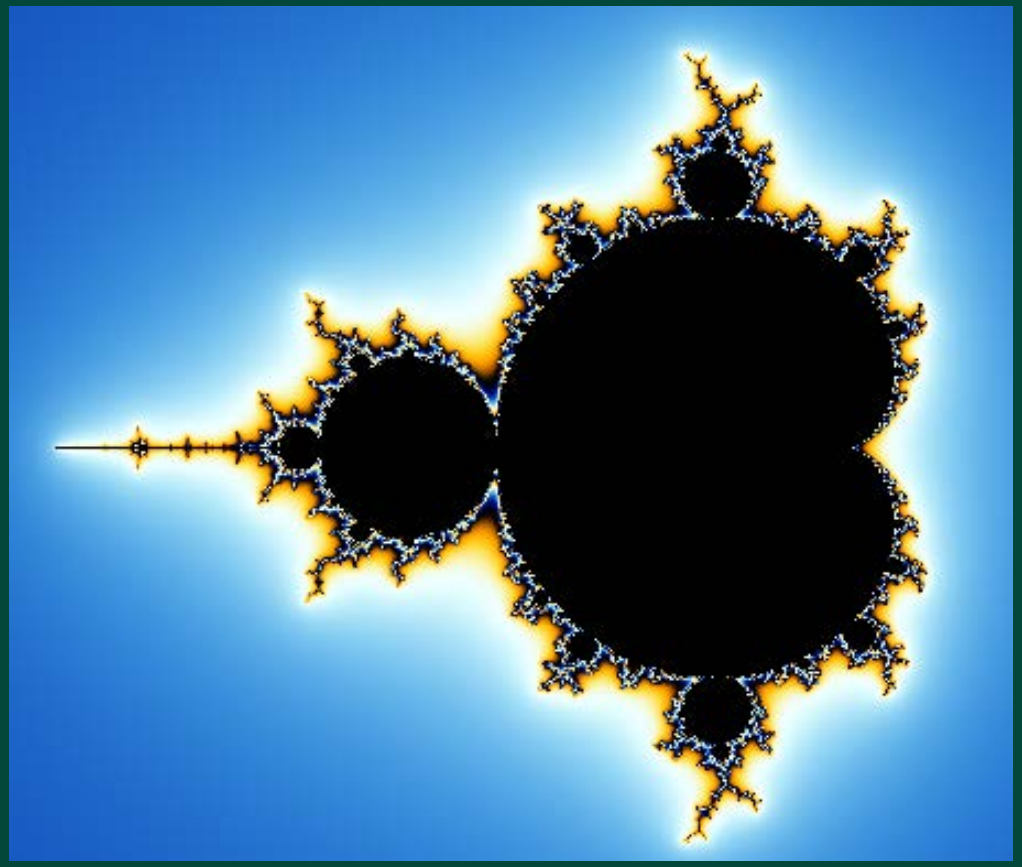
Tutoría 10
Física Computacional I
Grado en Física

Javier Carrasco Serrano, javcarrasco@madrid.uned.es

Física Computacional I, Las Tablas

Tema 13: Fractales

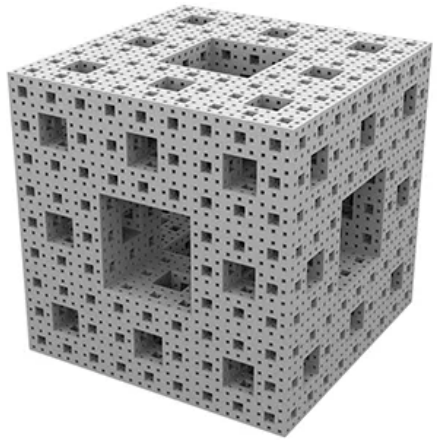


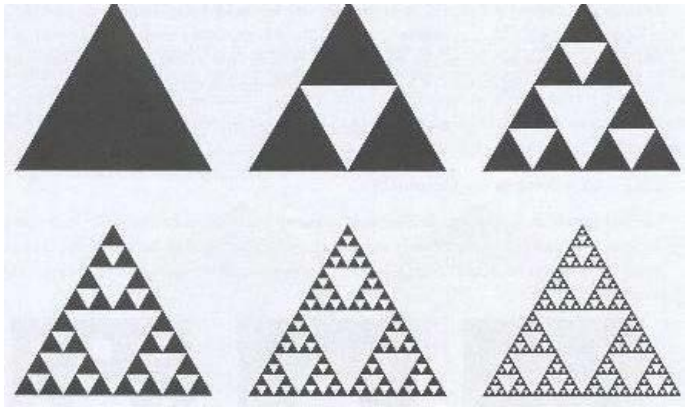
Adobe Acrobat Document

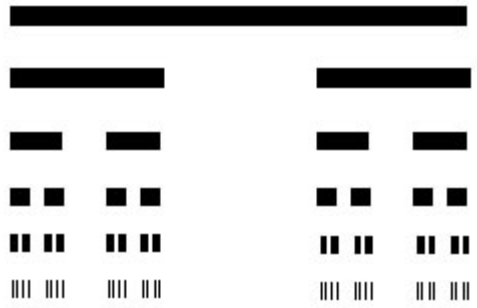
13

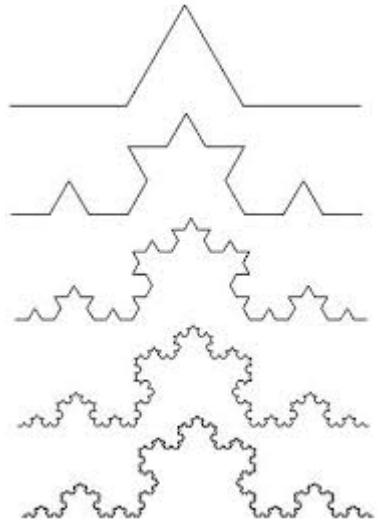
13.1. Fractales: Teoría y Fenomenología - Introducción

- Geometría irregular de la naturaleza, difíciles de medir con la geometría euclídea → geometría fractal; Mandelbrot dota a esta geometría de teoría matemática.
- Naturaleza: “las nubes no son esferas, las montañas no son conos, la costa no son círculos, corteza de los árboles no es lisa, los rayos no son líneas rectas”.
- No derivabilidad en ningún punto, algunas figuras con superficie nula pero longitud infinita…
- Conjunto de Cantor, curva de Koch, triángulo de Sierpinski, monstruo de Mandelbrot, esponja de menger.



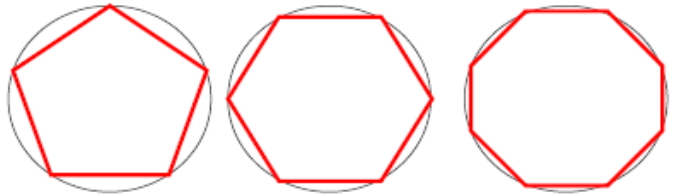






13.1.1. El proceso de medida

- ¿Cómo medir estos objetos de la naturaleza?
- Medimos segmentos rectos con reglas… independientemente de la longitud de la regla, un objeto siempre mide igual → longitud = longitud regla x número reglas.
- ¿Qué pasa si hacemos mediciones con reglas que no cubren bien las longitudes?



- Cuando la longitud de la regla tiende a 0, mide bien la longitud del círculo ($2\pi r$).
- El patrón de medida puede fallar:

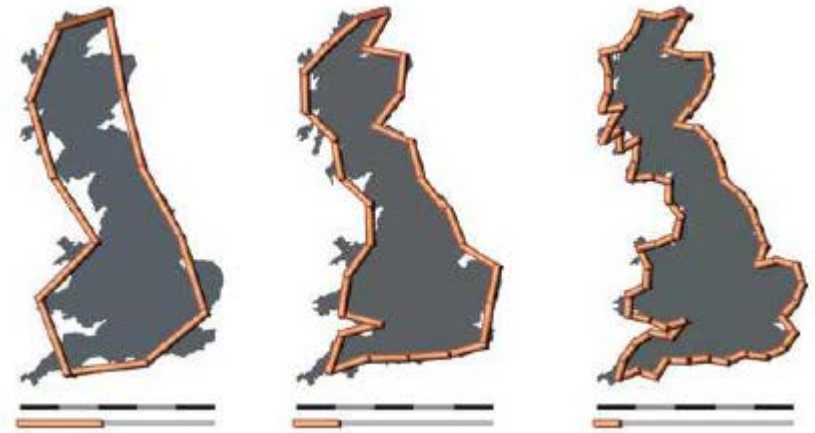
  l=200km → L=2.400km

  l=100km → L=2.800km

  l=50km → L=3.400km

  "realidad" → L=17.820km

  Marcus du Sautoy → infinito (limitación número de Plank)



"imposible medir una distancia más pequeña que 10 elevado a -34 sin crear un agujero negro que aspiraría consigo el instrumento de medir"

13.1.1. El proceso de medida

- ¿Tiene más km de costa Galicia o Andalucía?

  Andalucía: 886km

  Galicia: 1.676km

- El problema de las geometrías puede ser el patrón de medida.
- Dimensión de Hausdorff → dimensión “correcta” de medida.
  - Medida de Hausdorff → su medida asociada (longitud, superficie, volumen…).
- Se puede medir un volumen $V(l)$ de un objeto cubriéndolo con bolas de tamaño lineal l y volumen l^d. Así $V(l) = N(l) \cdot l^d$.
- “Dimensión Hausdorff” de un objeto → d → no cambia si cambiamos la medida l.

13.1.2. Geometría fractal

- ¿Qué ocurre cuando intentamos medir la longitud de un objeto irregular? → Costa Gran Bretaña, km frontera España - Portugal en atlas españoles vs portugueses → la medida con reglas tiende a infinito, la medida con círculos no (Hausdorff).
- ¿Dimensión correcta? Dimensión fractal:

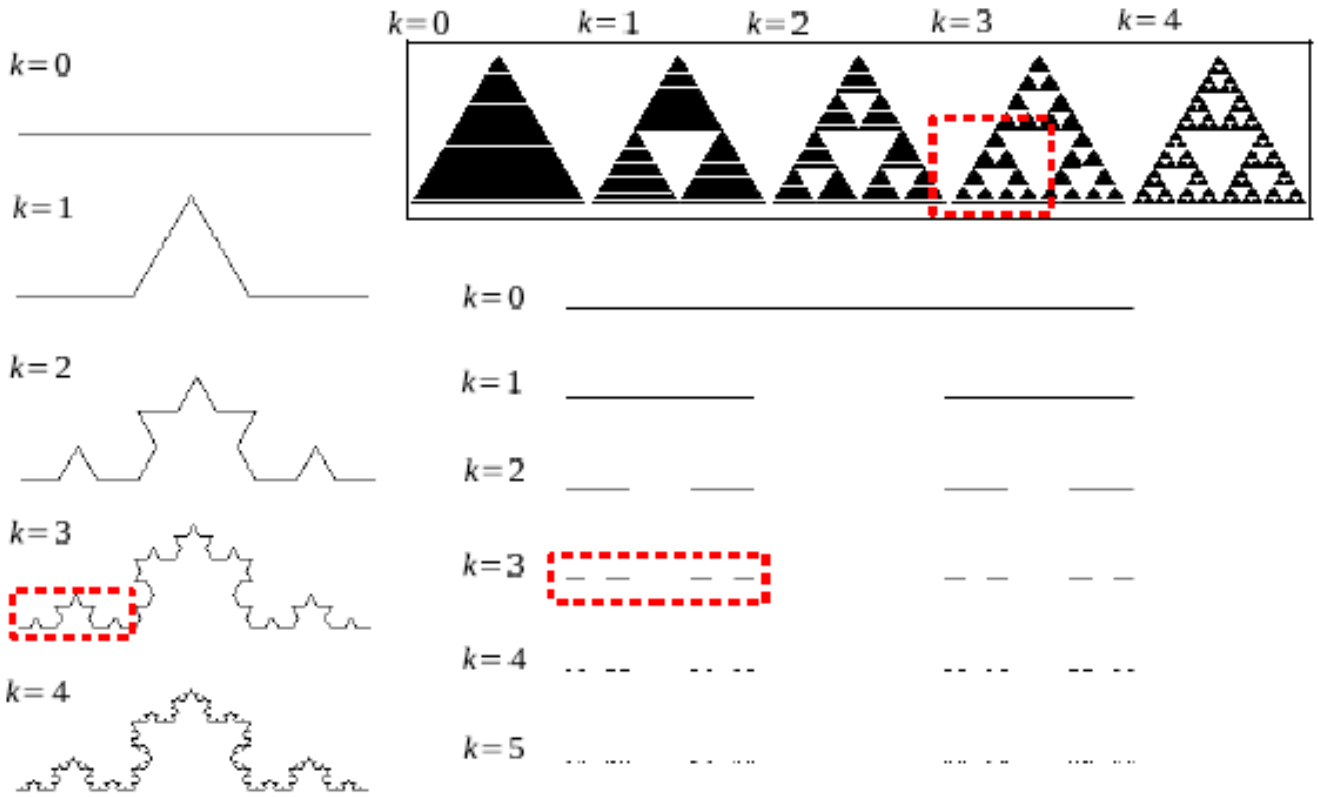
$$N(l) \sim l^{-d_f} \rightarrow d_f = \lim_{l \to 0} \frac{\log N(l)}{\log(1/l)}$$

Ejemplo Gran Bretaña → $d_f = 1.25$.

- Autosimilaridad → propiedad que define a los fractales: cualquier parte es similar al todo. Fractales deterministas cuando son exactas, fractales aleatorios cuando presentan misma estadística (árbol).
- Invarianza bajo cambios de escala → por lo anterior, los fractales no tienen escala o tamaño.

13.1.3. Generación de fractales: fráctales matemáticos

- Fractales matemáticos: fractales que pueden ser generados mediante iteración de sustitución de formas geométricas básicas. Estado inicial + regla geométrica. Regla constante → fractal determinista; regla aleatoria → fractal aleatorio.

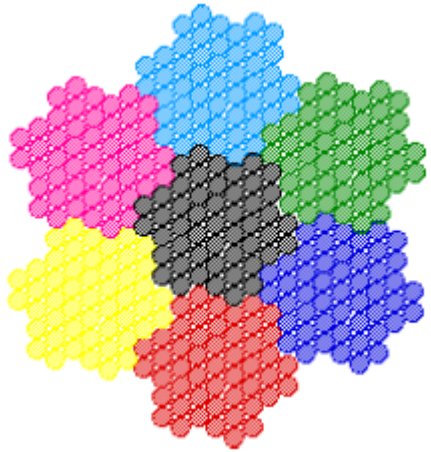


13.1.4. Cómo medir la dimensión fractal

- Método *sandbox* → $M(R) \sim R^{d_f}$ masa del fractal contenida en un círculo de radio R. Mide los puntos que caen dentro de círculos que se pintan dentro del fractal y hace la media.
- Método *box counting* → se mide mediante cajas (segmentos, cuadrados o cubos). $N(l)$ → cajas que cubren el fractal. Se hace para diferentes tamaños entre el máximo tamaño del fractal y un píxel o punto, y se obtienen $N(l) \sim l^{-d_f}$.

13.1.5. Fractales en la naturaleza

- Peculiaridades: longitud infinita que encierra área finita (copo Koch), curvas continuas no diferenciables en ningún punto,
- Medida infinita en una dimensión, encerradas en un espacio finito de dimensión superior.
- Problemas de optimización → solución a problemas de espacio.
- Los árboles optimizan el intercambio oxígeno – dióxido de carbono en un volumen limitado. Sistema nerviso, capilares sanguíneos, árbol bronquial.
- Autosimilitud → estructuras más robustas.
- Empaquetado fractal de fibras ópticas:



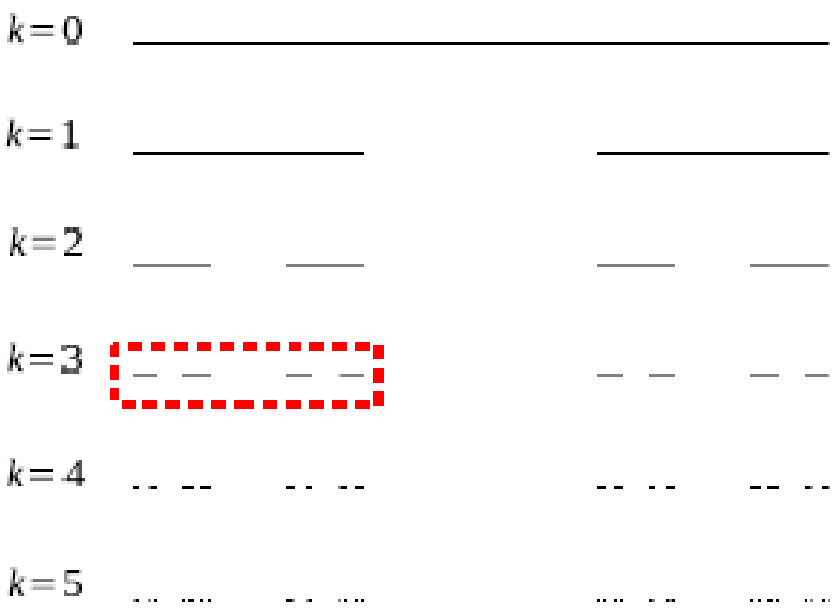
- En general pueden ayudar a resolver problemas de maximización de contacto o intercambio en un medio limitado o acotado.

13.2. Ejemplos y ejercicios en C - Conjunto de Cantor

- Conjunto de Cantor → intervalo inicial $[0,1]$; en cada iteración se divide en tres cada intervalo y se elimina la parte del medio.

En la primera iteración tendríamos los intervalos $\left[0,\frac{1}{3}\right] \cup [\frac{2}{3},1]$, que se vuelven a dividir en 3 y eliminar sus subintervalos medios.

- Se genera una sucesión de conjuntos que converge a un conjunto de puntos disjuntos.



Listado 13.1

Ejercicio 13.1, 13.2, 13.3

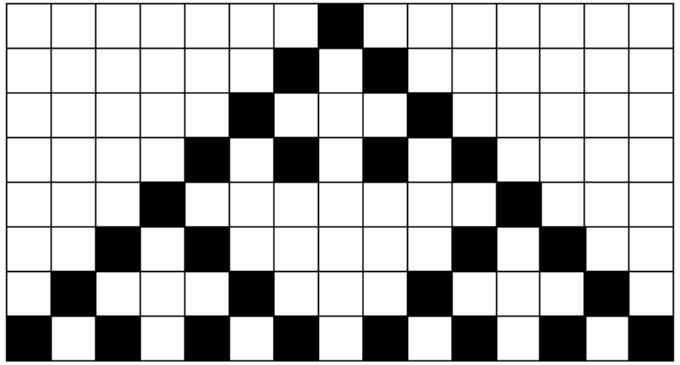
13.2. Ejemplos y ejercicios en C - Triángulo de Sierpinsky

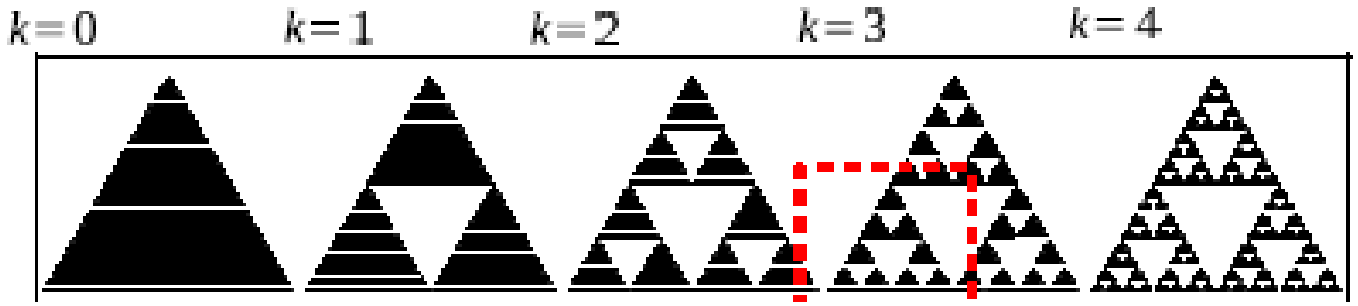
- Triángulo de Sierpinsky → triángulo inicial $\widehat{ABC}$; en cada iteración se eliminan los puntos del triángulo interior (determinado por los tres puntos medios de los lados).

  Se itera eliminando los triángulos centrales de cada triángulo, y converge a un conjunto de puntos fractal.

  Ejercicio 13.4

- Método alternativo: XOR → dibuja el triángulo por filas, y en cada fila decide (iteradamente) si el punto pertenece al conjunto o no. Matriz de 0s y 1s. En la primera línea se coloca un 1 en la posición central. Cada elemento de una fila posterior se construye como el XOR de las posiciones diagonales superiores.





Listado 13.2

Prueba evaluable de programación con C

Adobe Acrobat Document

Criterios Evaluación

Adobe Acrobat Document

- 55% de la nota final, se hace media para la calificación final a partir de 5.
- Septiembre: misma práctica C, corrige el Equipo Docente.
- 4 ejercicios, cada uno se puntúa de 0 a 10. La calificación de la Prueba de C es la media de los 4 ejercicios, “siempre y cuando cada ejercicio tenga al menos un 5”.
- Fecha límite: 19 Mayo 23h55, se sube al apartado “Entrega de trabajos” del campus virtual.
- Entregable: fichero comprimido que contenga:

  - Memoria explicativa del método de resolución y de los resultados obtenidos de cada ejercicio → “memoria_resultados.pdf”

  - Cada programa en un fichero de texto plano con nombre alumn@, en formato .c (ó .h) → “Ejercicio_1.c”, …, “Ejercicio_4.c”

Contenido memoria C

Estructura de cada ejercicio en la memoria:

- Introducción y objetivo → breve descripción de lo que se pretende calcular o simular.
- Metodología → breve explicación de las bases y funcionamiento del código desarrollado para la resolución del ejercicio.
- Resultados obtenidos → resultados de ejecutar el código, en forma de valores numéricos, tablas, gráficas o imágenes.
- Discusión → comentario breve de los resultados obtenidos y de su significado respecto a los objetivos planteados.

Extensión: hasta 5 folios por las dos caras, excluyendo códigos (no es necesario ponerlos en la memoria), y posibles imágenes de anexos. Aproximadamente márgenes de unos 3 cm, letra de 11 pt, interlineado de 1.5

Criterios corrección C

Memoria:

- Solución: explicación del programa C que lo resuelve (no es necesario incluir el código en la memoria).
- Resultados que se obtienen con el programa.
- Análisis/conclusiones. Explicación de los resultados, especialmente si no son los esperados.

Código C:

- Se valoran positivamente comentarios en el código → /* … */
- Comprobar que el código C compila !!! → si no compila y no genera un ejecutable, no se corrige.
- Muy importante: que los resultados coincidan con los que aparecen en la memoria.

Evaluación:

- 20% presentación → descripción objetivo, exposición metodología, presentación resultados.
- 50% resultados → resultado correcto, formato adecuado, análisis/discusión/conclusiones resultados.
- 30% código → cumple requisitos enunciado, estructura, comentarios.

Contenido memoria C

- La solución de cada ejercicio es un programa que puede hacer uso de una o varias funciones. Es importante entender que ahora sí que se deben ejecutar los programas/funciones y obtener resultados a partir de los inputs que se indican en el enunciado, al contrario de lo que sucedía en la práctica de Maxima.
- Pensar en pseudocódigo, ¿cómo lo haríamos nosotros o nuestro cerebro?, ¿qué entradas necesitamos?, ¿cuál es la salida?, ¿qué operaciones intermedias necesitamos hacer → variables o expresiones auxiliares?
- Pensar qué funciones o comandos de C nos ayudan a resolver cada uno de los pasos del problema que planteamos.
- Prueba a construir una función inicial más simple, y luego añade funcionalidades.
- Comentarios para describir cada paso de la función.
- Una vez hecho el programa, probar a guardarlo como un fichero de texto plano, cerrar la sesión, ejecutarlo y ver que funciona correctamente.
- No utilizar tildes, espacios, ni caracteres especiales (ñ, ç…) en el nombre del fichero.

Ejemplos: ejercicios resueltos

Ejercicios exámenes resueltos

- Enunciados:

Adobe Acrobat Document

- Soluciones:

  https://2019.cursosvirtuales.uned.es/dotlrn/grados/asignaturas/61041094-19/file-storage/index?package_key=file-storage&folder_id=42841785&return_url=%2fdotlrn%2fgrados%2fasignaturas%2f61041094-19%2ffile-storage%2f%3f

Carpeta comprimida (en zip

- Ejemplos código bien comentado: ejemplos que aparecen en los temas del Equipo Docente.

Tema 12: Métodos Monte Carlo

Adobe Acrobat Document

12

12.2. Números aleatorios

- Procesos aleatorios: dado, ruleta, bingo… sucesos en los que cada evento es equiprobable, pero impredecible (distribución uniforme). Las probabilidades suman 1.
- Variable aleatoria discreta que toma valores de su espacio muestral (finito).
- Método congruente lineal: genera números “aleatorios” uniformemente distribuidos en [0,1):

$$X_{n+1} = (AX_n + B)\ mod(C)$$

X_0 es la semilla (valor inicial entero), mod es el resto de la división entera.

Se genera en cada paso un entero entre 0 y C-1 → dividiendo entre C → [0, 1).

Como los números que van saliendo están determinados por los valores de la semilla y de la ecuación lineal, en ocasiones se saca un determinado número de “aleatorios” hasta que se elige uno de ellos como nueva semilla.

Ejemplo: Listado 12.1

→ útil para la práctica de C.

→ 1234567891LL quiere decir que es un *long long int.*

12.3. Números aleatorios continuos

- Paso de variables discretas a continuas → probabilidades asociadas a intervalos, cuando la longitud de los intervalos tiende a cero → integrales.
- Ejemplo: distribución normal (o gaussiana).
- Teorema Central de Límite. $Y = X_1 + \cdots + X_n$, X_i variables aleatorias, $N > 30$ para que la estimación de la varianza sea robusta.
- $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$
- $N(0,1)$

Ejemplo: Listado 12.2

Ojo: sqrt → indicación en la sección 10.2 del tema 10.

- Histograma → distribución gráfica de la frecuencia de un evento por intervalos. Muy útil en exploración de datos.

Ejemplo: Listado 12.3

12.4. Caminantes aleatorios y difusión browniana

- Proceso de difusión: desplazamiento de masa o energía de una región/cuerpo en el que hay más a otra en el que hay menos.
- Paseo aleatorio: movimiento sin dirección definida → "acaba" recorriendo todo el espacio (puede que en tiempo infinito…) → movimiento *browniano*. Muchas partículas → choques → difusión *browniana*.
- Suma de desplazamientos aleatorios → distribución normal de las partículas alrededor del punto de partida, D coeficiente de difusión:

$$p(x,t) = \frac{1}{\sqrt{4\pi Dt}} \exp\left(-\frac{x^2}{4Dt}\right)$$

- Difusión aleatoria en una dimensión → caminante aleatorio: avanza o retrocede un paso, 0.5 de probabilidad → distribución normal.
- Biblioteca *libprobabilidad*. Una vez construida, se pueden utilizar las funciones tantas veces como se quiera sin necesidad de volver a definirlas ☺
- La directiva *#ifndef* comprueba primero si una librería está definida (en realidad, el símbolo), y si no está, la procesa (la carga).

Listado 12.4, 12.5

12.4. Caminantes aleatorios y difusión browniana

- Compilación biblioteca *libprobabilidad*:

  gcc –c –o libprobabilidad.o libprobabilidad.c

- Cada vez que se use en un programa hay que incluir:

  #include libprobabilidad.h → en el código del programa

```
gcc -o browniano1d -lm libprobabilidad.o browniano1d.c
```

ejecutable | biblio. matemat. | biblio. probabilidad | programa fuente C

→ en el terminal.

- ¿Cómo simular un aleatorio de dos valores posibles (cara/cruz, 0/1,…)?
  - → Variable aleatoria con distribución uniforme en [0,1)
  - → Un valor si el resultado es < 0.5, el otro si es ≥ 0.5
  - → **PRIMER PASO DE LA PRÁCTICA C**
  - → **Resuelto en Listado 12.6**
  - → A partir de una distribución “aleatoria” (pseudoaleatoria), se está generando una distribución aleatoria.

Adobe Acrobat Document

Gracias!

Javier Carrasco Serrano

Física Computacional I, Las Tablas