo, a funciones trigonométricas, de modo que el usuario vea el efecto de los coeficientes en una función de la forma $y = a\,\mathrm{sen}(bx+c)$ o $y = a\cos(bx+c)$. El caso de líneas rectas, por supuesto, es más bien una simplificación de nuestro trabajo aquí.