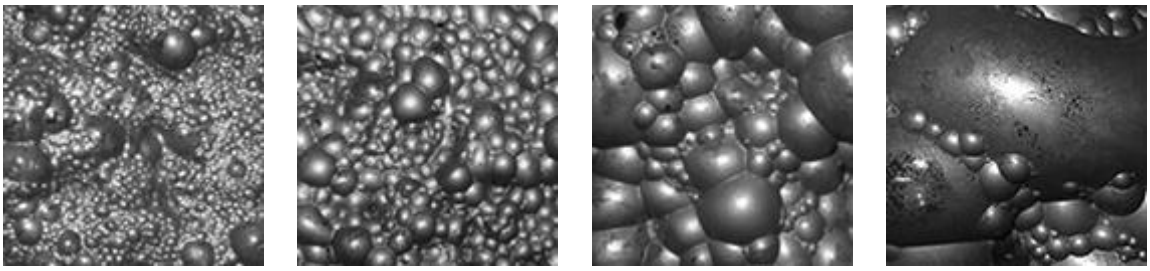
Taller introductorio a Matlab

Ejercicio 1. Las siguientes imágenes muestran distintos estados de un proceso de flotación para un mineral.



(a) (b) (c) (d)

Figura 1. Imágenes de un proceso de flotación de minerales.

Se adjunta un archivo `froth.mat` con las imágenes en forma de matrices de 130x130. Se pide calcular los siguientes valores para cada imagen y comparar los resultados: i) promedio, ii) mediana, iii) moda, iv) desviación estándar y la v) entropía, definida a continuación:

$$H(I) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Para una imagen I, p_i es la probabilidad de ocurrencia del valor de intensidad de un pixel, es decir, sería igual al número de pixeles con ese valor dividido en el total de pixeles de la imagen, y n es el número de niveles de intensidad de la imagen, para una imagen de 8 bits el número total es de $2^8 = 256$ niveles (en Matlab van de 0 a 255). Compare su resultado con la función `entropy()` de Matlab. ¿Qué método(s) usaría para una comparación efectiva entre las imágenes?

Ejercicio 2. Un número natural es un palíndromo si lee igual de izquierda a derecha y de derecha a izquierda. Por ejemplo, 35853 es un palíndromo, mientras que 12345 no lo es. Escriba un programa que indique si el número ingresado es o no palíndromo.

Ejercicio 3. Tres de los cuatro lados de una placa rectangular (a = 5 m, b = 4 m) se mantienen a una temperatura T = 0 °C, y el lado restante a una temperatura T1 = 80 °C, tal y como se muestra en la Fig. 2. Calcular y representar la distribución de la temperatura $T(x,y)$ en la placa.

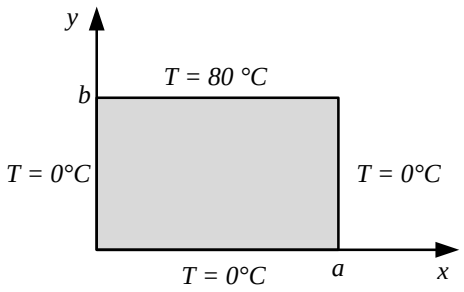


Figura 2. Imágenes de un proceso de flotación de minerales.

Nota: La distribución de la temperatura, $T(x,y)$, en la placa se puede calcular resolviendo la ecuación del calor en dos dimensiones. Para las condiciones de frontera dadas en el problema, $T(x,y)$ se puede expresar de forma analítica mediante la siguiente serie de Fourier:

$$T(x,y)=\frac{4T_1}{\pi}\sum_{n=1}^{\infty}\frac{sen\left[(2n-1)\cdot\frac{\pi x}{a}\right]}{(2n-1)}\frac{senh\left[(2n-1)\cdot\frac{\pi y}{a}\right]}{senh\left[(2n-1)\cdot\frac{\pi b}{a}\right]}$$

Considere k = 5 y k = 50 términos de la serie para el cálculo y compare los resultados.

Ejercicio 4. La dependencia térmica de las reacciones químicas puede ser calculada con la ecuación de Arrhenius:

$$k=Ae^{-E/(RT_a)}$$

donde k es la tasa de reacción en s^{-1}, A es un factor pre-exponencial (o frecuencia), E es la energía de activación en J/mol, R es la constante de los gases [8.314 J/(mol·K)], y T_a es la temperatura absoluta en K. Un compuesto tiene una $E = 1\times10^5$ J/mol y $A = 7\times10^{16}$. Usar Matlab para generar valores de tasas de reacción para temperaturas en el rango de 253K a 325K. Usar `subplot` para graficar lado a lado (a) k vs T_a (línea verde) y (b) $\log_{10}(k)$ (línea roja) versus $1/T_a$. Emplear la función `semilogy` para crear (b). Incluir las etiquetas de los ejes correspondientes en ambas sub-figuras. Interpretar los resultados.

Ejercicio 5. El juego del caos. Asumamos que tenemos tres puntos en los vértices de un triángulo equilátero, P_1 = (0, 0), P_2 = (0.5, $\sqrt{2}/2$), P_3 = (1, 0). Generar otro conjunto de puntos $p_i = (x_i, y_i)$ tal que p_1 = (0, 0) y p_{i+1} es el punto medio entre p_i y P_1 con un 33% de probabilidad, o el punto medio entre p_i y P_2 con un 33% de probabilidad, o el punto medio entre p_i y P_3 con un 33% de probabilidad. Escribir un programa que genere los puntos pi para i = 1, …, n. Por ejemplo, se ilustran gráficos generados para n = 100 y n = 10000. (Tip: la función `rand` puede ayudar)

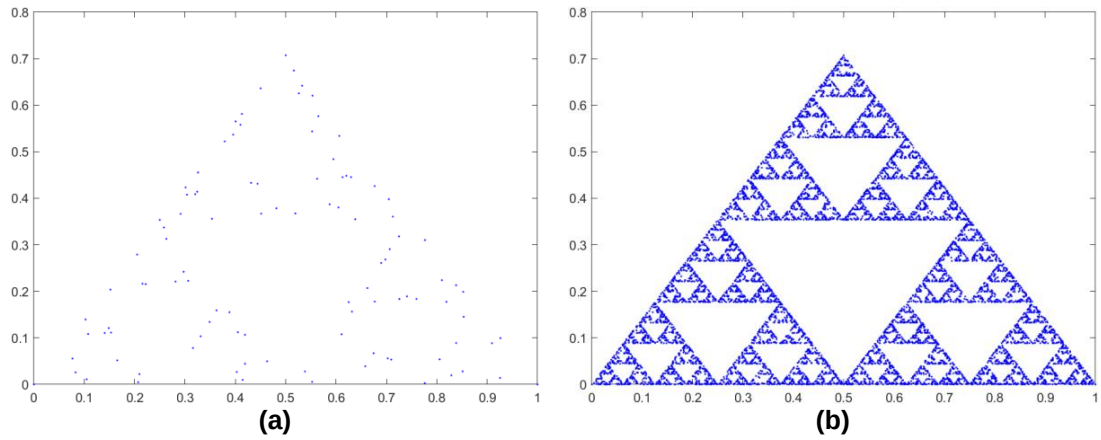


Figura 3. Triángulo de Sierpinski generado con puntos aleatorios para a) n = 100 puntos, b) n = 10000.

Nota: Más sobre el triángulo de Sierpinski en wiki.

Ejercicio 6. La expansión en series de Taylor de la función coseno se puede escribir de la siguiente forma:

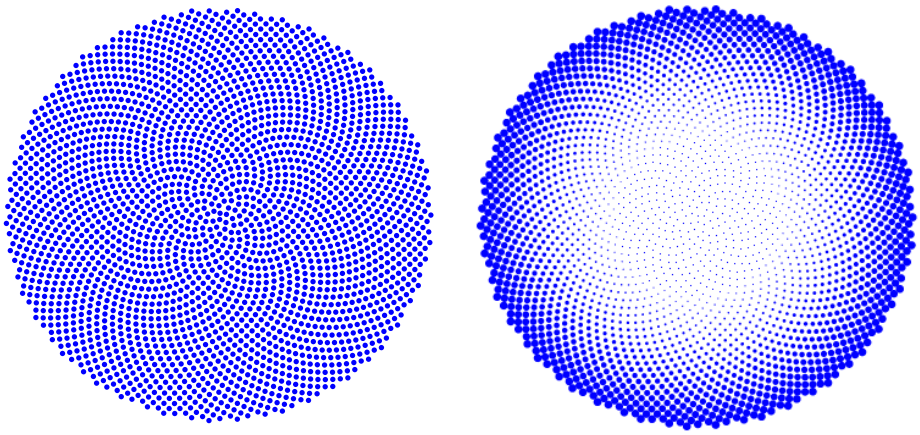
$$\cos(x) \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - .... = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}, \ \forall x, \forall n \in \mathbb{Z}_0^+$$

Usar Matlab para crear una función que aproxime cos(x) a un número N de términos, x puede ser un escalar o vector. La función debe llamarse `cos_Taylor()`. Usar la función factorial desarrollada en clases para calcular la serie. Para testear la función, crear un vector x de 100 puntos en el rango de 0 a $3\pi/2$ (usar función `linspace()`) y guardar en una matriz los vectores resultantes, para N = 10,50 y 100. Graficar en la misma figura las tres aproximaciones de cos(x) para cada valor de N (usar líneas discontinuas de diferente color) y la función `cos()` de Matlab (usar línea de color negro continua) para el mismo vector x. Etiquetar los ejes y gráfico de acuerdo a los datos. El rango del eje y debe estar entre -1 y 1, y los valores del eje x deben ser etiquetados como 0, $\pi/2$, π y $3\pi/2$ (usar apropiadamente los comandos `xticks` y `xticklabels`).

Ejercicio 7. Misceláneo. El siguiente código genera la imagen de una flor de maravilla como se muestra en la Figura 3(a). Modificar el código para obtener la representación mostrada en la Figura 3(b).

Código:

```
phi          = (sqrt(5)+1)/2;% Número áureo
golden_angle = 2*pi/phi;
max_angle    = 10000;
theta        = 1:golden_angle:max_angle;% ángulo
r            = sqrt(theta);% radio
[x,y]        = pol2cart(theta,r);% coordenadas polares a cartesianas
figure
plot(x,y,'.','MarkerSize',10);axis off;
```



(a) **(b)**

Figura 4. Flor de maravilla.

Ejercicio 8. Lanzamiento de una moneda. Escribir un script `coinTest.m` para simular el experimento de lanzar secuencialmente una moneda 5000 veces. Mantener un registro de cada vez que se obtenga 'cara' y graficar la estimación de la probabilidad de obtener cara con esta moneda de forma acumulada. Graficar esta frecuencia estadística acumulada junto con una línea horizontal del valor esperado de 0.5. Esto se puede hacer fácilmente sin un ciclo (funciones útiles: `rand`, `round` y `cumsum`).

Ejercicio 9. La *curva de la mariposa* está dada por las siguientes ecuaciones paramétricas:

$$x = sin(\mathrm{t})\left(e^{\cos(t)} - 2\cos(4t) - \sin^5(\frac{t}{12})\right)$$
$$y = cos(\mathrm{t})\left(e^{\cos(t)} - 2\cos(4t) - \sin^5(\frac{t}{12})\right)$$

Generar los valores de x e y para valores de t desde 0 a 100 con Δt = 1/16. Construir los gráficos de (a) x e y versus t y (b) x vs y. Usar `subplot` para apilar estos dos gráficos en forma vertical. Incluir títulos y etiquetas a los ejes sobre ambos gráficos y leyendas para (a). Para (a) emplear una línea discontinua y otro color para y para distinguirla de x.

Ejercicio 10. Graficar las siguientes ecuaciones paramétricas, experimentar con los rangos de valores para las variables.

a) $x(t) = 2\cos(t) + \sin(2t)\cos(60t),\ y(t) = \sin(2t) + \sin(60t)$
b) $x(t) = t - 1.6\cos(24t),\ y(t) = t - 1.6\sin(25t)$
c) $x(t) = 16\sin^3(t),\ y(t) = 13\cos(t) - 5\cos(2t) - 2\cos(3t) - \cos(4t)$
d) Integral seno versus integral coseno de Fresnel, S(t) vs C(t), usar las funciones `fresnels` y `fresnelc` de Matlab para evaluarlas, la curva resultante se conoce como la espiral del Cornu.

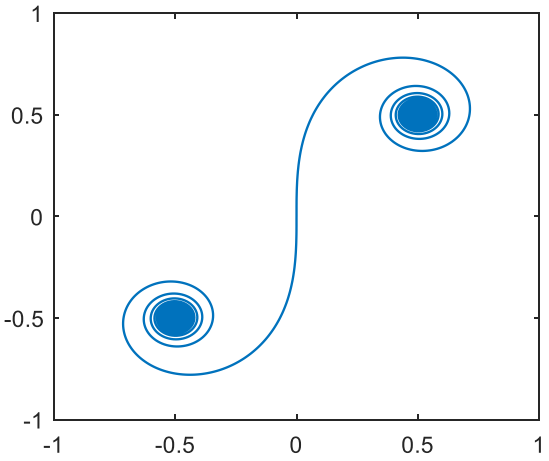


Figura 5. Espiral de Cornu.

Ejercicio 11. Datos con información espacial. En el archivo `llama.mat` se entrega un arreglo con datos de temperatura espacial de una llama de vela, se pide graficar un mapa de intensidad 2D de los datos junto con una barra de color (`colorbar`) con escala adecuada para visualizar de mejor forma los valores de temperatura. Finalmente, se pide graficar el histograma de los datos de temperatura de la llama (solo valores > 0). La siguiente figura muestra un ejemplo de lo que se pide.

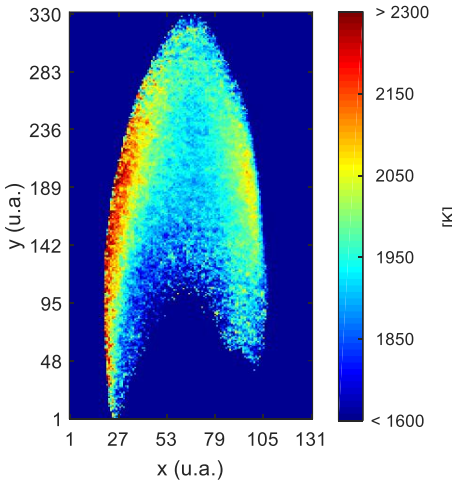


Figura 6. Mapa de temperatura de una llama.

Ejercicio 12. Análisis de datos con Matlab: Estimación de temperatura de la llama de combustión de minerales de interés en la industria del cobre. Es sabido que la temperatura de un proceso de combustión se puede estimar a partir de la medición de la intensidad de radiación visible emitida por la llama producida. Dentro de los métodos existentes para este fin, el de dos longitudes de onda es el más usado, para aplicarlo, es necesario medir dos muestras de la intensidad de la radiación, I_1 e I_2, en dos longitudes de onda particulares, λ_1 y λ_2 (**en metros**), para luego aplicar la siguiente ecuación y así estimar la temperatura (en *K*):

$$T = \frac{-c_2\left(\frac{1}{\lambda_1}-\frac{1}{\lambda_2}\right)}{\ln\left(\frac{I_1}{I_2}\right)+5\cdot\ln\left(\frac{\lambda_1}{\lambda_2}\right)}(K)$$

donde $c_2 = 1.438\cdot10^{-2}$(m·K) (segunda constante de Planck), la expresión anterior nace de la *Ley de Radiación de Planck*. En este problema se entrega un archivo `Intensidades.mat` con 400 muestras de pares de valores de intensidad de radiación medidos desde un proceso de combustión flash de Pirita y Calcopirita, 400 muestras para c/u en los vectores `Ipy` y `Icpy` (fila 1 corresponde a I_1, fila 2 corresponde a I_2). Se pide:

1. Si las intensidades I_1 e I_2 fueron medidas en $\boldsymbol{\lambda_1 = 650nm}$ y $\boldsymbol{\lambda_2 = 750nm}$ respectivamente, estimar para cada especie la temperatura de llama generada y guardar los valores en un arreglo `Tpy` y `Tcpy`. La **temperatura debe estar en grados Celsius**.
2. Calcular el valor de temperatura media μ_T, mediana med_T y desviación estándar σ_T para cada vector de temperatura estimado, tabular los valores calculados y comentar los

resultados. Programar sus propias versiones de las funciones para calcular los estadísticos mencionados.

3. Graficar los valores de temperatura vs el número de muestra (1 hasta 400). Etiquetar y ajustar adecuadamente los ejes. Los valores de temperatura mayores a $\mu_T + \sigma_T$ o menores a $\mu_T - \sigma_T$ graficarlos en rojo, azul en caso contrario. Además, graficar en las figuras las rectas y = μ_T, y = $\mu_T - \sigma_T$ e y = $\mu_T + \sigma_T$ como líneas negras discontinuas, en el mismo rango. Todo esto para cada especie por separado en distintas figuras.
4. Dentro de las técnicas de visualización estadística más usadas para resumir datos se encuentran los gráficos de caja o *boxplots*. Se pide realizar estos gráficos para los dos vectores de temperatura calculados, `Tpy` y `Tcpy`, mostrar los 2 gráficos correspondientes en la misma figura y comentar los resultados. ¿Qué información relevante acerca de los datos entregan estos gráficos?

Nota: En el paper: **Flash Smelting Copper Concentrates Spectral Emission Measurements**, *Sensors* **2018**, *18*(7). DOI: doi.org/10.3390/s18072009 pueden encontrar información extra para entender el contexto del problema.

Ejercicio 13. Animación Movimiento Browniano. Escribir una función con la siguiente declaración: `brown2D(N)`. La función toma como único argumento el valor `N`, el cual es un entero que especifica el número de partículas o puntos a simular. Todos los puntos debieran iniciar su movimiento en el origen (0,0). Graficar todos los puntos en una figura usando como marcador '**.**', y fijar los ejes a una región rectangular con límites desde -1 a 1 en ambos ejes. Para simular el movimiento Browniano de los puntos, escribir un ciclo de 1000 iteraciones el cual calculará una nueva posición de *x* e *y* para cada punto y desplegará estas nuevas posiciones como una animación (la sentencia `drawnow` puede ser de ayuda). La nueva posición de cada punto se obtiene al sumar una variable normalmente distribuida con una desviación estándar de 0.005 a cada valor de *x* e *y* (usar `randn`; si por ejemplo tienes 10 puntos, necesitarás sumar 10 valores aleatorios distintivos a los valores de *x* y 10 valores aleatorios distintivos a los valores de *y*). Cada vez que las nuevas posiciones de todos los puntos son calculadas, graficarlos sobre la figura.

Lo que verás es una simulación del fenómeno de difusión, donde las partículas se mueven aleatoriamente lejos del centro de la figura. Por ejemplo, las siguientes figuras muestran la simulación al inicio, mitad y final para 100 puntos:

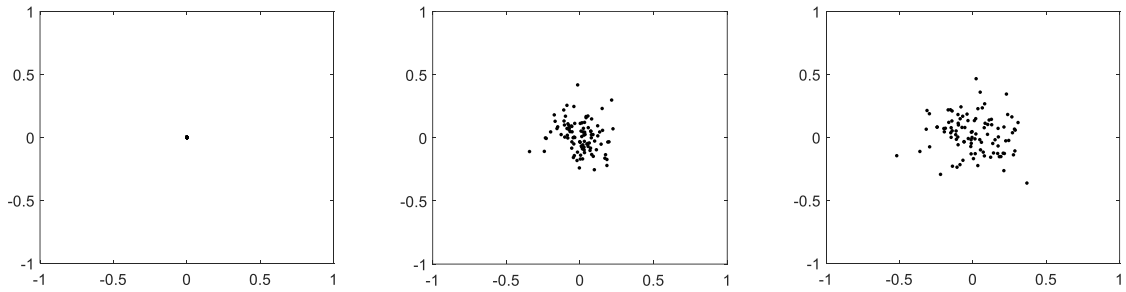


Figura 7. Simulación de movimiento Browniano.

Problemas opcionales

P.O.1. Divisores serie de Fibonacci. La serie de Fibonacci se construye utilizando la siguiente relación de recurrencia:

$$F_n = F_{n_1} + F_{n_2},$$

donde $F_1 = 1$ y $F_2 = 2$. Por ende, los primeros doce términos de esta serie son: 1, 2, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144. Ahora considere los divisores de estos términos:

n	F_n	**Divisores**
1	1	1
2	2	1, 2
3	2	1, 2
4	3	1, 3
5	5	1, 5
6	8	1, 2, 4, 8
7	13	1, 13
8	21	1, 3, 7, 21
9	34	1, 2, 17, 34
10	55	1, 5, 11, 55
11	89	1, 89
12	144	1, 2, 3, 4, 6, 8, 9, 12, 16, 18, 24, 36, 48, 72, 144

Como se puede ver, 144 es el primer número de la serie de Fibonnacci que tiene más de 10 divisores (de hecho, tiene 15). Se pide encontrar el primer número de Fibonacci F_n que tiene más de 1000 divisores.

P.O.2. Dígito Verificador. El algoritmo para obtener el dígito verificador del Rol Único Nacional (RUN) o el Rol Único Tributario (RUT) chilenos se conoce como Módulo 11. Para calcular el dígito verificador con este método, debemos seguir los siguientes pasos:

1. Obtener los dígitos del RUT (sin dígito verificador).
2. Invertir el número, ej. 18232155 a 5512381.
3. Multiplicar los dígitos por la secuencia 2, 3, 4, 5, 6, 7 y sumar el resultado, si quedan dígitos por multiplicar, se comienza la serie nuevamente. Para el ejemplo en 2, quedaría:

$$5\times2+5\times3+1\times4+2\times5+3\times6+8\times7+1\times2=115$$

4. Al resultado obtenido, en este caso 115, debemos sacarle el módulo 11, es decir, se divide por 11 y se determina el resto de la división, en Matlab puede usar la función `mod()`, nos quedaría el valor de **5**.
5. Con el resultado del paso anterior, debemos restarlo de 11: 11 - 5 = **6**.
6. Finalmente, si el resultado del paso 5): i) es **11**, el dígito verificador será **0**, ii) es **10**, el dígito verificador será K, en cualquier otro caso, el resultado será el propio digito verificador, en el ejemplo, el dígito es **6**.

P.O.3. *Pi Day*. El pasado 14 de marzo (3/14) se 'celebró' el día del número π, en relación con esto, ejecutar el script `Piday.m` para visualizar los dígitos de este número. (Este script fue propuesto por desarrolladores de la comunidad de Matlab y se dejan los créditos respectivos en el archivo). La gráfica de más abajo muestra la visualización generada. Modificar el script y realizar visualizaciones de otros números irracionales, ej. el número *e*, raíces de números primos, el número de oro, etc., experimentar también con la paleta de colores.

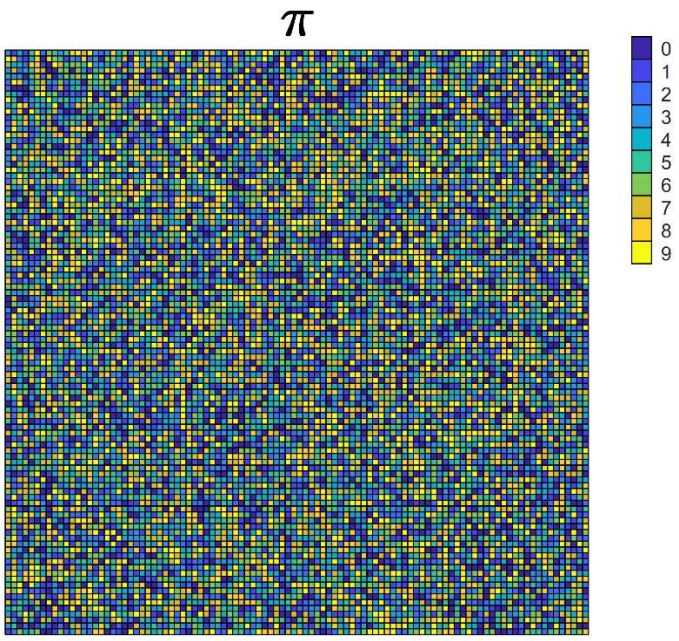


Figura 8. Visualización dígitos número pi.

Ejemplos Resueltos

Ejercicio R1. El juego del caos. Asumamos que tenemos tres puntos en los vértices de un triángulo equilátero, $P_1 = (0, 0)$, $P_2 = (0.5, \sqrt{2}/2)$, $P_3 = (1, 0)$. Generar otro conjunto de puntos $p_i = (x_i, y_i)$ tal que $p_1 = (0, 0)$ y p_{i+1} es el punto medio entre p_i y P_1 con un 33% de probabilidad, o el punto medio entre p_i y P_2 con un 33% de probabilidad, o el punto medio entre p_i y P_3 con un 33% de probabilidad. Escribir un programa que genere los puntos p_i para $i = 1, \ldots, n$. Por ejemplo, se ilustran gráficos generados para $n = 100$ y $n = 10000$. (Tip: la función `rand` puede ayudar)

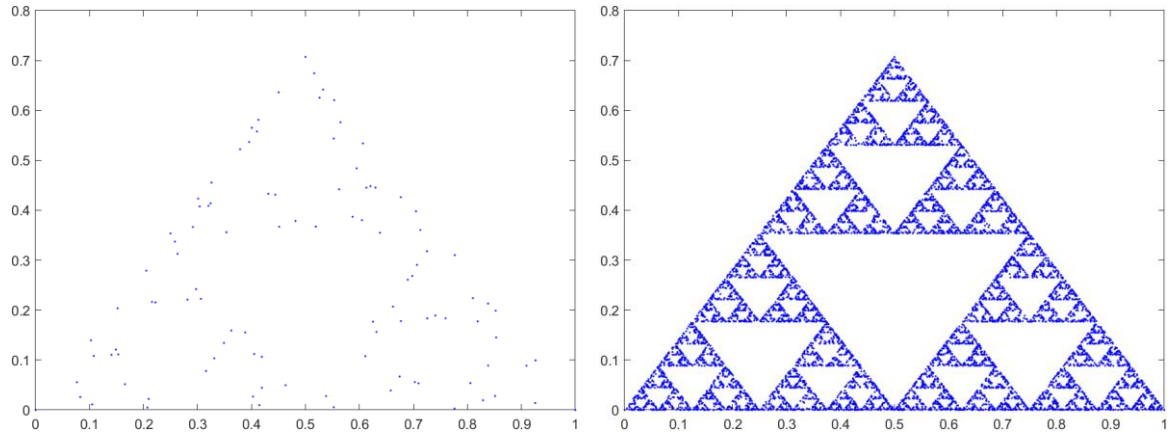


Figura 1. Triángulo de Sierpinski generado con puntos aleatorios para a) $n = 100$ puntos, b) $n = 10000$.

Nota: Más sobre el triángulo de Sierpinski en wiki.

Solución:

Este problema involucra conceptos de probabilidad y geometría. Para programar esta visualización deberemos seguir los siguientes pasos:

- Declarar los puntos iniciales.
- Crear un ciclo donde se vayan generando los siguientes puntos.
- Los puntos se generan con distancias medias entre el punto de la iteración anterior (o inicial p_1) y alguno de los que definen el triángulo equilátero inicial, ¿cuál? Elegir según las probabilidades dadas. Aquí será necesario el uso de condicionales `if-else` para realizar las comparaciones.
- Finalmente, podemos desplegar cada punto en cada iteración, o almacenarlos en un arreglo y desplegarlos al terminar el ciclo, lo primero tomará más tiempo, pero podremos seguir la actualización del fractal, esto es lo que haremos en este ejemplo.

Veamos la solución:

```
clear; clc; close

% El juego del caos, el triángulo de Sierpinski
clear; clc; close all;

% Primeros puntos que definen el triángulo equilátero
Px      = [0 0.5 1]; % Coordenadas x
Py      = [0 sqrt(2)/2 0]; % Coordenadas y

figure
plot(Px,Py,'.b','MarkerSize',3);
axis off; % borramos los ejes para visualizar solamente el fractal
set(gcf,'color','w'); % cambiamos el color de fondo de la figura actual

Npoints = 10000; % número de puntos a dibujar
p       = [0 0]; % punto inicial pi = (xi,yi)
hold on % escribir este comando acá, hará que toda gráfica dentro del
        % ciclo se guarde en la misma figura

for k = 1:Npoints
    prob  = rand;% números aleatorios entre 0 y 1 con distrib. uniforme

    % Actualización de los puntos según las probabilidades dadas
    if prob < 0.33
       p(1) = (Px(1)+p(1))/2;
       p(2) = (Py(1)+p(2))/2;

    elseif prob < 0.66
       p(1) = (Px(2)+p(1))/2;
       p(2) = (Py(2)+p(2))/2;

    else
       p(1) = (Px(3)+p(1))/2;
       p(2) = (Py(3)+p(2))/2;

    end

    plot(p(1),p(2),'.b','MarkerSize',3);
    drawnow
end
```