Revista digital
**Matemática, Educación e Internet**
(https://tecdigital.tec.ac.cr/revistamatematica/).
Vol 20, No 1. Setiembre – Febrero, 2020

Artículo de sección
ISSN 1659 -0643

# Desarrollo de Documentos con un Formato Computable Utilizando el Software Wolfram Mathematica

## Document Development with a Computable Format Using the Wolfram Mathematica Software

Enrique Vílchez Quesada
evq1529@una.ac.cr
Escuela de Informática
Universidad Nacional de Costa Rica
Costa Rica

Juan Félix Avila Herrera
delagarita@gmail.com
Escuela de Informática
Universidad Nacional de Costa Rica
Costa Rica

Recibido: 10 julio 2018 Aceptado: 9 mayo 2019

**Resumen.** Este es un tutorial sobre la construcción de *CDF*'s (*Computable Document Format*). Se explica en detalle cómo elaborar este tipo de archivos mostrando, entre otras cosas, definiciones, ejemplos ilustrativos, y ejercicios, la descripción de los componentes visuales (o controladores) más importantes y sus opciones. Se supone que el lector tiene un conocimiento medio en el lenguaje *Wolfram*.

**Palabras clave**: *CDF*, *Wolfram*, *Mathematica*, comando **`Manipulate`**, botones, cajas de texto, cajas de chequeo, radio botones, *popmenus*, deslizadores.

**Abstract.** This is a tutorial on the construction of *CDF*'s (*Computable Document Format*). It is explained in detail how to elaborate this type of files showing, among other things, definitions, examples and exercises, the description of the most important visual components (or controllers) and their options. It is assumed that the reader has a medium knowledge in the *Wolfram* language.

**KeyWords**: *CDF*, *Wolfram*, *Mathematica*, **`Manipulate`** command, buttons, text boxes, check boxes, radio buttons, *popmenus*, sliders.

## 1.1 Introducción

En el año 2008 la empresa *Adobe Systems* revolucionó la publicación de documentos digitales con un formato estándar llamado *PDF* (portable *document* format por sus siglas en inglés). Si bien es cierto que el impacto positivo de este tipo de archivos ha sido incuestionable, permitiendo compartir todo tipo de información para su lectura, en la actualidad el funcionamiento de los *PDF*s resulta insuficiente al encontrarnos en una sociedad donde normalmente los usuarios adoptan la necesidad de convertirse en prosumidores (Islas-Carmona, 2008).

La empresa *Wolfram Research* notando estos cambios y la necesidad de requerimientos empresariales más flexibles para la toma de decisiones, desarrolló un tipo de archivo propietario denominado *CDF* (acrónimo de: *computable document format*) el cual establece un formato de documento capaz de ejecutarse de manera gratuita, en muchos casos, a través de distintos sistemas operativos. Los *CDF*'s pueden ser desplegados para su uso, mediante la instalación de un *plug in* gratuito multiplataforma (para los sistemas operativos *Windows*, *MAC* y *Linux*) disponible en la dirección *URL*: http://www.wolfram.com/cdf (Vílchez-Quesada, 2015b). La arquitectura tecnológica de un documento con un formato computable se sustenta en el poder computacional provisto por el lenguaje *Wolfram*. Con muy pocas excepciones, prácticamente todo lo que es posible desarrollar en un cuaderno convencional de *Mathematica*, será un recurso fácil de exportar como un *CDF*.

Desde un punto de vista educativo, los documentos con un formato computable facilitan la exploración de ideas y el estímulo de la intuición empleando como principal ruta didáctica la convergencia de una amplia interactividad, no necesariamente prevista al momento del desarrollo del *CDF*. Por ejemplo, el estudiante es capaz de interactuar con objetos dinámicos visualizando en tiempo real los cambios producidos por valores paramétricos, lógicos o de control secuencial (Honan, 2012). Las aplicaciones de los *CDF*'s abarcan un amplio rango de posibilidades que van desde la elaboración de un reporte automatizado hasta el diseño de un artículo científico en el que el lector pueda operar los datos del estudio.

El usuario encontrará en el presente tutorial los fundamentos básicos que deben ser considerados en el diseño de documentos con un formato computable. Se brinda un recorrido amplio sobre su forma de elaboración, atributos, exportación y ejemplos de uso. No obstante, se asume que el lector posee conocimientos previos a un nivel medio sobre el uso del software *Wolfram Mathematica*, por lo que no se explicarán aquí aspectos relacionados con las utilidades que caracterizan a esta herramienta. En el trabajo se emplea un conjunto de recursos iconográficos que facilitan su consulta, la iconografía integrada es la siguiente:


Consulte el software *Wolfram Mathematica*


Descargue un *CDF*


Descargue un *.nb*

El enfoque utilizado en este tutorial se centra en resaltar las oportunidades que los *CDF*'s brindan para la enseñanza y el aprendizaje de la matemática y otras disciplinas afines, particularmente en el desarrollo de textos digitales interactivos.

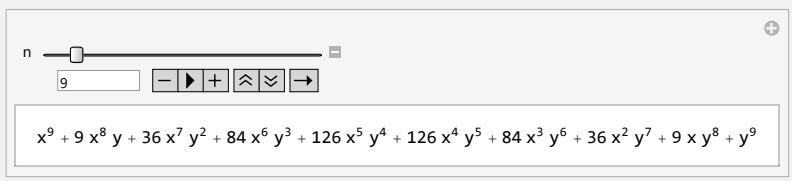
## 1.2 Comando `Manipulate`

• El comando **`Manipulate`** (puede traducirse como *Manipular*) es una instrucción nativa del software *Wolfram Mathematica* y constituye la sentencia base mediante la cual es posible crear cualquier tipo de documento con un formato computable.

**Ejemplo 1.1**

Por ejemplo, al ejecutar en el software *Mathematica*:

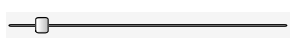
```
:= In[~]
 Manipulate[Expand[(x + y)^n], {n, 1, 100, 1, Appearance -> "Open"},
 ControlPlacement -> Top]

= Out[~]
```




Descargar CDF

La instrucción **`Manipulate`** ha construido un componente que llamaremos manipulador y que permite mostrar la expansión de un binomio con distintas potencias comenzando en 1 y terminando en 100. La secuencia "$n$,1,100,1", indica que el contador empieza en 1, termina en 100 y se incrementa en una unidad en cada iteración. La opción **`Appearance`** `->` **`"Open"`** muestra los controladores de animación del manipulador (en este caso, correspondientes a los valores que va tomando el parámetro) y **`ControlPlacement`** `->` **`Top`** especifica la ubicación de la barra deslizadora en la parte superior.

N Es importante aclarar que *Wolfram* cuenta con dos tipos de controladores muy similares: los *manipuladores* y los *deslizadores*. Ambos usan un elemento de la forma:



Sin embargo, el *deslizador* o *slider* no presenta las siguientes opciones de animación:

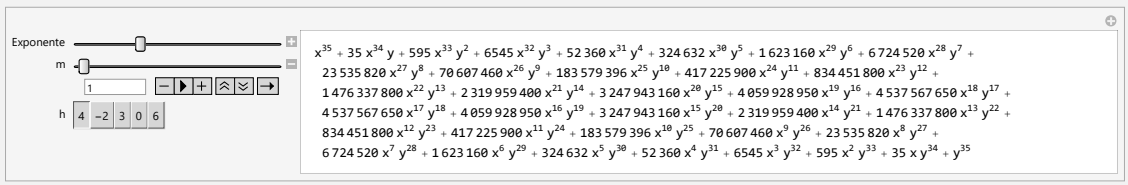
En ellas, el "−" resta un incremento a la variable $n$, el "+" suma un incremento, el *play* activa de manera automática las variaciones y finalmente los otros botones facilitan definir una velocidad de recorrido y su dirección.

- **Manipulate** [Expresión, {Parámetro, Valor inicial, Valor mínimo, Valor máximo}] : da un valor inicial al parámetro con el cual comenzará a mostrarse el objeto dinámico indicado por Expresión.
- **Manipulate** [Expresión, {{Parámetro, Valor inicial, "Etiqueta"}, ⋯ }]: añade una etiqueta.
- **Manipulate** [Expresión, {Parámetro, {Valor 1, Valor 2, Valor 3, ⋯ }}]: agrega valores discretos al parámetro, esta opción en *Wolfram Mathematica* se denomina *SetterBar*.
- **Manipulate** [Expresión,{Parámetro 1 , ⋯ }, {Parámetro 2 , ⋯ }, {Parámetro 3 , ⋯ }, ⋯ ]: muestra varios manipuladores simultáneamente.

**Ejemplo 1.2**

Considere el código:

```
:= In[~]

Manipulate[Expand[m (x + y)^(n + h)], {{n, 30, "Exponente"}, 1, 100, 1}, {m, 1,
1000, Appearance -> "Open"}, {h, {4, -2, 3, 0, 6}}, ControlPlacement -> Left]


= Out[~]
```




Descargar CDF

Se sugiere al lector analizar el código anterior y su salida antes de continuar con el estudio de este tutorial.

Consulte aquí una guía rápida sobre la creación de *CDF*'s, en el siguiente enlace: http://www.wolfram.com/training/videos/CDF001

QR del video:

▶ Resuelva los siguientes ejercicios:

1. Genere una animación que grafique la función $y = \text{sen}(n \cdot x)$ en el intervalo $[-10, 10]$.
2. Construya un objeto dinámico con distintos grafos completos (es decir, grafos sin lazos, con aristas entre cada par de vértices) en el orden de 1 a 30. Etiquete al parámetro con el nombre de

"**Orden**". Sugerencia: utilice la instrucción **CompleteGraph** de *Mathematica*. A manera de ejemplo corra en el programa la línea: **CompleteGraph**[5].

3. Realice una animación parametrizando los coeficientes numéricos de la expresión $ax^2 + bx + c$, donde se muestre **True** si el trinomio es un cuadrado perfecto, o bien, **False** en caso contrario. Sugerencia: recurra al uso del comando **Discriminant** .

4. El matemático *Leonard Euler* conjeturó en algún momento que el polinomio $x^2 + x + 41$ constituía un generador de números primos (Dunham, 1999). Desarrolle un *CDF* que verifique la veracidad o falsedad de esta afirmación.

5. Compruebe a través de un *CDF* que el cociente $\frac{a_{n+1}}{a_n}$, siendo $a_n$ el $n$-ésimo número de *Fibonacci*, tiene un valor "cercano" al número de oro $\phi = \frac{1+\sqrt{5}}{2}$, conforme $n$ "crece".

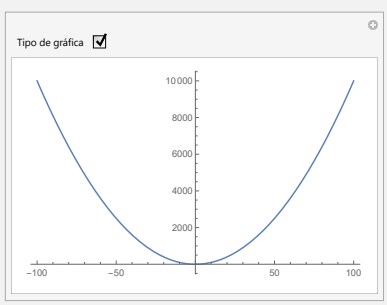
## 1.3 Controladores

El comando **Manipulate** presenta otros tipos de controladores distintos a la barra deslizadora propia de un manipulador. A través de la instrucción **Manipulate** se pueden crear específicamente: componentes de verificación (también llamados *checkbox*), barras de selectores (*setterbar*), menús desplegables (*popupmenu*), botones, campos de texto, *slider* 2D, puntos de localización y deslizadores de color. En esta sección se explicará el uso de cada uno. Comencemos.

• **Checkbox**: agrega un componente de verificación. Por ejemplo:

**Ejemplo 1.3**

```
:= In[~]

Manipulate[If[n == True, Plot[x^2, {x, 0, 100}], Plot[Log[x], {x, 0, 100}]],
{{n, False, "Tipo de gráfica"}, {True, False}}]


= Out[~]
```



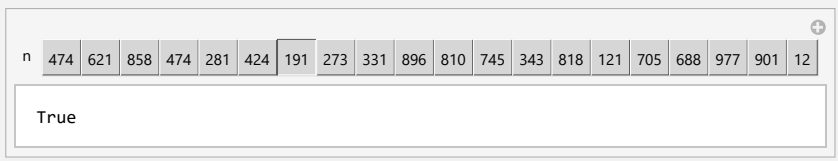

Al marcar el *checkbox* se muestra la gráfica de la función $f(x) = x^2$ y si su valor lógico es **False**, aparece la gráfica de $f(x) = \ln(x)$. Por defecto, un componente de verificación toma el valor lógico **True**.

Descargar CDF

• **SetterBar**: crea una barra de selectores con valores discretos para el parámetro que controla el objeto dinámico. Dentro del **Manipulate** la instrucción {**n**, {**S1**, **S2**}} añade a la barra dos selectores con nombres **S1** y **S2** (pueden ser más), respectivamente. Veamos el siguiente ejemplo:

**Ejemplo 1.4**

```
:= In[~]
Manipulate[PrimeQ[n], {n, RandomInteger[{1, 1000}, 20]}, ControlType -> SetterBar]

= Out[~]
```




Descargar CDF

En el código se solicita a *Mathematica* un vector pseudoaleatorio con números enteros en el rango de 1 a *n* y la longitud del vector es igual a 20. El objeto ofrece como salida **True** si el número seleccionado de la barra es primo y en caso contrario, arroja un **False**. Hemos indicado explícitamente el tipo de controlador mediante la instrucción **ControlType -> SetterBar**, pues por la cantidad de elementos del vector, el software crea por defecto un *popupmenu*.

N Los selectores se pueden etiquetar empleando "-> " que corresponde a la sentencia de generación de reglas o asignaciones en *Wolfram Mathematica*. Si se desea etiquetar dos selectores **S1** y **S2**, como "**Selector1**" y "**Selector2**", respectivamente, se agrega en el manipulador {**n**, {**S1** -> "Selector1", **S2** -> "Selector2"}}. En el ejemplo anterior, es posible reemplazar los números enteros pseudoaleatorios del vector en la barra de selectores, por caracteres consecutivos del abecedario de la siguiente manera:

**Ejemplo 1.5**

```
:= In[~]
Manipulate[PrimeQ[Valor], {Valor, Table[RandomInteger[{1, 1000}, 20][[i]] ->
Table[Alphabet[][[j]], {j, 20}][[i]], {i, 20}]},
ControlType -> SetterBar, LocalizeVariables -> False]

= Out[~]
```


Descargar CDF

La instrucción **LocalizeVariables** -> **False** se utiliza para declarar la variable del manipulador (**Valor**) como global, es decir, el parámetro conserva su valor dentro y fuera del **Manipulate**. Por defecto **Manipulate** siempre construye variables locales. Al ser **Valor** un parámetro global, en una celda aparte, se puede referenciar este identificador con el objetivo de conocer cuál es el número entero que toma en tiempo real, es decir, por ejemplo, en el **Out**[ ] anterior:

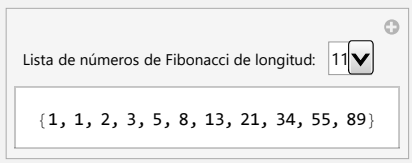
**Ejemplo 1.6**

```
:= In[~]
Valor

= Out[~]
680
```

• **PopupMenu**: la línea de comandos {**n**, {**P1**, **P2**,..., **Pm**}} dentro de un **Manipulate** crea un menú desplegable con "*m*" opciones. Esta clase de menú típicamente es conocida con el nombre de *popupmenu* y genera un combo de selección de valores para asignar a un parámetro. El ejemplo que prosigue muestra una lista de números de *Fibonacci* consecutivos de un tamaño seleccionable al interior de un menú desplegable:

**Ejemplo 1.7**

```
:= In[~]
Manipulate[Table[Fibonacci[j], {j, i}], {{i, 1, "Lista de números de Fibonacci
de longitud:"}, 1, 20, 1}, ControlType -> PopupMenu]

= Out[~]
```




Descargar CDF

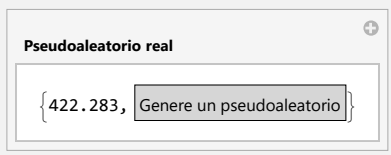
N Hay que recordar que el comando **Fibonacci** es propio del software *Wolfram Mathematica* y retorna el número de *Fibonacci* que se encuentra en la posición indicada en su argumento.

En muchas ocasiones al asignar un número significativo de valores discretos a un parámetro dentro de la instrucción **Manipulate**, se crea automáticamente un *popupmenu* (como ya fue mencionado antes), por lo que es posible prescindir de **ControlType** -> **PopupMenu**.

• **Button**: este comando inserta un botón añadiendo el conjunto de instrucciones a ejecutar cuando el mando es activado. Se aclara que el uso de botones no es exclusivo del comando **Manipulate** y de hecho pueden ser utilizados de manera independiente. Se insta al lector, para corroborarlo, correr por ejemplo en un cuaderno de *Mathematica* la sentencia **Button**["OK", **Print**[10]]. A continuación, se presenta un generador de un número real pseudoaleatorio entre 1 y 1000 mediante el uso de un botón:

**Ejemplo 1.8**

```
:= In[~]
Manipulate[{n, Button["Genere un pseudoaleatorio", n = RandomReal[{1, 1000}]]},
Style["Pseudoaleatorio real", Bold]]

= Out[~]
```



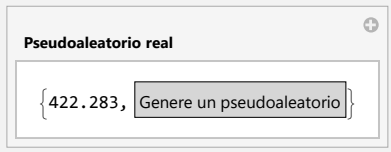

Descargar CDF

N Al presionar el botón "**Genere un pseudoaleatorio**" aparece en sustitución de **n** el pseudoaleatorio correspondiente.

Un comando que brinda un efecto interesante sobre un botón lo constituye **Mouseover**. Este permite un cambio de aspecto del botón cuando el ratón se encuentra sobre él. Veamos:

**Ejemplo 1.9**

```
:= In[~]
Manipulate[{n, Button[Mouseover["Genere un pseudoaleatorio", "Generar"],
n = RandomReal[{1, 1000}]]}, Style["Pseudoaleatorio real", Bold]]

= Out[~]
```

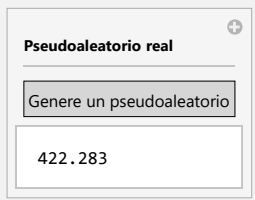



Descargar CDF

Aquí la etiqueta del botón muestra el mensaje "**Generar**" al colocar el cursor sobre el controlador. También, el botón pudo haber sido colocado fuera del área de visualización de resultados, tal y como se muestra a continuación:

**Ejemplo 1.10**

```
:= In[~]

Manipulate[n, Style["Pseudoaleatorio real", Bold], Delimiter, Button[Mouseover[
"Genere un pseudoaleatorio", "Generar"], n = RandomReal[{1, 1000}]]]

= Out[~]
```




Descargar CDF

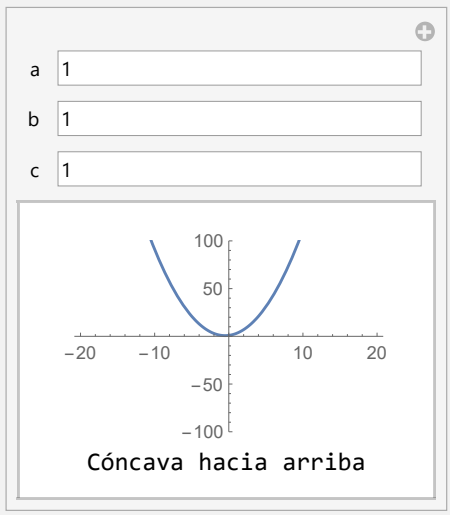
La opción **Style["Pseudoaleatorio real", Bold]** añadió la línea de texto "**Pseudoaleatorio real**" con un estilo en negrita, antes del botón.

Analice, ¿cuál es la función del comando **Delimiter** en este código? (sugerencia: una explicación sobre el uso de esta instrucción se provee en la sección cuatro del presente documento: "Atributos del **Manipulate**").

• Campos de texto: otro tipo de controlador que es posible crear usando **Manipulate**, consiste en un campo de texto o *input field* empleado por el usuario para ingresar de manera más personalizada cambios dentro de un parámetro que caracteriza a un objeto dinámico. Por ejemplo, en el siguiente *CDF* se varían los coeficientes constantes de la función con criterio $f(x) = ax^2 + bx + c$, desplegando su gráfica y tipo de concavidad:

**Ejemplo 1.11**

```
:= In[~]

CellPrint[TextCell[Manipulate[Column[{Plot[a x^2 + b x + c, {x, -20, 20},
PlotRange -> 100],  If[a > 0, "Cóncava hacia arriba", If[a < 0,
"Cóncava hacia abajo", "No hay concavidad"]]}, Center],
{{a, 1}}, {{b, 1}}, {{c, 1}}, Alignment -> Center], "Text"]]


= Out[~]
```




Descargar CDF

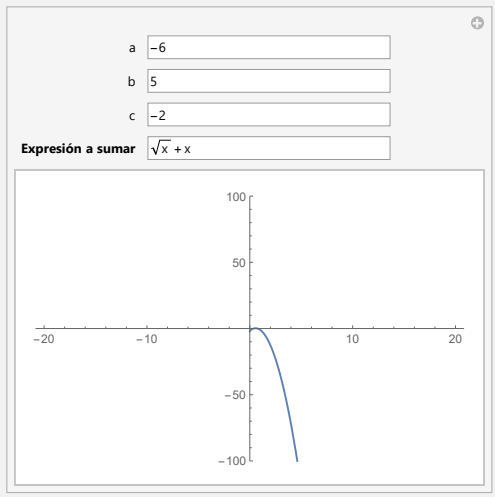
La instrucción **Column** ha acomodado la información de salida en una columna con dos filas. Una contiene la gráfica de la función cuadrática o lineal y la otra un mensaje que indica si la concavidad es hacia arriba, hacia abajo, o bien, si no hay concavidad. Los comandos **CellPrint** y **TextCell** se han utilizado para evitar un doble **Out[ ]** cuando el usuario del *CDF* actualiza un valor dentro de uno de los campos de texto, al presionar el "*enter*" del teclado numérico (este tipo de botón ejecuta en *Mathematica* la secuencia: *shift*+*enter*). Se recomienda el uso de **CellPrint** y **TextCell** cada vez que se crea un campo de texto en un documento con un formato computable.

¿Qué ocurre en la salida del *CDF* anterior, si se toma el valor del parámetro "**a**" igual a cero?

Por otra parte, es esencial indicar que el *plug in* señalado en la introducción del presente documento, *Wolfram CDF Player Free*, solo acepta datos exclusivamente numéricos en los campos de texto. Por ejemplo, si añadiéramos al ejercicio anterior un *input field* para sumar a la fórmula de la función cuadrática alguna otra expresión, ésta será leída por el *plug in* únicamente si lo ingresado por el usuario corresponde a un número:

**Ejemplo 1.12**

```
:= In[~]

CellPrint[TextCell[Manipulate[Plot[a x^2 + b x + c + d, {x, -20, 20},
PlotRange -> 100], {{a, 1}}, {{b, 1}}, {{c, 1}},
{{d, 1, Style["Expresión a sumar", Bold]}}, Alignment -> Center], "Text"]]

= Out[~]
```



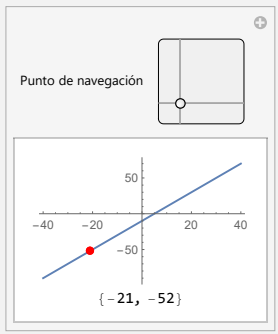

Descargar CDF

N En el **Out[ ]** de acuerdo con lo explicado anteriormente, la expresión $\sqrt{x}+x$ no será interpretada de forma exitosa por el *Wolfram CDF Player Free*. Existe otro tipo de *plug in* de pago, llamado *Wolfram CDF Player Pro*, que es libre de restricciones y que también es distribuido por la empresa *Wolfram Research*. Este programa consiste en una versión comercial más potente de visualización y manipulación de aplicaciones *CDF*'s. Para más información sobre esta clase de licencia se puede visitar: https://www.wolfram.com/player-pro/licensing-options.html. También, si se desea conocer todas las limitaciones del *Wolfram CDF Player Free*, se recomienda la dirección *URL*: https://www.wolfram.com/player-pro/how-player-pro-compares.html.

• *Slider* 2D: en un **Manipulate** la línea de sentencias `{{n, {0, 0}, "Punto"}, {xmin, ymin}, {xmax, ymax}}` construye un *slider* 2D ubicando un par ordenado manipulable en la posición $(0,0)$. Un *slider* 2D es un controlador sobre un plano cartesiano con variación en el eje de las abscisas en el rango **xmin**−**xmax** y en el eje de las ordenadas en el rango dado por **ymin**−**ymax**, donde al arrastrar el punto móvil, se obtiene un par ordenado como resultado de la navegación. En el ejemplo que sigue se usa un *slider* 2D. Si el punto del *slider* reposa sobre la gráfica de la función con criterio $f(x)=2x-10$, cambia de color y de tamaño:

**Ejemplo 1.13**

```
:= In[~]

Manipulate[Column[{Show[Plot[2x-10, {x, -40, 40}], If[(2x-10/. x -> Par[[1]])
== Par[[2]], ListPlot[{Par}, PlotStyle ->{PointSize[0.04], Red}], ListPlot[{Par},
PlotStyle -> PointSize[0.02]]]], Par}, Center], {{Par, {0, 0},
"Punto de navegación"}, {-40, -90}, {40, 70}, 1}, Alignment -> Center]


= Out[~]
```

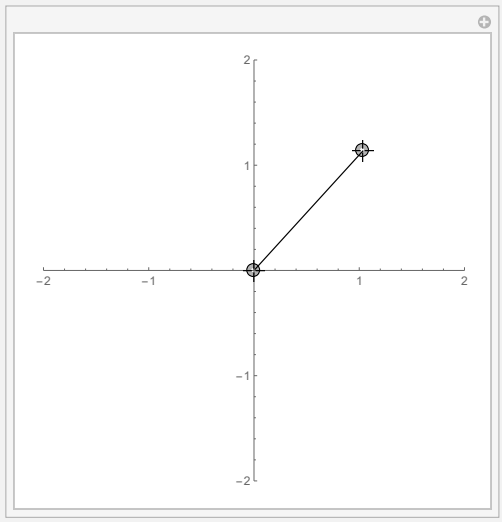



Descargar CDF

• Puntos de localización: un punto de localización consiste en un punto en el plano, manipulable por el usuario mediante el *mouse*. En un **Manipulate** la línea de código {{**Puntos**, **P1**, **P2**, ..., **Pn**}, **Locator** } genera $n$ puntos de localización donde **P1**, **P2**, $\cdots$, **Pn** corresponden a las coordenadas de los pares ordenados que serán almacenados en la variable "**Puntos**". El controlador funciona siempre y cuando lo que se procese para efectos de visualización sea una instrucción que reciba como argumento una lista de pares ordenados. El ejemplo mostrado a continuación construye un segmento de recta donde sus extremos son puntos de localización:

**Ejemplo 1.14**

```
:= In[~]

Manipulate[Graphics[Line[{Punto1, Punto2}], PlotRange -> 2, Axes -> True],
{{Punto1, {0, 0}}, Locator}, {{Punto2, {1, 1}}, Locator}]


= Out[~]
```



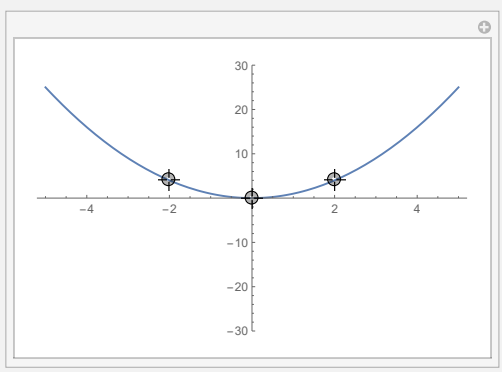

Descargar CDF

El lector puede comprobar al abrir el *CDF*, cómo los puntos de localización constituyen puntos móviles capaces de ser arrastrados con el ratón.

Otro ejemplo interesante se comparte mediante el siguiente *CDF*. En él se utilizan tres puntos de localización para crear una interpolación interactiva al mover los pares ordenados:

**Ejemplo 1.15**

```
:= In[~]

Manipulate[Plot[InterpolatingPolynomial[Puntos, x], {x, -5, 5}, PlotRange -> 30],
{{Puntos, {{-2, 4}, {0, 0}, {2, 4}}}, Locator}]

= Out[~]
```

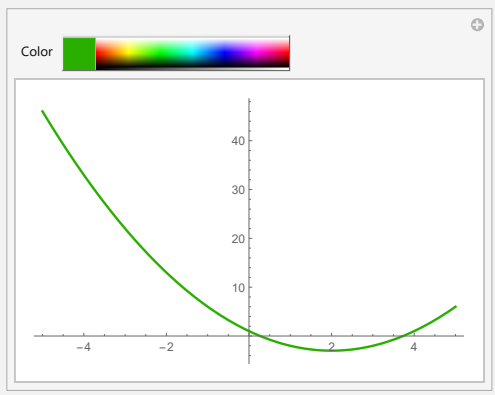



Descargar CDF

• Deslizador de color: este tipo de controlador construye una barra de color (como la que se aprecia en el dibujo siguiente). Al integrar en un **Manipulate** la línea {**Color**, **Nombre**} donde **Color** es la variable que almacena el color especificado en **Nombre** (el nombre debe corresponder al color en inglés) es posible cambiar dinámicamente de color a un objeto, al navegar sobre el deslizador. Veamos el siguiente ejemplo:

**Ejemplo 1.16**

```
:= In[~]
Manipulate[Plot[x^2-4x+1, {x, -5, 5}, PlotStyle-> {Color, Thick}], {Color, Blue}]

= Out[~]
```




Descargar CDF

Aquí la gráfica de la función cuadrática toma el color azul por defecto y al navegar sobre el deslizador **Color**, su color cambia de manera automática.

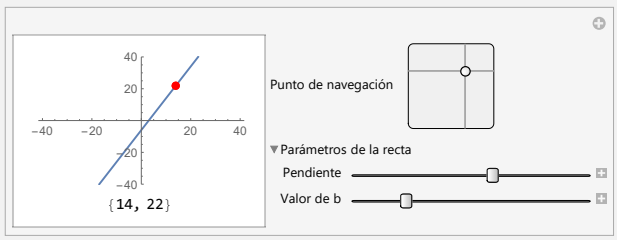
Muchos de los controladores mencionados en esta sección se pueden generar utilizando la instrucción **ControlType** dentro del comando **Manipulate**, a saber: **ControlType** -> **Manipulator** crea un manipulador, **ControlType** -> **PopupMenu** genera un menú desplegable, **ControlType** -> **SetterBar** muestra una barra de selectores, **ControlType** -> **Slider2D** un *slider* 2D, **ControlType** -> **Locator** construye puntos de localización, **ControlType** -> **ColorSlider** crea un deslizador de color, **ControlType** -> **RadioButtonBar** una barra de botones circulares (se recomienda al lector explorar este tipo de controlador), finalmente tenemos que **ControlType** -> **Automatic** asiente a *Mathematica* escoger el tipo de controlador automáticamente, entre otros.

Otra opción útil la provee el comando **OpenerView**. En el ejemplo siguiente se utiliza esta instrucción para crear un combo con dos manipuladores que varían los parámetros de la ecuación de una recta, a

saber, su pendiente y la intersección con el eje $y$:

**Ejemplo 1.17**

```
:= In[~]

Manipulate[Column[{Show[Plot[m x + b, {x, -40, 40}, PlotRange -> -40],
If[(m x + b /. x -> Par[[1]]) == Par[[2]], ListPlot[{Par}, PlotStyle ->
{PointSize[0.04], Red}],  ListPlot[{Par}, PlotStyle -> PointSize[0.02]]]], Par},
Center], {{Par, {0, 0}, "Punto de navegación"}, {-40, -90}, {40, 70}, 1},
OpenerView[{"Parámetros de la recta", Column[{Control[{{m, 2, "Pendiente"},
-10, 10, 1}], Control[{{b, -10, "Valor de b"}, -10, 10, 1}]}]}],
Alignment -> Center, ControlPlacement -> Right]


= Out[~]
```




Descargar CDF

Como se aprecia **OpenerView** emplea dos argumentos: una etiqueta con el nombre que llevará este objeto y los controladores que se añadirán mediante el uso del comando **Control**. La instrucción **Manipulate** cuenta con una serie de propiedades o atributos que serán estudiados en la siguiente sección.

## 1.4 Atributos Manipulate

En esta sección se mostrarán una serie de atributos que posee el comando **Manipulate**. Estas propiedades pueden ser empleadas de manera optativa al crear un *CDF*. Los atributos se clasifican en varias categorías: apariencia, ordenamiento, estilo y control. A continuación, se describe sus funcionalidades.

### 1.4.1 Apariencia

- **Alignment**: alínea el contenido del cuadro donde se muestra la salida de un *CDF*. Puede tomar las siguientes opciones: **Left**, **Center** o **Right**.

- **Appearance**: brinda distintas alternativas para mostrar el valor de un parámetro del cual depende un objeto dinámico. Solo funciona con controladores que posean un deslizador. Por ejemplo,

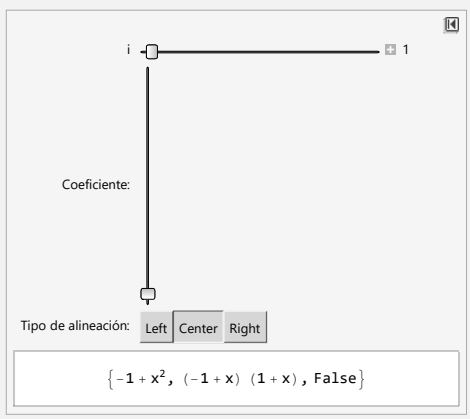
`Appearance -> "Open"` abre los botones de animación del *slider* y `Appearance -> "Labeled"` muestra una etiqueta que actualiza en tiempo real el valor del parámetro.

- `VerticalSlider`: ubica un *slider* en posición vertical. Cabe destacar que *Mathematica* por defecto, posiciona cualquier deslizador de manera horizontal.
- `AppearanceElements`: permite agregar un botón de restauración en la parte superior derecha de un *CDF*. Al presionar este botón, se actualizan a sus valores iniciales, todos los parámetros del *CDF*. La sintaxis que se debe emplear es: `AppearanceElements -> "ResetButton"`.

Se sugiere al lector para comprender mejor los atributos ya mencionados, explorar con el siguiente *CDF* a través de su descarga. Este objeto brinda como salida un vector con tres componentes: la primera muestra el polinomio $bx^2 - i$, la segunda despliega su factorización y la tercera genera un valor lógico `True` si el polinomio es irreductible o `False`, en caso contrario.

**Ejemplo 1.18**

```
:= In[~]

Manipulate[{b x^2 - i, Factor[b x^2 - i], IrreduciblePolynomialQ[b x^2 - i]},
{i, 1, 100, 1, Appearance -> "Labeled"}, {{b, 1, "Coeficiente:"}, 1, 100, 1,
VerticalSlider}, {{a, Center, "Tipo de alineación:"}, {Left, Center, Right}},
Alignment -> a, AppearanceElements -> "ResetButton"]


= Out[~]
```




Descargar CDF

Una explicación en video sobre este *CDF* se puede obtener mediante el siguiente enlace: https://youtu.be/CgmfNV3Q83Q

QR del video:

▶ Construya el mismo *CDF* sin el botón de reseteo, con ambos *slider* verticales y con los controladores de animación abiertos. ¿Es posible realizarlo?

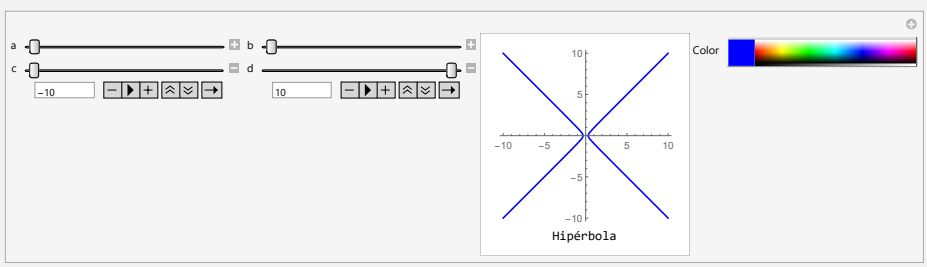
### 1.4.2 Ordenamiento

- **Row**: ordena el contenido de un *CDF* en filas. Dicho contenido puede corresponder a lo mostrado en la ventana de resultados, o bien, a un conjunto de controladores. Si se emplea para ordenar controladores, cada uno debe ser integrado mediante el comando **Control**.
- **Column**: presenta el mismo funcionamiento de la instrucción **Row**, pero ordenando en columnas. A diferencia de **Row** admite el atributo: **Left**, **Center** o **Right**.
- **Grid**: ordena el contenido en filas y columnas. Se opera de forma equivalente a las instrucciones **Row** y **Column**.
- **ControlPlacement**: da ubicación a los controladores u otros elementos de un *CDF*. Admite los atributos: **Left**, **Right**, **Top** y **Bottom**.

N El ejemplo que sigue despliega la gráfica de diversas curvas elípticas o hiperbólicas mostrando el tipo de cónica que se despliega en el plano cartesiano. Este es un ejemplo mucho más elaborado y en él, se han utilizado las distintas opciones de ordenamiento ya citadas. ¡Se recomienda tomar el tiempo debido para su revisión y prestar mucha atención al código!

**Ejemplo 1.19**

```
:= In[~]

Manipulate[If[d < 0 && c < 0, d = -d; Column[{ContourPlot[d x^2/a^2 + c y^2/b^2
== 1, {x, -10, 10}, {y, -10, 10}, Frame -> False, Axes -> True], If[Positive[d]&&
Positive[c], "Elipse", If[c == 0 || d == 0, "No es una cónica", "Hipérbola"]]},
Center], Column[{ContourPlot[d x^2/a^2 + c y^2/b^2 == 1, {x, -10, 10},
{y, -10, 10}, Frame -> False, Axes -> True, ContourStyle -> Color],
If[Positive[d] && Positive[c], "Elipse", If[c == 0 || d == 0, "No es una cónica",
"Hiperbola"]]}, Center]], Grid[{{Control[{a, 1, 30}], Control[{b, 1, 30}]},
{Control[{c, -10, 10, Appearance -> "Open"}], Control[{d, -10, 10, Appearance ->
"Open"}]}}], {Color, Blue}, Alignment->Center, ControlPlacement->{Left, Right}]


= Out[~]
```




Descargar CDF

El **If** al inicio del **Manipulate**, evita que las variables **d** y **c** sean negativas simultáneamente. Otro aspecto interesante en este *CDF* reside en la manera en cómo el comando **ControlPlacement** ha acomodado el

*grid* de manipuladores al lado izquierdo y el deslizador de color al lado derecho.

Observe un video sobre el contenido de este *CDF* en el siguiente enlace:
https://youtu.be/967D_ObVI8M

QR del video:

### 1.4.3 Estilo

- **LabelStyle**: especifica un estilo de etiqueta que se asignará a todos los nombres que aparecen dentro de un objeto dinámico. El estilo admite distintas propiedades relacionadas con el tamaño de fuente, clase de tipografía y color. Algunas opciones son: {**Blue**, **Bold**, **Large**, **FontSize**->36, **FontFamily**->"Courier"}. Si se desean agregar varias instancias al mismo tiempo, se debe emplear la instrucción **Directive**.

- **Delimiter**: añade una línea horizontal denominada delimitador, para acomodar las secciones en las cuales se mostrarán los controladores.

- "**Mensaje**": permite incluir el texto **Mensaje** en el área de salida de un *CDF*, o bien, en la sección de controladores. Admite atributos similares a los procesados por el comando **LabelStyle**. Si se requiere detallar un estilo, se debe usar la instrucción **Style**.

- **Item**: representa un elemento dentro de una construcción generada por **Manipulate**. Es un recurso útil cuando se desea añadir texto alineado (usando el comando **Alignment**), sea a la izquierda, centrado, o a la derecha.

- **Framed**: muestra dentro de un recuadro, cualquier tipo de contenido.

- **FrameLabel**: integra una o varias leyendas a un objeto dinámico, utilizando la sintaxis: **FrameLabel** -> {"Leyenda abajo", "Leyenda izquierda", "Leyenda superior", "Leyenda derecha"}]. El comando **None** puede reemplazar cualquiera de las leyendas en **FrameLabel** para prescindir de ellas.

El *CDF* presentado a continuación mezcla de forma interesante diversas opciones de estilo.

**Ejemplo 1.20**

```
:= In[~]

Manipulate[Column[{DiscretePlot[Prime[i], {i, 1, n}, PlotStyle -> Color],
Framed[Style["Gráfica discreta", Orange, 30, "Courier"]]}, Center],
Item[Style["Primera sección", Italic, 30], Alignment -> Center],
Style["Gráfica", 30, Bold, Blue], {n, 10, 100, 1}, {Color, Blue},
Delimiter, Item[Style["Segunda sección", Italic, 30, Magenta],
Alignment -> Right], Style["Estilo del texto", 30, Bold, Orange],
{{m, "Arial", "Tipo de familia"}, $FontFamilies}, LabelStyle ->
 Directive[Bold, Large, Italic, FontFamily -> m], FrameLabel ->
{"Ejemplo", None, Style["Números primos", 30, Bold, Blue]}]


= Out[~]
```


Descargar CDF

La gráfica que aparece en el objeto dinámico anterior, corresponde a una lista de pares ordenados, en la cual, la primera coordenada es un número natural consecutivo comenzando en 1 y la segunda componente, la variación en un vector de números primos también consecutivos desde 1 hasta $n$. Es importante aclarar, además, que la sentencia **`$FontFamilies`** dentro del código, construye un *array* con todas las fuentes disponibles en la computadora del usuario.

Una explicación complementaria de este *CDF* se encuentra en el siguiente enlace: https://youtu.be/CyDrDlVjBaU

QR del video:

### 1.4.4 Control

- **`AutoAction`**: esta propiedad facilita el arrastre de un deslizador con solo colocar el *mouse* sobre la barra del *slider*, creando un efecto magnético entre el cursor y el controlador. Utiliza la sintaxis: **`AutoAction`** `->` **`True`**.

- **`ContinuousAction`** `->` **`False`**: actualiza el valor de un parámetro asociado a un controlador únicamente cuando se lanza dicha herramienta.

- **`LocalizeVariables`** `->` **`False`**: hace que las variables dentro de un manipulador sean globales.

- **`Initialization`** `:> ()`: permite inicializar variables o generar funciones dentro de un *CDF*. Se emplea el punto y coma para separar todas las sentencias que se incorporan dentro.

- **`SaveDefinitions`** `->` **`True`**: esta instrucción salva cualquier definición de funciones o inicialización de variables que se haya realizado fuera de un **`Manipulate`** al crear un *CDF*. Su funcionamiento es similar al atributo **`Initialization`** con la diferencia de que **`SaveDefinitions`** salva sin necesidad de tener que crear las definiciones dentro del documento con un formato computable.

- **`TrackedSymbols`** `:> {}`: especifica los parámetros (símbolos de rastreo) que serán usados para actualizar un objeto dinámico. Los símbolos de rastreo tienen la característica de que al ser modificados, activan, de manera automática, los cambios de las otras variables.

- **Dynamic**: este interesante atributo construye elementos que cambian dinámicamente en función de otros incorporados dentro de un *CDF*, lo cual favorece la generación de actualizaciones en tiempo real de los distintos parámetros o variables que puede contener el documento con un formato computable.

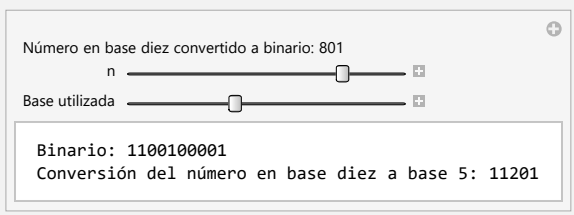
- **Deployed** -> **True**: restringe la interactividad de un objeto dinámico a los controladores del *CDF*. Por ejemplo, una gráfica 3D por defecto puede rotarse al mover el *mouse* sobre ella. Usando este atributo se podría volver nulo dicho comportamiento.

- **ControllerLinking** -> **False**: hace que un *CDF* nunca responda a controladores externos, como por ejemplo, un *joystick* o un *gamepad*.

A continuación, se comparte un *CDF* donde se utilizan algunos de los atributos de control ya mencionados.

**Ejemplo 1.21**

```
:= In[~]

DecimalToBase[decimal_, base_] := StringJoin[ToString /@ IntegerDigits[decimal,
base]]
Manipulate[Column[{Row[{"Binario: ", DecimalToBinario[n]}], Row[{
"Conversión del número en base diez a base ", m, ": ", DecimalToBase[n, m]}]}],
Row[{"Numero en base diez convertido a binario: ", Dynamic[n]}], {n, 1, 1000, 1},
{{m, 2, "Base utilizada"}, 2, 10, 1}, ContinuousAction->False, AutoAction->False,
Deployed -> True, SaveDefinitions -> True, TrackedSymbols :> {n}, Initialization
:> (DecimalToBinario[n_] := StringJoin[ToString /@ IntegerDigits[n, 2]])]


= Out[~]
```




Descargar CDF

N El objeto dinámico convierte un número entero positivo en el rango de 1 hasta 1000, a un número binario. Además, mediante el segundo deslizador realiza la conversión a cualquier base desde 2 hasta 10. Se han creado funciones que automatizan estas conversiones. Se insta al lector a explorar detenidamente el *CDF* y a contestar las siguientes preguntas: ¿qué ocurre si se le atribuye a **AutoAction** el valor lógico **True**?, ¿qué sucede si se sustituye en **Deployed True** por **False**?, ¿qué ocurre si el parámetro no se evalúa en **Dynamic**?

Un video sobre el *CDF* anterior se comparte en el siguiente enlace: https://youtu.be/4nKVelCUB6s

QR del video:

Consulte, además, información adicional sobre cómo crear aplicaciones *CDF*'s con utilidades reales, en el siguiente enlace: http://www.wolfram.com/training/videos/CDF003

QR del video:

## 1.5 Antes de crear un *CDF*

El proceso de creación de un archivo *CDF* debe seguir un determinado protocolo con el objetivo de evitar errores en la aplicación final. Antes de salvar un documento *.nb* de *Mathematica* al formato *CDF* se recomienda seguir los siguientes pasos para limpiar las variables y/o funciones que fueron utilizadas previamente.

1. Ejecute el comando `Quit[ ]`, o bien, reinicialice el *kernel* en el menú: `Evaluation -> Quit kernel -> Local`.

2. Desactive la barra de sugerencias en: `Edit -> Preferences` y desmarque la opción "*Show Suggestions Bar after last output*".

3. Corra el *notebook*.

4. Esconda el código fuente haciendo doble *click*.

5. Desactive la edición de las celdas, en un cuaderno aparte realice:

   (a) `nb=First[Notebooks["Nombre de su archivo"]]`

   (b) `SetOptions[nb, Editable -> False, ShowCellBracket -> False]`

6. Salve como un *CDF*.

Consulte aquí una explicación en video del procedimiento anterior, en el siguiente enlace: https://www.youtube.com/watch?v=iRMxhoJCQJE

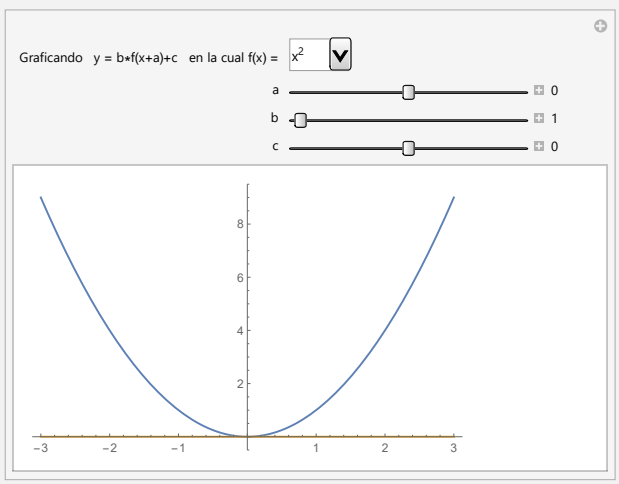
QR del video:

## 1.6 Ejemplos de *CDF*'s

En esta sección se presentan una serie de ejemplos de documentos con un formato computable elaborados por los autores del presente trabajo. La principal finalidad de esta sección consiste en mostrar los alcances desde un punto de vista didáctico, de las aplicaciones *CDF*'s en diversos campos de estudio. Los ejemplos presentados a continuación son aplicaciones más elaboradas y, por ende, no se comparte en este documento el código fuente que les dio origen, teniendo el lector la posibilidad de descargar los archivos con toda la programación correspondiente en lenguaje *Wolfram*, *.nb* para su exploración minuciosa.

**Ejemplo 1.22**

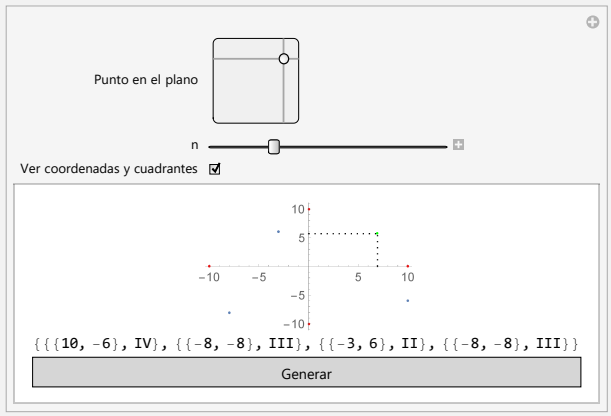
Presenta un graficador de la función $y = bf(x+a)+c$ con $f(x) \in \left\{x, x^2, x^3, \sqrt{x}, \frac{1}{x}, \frac{1}{x^2}, |x|, \sqrt[3]{x}, 2^x, \left(\frac{1}{2}\right)^x\right\}$.




Descargar .nb

**Ejemplo 1.23**

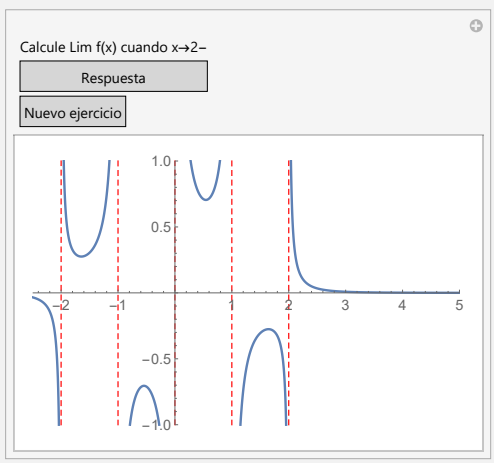
El *CDF* crea de forma pseudoaleatoria $n$ pares ordenados en el plano cartesiano. El alumno debe encontrar visualmente sus coordenadas y el cuadrante donde se ubican, teniendo la posibilidad de generar la solución.




Descargar .nb

## Ejemplo 1.24

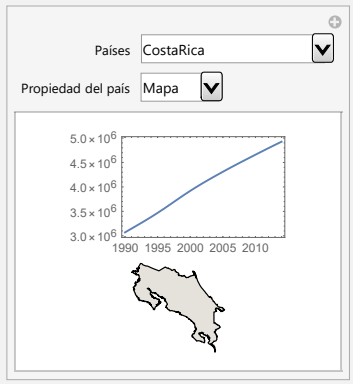
El objeto dinámico provee un generador de ejercicios para estimar gráficamente el valor de un límite unilateral. El documento computable le permite al estudiante obtener el resultado en cada caso. Una limitación observable en este ejemplo, reside en la escritura de texto matemático dentro de un *CDF*, como se aprecia, esto no es posible hacerlo empleando código *Latex*.




Descargar .nb

## Ejemplo 1.25

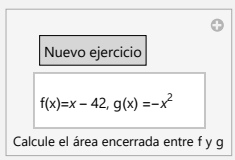
El *CDF* muestra como salida una gráfica con el crecimiento o decrecimiento poblacional de un país seleccionado, desde el año 1990 al 2014. Además, adiciona una propiedad del país, escogida de un combo, a saber: el código de área, su capital, la bandera, el área territorial, su mapa geográfico o sus principales regiones o provincias.

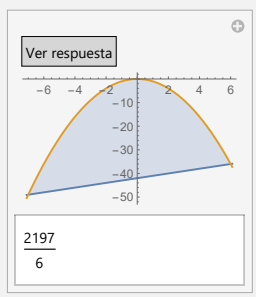



Descargar .nb

## Ejemplo 1.26

El documento con un formato computable construye ejercicios pseudoaleatorios para determinar el área entre dos curvas. Ofrece una ventana que cambia el ejemplo a resolver y otra donde se muestra su solución.

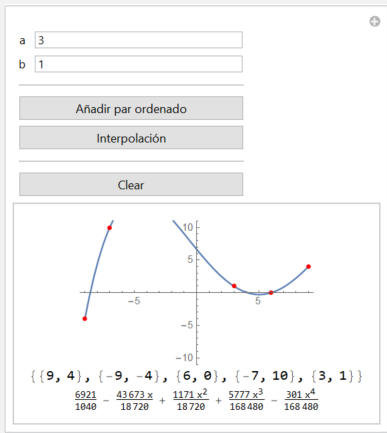



Descargar .nb

## Ejemplo 1.27

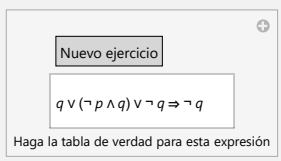
Se presenta un objeto dinámico que permite ingresar componente a componente, una lista de pares ordenados para buscar un polinomio de interpolación, mostrándose a su vez en el plano cartesiano (si es que el polinomio existe), la gráfica del polinomio y los pares ordenados que le dieron origen.

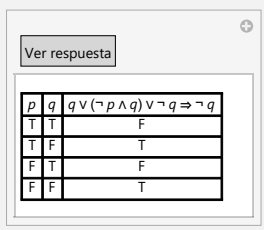



Descargar .nb

## Ejemplo 1.28

El *CDF* construye de forma pseudoaleatoria una proposición con diversas conectivas lógicas, solicitando al estudiante realizar su tabla de verdad. En una ventana independiente el alumno puede ver la respuesta del ejercicio.

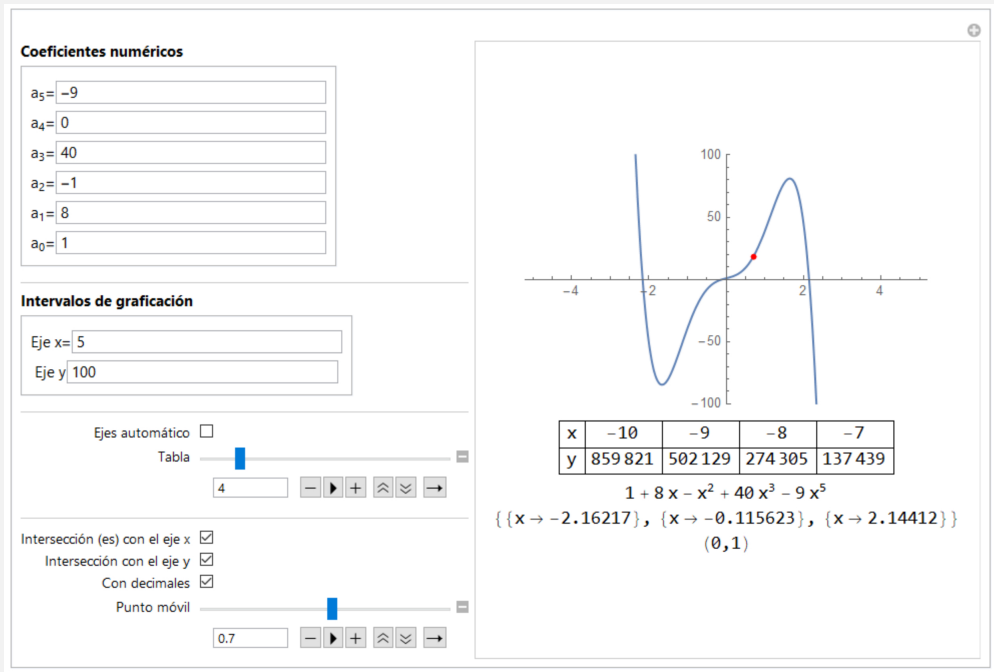



Descargar .nb

## Ejemplo 1.29

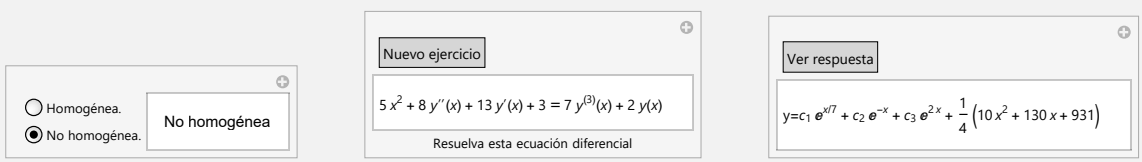
El *CDF* permite analizar una función polinomial de grado a lo sumo cinco, es decir, aparece la gráfica de la función, una tabla de valores, las intersecciones con los ejes coordenados y un punto móvil, pudiendo personalizar los rangos de graficación.




Descargar .nb

## Ejemplo 1.30

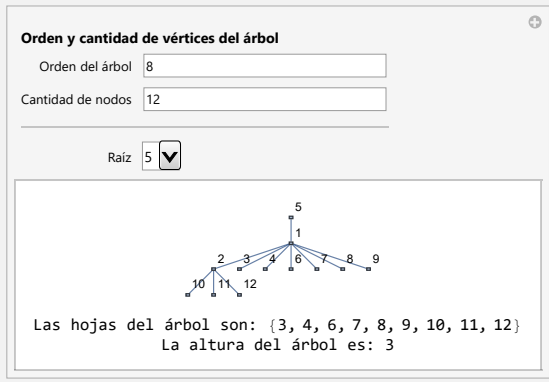
Se presenta un objeto dinámico que sirve para ejercitarse en la resolución de ecuaciones diferenciales homogéneas y no homogéneas con coeficientes constantes. El *CDF* contiene tres ventanas: una donde se elige si la ecuación será homogénea o no homogénea, otra con el ejercicio a resolver y finalmente, una ventana para visualizar la solución de la ecuación.




Descargar .nb

## Ejemplo 1.31

El *CDF* muestra un árbol de orden personalizado con una cantidad de vértices escogidos por el estudiante. La animación pretende servir de base para introducir conceptos básicos de la teoría de árboles, como por ejemplo, la definición de raíz, hojas y altura.




Descargar .nb

Otro tema de importancia lo circunscribe la publicación de *CDF*'s en la *web*, aspecto que será desarrollado en la siguiente sección.

## 1.7 Publicación de *CDF*'s en la *web*

Los documentos con un formato computable pueden ser publicados en la *web* utilizando un servicio denominado *Wolfram Cloud*, el cual es provisto por la empresa *Wolfram Research*. Se advierte que para poder emplear la nube de la compañía *Wolfram Research* es indispensable contar con una licencia autorizada del software *Wolfram Mathematica*. Lo anterior podría constituirse en una importante limitación, si el usuario no cuenta con esta licencia de paga.
En *Mathematica* si se desea exportar un *CDF* a *Wolfram Cloud* se ejecuta:

```
CloudDeploy[Manipulate[...], "File", Permissions -> "Public"]
```

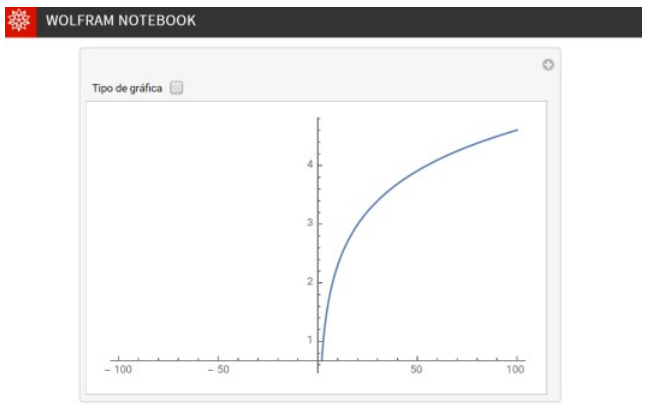
La instrucción solicita las credenciales del usuario en el portal de *Wolfram* y verificando su identidad, retorna como salida una dirección *URL* que carga el *CDF* en una página *web* de la empresa. El archivo se salvará con el nombre especificado en `"File"`. Por ejemplo:

**Ejemplo 1.32**

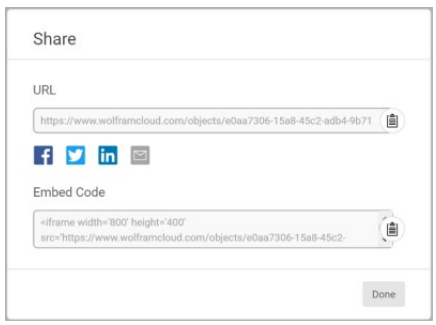
```
:= In[~]

CloudDeploy[Manipulate[If[n == True, Plot[x^2, {x, -100, 100}],
Plot[Log[x], {x, -100, 100}]], {{n, False, "Tipo de gráfica"},
{True, False}}], "Tipo de grafica", Permissions -> "Public"]


= Out[~]

 CloudObject["https://www.wolframcloud.com/objects/enrique.vilchez.quesada/Tipo de
                                        grafica"]
```

Descargar .nb

La dirección provista por **`CloudDeploy`** denota la exportación exitosa del *CDF* en la nube:




En este sitio es posible compartir el recurso al presionar: Share
Esto abre la ventana:




Donde se cuenta con dos opciones de socialización:

1. *URL*: para compartir directamente la dirección del documento con un formato computable.
2. *Embed Code*: que sirve para integrar el objeto dinámico en una página *web* externa (por ejemplo, en un aula virtual).

Un aspecto interesante reside en emplear la *URL* del *CDF* como un medio de generación de realidad aumentada, al guardar la dirección en un código *QR* facilitando la consulta del documento computable en un dispositivo móvil. Por ejemplo, este es el código *QR* del *CDF* "**`Tipo de gráfica`**":

El lector puede corroborar que se despliega el objeto, al abrir el código en un celular que cuente con una *app* lectora de *QR* .

Como una observación importante, cabe destacar que en *Wolfram Mathematica* es posible crear códigos *QR* usando la sentencia: `BarcodeImage["Dirección URL", "QR"]`.

Para más información sobre publicación de *CDF*'s en dispositivos móviles, se dispone del video mostrado mediante el siguiente enlace: https://www.youtube.com/watch?v=a2g3m_7otU0

QR del video:

## 1.8 Conclusiones

El presente trabajo constituye un esfuerzo que provee un conjunto de recursos didácticos con la intención de introducir al usuario en el diseño de documentos con un formato computable. El recorrido realizado describe aspectos básicos sobre cómo se pueden generar diversos tipos de controladores en la construcción de un *CDF*, abarcando, además, un nivel de especificidad medio en la descripción y usos de muchos de sus atributos.

Es reconocible que los documentos con un formato computable son una excelente alternativa para construir objetos de aprendizaje en el contexto de la educación matemática, sin embargo, una limitación clara para sus potenciales desarrolladores, reside en la necesidad de contar con una licencia del software *Wolfram Mathematica*.

Lo expuesto en este documento se espera, logre inspirar a otros colegas en el área de la educación matemática o afines, a innovar en el desarrollo e implementación de documentos con un formato computable. Se recomienda al lector ser paciente y perseverante con relación al uso del lenguaje *Wolfram*, esto le garantizará una labor de programación exitosa.

Las ideas aquí propuestas conforman piezas segregadas que, en manos de un buen artesano didáctico, conducirán en el mejor de los casos, a promover procesos de enseñanza y aprendizaje originales y disruptivos.

## Bibliografía

[1] Dunham, W. (1999). Euler: The Master of Us All. USA: Mathematical Association of America.

[2] Honan, T. [Wolfram] (2012, noviembre 15). Using Mathematica and *CDF* to Create and Distribute Interactive Physics Lecture Notes [Video file]. Recuperado de https://www.youtube.com/watch?v=YoJRj9VRWd0

[3] Islas-Carmona, J. (2008). El prosumidor. El actor comunicativo de la sociedad de la ubicuidad. *Revista Palabra Clave*, *11*(1), 29-39. Recuperado de https://dialnet.unirioja.es/descarga/articulo/2709722.pdf

[4] Vílchez-Quesada, E. (2015a). Estructuras discretas con *Mathematica*. México: Editorial Alfaomega.

[5] Vílchez-Quesada, E. (2015b). Creación de *CDF*'s para la enseñanza de funciones con *Wolfram Mathematica*. En J. Córica (Presidencia), *VI Congreso Virtual Iberoamericano de Calidad en Educación Virtual y a Distancia* (EduQ@2015, pp. 1-30). Congreso virtual organizado por la Fundación Latinoamericana para la Educación a Distancia. Mendoza, Argentina. Recuperado de http://www.eduqa.net/eduqa2015/images/ponencias/eje1/1_t_Vilchez_Quesada_Enrique_Creacion_de_CDF_s_para_la_ensenanza_del_tema_de_funciones_con_Wolfram_Mathematica_-_copia.pdf

[6] *Wolfram Research*. (2019). *Wolfram Language & System* Documentation Center. USA: Wolfram. Recuperado de http://reference.wolfram.com/language