Instituto Politécnico Nacional
Escuela Superior de Cómputo
Materia: GENETIC ALGORITHMS

Lab Session 1. Introduction to python

Alumno: Torres Abonce Luis Miguel
Profesor: Jorge Luis Rosas Trigueros
Fecha de Realización: 09 Septiembre 2024
Fecha de Entrega: 10 Septiembre 2024

Marco Teórico.

En matemáticas y programación, el uso de funciones trigonométricas y la generación de números aleatorios son fundamentales para diversos campos, como la simulación, modelado y análisis de datos. Las funciones seno y coseno son particularmente importantes en el estudio de fenómenos periódicos, tales como ondas sonoras, oscilaciones y movimientos cíclicos. En este contexto, la función $f(t)=\sin((2\pi t)/5+0.2$ representa una onda senoidal desplazada en fase, lo cual es útil para modelar comportamientos periódicos con una frecuencia específica.
En matemáticas y programación, el uso de funciones trigonométricas y la generación de números aleatorios son fundamentales para diversos campos, como la simulación, modelado y análisis de datos. Las funciones seno y coseno son particularmente importantes en el estudio de fenómenos periódicos, tales como ondas sonoras, oscilaciones y movimientos cíclicos. En este contexto, la función $f(t)=\sin((2\pi t)/5+0.2)$ representa una onda senoidal desplazada en fase, lo cual es útil para modelar comportamientos periódicos con una frecuencia específica.
La generación de números aleatorios se realiza comúnmente utilizando librerías como NumPy en Python, que permite crear listas de números con características específicas (en este caso, valores flotantes entre -1000 y 1000). Estos números son esenciales en simulaciones estadísticas, en la generación de muestras de datos y en el análisis de grandes conjuntos de datos donde se necesita aleatoriedad controlada.
El análisis de las pendientes de una función, utilizando la derivada o aproximaciones mediante secantes, permite identificar puntos críticos en una gráfica, como máximos, mínimos y puntos de inflexión. En este experimento, se buscó el punto donde la pendiente de la función es más cercana a cero, lo cual indica un cambio mínimo en la tasa de la función, siendo este punto relevante en el estudio de estabilidad o equilibrio en sistemas físicos y matemáticos.

Material y Equipo

- **Hardware:** Computadora con capacidad para ejecutar Python.
- **Software:**
  - Python 3.x
  - Google Colab.

Desarrollo de la practica

1. **Generación de Números Aleatorios:** Se utilizó la función np.random.uniform(-1000, 1000, 100) de la librería NumPy para crear una lista de 100 números aleatorios entre -1000 y 1000. Posteriormente, se identificaron los dos valores más pequeños y más grandes de la lista mediante np.sort.
2. **Cálculo de la Función f(t)f(t)f(t):** Se definió un conjunto de valores de ttt para dos periodos de la función $f(t)=\sin((2\pi t)/5+0.2)$. Se utilizó np.linspace para generar estos valores y np.sin para calcular los correspondientes valores de la función.

3. **Identificación de Pendientes Casi Cero:** Se aproximó la derivada de la función mediante secantes calculadas con la diferencia entre puntos consecutivos (np.diff). El punto con la pendiente más cercana a cero fue identificado utilizando la función np.argmin(np.abs(slopes)).
4. **Gráficas y Resultados:** La gráfica de la función f(t)f(t)f(t) se generó utilizando Matplotlib, mostrando los puntos críticos donde la pendiente es más cercana a cero.

La principal dificultad fue asegurar la precisión en la selección de los valores extremos. Se solucionó ordenando el arreglo y seleccionando los primeros y últimos dos elementos. Ajustar el cálculo de la pendiente para coincidir con el punto medio de los valores t y f(t). Se resolvió calculando el promedio de los puntos involucrados.

Diagramas, gráficas y pantallas.

Figura 1. Se muestra la elección de los 2 números más chicos y los 2 más grandes.

```
(array([-937.06574209, -936.70062401]), array([982.61335377, 996.37801578]))
```

Figura 1.

Figura 2.

Gráfica de la función $f(t)=\sin(2\pi t5+0.2)$ con el punto donde la pendiente es cercana a cero marcado en rojo.

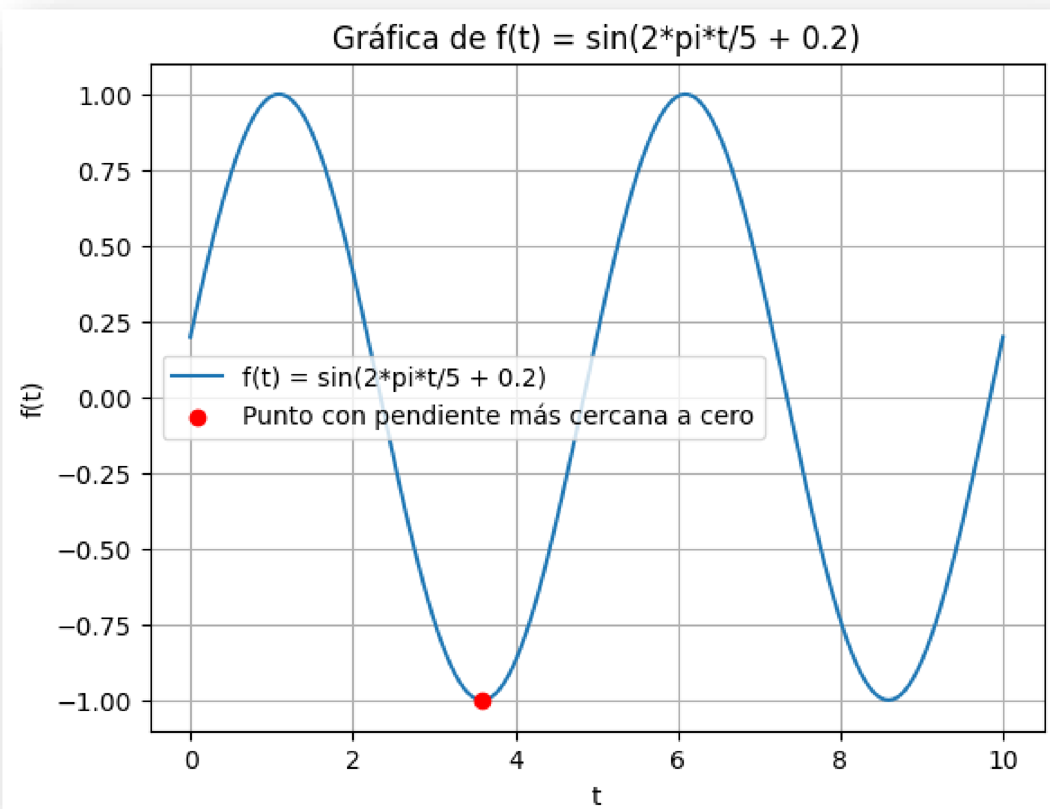

Figura 2.

Conclusiones

El experimento permitió analizar el comportamiento de una función periódica y la identificación de puntos críticos basados en la pendiente. Este análisis es útil en diversas aplicaciones prácticas, como la evaluación de estabilidad en sistemas físicos. Se puede explorar con diferentes frecuencias y fases para observar cómo cambian los puntos de pendiente mínima y el impacto en la forma de la función. Además, ajustar la densidad de puntos puede mejorar la precisión al aproximar tangentes.

Bibliografía

-NumPy Documentation
  Autor: NumPy Developers
  Fecha: 2024
  Fuente: https://numpy.org/doc/

- https://colab.research.google.com/

Anexo:

```
import numpy as np
import matplotlib.pyplot as plt

# Paso 1: Definir una lista con 100 números aleatorios flotantes entre -1000 y 1000
random_numbers = np.random.uniform(-1000, 1000, 100)

# Encontrar los dos números más pequeños y más grandes
smallest_two = np.sort(random_numbers)[:2]  # Los dos más pequeños
largest_two = np.sort(random_numbers)[-2:]  # Los dos más grandes

# Paso 2: Producir una lista con dos periodos de la función f(t) = sin(2*pi*t/5 + 0.2)
t_values = np.linspace(0, 10, 200)  # Dos periodos con muestreo fino
f_values = np.sin(2 * np.pi * t_values / 5 + 0.2)  # Valores de la función

# Paso 3: Graficar los puntos (t, f(t))
plt.plot(t_values, f_values, label='f(t) = sin(2*pi*t/5 + 0.2)')
plt.xlabel('t')  # Etiqueta del eje x
plt.ylabel('f(t)')  # Etiqueta del eje y
plt.title('Gráfica de f(t) = sin(2*pi*t/5 + 0.2)')  # Título de la gráfica

# Paso 4: Encontrar los puntos donde la tangente a f(t) tiene una pendiente más cercana a
cero
# Aproximar la tangente usando la secante
slopes = np.diff(f_values) / np.diff(t_values)  # Cálculo de pendientes entre puntos
consecutivos
zero_slope_index = np.argmin(np.abs(slopes))  # Índice de la pendiente más cercana a cero

# Puntos donde la pendiente es más cercana a cero
t_zero_slope = (t_values[zero_slope_index] + t_values[zero_slope_index + 1]) / 2
```

```
f_zero_slope = (f_values[zero_slope_index] + f_values[zero_slope_index + 1]) / 2

# Marcar el punto con la pendiente más cercana a cero en la gráfica
plt.scatter(t_zero_slope, f_zero_slope, color='red', zorder=5, label='Punto con pendiente
más cercana a cero')

plt.legend()  # Mostrar leyenda
plt.grid(True)  # Mostrar la cuadrícula
plt.show()

(smallest_two, largest_two)  # Mostrar los valores más pequeños y más grandes encontrados
```