

Diplomado En Programación Básica

Universidad Autónoma de Chiapas
Centro Mesoamericano de Física Teórica

Michael Steven Paucar Rojas

MATHEMATICA



WOLFRAM

1. Introducción

El presente cuaderno constituye un recurso de apoyo para el aprendizaje de Mathematica orientado a la programación y al uso de sus principales funciones en contextos académicos y prácticos. El contenido se organiza de manera progresiva iniciando con operaciones básicas sobre listas, expresiones matemáticas y representaciones gráficas para avanzar hacia temas más complejos como manejo de entidades, conversiones de unidades, generación de visualizaciones interactivas y aplicaciones en análisis de datos.

El enfoque seguido combina teoría con ejemplos prácticos que buscan ilustrar no solo la sintaxis del lenguaje sino también la lógica detrás de cada comando. Se ha procurado mantener una estructura clara donde cada sección incluye subtítulos, descripciones y comentarios en el código para facilitar la comprensión. Esto permite que el material pueda ser utilizado tanto por estudiantes en formación como por interesados en explorar las capacidades del software en distintos escenarios.

Cabe señalar que el documento reúne apuntes propios sistematizados a partir del estudio y la práctica personal. Estos apuntes no reemplazan la documentación oficial de Mathematica pero sí constituyen un complemento útil para guiar el aprendizaje y servir como referencia en la resolución de ejercicios y proyectos futuros.

2. Tabla de contenidos

1. Introducción

2. Tabla de contenidos

3. Clase 1 — Introducción a Wolfram Mathematica

3.1. Captura y análisis de imagen

4. Clase 2 — Comandos básicos, listas y entidades

4.1. Comandos del sistema

4.2. Comandos interactivos

4.3. Entidades: países y banderas

4.4. Exploración planetaria

4.5. Conversiones de unidades y monedas

4.6. Listas: creación y operaciones básicas

4.7. Funciones para secuencias y combinación de listas

4.8. Manipulación avanzada de listas

4.9. Funciones adicionales sobre listas

5. Clase 3 — Gráficos, colores y funciones trigonométricas

5.1. Gráficas estadísticas (barras y pastel)

5.2. Selección y manipulación de datos para visualización

5.3. Colores y estilos gráficos (paletas y transformaciones)

5.4. Funciones matemáticas básicas y plots elementales

6. Clase 4 — Funciones Trascendentes

6.1. Expansión de expresiones trigonométricas

6.2. Números complejos

6.3. Logaritmos

6.4. Exponenciales

6.5. Series

6.6. Límites

6.7. Funciones

6.8. Derivadas

6.9. Integrales

6.10. Notación de Lagrange

6.11. Integración Numérica

6.12. Tablas

6.13. Gráfica de Tablas

7. Clase 5 — Visualización Matemática Interactiva

7.1. Gráficas Bidimensionales (2D)

7.2. Gráficas Tridimensionales (3D)

7.3. Manipuladores Interactivos

8. Clase 6 — Álgebra Simbólica y Series Numéricas

8.1. Solución de ecuaciones

8.2. Manipulación algebraica

8.3. Series Numéricas

9. Clase 7 — Variable Compleja

9.1. Números Complejos

9.2. Conversión de la forma Polar a Rectangular

9.3. Conversión de la forma Rectangular a Polar

9.4. Gráficas de Números Complejos

10. Clase 8 — Álgebra Lineal

10.1. Definición y creación de matrices

10.2. Operaciones básicas con matrices

10.3. Acceso a elementos

10.4. Operaciones avanzadas con matrices y vectores

10.5. Programación básica en Mathematica

11. Clase 9 — Álgebra Lineal

11.1. Operadores condicionales

11.2. Condicional if

12. Tareas

12.1. Tarea 1 — Cálculos Numéricos y Funciones en Mathematic

12.2. Tarea 2 — Formato de Notebook

12.3. Tarea 3 — Aplicaciones de Funciones Trascendentes

12.4. Tarea 4 — Esferas 3D

12.5. Tarea 5 — Repaso general en Mathematica

12.6. Tarea 6 — Solución de ecuaciones

12.7. Tarea 7 — Variable Compleja

12.8. Tarea 8 — Reto Matrices

12.9. Tarea 9 — Aplicación del condicional if

13. Apéndice

13.1. Comandos comunes

Tareas

⚡ **Instrucciones:** En esta sección se agrupan las tareas asignadas.

Tarea 9 – Aplicación del condicional if

📅 2025/10/20

1. Definir una lista de 10 números. Usando el comando *if*, decir al usuario si el número 6 está en la lista.

- Si está que diga la frase “El número está contenido en la lista”.
- Si no está, que lo agregue a la lista.

Si {6} no se encuentra en la lista:

```
In[ ]:= lista = Input["Ingrese una lista de 10
    |entra
    números (por ejemplo: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}): "];
If[MemberQ[lista, 6],
|si |¿contenido en?
    Print["El número está contenido en la lista"], (lista = Join[lista, {6}]);
|escribe |junta
    Print["El número no estaba en la lista, se ha agregado."]];
|escribe
Print["Lista actual: ", lista];
|escribe
```

El número no estaba en la lista, se ha agregado.

Lista actual: {1, 7, 9, 3, 4, 5, 1, 11, 113, 20, 6}

Si {6} se encuentra en la lista:

```
In[ ]:= lista = Input["Ingrese una lista de 10
    |entra
    números (por ejemplo: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}): "];
If[MemberQ[lista, 6],
|si |¿contenido en?
    Print["El número está contenido en la lista"], (lista = Join[lista, {6}]);
|escribe |junta
    Print["El número no estaba en la lista, se ha agregado."]];
|escribe
Print["Lista actual: ", lista];
|escribe
```

El número está contenido en la lista

Lista actual: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

2. Definir un número complejo $z = a + bi$

- Si el módulo de este número es menor que 4, que diga la oración “el módulo es: ”.
- Si es mayor que 4, que diga “No hago nada”.

```
# Si el módulo es menor que 4
```

```
In[ ]:= z = Input["Ingrese un número complejo (por ejemplo: 2 + 3 I): "];
          |entra                               |número i
          |
          If[Abs[z] < 4, Print["El módulo es: ", Abs[z]], Print["No hago nada"]]
          |si |valor absoluto |escribe          |valor abs... |escribe
```

El módulo es: $\sqrt{13}$

```
# Si el módulo es mayor que 4
```

```
In[ ]:= z = Input["Ingrese un número complejo (por ejemplo: 2 + 3 I): "];
          |entra                               |número i
          |
          If[Abs[z] < 4, Print["El módulo es: ", Abs[z]], Print["No hago nada"]]
          |si |valor absoluto |escribe          |valor abs... |escribe
```

No hago nada

3. Definir la lista de números aleatorios.

- Si la longitud de esta lista es menor de 5, hacer una gráfica de barras.
- Si la longitud es mayor que 5, hacer una gráfica circular.

```
# Si la longitud de la lista menor a 5
```

```

In[*]:= Clear[lista];
borra

lista = RandomInteger[{1, 10}, RandomInteger[{2, 10}]];
entero aleatorio      entero aleatorio

Print["Lista generada: ", lista];
escribe

If[Length[lista] < 5,
si longitud

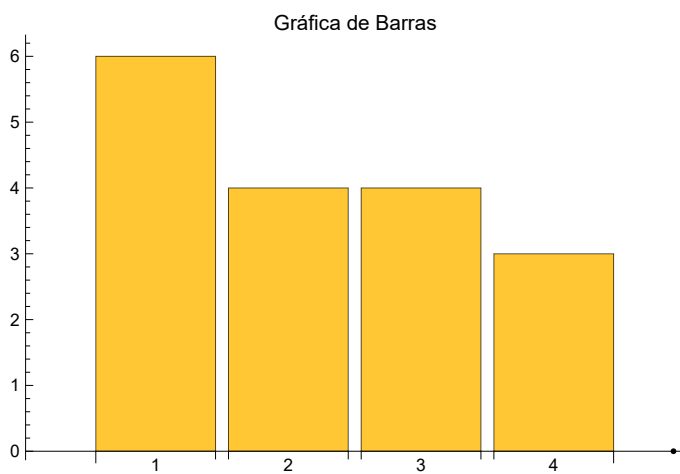
  BarChart[lista, ChartLabels → Range[Length[lista]], PlotLabel → "Gráfica de Barras"],
diagrama de barras  etiquetas de dia... rango longitud  etiqueta de representación

  PieChart[lista, ChartLabels → Range[Length[lista]], PlotLabel → "Gráfica Circular"]
diagrama circular  etiquetas de dia... rango longitud  etiqueta de representación

```

Lista generada: {6, 4, 4, 3}

Out[*]=



Si la longitud de la lista mayor a 5

```

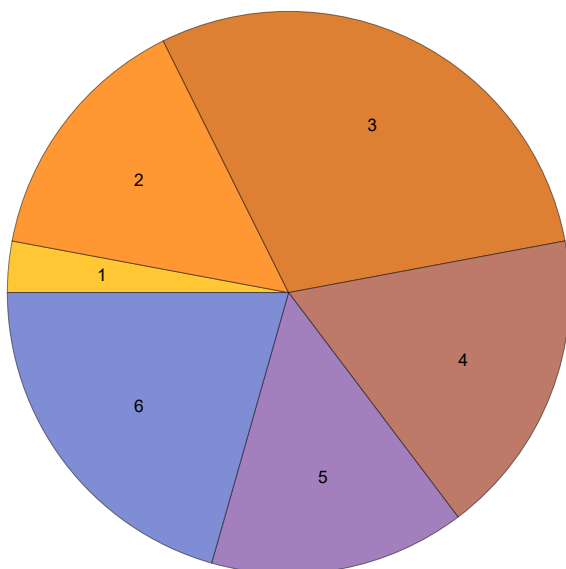
In[*]:= Clear[lista];
borra
lista = RandomInteger[{1, 10}, RandomInteger[{2, 10}]];
entero aleatorio entero aleatorio
Print["Lista generada: ", lista];
describe
If[Length[lista] < 5,
si longitud
BarChart[lista, ChartLabels -> Range[Length[lista]], PlotLabel -> "Gráfica de Barras"],
diagrama de barras etiquetas de diagrama de barras rango longitud etiqueta de representación
PieChart[lista, ChartLabels -> Range[Length[lista]], PlotLabel -> "Gráfica Circular"]
diagrama circular etiquetas de diagrama de barras rango longitud etiqueta de representación

```

Lista generada: {1, 5, 10, 6, 5, 7}

Out[*]=

Gráfica Circular



4. Definir r_1 , si:

- Si es número, que lo divida entre 2.
- Si no que, expanda la expresión.

Ingreso la expresión $\frac{2}{5}$, se divide por 2

```

r1 = Input["Ingrese un número o una expresión algebraica: "];
entra
If[NumericQ[r1], Print["El resultado de dividir entre 2 es: ", r1/2],
si expresión numérica describe
Print["La expresión expandida es: ", Expand[FullSimplify[r1]]];
describe expand simplifica completamente

```

El resultado de dividir entre 2 es: $\frac{1}{5}$

Ingreso la expresión $\cos^2[x] + \sin^2[x]$ se espera que la salida sea 1:

Se usa EXPAND y fullsimplify, porque no todas las expresiones que ingrese el usuario podrán expandirse

```
In[ ]:= Clear[r1];
         |borra
r1 = Input["Ingrese un número o una expresión algebraica: "];
         |entra

If[NumericQ[r1],
  |si |expresión numérica?
  Print["El resultado de dividir entre 2 es: ", r1/2],
  |escribe
  Print["La expresión expandida es: ", Expand[FullSimplify[r1]]];
  |escribe |expand |simplifica completamente
```

La expresión expandida es: 1

5. Defina la función para la delta de Dirac:

$$\delta(x - c) = \begin{cases} \infty & \text{si } x = c \\ 0 & \text{si } x \neq c \end{cases}$$

Grafique la expresión obtenida de -10 a 10.

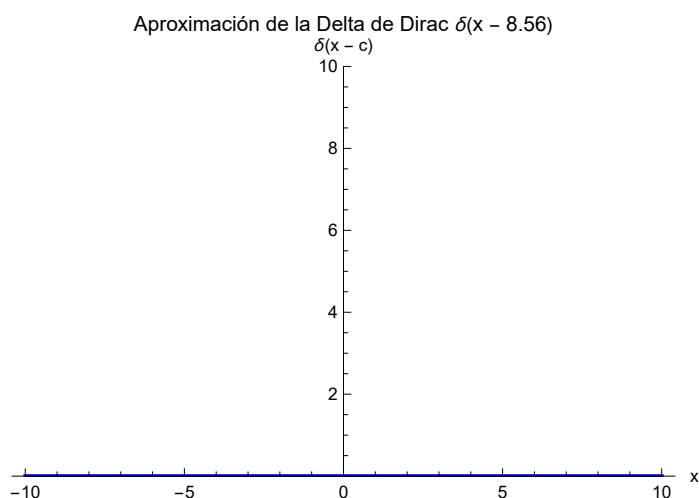
Cumple con lo pedido por definición cuando $x=c \rightarrow \infty$;
si no cumple, $x \neq c \rightarrow 0$

```
In[ ]:= c = Input["Ingrese el valor de c para la delta de Dirac  $\delta(x - c)$ : "];
         |entra

delta[x_] := If[x == c,  $\infty$ , 0];
         |si

Plot[delta[x], {x, -10, 10}, PlotRange -> {0, 10},
  |representación gráfica |rango de representación
  PlotLabel -> "Aproximación de la Delta de Dirac  $\delta(x - "$  <> ToString[c] <> ")",
  |etiqueta de representación |convierte a cadena de cara
  AxesLabel -> {"x", " $\delta(x - c)$ "}, PlotStyle -> Blue, Exclusions -> None]
  |etiqueta de ejes |estilo de repr... |azul |exclusiones |ninguno
```

Out[]:=



Delta de Dirac como función gaussiana, para observar una campana

Un valor pequeño para epsilon hace que el pico sea más estrecho y alto.

- Para que pueda visualizarse la gráfica, usamos la delta de Dirac como función gaussiana: $\delta_\epsilon(x - c) = \frac{1}{\sqrt{2\pi\epsilon}} e^{-\frac{(x-c)^2}{2\epsilon}}$

```
In[ ]:= c = Input["Ingrese el valor de c para la delta de Dirac (centro): "];
      entra
epsilon = Input["Ingrese el valor de epsilon (ejemplo: 0.01): "];
      entra
delta[x_] := (1 / Sqrt[2 Pi epsilon]) Exp[-(x - c)^2 / (2 epsilon)];
      raíz c... número pi exponencial

Plot[delta[x], {x, -10, 10}, PlotRange -> All,
      representación gráfica rango de rep... todo
  PlotLabel -> "Aproximación Gaussiana de la Delta de Dirac en x = "<> ToString[c],
      etiqueta de representación convierte a caden:
  AxesLabel -> {"x", "\delta(x - c)"}, PlotStyle -> Thick]
      etiqueta de ejes estilo de repr... grueso
```

Out[]:=

