
Análisis Real

Sonia Acinas y Fernando Mazzone





30 de abril de 2019

Índice general

1. Breve introducción a Python y SymPy	4
1.1. Descripción	4
1.2. Instalación	4
1.2.1. Anaconda	4
1.2.2. Windows	5
1.2.3. linux	5
1.2.4. Android	5
1.2.5. Computación en la nube	5
1.3. Forma de trabajo: por medio de scripts e interactiva	5
1.4. Características sobresalientes del lenguaje	5
1.5. Elementos del Lenguaje	6
1.5.1. Comentarios	6
1.5.2. Variables	6
1.5.3. Tipo de datos	6
1.5.4. Listas y tuplas	7
1.5.5. Diccionarios	8
1.5.6. Listas por comprensión	8
1.5.7. Funciones	8
1.5.8. Condicionales	9
1.5.9. Bucles	9
2. Sucesiones, series de funciones y sus amigos	12
3. Integral de Riemann	13
3.1. Introducción	13
3.2. Área de figuras elementales planas	13
3.3. Integral de Riemann	15
3.4. Integrabilidad y continuidad	19

Prólogo

El significado de las imágenes en los márgenes es el siguiente:

	Enlace de Internet
	Prestar atención
	Lectura adicional
	Actividad práctica

Bibliografía

Capítulo 1

Breve introducción a Python y SymPy

Descripción

Python es un lenguaje de programación interpretado, abierto, fácil de aprender, potente y portátil. Es utilizado en proyectos de todo tipo, no sólo aplicaciones científicas.

SciPy, Python científico, es un conjunto de módulos de python para distintos tipos de cálculos. Está integrado por los módulos, SymPy (para cálculos simbólicos), numpy (cálculos numéricos), matplotlib (gráficos) entre otros. En este curso sólo usaremos SymPy.

SymPy es una biblioteca de Python para matemática simbólica. Su objetivo es convertirse en un sistema de álgebra computacional (SAC) completo, manteniendo el código lo más simple posible para que sea comprensible y fácilmente extensible. SymPy está escrito enteramente en Python y no requiere de ninguna biblioteca externa.

Matplotlib es una biblioteca de trazado de gráficos de Python que produce figuras de calidad de publicación en una variedad de formatos impresos y entornos interactivos a través de plataformas. Matplotlib se puede utilizar en scripts Python, en el shell Python e IPython, el portátil jupyter, servidores de aplicaciones web.

Instalación

Son muchas las componentes requeridas para poder ejecutar los programas con los que trabajaremos en esta asignatura. Hay que instalar un interprete de python, los módulos que utilizaremos (sympy, matplotlib), es útil utilizar entornos integrados de desarrollo (IDE), que facilitan al usuario editores de código fuente (especializados con la sintaxis de python), consolas de comandos mejoradas (ipython, qt, etc). Otro recurso que se dispone son las notebooks, de las cuales hablaremos más adelante. Sería engorroso instalar todas estas componentes, que muchas veces tienen orígenes en desarrolladores diferentes, de manera independiente. Para nuestra fortuna existen, las así llamadas, *distribuciones*. Estas en algunos casos son archivos ejecutables que instalan todas las componentes necesarias, o al menos muchas de ellas, de un determinada aplicación. Recomendamos las siguientes distribuciones.

Anaconda

La versión de código abierto de Anaconda es una distribución de alto rendimiento de Python y R e incluye más de 100 de los paquetes científicos más populares asociados a estos lenguajes. Además, se puede acceder a más de 720 paquetes que pueden ser fácilmente instalados con Conda, un programa incluido en Anaconda para la gestión de paquetes. Anaconda tiene licencia BSD que da permiso para utilizar Anaconda comercialmente y para su redistribución. Al día que se escriben estas líneas, anaconda parece la opción más sencilla y completa para instalar todos los recursos necesarios para desarrollar los contenidos de estas notas. Existen versiones para linux, OS X y Windows.



matplotlib



Windows

Hay distribuciones específicas para distintos sistemas operativos. La distribución python(x,y) instala el interprete de python y todos los módulos de scipy. Además el entorno de desarrollo integrado (IDE) spyder.

linux

Aquí todo es más sencillo, el interprete de python suele venir con la distribución del SO y se pueden instalar los módulos, SymPy, NumPy, etc, recurriendo al administrador de paquetes o tipeando la sentencia adecuada en la línea de comandos.

Android

Ppython es una aplicación que permite ejecutar código python y una versión básica de sympy desde tablets y smartphones. Se descarga desde la plataforma google play. .

Computación en la nube

En los últimos tiempos se ha popularizado el uso de la computación en la nube. Esto se trata de servidores que algunas empresas o asociaciones sin fines de lucro facilitan en la web para ejecutar programas en diversos lenguajes. Citamos como ejemplo cocalc. Una vez registrado en el sitio se pueden subir o crear notebooks de jupyter. Soporta varios lenguajes, incluido Python y sus librerías. Como uno utiliza los recursos instalados en el servidor, no se necesita tener instalado ningún interprete de los lenguajes. Sólo se necesita un navegador web actualizado. De este modo pueden ejecutarse programas desde un smartphone o tablet.

Forma de trabajo: por medio de scripts e interactiva

Se puede trabajar de tres formas

1. Interactivamente, ingresando sentencias, de a una por vez, en la línea de comandos y obteniendo respuestas. Se requiere una consola.
2. Haciendo un script (programa) donde se guardan todas las sentencias que se desea ejecutar. Posteriormente este script se puede ejecutar, ya sea desde la línea de comandos o desde un IDE (spyder) oprimiendo un botón de ejecución.
3. En una notebook. Se hacen celdas que contienen porciones de código que pueden ejecutarse.

Características sobresalientes del lenguaje



Seguiremos en esta exposición a [?] de manera cercana. Las principales características del lenguaje son:

- Interpretado. Es necesario un conjunto de programas, el interprete, que entienda el código python y ejecute las acciones contenidas en él.
- Implementa tipos dinámicos.
- Multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.
- Multiplataforma.
- Es comprendido con facilidad. Usa palabras donde otros lenguajes utilizarían símbolos. Por ejemplo, los operadores lógicos !, || y \&\& en Python se escriben not, or y and, respectivamente.



- El contenido de los bloques de código (bucles, funciones, clases, etc.) es delimitado mediante espacios o tabuladores.
- Empieza a contar desde cero (elementos en listas, vectores, etc).

Elementos del Lenguaje

Comentarios

Hay dos formas de producir comentarios, texto que el interprete no ejecuta y que sirve para entender un programa.

Para comentarios largos se utilizan las tildes: `''' comentario '''`.

La segunda notación utiliza el símbolo #, no necesita símbolo de finalización pues se extiende hasta el final de la línea.

```
'''
Comentario largo en un script de Python
'''
print "Hola mundo" # Comentario corto
```

El intérprete no tiene en cuenta los comentarios, lo cual es útil si deseamos poner información adicional en nuestro código como, por ejemplo, una explicación sobre el comportamiento de una sección del programa.

Variables

Las variables se definen de forma dinámica, lo que significa que no se tiene que especificar cuál es su tipo de antemano y que una variable puede tomar distintos valores en distintos momentos de un programa, incluso puede tomar un tipo diferente al que tenía previamente. *Se usa el símbolo = para asignar valores a variables.* Es importante distinguir este = (de asignación) con el igual que es utilizado para definir igualdades en sympy, para ecuaciones por ejemplo.

```
x = 1
x = "texto"
```

Esto es posible porque los tipos son asignados dinámicamente

Tipo de datos

Python implementa diferentes tipos de datos. Para la noción de *tipos de datos* en general ver [?] . A continuación describimos sumariamente algunos de los tipos más comunes presentes en Python. Cuando se utilizan módulos específicos (p. ej. sympy) la diversidad de tipos de datos se expande, con la incorporación de tipos con significación matemática, p.ej. matrices, expresiones algebraicas, etc.

Tipo	Clase	Notas	Ejemplo
str	Cadena	Inmutable	'Cadena'
list	Secuencia	Mutable, puede contener objetos de diversos tipos	[4.0, 'Cadena', True]
tuple	Secuencia	Inmutable, puede contener objetos de diversos tipos	(4.0, 'Cadena', True)
dict	Mapping	Grupo de pares clave:valor	{'key1': 1.0, 'key2': False}
int	Número entero	Precisión fija, convertido en long en caso de overflow.	42
long	Número entero	Precisión arbitraria	42L ó 456966786151987643L
float	Número decimal	Coma flotante de doble precisión	3.1415927
complex	Número complejo	Parte real y parte imaginaria j.	(4.5 + 3j)
bool	Booleano	Valor booleano verdadero o falso	True o False

Se clasifican en:

Mutable si su contenido puede cambiarse.

Inmutable si su contenido no puede cambiarse.

Se usa el comando `\type` para averiguar que tipo de dato contiene una variable

```
>>> x=1
>>> type(x)
<type 'int'>
>>> x='Ecuaciones'
>>> type(x)
<type 'str'>
```

Listas y tuplas

- Es una estructura de dato, que contiene, como su nombre lo indica, listas de otros datos en cierto orden. Listas y tuplas son muy similares.
- Para declarar una lista se usan los corchetes [], en cambio, para declarar una tupla se usan los paréntesis (). En ambos casos los elementos se separan por comas, y en el caso de las tuplas es necesario que tengan como mínimo una coma.
- Tanto las listas como las tuplas pueden contener elementos de diferentes tipos. No obstante las listas suelen usarse para elementos del mismo tipo en cantidad variable mientras que las tuplas se reservan para elementos distintos en cantidad fija.
- Para acceder a los elementos de una lista o tupla se utiliza un índice entero (empezando por "0", no por "1"). Se pueden utilizar índices negativos para acceder elementos a partir del final.
- Las listas se caracterizan por ser mutables, mientras que las tuplas son inmutables.

```
>>> lista = ["abc", 42, 3.1415]
>>> lista[0] # Acceder a un elemento por su indice
'abc'
>>> lista[-1] # Acceder a un elemento usando un indice negativo
3.1415
>>> lista.append(True) # Agregar un elemento al final de la lista
>>> lista
['abc', 42, 3.1415, True]
>>> del lista[3] # Borra un elemento de la lista usando un indice
>>> lista[0] = "xyz" # Re-asignar el valor del primer elemento
>>> lista[0:2] # elementos del indice "0" al "1"
['xyz', 42]
>>> lista_anidada = [lista, [True, 42L]] #Es posible anidar listas
>>> lista_anidada
[['xyz', 42, 3.1415], [True, 42L]]
>>> lista_anidada[1][0] #accede lista dentro de otra lista
True

>>> tupla = ("abc", 42, 3.1415)
>>> tupla[0] # Acceder a un elemento por su indice
'abc'
>>> del tupla[0] # No es posible borrar ni agregar
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object doesn't support item deletion
>>> tupla[0] = "xyz" # Tampoco es posible re-asignar
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> tupla[0:2] # elementos del indice "0" al "2" sin incluir
('abc', 42)
>>> tupla_anidada = (tupla, (True, 3.1415)) # es posible anidar
>>> 1, 2, 3, "abc" # Esto tambien es una tupla
(1, 2, 3, 'abc')
>>> (1) # no es una tupla, ya que no posee al menos una coma
1
>>> (1,) # si es una tupla
(1,)
>>> (1, 2) # Con mas de un elemento no es necesaria la coma final
```



```
(1, 2)
>>> (1, 2,) # Aunque agregarla no modifica el resultado
(1, 2)
```

Diccionarios

- Para declarar un diccionario se usan las llaves {}. Contienen elementos separados por comas, donde cada elemento está formado por un par clave:valor (el símbolo : separa la clave de su valor correspondiente).
- Los diccionarios son mutables, es decir, se puede cambiar el contenido de un valor en tiempo de ejecución.
- En cambio, las claves de un diccionario deben ser inmutables. Esto quiere decir, por ejemplo, que no podremos usar ni listas ni diccionarios como claves.
- El valor asociado a una clave puede ser de cualquier tipo de dato, incluso un diccionario.

```
>>> dicci = {"cadena": "abc", "numero": 42, "lista": [True, 42L]}
>>> dicci["cadena"] # Usando una clave, se accede a su valor
'abc'
>>> dicci["lista"][0]
True
>>> dicci["cadena"] = "xyz" # Re-asignar el valor de una clave
>>> dicci["cadena"]
'xyz'
>>> dicci["decimal"] = 3.1415927 # nuevo elemento clave:valor
>>> dicci["decimal"]
3.1415927
>>> dicci_mixto = {"tupla": (True, 3.1415), "diccionario": dicci}
>>> dicci_mixto["diccionario"]["lista"][1]
42L
>>> dicci = {("abc",): 42} # tupla puede ser clave
>>> dicci = {"abc": 42} # una clave no puede ser lista
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

Listas por comprensión

Una lista por comprensión es una expresión compacta para definir listas. Al igual que el operador lambda, aparece en lenguajes funcionales. Ejemplos:

`range(n)` devuelve una lista, empezando en 0 y terminando en $n - 1$.

```
>>> range(5) #
[0, 1, 2, 3, 4]
>>> [i*i for i in range(5)]
[0, 1, 4, 9, 16]
>>> lista = [(i, i + 2) for i in range(5)]
>>> lista
[(0, 2), (1, 3), (2, 4), (3, 5), (4, 6)]
```

Funciones

- Las funciones se definen con la palabra clave `def`, seguida del nombre de la función y sus parámetros. Otra forma de escribir funciones, aunque menos utilizada, es con la palabra clave `lambda` (que aparece en lenguajes funcionales como Lisp). Generalmente esta forma es apropiada para funciones que es posible definir en una sola línea.
- El valor devuelto en las funciones con `def` será el dado con la instrucción `return`.

```
>>> def suma(x, y = 2): #el argumento y tiene un valor por defecto
...     return x + y # Retornar la suma
...
>>> suma(4) # La variable "y" no se modifica, siendo su valor: 2
6
>>> suma(4, 10) # La variable "y" si se modifica
14

>>> suma = lambda x, y = 2: x + y
>>> suma(4) # La variable "y" no se modifica
6
>>> suma(4, 10) # La variable "y" si se modifica
14
```

Condicionales

Una sentencia condicional (**if** condicion) ejecuta su bloque de código interno sólo si condicion tiene el valor booleano **True**. Condiciones adicionales, si las hay, se introducen usando **elif** seguida de la condición y su bloque de código. Todas las condiciones se evalúan secuencialmente hasta encontrar la primera que sea verdadera, y su bloque de código asociado es el único que se ejecuta. Opcionalmente, puede haber un bloque final (la palabra clave **else** seguida de un bloque de código) que se ejecuta sólo cuando todas las condiciones fueron falsas.

```
>>> verdadero = True
>>> if verdadero: # No es necesario poner "verdadero == True"
...     print "Verdadero"
...
Verdadero
>>> else:
...     print "Falso"
...
File "<stdin>", line 1
    else:
    ^
SyntaxError: invalid syntax

>>> print "Falso"
File "<stdin>", line 1
    print "Falso"
    ^
IndentationError: unexpected indent
```

```
>>> lenguaje = "Python"
>>> if lenguaje == "C":
...     print "Lenguaje de programacion: C"
... elif lenguaje == "Python": # tantos "elif" como se quiera
...     print "Lenguaje de programacion: Python"
... else:
...     print "Lenguaje de programacion: indefinido"
...
Lenguaje de programacion: Python
>>> if verdadero and lenguaje == "Python":
...     print "Verdadero y Lenguaje de programacion: Python"
...
Verdadero y Lenguaje de programacion: Python
```

Bucles

El bucle **for** es similar a otros lenguajes. Recorre un objeto *iterable*, esto es una lista o una tupla, y por cada elemento del iterable ejecuta el bloque de código interno. Se define con la palabra clave **for** seguida de un nombre de variable, seguido de **in** seguido del iterable, y finalmente el bloque de código interno. En cada iteración, el elemento siguiente del iterable se asigna al nombre de variable especificado:

```
>>> lista = ["a", "b", "c"]
>>> for i in lista: # Iteramos sobre una lista, que es iterable
```

```
...     print i
...
a
b
c
>>> cadena = "abc"
>>> for i in cadena: # Iteramos sobre una cadena, que es iterable
...     print i # una coma al final evita un salto de línea
...
a
b
c
```

El bucle **while** evalúa una condición y, si es verdadera, ejecuta el bloque de código interno. Continúa evaluando y ejecutando mientras la condición sea verdadera. Se define con la palabra clave **while** seguida de la condición, y a continuación el bloque de código interno:

```
>>> numero = 0
>>> while numero < 3:
...     print numero
...     numero += 1
...
0
1
2
```

Bibliografía

Capítulo 2

Sucesiones, series de funciones y sus amigos

Capítulo 3

Integral de Riemann

Introducción

« Bernard Riemann recibió su doctorado en 1851, su *Habilitación* en 1854. La *habilitación* confiere el reconocimiento de la capacidad de crear sustanciales contribuciones en la investigación más allá de la tesis doctoral, y es un requisito necesario para ocupar un cargo de profesor en una universidad Alemana. Riemann eligió como tema de *habilitación* el problema de las series de Fourier. Su tesis fue titulada *Über die Darstellbarkeit einer Function durch eine trigonometrische Reihe* (Sobre la representación de una función por series trigonométricas) y respondía la pregunta: Cuándo una función definida en el intervalo $(-\pi, \pi)$ puede ser representada por la serie trigonométrica $a_0/2 + \sum_{n=1}^{\infty} [a_n \cos(nx) + b_n \sin(nx)]$? En este trabajo es donde hallamos la Integral de Riemann, introducida en una sección corta antes del núcleo principal de la tesis, como parte del trabajo preparatorio que él necesitó desarrollar antes de abordar el problema de representabilidad por series trigonométricas. »



Bernhard Riemann
1826-1866



David M. Bressoud
A Radical Approach to Lebesgue's Theory of Integration.

En este capítulo vamos a desarrollar el concepto de la integral de Riemann. Vamos a exponer la definición de la integral debida a Riemann y la ideada por J. G. Darboux. Mostraremos la equivalencia de las dos definiciones y discutiremos las propiedades de la integral, sus alcances y límites. Preparamos así el camino para la introducción de la integral de Lebesgue.



Debemos advertir al alumno que en este curso dejaremos un poco de lado las cuestiones procedimentales de cómo calcular integrales, aspecto que seguramente abordó en cursos anteriores y del cual nos vamos a valer. Tampoco debe esperar que las actividades prácticas se centren en esa dirección. Nuestro principal objetivo aquí es discutir la materia conceptual ligada a la integral y cómo es previsible las actividades prácticas estarán orientadas con ese propósito.

El concepto de integral encuentra su motivación en diversos problemas. Aparece cuando se busca el centro de masas de un determinado cuerpo, cuando se quieren hallar longitudes de arco, volúmenes, cuando se quiere reconstruir el movimiento de cuerpo conocida su velocidad, etc. La integral es utilizada en incontables otros conceptos matemáticos, como ser el mencionado más arriba relativo a las series de Fourier.

Quizás el problema más simple donde aparece la integral es el que utilizaremos como motivación para introducirla y es el concepto de área. Vamos a tratar de reconstruir este concepto desde su base, esto es analizando la noción de área de figuras tan simples como rectángulos, triángulos, etc.

Área de figuras elementales planas

El cálculo de áreas es necesario en multitud de actividades humanas, por ejemplo con el comercio. La cantidad de muchos productos y servicios se estima en medidas de área, por ejemplo: las telas, el trabajo de un colocador de pisos, el precio de la construcción, el valor de las extensiones de tierra, etc.



Jean G. Darboux
1842-1917

Por figuras elementales planas nos referimos a rectángulos, triángulos, trapecios, etc. Sin duda el alumno debe estar muy familiarizado con las áreas de estas figuras, el área de un rectángulo viene dada por la conocida fórmula $b \times h$, donde b es la base del rectángulo y h su altura. Ahora bien, ¿Cómo se llega a esta fórmula? Porque esta fórmula es apropiada para calcular el precio de un terreno por ejemplo. En esta sección vamos a justificar esta fórmula a partir de algunos hechos elementales.

Vamos a considerar un plano \mathcal{P} . En este plano \mathcal{P} supondremos fijada una unidad de longitud. Pretendemos asignar un área a las figuras, es decir a los subconjuntos, de \mathcal{P} . De ahora en más, cómo es usual en esta materia nos referiremos a *medida* en lugar de área. La medida es un concepto más general que el concepto de área. No obstante en el contexto en que estamos actualmente son sinónimos.

Queremos construir pues una función m tal que $m(A)$ represente la medida de $A \subset \mathcal{P}$. Ahora bien ¿qué podemos usar de guía con ese objetivo? Si, como dijimos, desconocemos todas las fórmulas previamente aprendidas, sobre que partimos para construir la medida o área. La respuesta es que tomaremos como principio rector ciertas propiedades que son deseables que una medida satisfaga. Ellas son las siguientes.

Positividad. debería ser una magnitud no negativa.

Invariancia por movimientos rígidos. Si una región es transformada en otra por medio de un movimiento rígido, ambas regiones deberían tener la misma área. Otra manera de expresar esta propiedad es diciendo que dos figuras *congruentes* tienen la misma área.

Aditividad. Si una región es la unión de cierta cantidad de regiones más chicas mutuamente disjuntas

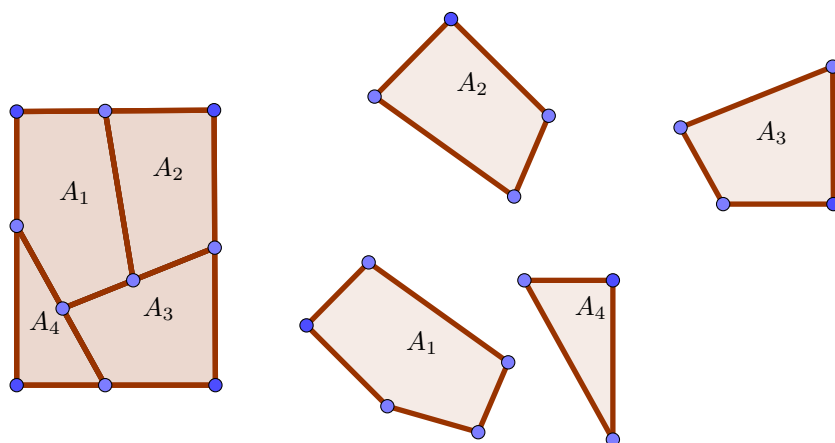


Figura 3.1: El área del rectángulo es la suma de sus partes

Utilizando la segunda y tercer propiedad se pueden relacionar el área del rectángulo de la figura 3.1 con las cuatro regiones en la que es dividido.

Como veremos a lo largo de la materia la propiedad de aditividad debe ser estudiada con cuidado, esto ocurre por las intrincadas maneras en que una región puede ser unión de otras regiones. A lo largo de esta materia elaboraremos una teoría que nos dará una descripción precisa de a que conjuntos podemos asignarle una medida de modo que las propiedades previas sean ciertas.

Por el momento veamos como las propiedades anteriores determinan prácticamente de manera unívoca la medida de regiones elementales planas.

Hablando de propiedades de la medida, supongamos que A y B son dos regiones con $A \subset B$. Entonces como $B = A \cup (B - A)$ y por la propiedad de aditividad y positividad

$$m(B) = m(A) + m(B - A) \geq m(A).$$

Descubrimos así que nuestra medida deberá tener adicionalmente la siguiente propiedad:

Monotonía. Si $A \subset B$ entonces $m(A) \leq m(B)$.

Es claro que si logramos construir una medida que satisfaga las propiedades anteriores cualquier múltiplo por un número real positivo de ella seguirá cumpliendo las propiedades. Esto es una manera de expresar el hecho que podemos usar diferentes unidades de medición. Esta cuestión se sortea proponiendo la unidad de medida. Esta unidad es completamente arbitraria, ud. podría elegir su figura plana preferida como unidad de área. Como es habitual, elijamos el cuadrado cuyos lados miden la unidad de longitud previamente fijada.

Supongamos ahora que tenemos un rectángulo de un lado igual a la unidad y el otro de lado un racional n/m , $n, m \in \mathbb{N}$. Veamos que la aditividad, la invariancia por movimientos rígidos y el hecho que decidimos que el cuadrado de lados igual a la unidad determinan el área de este rectángulo. Primero observar que si dividimos el lado de cuadrado unidad en m segmentos iguales de longitud. Queda dividido el cuadrado en m rectángulos R_1, \dots, R_m (ver figura en el margen), todos ellos congruentes entre sí, de modo que todos tienen la misma medida, digamos $m(R_1)$. La unión de ellos es el cuadrado que por convención dijimos que tiene medida 1. De modo que por la aditividad debe ocurrir que $m(R_1) = \dots = m(R_m) = 1/m$. Recordemos nuestra pretensión de inferir la medida de un rectángulo R de lado 1 y otro n/m . Este rectángulo está compuesto de n rectángulos congruentes a los R_i , $i = 1, \dots, m$, nuevamente por la aditividad inferimos que $m(R) = n/m$.

Sea ahora un rectángulo R con un lado unidad y el otro un real cualquiera $l > 0$. Existen sucesiones $0 < q_k, p_k \in \mathbb{Q}$, $k \in \mathbb{N}$, tales que $q_1 \leq q_2 \leq \dots \leq l \leq \dots \leq p_2 \leq p_1$ y $\lim_{k \rightarrow \infty} q_k = \lim_{k \rightarrow \infty} p_k = l$. Consideremos una dos sucesiones de rectángulos R_k y S_k que comparten el lado de R igual a la unidad, mientras que el otro lado de R_k y S_k es igual a q_k y p_k respectivamente. Luego por la monotonía

$$q_k = m(R_k) \leq m(R) \leq m(S_k) \leq p_k.$$

Tomando límite cuando $k \rightarrow \infty$ inferimos que $m(R) = l$.

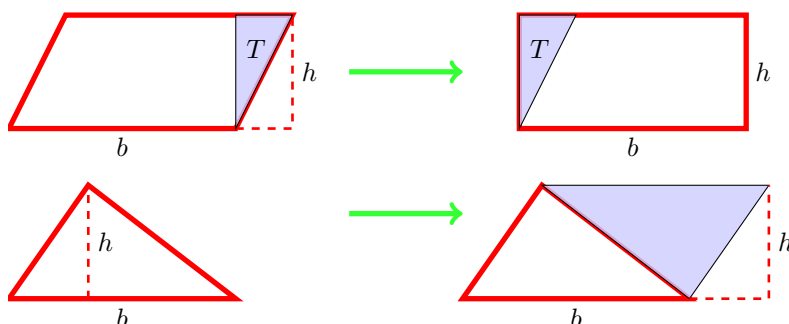


Figura 3.2: Áreas de otras figuras elementales.

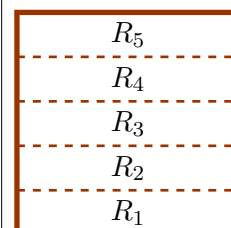
A partir de las propiedades fundamentales que postulamos para la medida o área inferimos la famosa fórmula del área de un rectángulo en el caso que uno de los lados sea igual a la unidad. Para un rectángulo arbitrario. En la figura 3.2 se muestra como relacionar el área de un paralelepípedo con la de un rectángulo y la de un triángulo con la de un paralelepípedo para inferir las conocidas fórmulas para estas figuras.

Integral de Riemann

En esta sección abordaremos el problema del área de regiones planas. Vamos a contextualizarnos dentro del marco conceptual que nos brinda la geometría analítica. Mediante coordenadas cartesianas ortogonales los puntos del plano se identifican con pares ordenados $(x, y) \in \mathbb{R}^2$ y el plano con el conjunto \mathbb{R}^2 . Nuestro propósito es entonces definir la medida de subconjuntos de \mathbb{R}^2 . La geometría analítica abre así nuevas posibilidades para abordar el problema del área.

Nuestra primera aproximación será la que propuso Bernhard Riemann en 1854, pero seguiremos el enfoque de Jean Darboux. En esta parte de nuestra exposición consideraremos subconjuntos de \mathbb{R}^2

Podríamos por ejemplo elegir el círculo de radio uno como unidad de área. Así ya no tendríamos el problema de ese número raro π que aparece en la fórmula del área del círculo. ¡El área de cualquier círculo sería igual a su radio al cuadrado! Claro que aparecería π en la fórmula del área del cuadrado de lado 1. Nos tapamos los pies y se destapa el cuerpo.



Descomposición rectángulo R

de un tipo especial, concretamente a conjuntos que quedan encerrados entre la gráfica de una función y del eje coordenadas x . Esto nos lleva al concepto de integral.

Definición 1 (Partición)

Sea $[a, b]$ un intervalo. Una *partición* P es un conjunto ordenado y finito de puntos, donde el primer elemento es a y el último b . Es decir $P = \{x_0, x_1, \dots, x_n\}$, donde $a = x_0 < x_1 < \dots < x_n = b$.

Definición 2 (Sumas de Darboux)

Sea $f : [a, b] \rightarrow \mathbb{R}$ una función acotada y $P = \{x_0, x_1, \dots, x_n\}$ una partición de $[a, b]$. Consideremos las siguientes magnitudes

$$m_i := \inf\{f(x) | x \in [x_{i-1}, x_i]\}$$

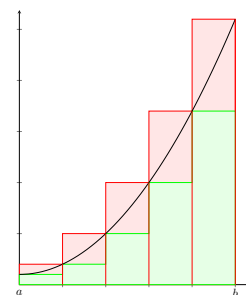
$$M_i := \sup\{f(x) | x \in [x_{i-1}, x_i]\}$$

Definimos la *Suma superior de Darboux* como

$$\overline{S}(P, f) = \sum_{i=1}^n M_i(x_i - x_{i-1}),$$

y la *Suma inferior de Darboux* como

$$\underline{S}(P, f) = \sum_{i=1}^n m_i(x_i - x_{i-1}),$$



Sumas de Darboux.

Lema 1 (Monotonía sumas de Darboux)

Sea $f : [a, b] \rightarrow \mathbb{R}$ una función acotada y $P = \{x_0, x_1, \dots, x_n\}$ una partición de $[a, b]$. Supongamos que P' es otra partición que tiene un pnto más que P . Entonces

$$\underline{S}(P', f) \geq \underline{S}(P, f) \quad \text{y} \quad \overline{S}(P', f) \leq \overline{S}(P, f)$$

Ejercicio

Sea $f : [a, b] \rightarrow \mathbb{R}$ una función acotada y P, P' particiones de $[a, b]$ con $P \subset P'$. Demostrar que

$$\underline{S}(P, f) \leq \underline{S}(P', f) \quad \text{y} \quad \overline{S}(P', f) \leq \overline{S}(P, f).$$

Inferir que para cualesquiera P, P' (sin importar que una este o no contenida dentro de la otra)

$$\underline{S}(P, f) \leq \overline{S}(P, f).$$

Definición 3 (Funciones integrables)

Sea $f : [a, b] \rightarrow \mathbb{R}$ una función acotada. Diremos que f es *integrable Riemann* si

$$\sup \{ \underline{S}(P, f) \mid P \text{ partición de } [a, b] \} = \inf \{ \overline{S}(P, f) \mid P \text{ partición de } [a, b] \} \quad (3.1)$$

En caso que f sea integrable llamamos *integral* entre a y b de f al valor de los dos miembros de (3.1) y este número se denota

$$\int_a^b f(x) dx.$$

Teorema 1 (Primer criterio de integrabilidad)

Sea $f : [a, b] \rightarrow \mathbb{R}$ una función acotada. Entonces f sea integrable si para todo $\varepsilon > 0$ existe una partición P tal que

$$\overline{S}(P; f) - \underline{S}(P; f) < \varepsilon. \quad (3.2)$$

Dem. La demo

□

Ejemplo 3.0. Sea $0 \leq a < b$ veamos que

$$\int_a^b x dx = \frac{b^2}{2} - \frac{a^2}{2}.$$

Ejercicio

Sea $0 \leq a < b$ veamos que

$$\int_a^b x^2 dx = \frac{b^3}{3} - \frac{a^3}{3}.$$

Ayuda: Usar particiones uniformes y la fórmula $\sum_{i=1}^n n^2 = n(n+1)(2n+1)/6$.

Ejemplo 3.1. Sea $0 \leq a < b$ y $n \in \mathbb{N}$, veamos que

$$\int_a^b x^n dx = \frac{b^{n+1}}{n+1} - \frac{a^{n+1}}{n+1}.$$

Usamos particiones no uniformes

Ejercicio

Sea $0 \leq a < b$ y n un entero negativo, veamos que

$$\int_a^b x^n dx = \begin{cases} \frac{b^{n+1}}{n+1} - \frac{a^{n+1}}{n+1} & \text{si } n \neq -1 \\ \ln(b) - \ln(a) & \text{si } n = -1 \end{cases}$$

Ejemplo 3.2. Sea $0 \leq a < b \leq \pi/2$, veamos que

$$\int_a^b \sin x dx = -(\cos(b) - \cos(a)).$$

Ejercicio

Sea $0 \leq a < b \leq \pi$, veamos que

$$\int_a^b \sin x dx = -(\cos(b) - \cos(a)).$$

Ejemplo 3.3. Usamos SymPy y sumas de Darboux aproximar el valor de π . Utilizamos el hecho que $\pi/4$ es el área de un cuarto de círculo de radio 1. Entonces

$$\pi = 4 \int_0^1 \sqrt{1-x^2} dx.$$

```
from sympy import *
N=1000.0
lim=int(N+1)
x=symbols('x')
f=sqrt(1-x**2)
Sinf=sum([ f.subs(x,i/N)*1/N for i in range(1,lim)])
Ssup=sum([f.subs(x,(i-1)/N)*1/N for i in range(1,lim)])
```

Encontramos la estimación

$$3,13955546691103 \leq \pi \leq 3,14355546691103$$

Ejercicio

Usando SymPy estimar las siguientes integrales

$$\int_1^2 \frac{1}{x} dx,$$

comparar con $\ln(2)$,

$$\int_{-1}^1 x^2 dx$$

¿A qué parece aproximarse las sumas inferiores y superiores?

$$\int_0^{\frac{1}{2}} \frac{1}{\sqrt{1-x^2}} dx,$$

¿Por qué el resultado puede usarse para aproximar π ?

Teorema 2 (Propiedades elementales de la integral)

Sean $f, g : [a, b] \rightarrow \mathbb{R}$ integrables, $\alpha, \beta \in \mathbb{R}$ y $c \in (a, b)$. Entonces

Linealidad $\alpha f + \beta g$ es integrable y

$$\int_a^b \alpha f(x) + \beta g(x) dx = \alpha \int_a^b f(x) dx + \beta \int_a^b g(x) dx.$$

Monotonía Si $f(x) \leq g(x)$ para $x \in [a, b]$ entonces

$$\int_a^b f(x) dx \leq \int_a^b g(x) dx.$$

Aditividad del Intervalo

$$\int_a^b \alpha f(x) dx = \int_a^c \alpha f(x) dx + \int_c^b \alpha f(x) dx.$$

Observación: Las propiedades anteriores son compatibles con las propiedades que habíamos propuesto para el concepto de área en la sección 3.2.

Integrabilidad y continuidad

Teorema 1 (Segundo criterio de integrabilidad)

Sea $f : [a, b] \rightarrow \mathbb{R}$ una función acotada. Entonces f sea integrable si para todo $\varepsilon > 0$ existe un $\delta > 0$ tal que para cualquier partición P que satisface

$$\max_i \{x_i - x_{i-1}\} < \delta,$$

se tiene que

$$\overline{S}(P; f) - \underline{S}(P; f) < \varepsilon. \quad (3.3)$$

Dem. Agarrate catalina

□

Teorema 2 (Continuidad implica integrabilidad)

Si $f : [a, b] \rightarrow \mathbb{R}$ es una función continua entonces es integrable.

Dem. hacer □

¿Qué ocurre con las funciones discontinuas?

Ejemplo 3.4. [Función de Heavside] Es la función

$$H(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}.$$

Es discontinua en $[-1, 1]$ pero integrable.

Ejemplo 3.5. [Función de Dirichlet] Es la función $f : [0, 1] \rightarrow \mathbb{R}$ definida por

$$f(x) = \begin{cases} 1 & \text{si } x \in \mathbb{Q} \\ 0 & \text{si } x \notin \mathbb{Q} \end{cases}$$

Veamos que f es discontinua en todo punto y no integrable.

Ejemplo 3.6. [Función de Thomae] Es la función $f : [0, 1] \rightarrow \mathbb{R}$ definida por

$$f(x) = \begin{cases} \frac{1}{q} & \text{si } x = \frac{p}{q}, p, q \in \mathbb{Z}, \text{m.c.d.}(p, q) = 1 \\ 0 & \text{si } x \notin \mathbb{Q} \end{cases}$$

Para graficarla

```
from matplotlib import pyplot as plt
total=500
q=[]
f=[]
for i in range(1,total):
    for j in range(1,i):
        if gcd(j,i)==1:
            q.append(Rational(j,i))
            f.append(1.0/i)
plt.plot(q,f,'.',markersize=12)
```

Veamos que es discontinua en todo punto racional y es integrable.

Ejemplo 3.7. [Escalera discontinua] Sea $\mathbb{Q} \cap [0, 1] = \{q_1, q_2, \dots\}$ una numeración de los racionales del $[0, 1]$. Definamos $f : [0, 1] \rightarrow \mathbb{R}$ como

$$f(x) = \sum_{n=1}^{\infty} H(x - q_n),$$

donde H es la función de *Heavside*.

```
q=[]
f=[]
total=20
for i in range(1,total):
    for j in range(1,i):
        if gcd(j,i)==1:
            q.append(float(Rational(j,i)))
x=symbols('x')
Heavside=Piecewise((0,x<0),(1,x>=0))
f=sum([Heavside.subs(x,x-q[n])/2**n for n in range(len(q))])
plot(f,(x,0,1))
```

Veamos que f es monotona no decreciente y discontinua en todo punto de $[0, 1] \cap \mathbb{Q}$. Además f es integrable.

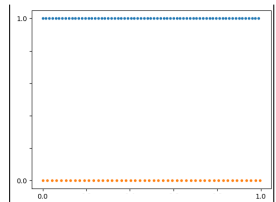
Definición 1 (Oscilación sobre un intervalo)

Sea $f : [a, b] \rightarrow \mathbb{R}$ acotada y $I = [\alpha, \beta] \subset [a, b]$. Definimos la *oscilación* de f en I por

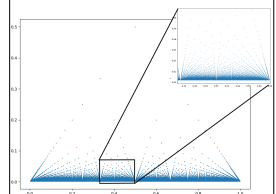
$$w(f, I) = \sup\{f(x) | x \in I\} - \inf\{f(x) | x \in I\}.$$

Ejemplo 3.8.

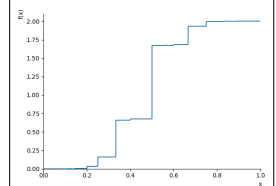
1. Para la función de Dirichlet $w(f, I) = 1$ para todo I con interior no vacío.



Función de Dirichlet



Función de Thomae



Función creciente y discontinua en $\mathbb{Q} \cap [0, 1]$

2. Para la función de Heavside e $I = [\alpha, \beta]$

$$w(f, I) = \begin{cases} 1 & \text{si } 0 \in (\alpha, \beta] \\ 0 & \text{si } 0 \notin (\alpha, \beta] \end{cases}$$

3. Si $I^\circ \neq \emptyset$, f la función de Thomae e $I \subset [0, 1]$ entonces $w(f, I) = 1/q^*$, donde q^* es el mínimo valor de q para el que existe $p \leq q$ tal que $p/q \in I$.

4. Para la función escalera discontinua e $I \subset [0, 1]$

$$w(f, I) = \sum_{q_n \in I} \frac{1}{2^n}.$$

Definición 2

Sea $f : [a, b] \rightarrow \mathbb{R}$ acotada, $\sigma > 0$ y $P = \{x_0, x_1, \dots, x_n\}$ una partición. Definimos

$$I_\sigma := \{i \in \{1, \dots, n\} | w(f, [x_{i-1}, x_i]) > \sigma\}.$$

y

$$R(P, f, \sigma) = \sum_{i \in I_\sigma} (x_i - x_{i-1}).$$

Proposición 1

Si f es continua en $[a, b]$ para todo $\sigma > 0$ existe $\delta > 0$ tal que

$$\max_i (x_i - x_{i-1}) < \delta \Rightarrow I_\sigma = \emptyset \Rightarrow R(P, f, \sigma) = 0.$$

Ejemplo 3.9. Para la función de Dirichlet y para todo $0 < \sigma < 1$ y para toda partición de $[0, 1]$ tenemos $I_\sigma = \{x_1, \dots, x_n\}$ y $R(P, f, \sigma) = [0, 1]$

Ejemplo 3.10. Para la función de Heavside, para todo $0 < \sigma < 1$ y para toda partición de $[0, 1]$ tenemos $I_\sigma = i$, donde i es el índice para el que $i \in (x_{i-1}, x_i]$ y $R(P, f, \sigma) = x_i - x_{i-1}$.

Teorema 3 (Criterio de integrabilidad de Riemann)

Sea f acotada en $[a, b]$ entonces f es integrable si y sólo si para todo $\varepsilon > 0$ y $\sigma > 0$ existe $\delta > 0$ talque $R(P, f, \sigma) < \varepsilon$.

Ejemplo 3.11. Discutir los ejemplos Dirichlet, Heavside, Continuas, escalera discontinua

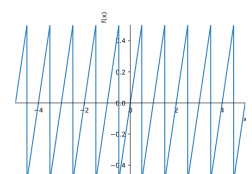
Ejemplo 3.12. [Función de Riemann] Definimos

$$((x)) = x - [x + 0,5]$$

```
x=symbols('x')
g=x-floor(x+.5)
plot(g, (x, -5, 5))
```

Definimos la función de Riemann Porque

$$f(x) = \sum_{n=1}^{\infty} \frac{((x))}{n^2}.$$



Función serrucho

```
f=sum([g.subs(x,n*x)/n**2 for n in range(1,20)])
plot(f,(x,0,1))
```

Demostramos que la función de Riemann es discontinua en los racionales p/q donde $\text{m.c.d.}(p, q) = 1$ y q par. Es integrable en $[0, 1]$.

Definición 3 (Oscilación de una función en un punto)

Sea $f : [a, b] \rightarrow \mathbb{R}$ y $x \in [a, b]$ acotada definimos la *oscilación de f en x* como

$$w(f; x) = \inf_{x \in I^\circ} w(f, I),$$

donde el ínfimo se toma sobre todos los intervalos que contienen a x en su interior.

Ejercicio

f es continua en x si y solo si $w(f; x) = 0$.

Definición 4 (Contenido exterior)

Sea $S \subset \mathbb{R}$. Un *cubrimiento finito* de S es una colección de intervalos $\{[x_{i-1}, x_i]\}_{i=1, \dots, n}$ tal que $S \subset \cup_{i=1}^n [x_{i-1}, x_i]$.

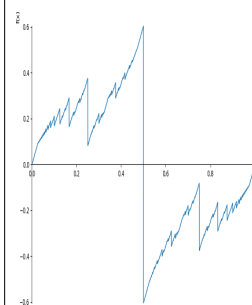
El *contenido exterior* de S se define por

$$c_e(S) = \inf \sum_{i=1}^n (x_i - x_{i-1}),$$

donde el ínfimo es tomado sobre todos los cubrimientos finitos de S .

Teorema 4 (Criterio de integrabilidad de Hankel)

Sea f acotada en $[a, b]$ entonces f es integrable si y sólo si para todo $\sigma > 0$ el conjunto $S_\sigma := \{x \in [a, b] | w(f, x) > \sigma\}$ tiene contenido exterior igual a 0 ($c_e(S_\sigma) = 0$).



Función de Riemann

Índice alfabético

congruencia, 14
Contenido exterior, 22
cubrimiento finito, 22

Heavside, 20

Integrable Riemann, 17
Integral, 17

medida, 14

oscilación, 20, 22

Partición, 16

Suma inferior, 16
Suma superior, 16