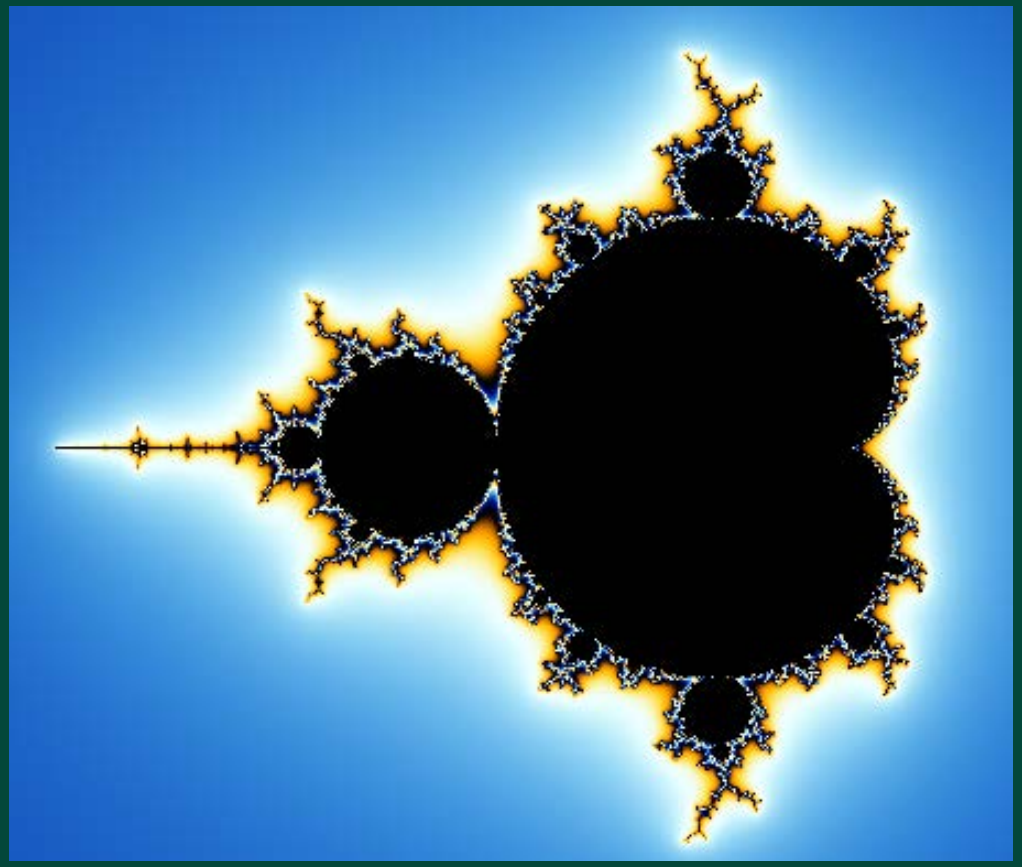
Tutoría 11
Física Computacional I
Grado en Física

Javier Carrasco Serrano, javcarrasco@madrid.uned.es

Física Computacional I, Las Tablas

Tema 14:
Sistemas dinámicos



Adobe Acrobat Document

14

14.1. Introducción

- Sistema Dinámico → modelo matemático que describe la evolución temporal de un proceso.
- Observaciones del proceso + leyes que definen su evolución.
- Población de una especie, **población depredador-presa**, propagación enfermedades, concentraciones químicas, energía, temperatura…
- Sistemas dinámicos discretos (series finitas) o continuos (tiempo continuo).
- Sistemas deterministas (determinados por el estado inicial) → Autómatas celulares (Tema 15); y estocásticos (componente aleatorio en la evolución de un estado a otro) → caminantes aleatorios.
- Espacio de fases → conjunto de valores que pueden tomar las variables de sistema.
- Condición inicial (X_0, Y_0), evolución temporal del sistema $X(t), Y(t)$, trayectoria $S(t) = (X(t), Y(t))$.
- Modelo depredador – presa de Lotka-Volterra:

https://es.wikipedia.org/wiki/Ecuaciones_Lotka%E2%80%93Volterra

$$\frac{dx}{dt} = x(\alpha - \beta y)$$

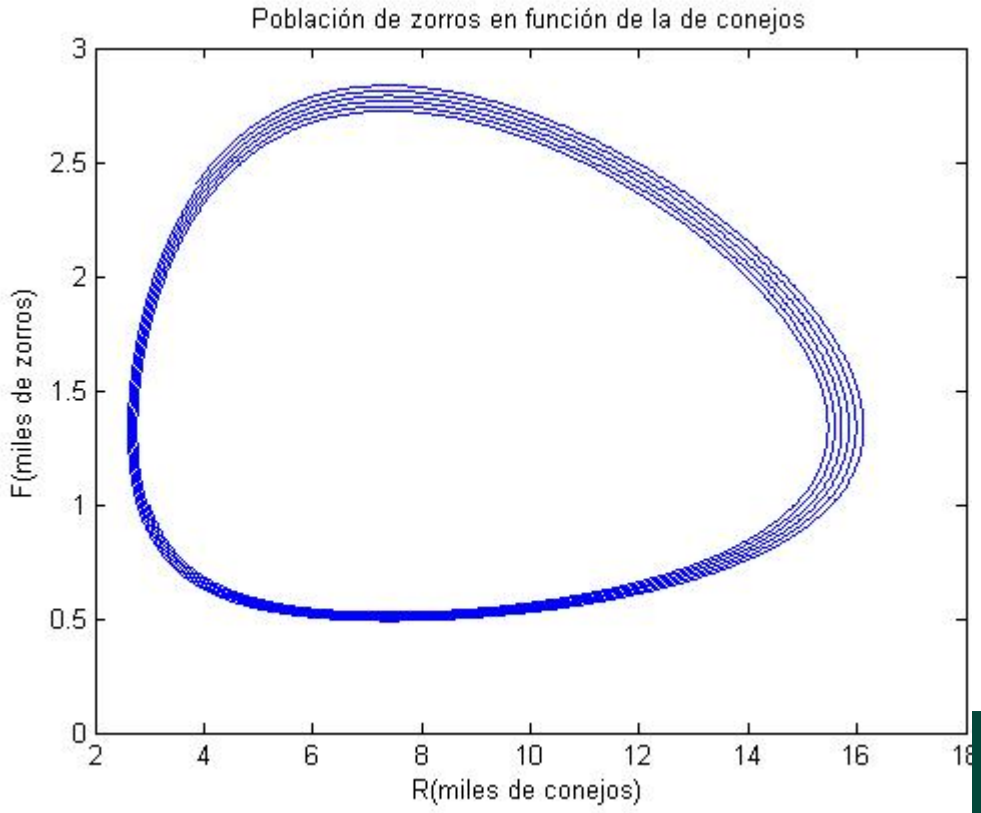
- Teoría cualitativa de ecuaciones diferenciales.

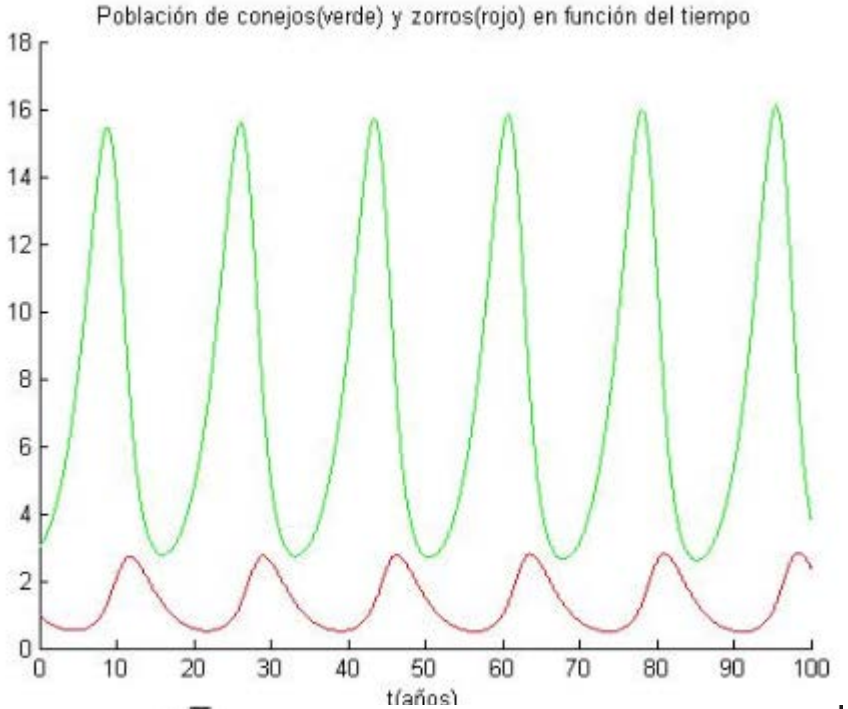
$$\frac{dy}{dt} = -y(\gamma - \delta x)$$

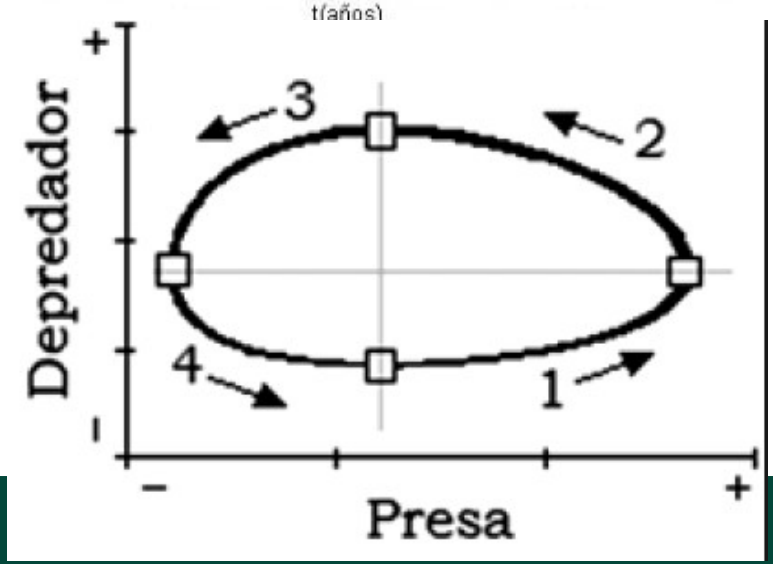
14.1. Introducción

$$\frac{dx}{dt} = x(\alpha - \beta y)$$

$$\frac{dy}{dt} = -y(\gamma - \delta x)$$

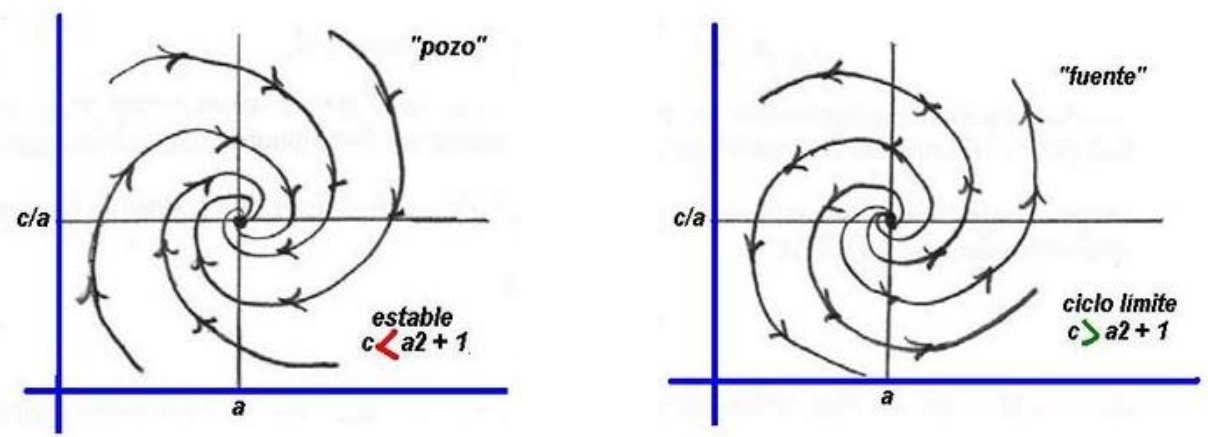




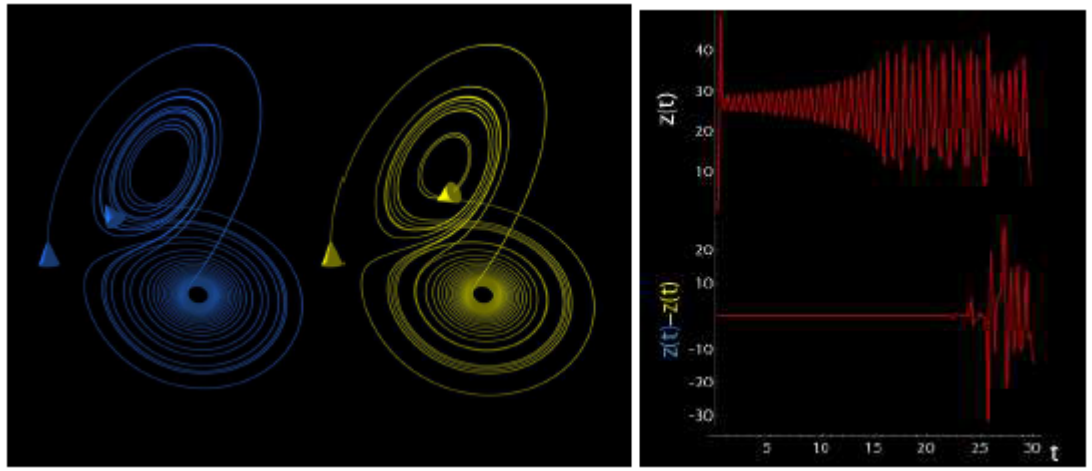


14.1. Introducción

- Comportamiento estable, comportamiento inestable, comportamiento caótico (estructura fractal) → impredecible.



- Sistemas inestables, dependen de la condición inicial, pero predecibles; sistemas caóticos no.
- Atractor de Lorenz (predecir clima), efecto mariposa → sensibilidad de los sistemas caóticos a las condiciones iniciales: “perturbación extremadamente insignificante en las condiciones iniciales, como por ejemplo la variación en la temperatura y presión local producida por el aleteo de una la previsión del tiempo en Texas cambiase de un tiempo Apacible a un violento tornado.



14.2. Sistemas dinámicos continuos y discretos

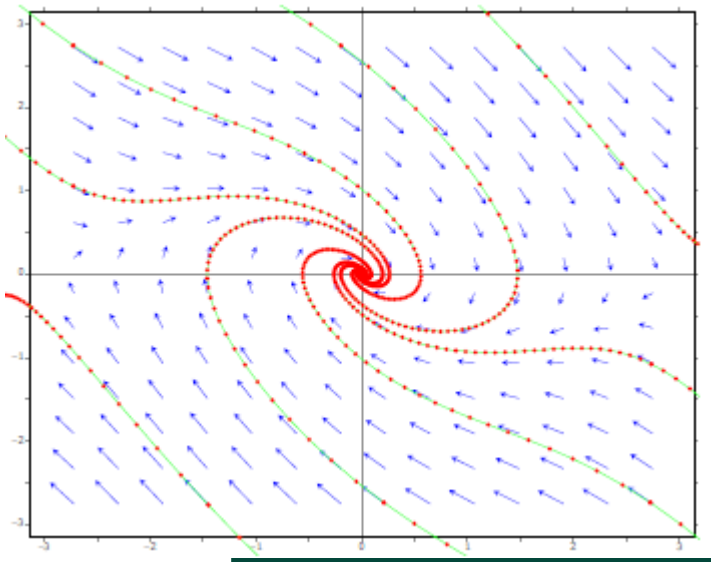
- Regla de evolución del sistema → MRU

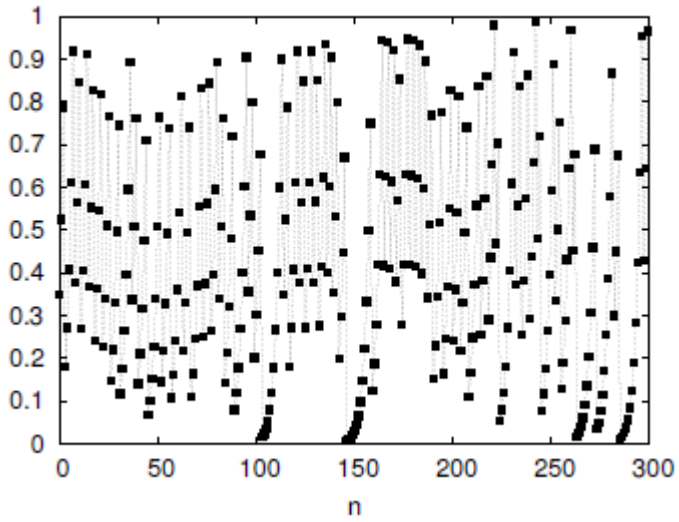
$$x(t) = x_0 + v * t$$

$$\frac{d}{dt}x(t) = v \rightarrow \text{continuo} \rightarrow \text{flujo}$$

$$x(n+1) = x(n) + v\Delta t \rightarrow \text{discreto} \rightarrow \text{aplicación (map)}$$

- Movimiento de un péndulo en un fluido: $\ddot{x} + \beta\dot{x} + \dot{\sin} x = 0$. Representación velocidad vs aceleración.
- Aplicación transformación de intervalos: $x(n+1) = \frac{3}{2}x(n)\mathrm{mod}(1)$





Trayectoria puntos intervalos

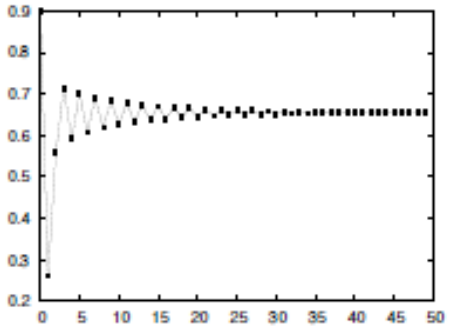
14.2. Sistemas dinámicos continuos y discretos

Descripción cualitativa:

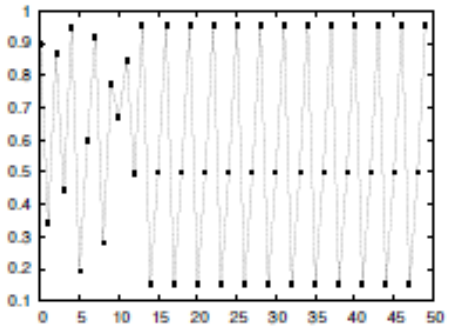
- Estados estacionarios (no cambian en t) → punto fijo.
- Estados que cambian pero vuelven a valor inicial (período) → órbita periódica.
- Estados que se repiten aproximadamente cada cierto intervalo → cuasiperiódica.
- Estados que no se repiten nunca → caos.

- Aplicación logística: $x_{n+1} = \mu x_n(1 - x_n)$, estable, órbita periódica, órbita caótica.

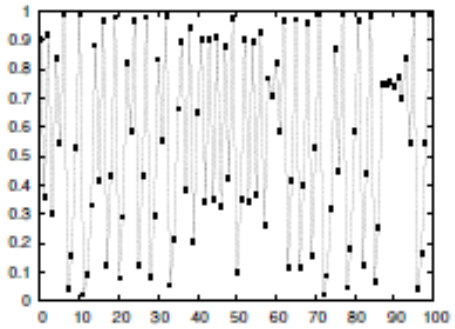
(a) $\mu = 2{,}9$



(b) $\mu = 3{,}831874$

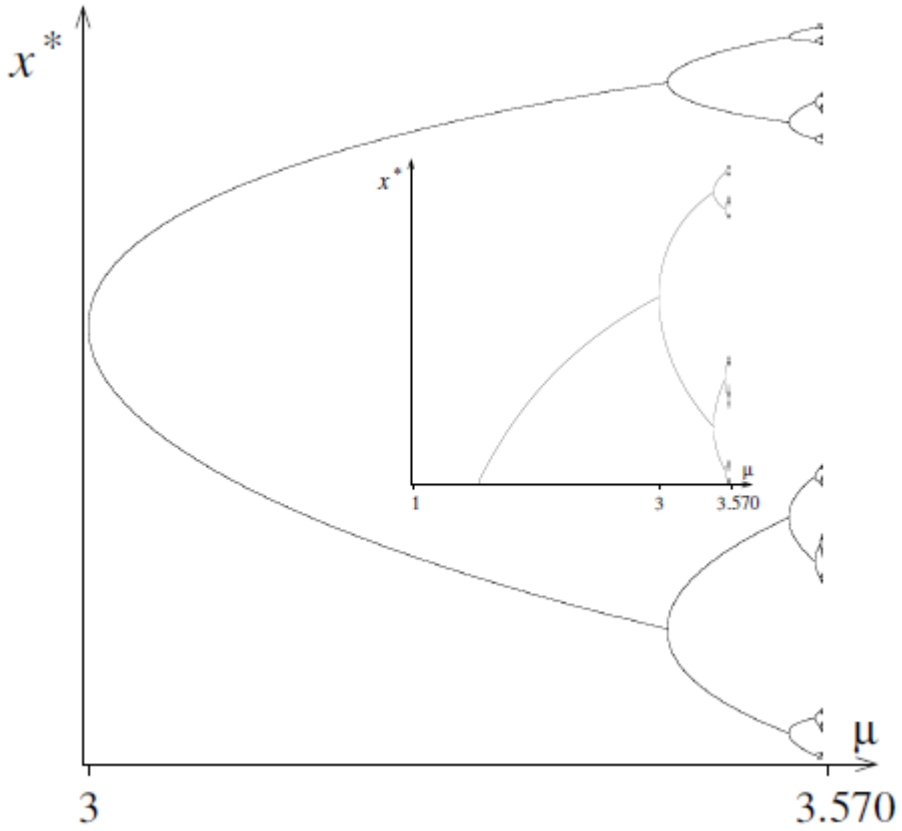


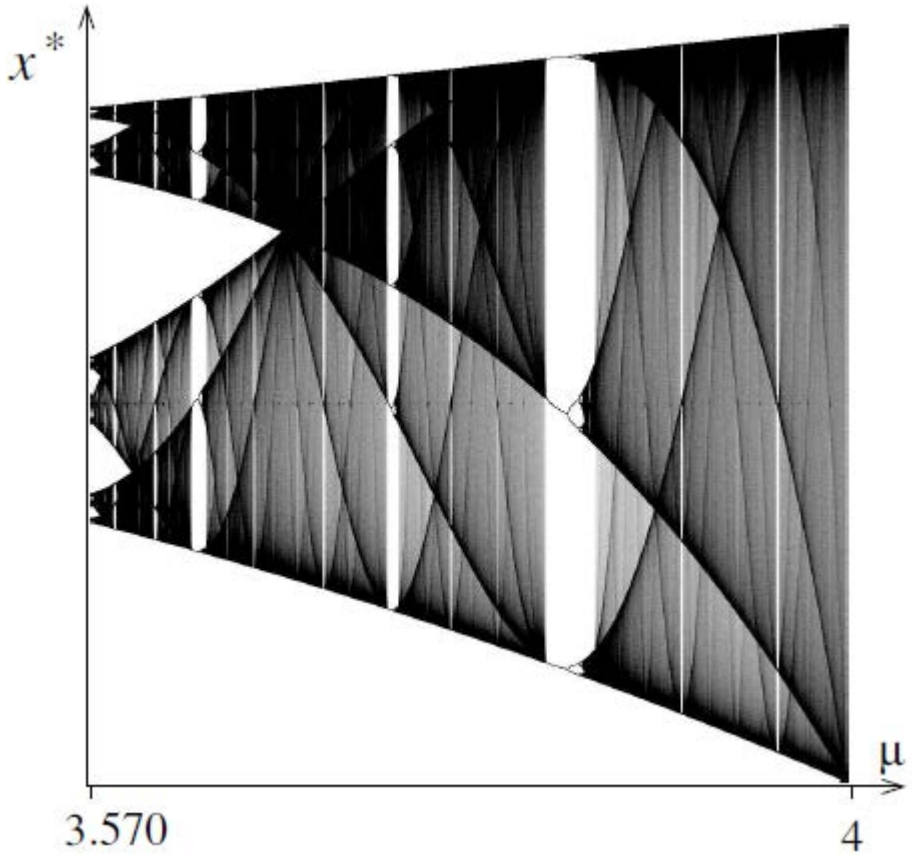
(c) $\mu = 3{,}99$



14.2. Sistemas dinámicos continuos y discretos

- Diagrama de bifurcación: el comportamiento (soluciones estacionarias, estabilidad) depende de μ.
- Cascada de bifurcaciones de Feigenbaum.





14.3. Aplicaciones unidimensionales: Aplicación logística

Función logística. Listado 14.1

- Función logística.
- Número iteraciones.
- Ejecución iteración (si converge), exportación pares ordenados (x_n, x_{n+1}).

Ejercicio 14.1, 14.2.

Período aplicación recursiva. Listado 14.2

- Se define un radio y se comprueba que los puntos nuevos no se acercan a los anteriores.
- Array circular (guarda N valores, y va desplazando el más antiguo por el nuevo) o shift sobre el array.

Ejercicio 14.3, 14.4.

Diagrama de bifurcaciones:

- Sistemas de la misma naturaleza que la logística tienen su mismo diagrama de bifurcaciones. Aplicación de Poincaré del sistema de Lorenz.

Ejercicio 14.5, Listado 14.3.

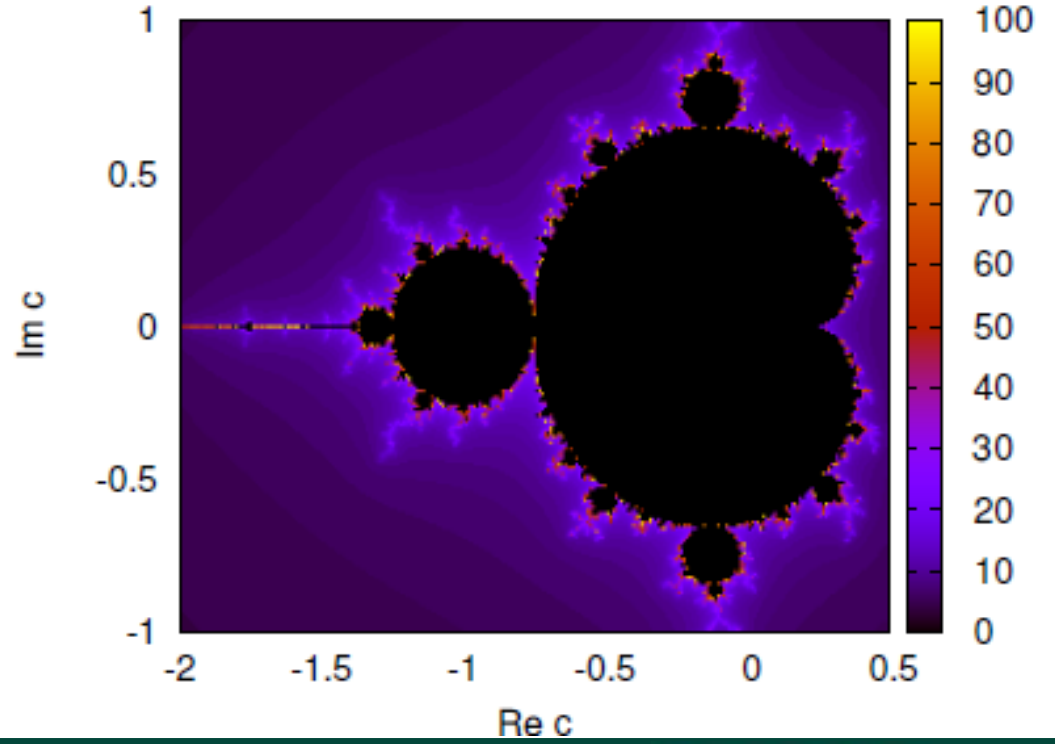
14.4. Aplicaciones bidimensionales: Conjunto de Mandelbrot

Bidemensional → representación de los complejos en dos dimensiones, base $(1, i)$.

$$z_{n+1} = z_n^2 + c$$

- c es el parámetro inicial (μ).
- Órbita estable si está acotada: $|z_n| < R$. Por ejemplo, $R = 20, N = 500$.
- Conjunto de Mandelbrot: conjunto de los puntos que pasan de ser acotados a diverger.

Ejercicio 14.6, 14.7.



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.

Algunas indicaciones foros generales

- Representación:
  https://2019.cursosvirtuales.uned.es/dotlrn/grados/asignaturas/61041094-19/uforums/thread-view?message_id=42164477
- Inputs, gnuplot:
  https://2019.cursosvirtuales.uned.es/dotlrn/grados/asignaturas/61041094-19/uforums/thread-view?message_id=42856494
- Número simulaciones:
  https://2019.cursosvirtuales.uned.es/dotlrn/grados/asignaturas/61041094-19/uforums/thread-view?message_id=42434837
- Aleatoriedad:
  https://2019.cursosvirtuales.uned.es/dotlrn/grados/asignaturas/61041094-19/uforums/thread-view?message_id=42781585
- Aclaraciones:
  https://2019.cursosvirtuales.uned.es/dotlrn/grados/asignaturas/61041094-19/uforums/thread-view?message_id=42673173
  https://2019.cursosvirtuales.uned.es/dotlrn/grados/asignaturas/61041094-19/uforums/thread-view?message_id=42557971

Apéndice: “trucos” C

- **ESTUDIAR EL TEMA 12.**
- Ahorro de memoria → tipo *char* (-127, 128) o tipo *unsigned char* (0, 255).
- Reloj del sistema como semilla.
  - Si se usa una semilla no dada por el reloj, estudiar resultados con una semilla diferente !!!
- Generador de números aleatorios en C, ejemplo semilla, valores en [0,1):

```
srand ( 1234 ) ;
srand(time(NULL));
double rnd=(double ) rand ( ) / ( ( double )RAND_MAX+1) ;
```

Bucles y condicionales

- Pseudocódigo: https://es.wikipedia.org/wiki/Pseudoc%C3%B3digo
- Bucle: repetir una o varias sentencias un número determinado (o no) de veces.
  1. Para i=valor_inicial hasta i=valor_final ejecutar {sentencias}
  2. Para cada i en los elementos de una lista ejecutar {sentencias}
  3. Para i=valor_inicial hasta i=valor_final ejecutar {sentencias} mientras se cumpla condición
  4. Para i=valor_inicial hasta i=valor_final ejecutar {sentencias} hasta que se cumpla condición

- Sentencia condicional: ejecutar una o varias sentencias dependiendo si se cumple o no una condición.
  1. Si {condición lógica} entonces {sentencias}
  2. Si {condición lógica} entonces {sentencias}, si no {sentencias}

- Combinaciones de sentencias condicionales y bucles.

11.3. Control de flujo: “if. . . else”

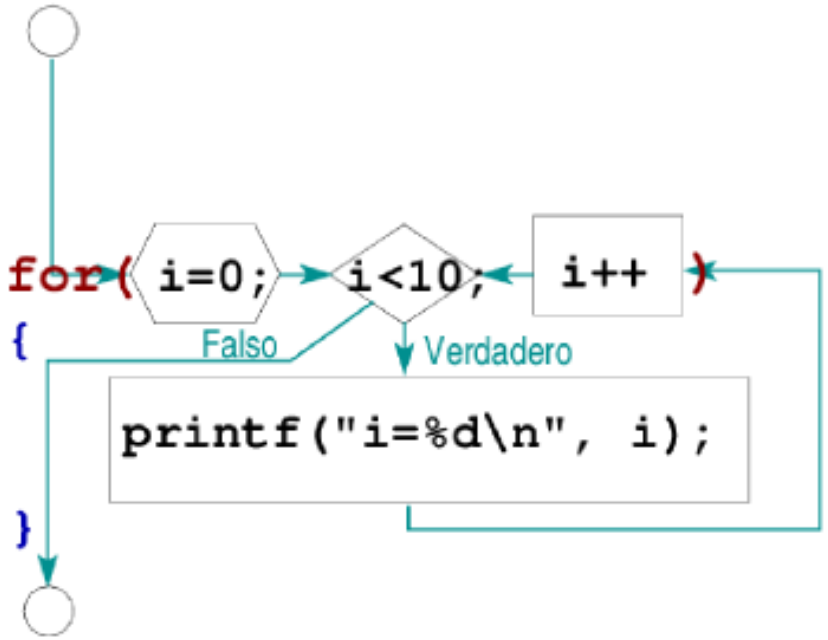
```
If (condición) {
        sentencia1;
} else {sentencia2}
```

Operadores de comparación: <, >, >=, <=, ==, !=

Operadores lógicos: ||, &&, ^, !

Ejemplo: Listado 11.4

11.4. Bucle “for”

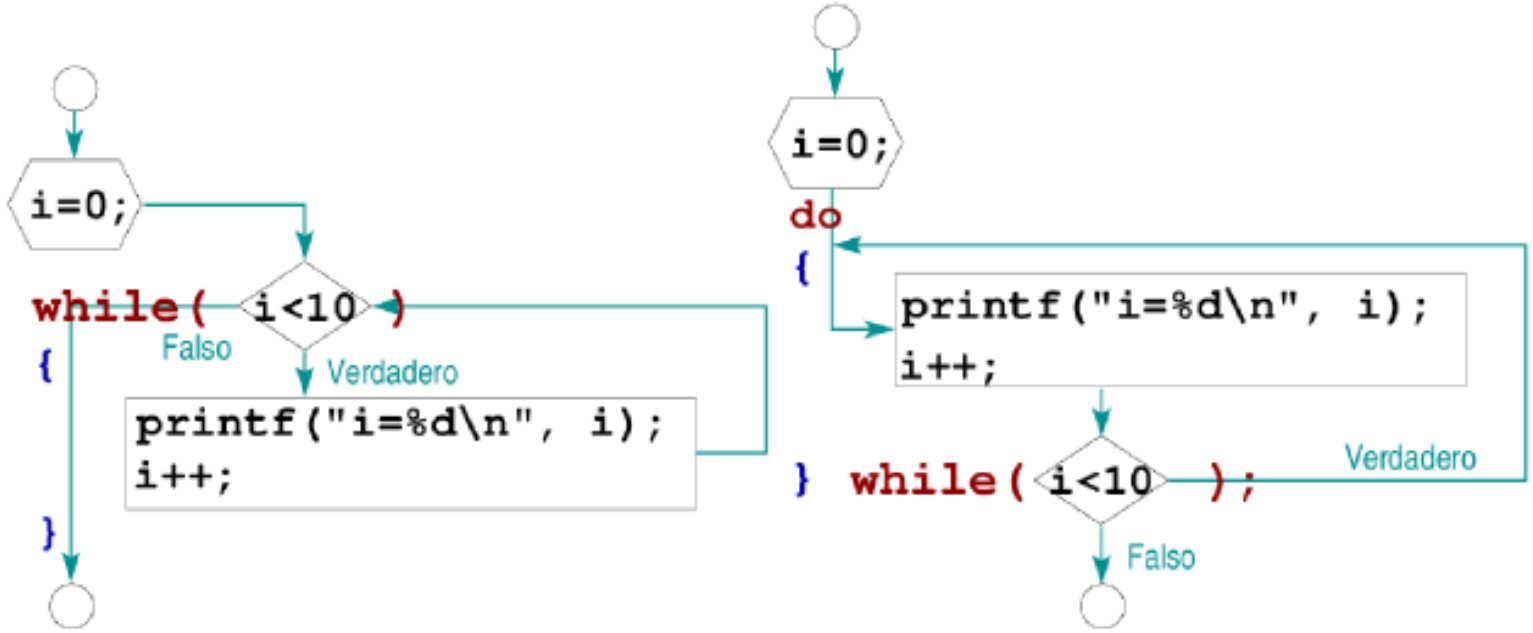


i++ → suma de 1 en 1

i+=m → suma de m en m

Ejemplo: Listado 11.5

11.5. Bucles “while” y “do…while”



do…while ejecuta al menos una vez

Ejemplo: Ejercico 11.9, Ejercicio 11.10

11.6. Selección con “switch…case…default”

```
if ( x == 1 ) {
      ...
} else if ( x==2 ) {
      ...
} else if ( x==3 ) {
      ...
} else {
      ...
}
```

```
switch( x ) {
case 1:
      ...
      break;
case 2:
      ...
      break;
case 3:
      ...
      break;
default:
      ...
}
```

En lugar de anidar else-if, se crean tablas de llaves-valores;

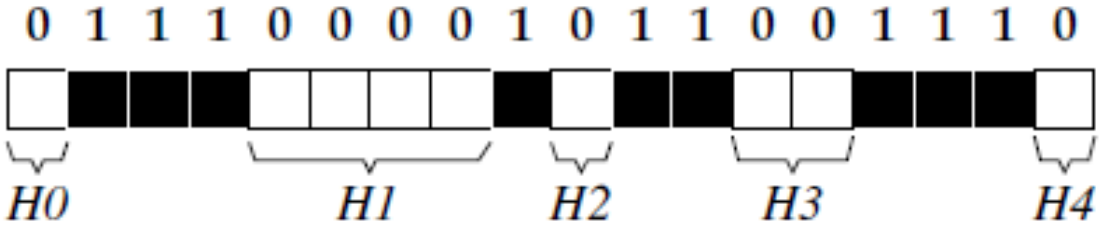
default → cuando ocurra un valor sin llave.

break → sirve para salir de la estructura y que no la siga recorriendo a ver si se cumple la condición.

Ejemplo: Listado 11.6

Prueba C – Ejercicio 1

- 0 no ocupado, 1 ocupado.
- Disposición aleatoria en un array.
- Porosidad = número 0’s / número 1’s
- ¿Qué se pide? Poroso longitud L y porosidad r dados por el usuario. Guardar en archivo texto y representar con gnuplot. Utilizar simulaciones con L = 250, 1250; r = 0.25, 1, 4. Es decir, tendríamos 6 simulaciones (L,r) = {(250, 0.25), (250, 1), (250, 4), (1250, 0.25), (1250, 1), (1250, 4)}
- ¿Qué quiere decir r=0.25, r=1, r=4?
- Resultado en una tabla de 2x3 gráficas.
- ¿Corte del aleatorio generado en > ó en >=?
- **PROBABILIDAD DE QUE UN SITIO ESTÉ OCUPADO:** $p = \frac{1}{(1+r)} \rightarrow p = ¿?$



Prueba C – Ejercicio 1

Javier Carrasco Serrano

UNED

Prueba C – Ejercicio 2

- Longitud de un hueco: número de 0's consecutivos.
- ¿Cómo calcular el número de 0's consecutivos?
- Histograma huecos: número de huecos de tamaño 1, de tamaño 2, … , de tamaño L.
- Histograma: Listado 12.3.
- Con el histograma se puede saber/estimar la probabilidad de que un 0 dado pertenezca a un hueco de longitud l. ¿Cómo? ¿Salida del histograma en número o en porcentaje?
- ¿Qué se pide? Calcular el histograma de la longitud de los huecos de tamaño 1, 2, … , L/2, dados L, r y N (número de cadenas a simular). Utilizar N = 1024, L = 64, 1024, r = 0.25, 1, 4.
- Distribución del histograma ("en el infinito"):

$$h(l,r) = r^l/(1+r)^{2+l}$$

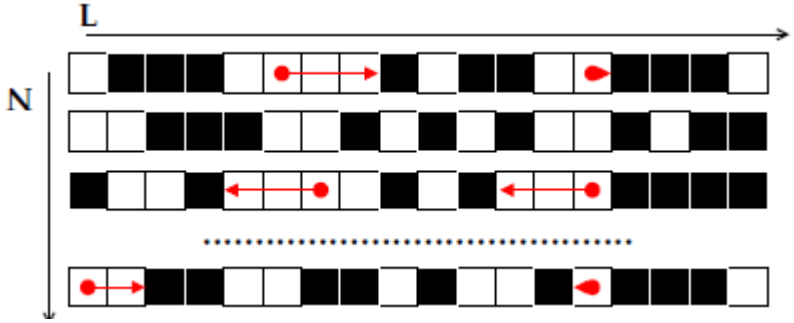
- ¿Qué sucede para los distintos tamaños (L)?

Prueba C – Ejercicio 2

Javier Carrasco Serrano

UNED

Prueba C – Ejercicio 3



- Valores medios → media.
- Distancia que se puede recorrer en un 0 elegido al azar (hacia un lado o hacia el otro) hasta chocar con un 1.
- ¿Qué información nos da el histograma que podamos utilizar?
  - Número de cadenas de cada tamaño.
  - Tamaño de cada una de las cadenas → ¿qué sitios puede ocupar el 0?, ¿cuánto recorre?
- ¿Y el histograma teórico?
- Distancia media ≡ longitud de correlación
- ¿Qué se pide? ¿Ser un bruto? Es una posibilidad. ¿Es la única? ¿Es la más eficiente?
- Se pide recorrer cada cadena, y cada vez que se encuentre un 0, contar los 0s que hay a su derecha y a su izquierda.
- Hacerlo para L = 1024, N = 1024, r = 0.1, 0.25, 0.5, 1, 2, 4, 9 → ¿p?
- Representar longitud de correlación (distancia media) en función de r.
- ¿Depende de L? → probar con L = 32 → ¿qué quiere decir?

Prueba C – Ejercicio 3

Javier Carrasco Serrano

UNED

Prueba C – Ejercicio 4

- Distancias medias mediante Monte-Carlo.
- Hacerlo para L = 1024, N = 1024, r = 0.1, 0.25, 0.5, 1, 2, 4, 9 → ¿p?
- Elegir M=N/16 de los 0s de cada cadena (al azar). Elegir el sentido (izquierda, derecha) al azar. ¿Cómo? Calcular la distancia hasta el primer 1. Hacer la media de las distancias calculadas y comparar con el ejercicio anterior. ¿Conclusiones? ¿Es Monte-Carlo robusto? ¿Merece la pena (recursos) una búsqueda exhaustiva?

Prueba C – Ejercicio 4

Javier Carrasco Serrano

UNED

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