ESCUELA
POLITÉCNICA
NACIONAL

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

MÉTODOS NUMÉRICOS (GR1CC)

PROYECTO IB

Alumnos:

Ivonne Carolina Ayala Coloma
Boris Gabriel Garcés Proaño
David Esteban Morales Estrella
Alicia Dayana Pereira Tuqueres

PROFESOR: Jonathan Alejandro Zea G.

FECHA DE ENTREGA: 30/11/2024

Índice

1. Objetivos

1.1. Objetivo General

Desarrollar una calculadora de ahorro moderno en Python que simule el crecimiento de una cuenta de ahorros, incorporando depósitos iniciales y aportes regulares, además de determinar la tasa de interés mediante el método de la secante y representar los resultados gráficamente.

1.2. Objetivos Específicos

- Calcular numéricamente la tasa de interés que satisface las condiciones del sistema, asegurando precisión y eficiencia.
- Desarrollar una aplicación que permita al usuario ingresar parámetros personalizables, como depósito inicial, monto y frecuencia de aportes, y tasa de interés.
- Visualizar el comportamiento del capital y los intereses acumulados mediante gráficos interactivos y tablas comprensibles.

2. Introducción

El interés compuesto representa un principio fundamental en las finanzas, dado que refleja cómo un capital inicial puede crecer exponencialmente mediante la acumulación de intereses sobre intereses. No obstante, el cálculo tradicional del interés compuesto no contempla el impacto de aportes regulares, un escenario típico en programas de ahorro modernos.

En este contexto, un banco requirió el desarrollo de una calculadora que permitiera simular el comportamiento de una cuenta de ahorros, considerando no solo un depósito inicial, sino también aportes regulares y la acumulación de intereses. Además, el programa debía ser lo suficientemente flexible para soportar diferentes frecuencias de aportes (semanales, mensuales, trimestrales, etc.) y facilitar el análisis de diversos escenarios financieros.

El problema, abordado mediante el concepto de progresión geométrica, implicó la formulación de una expresión matemática precisa que incorporara los aportes regulares en el cálculo del interés compuesto. Adicionalmente, para determinar la tasa de interés efectiva en cada escenario, se empleó el método de la secante, un enfoque numérico iterativo que permitió resolver la ecuación propuesta de forma eficiente.

Finalmente, la solución fue implementada en Python, aprovechando sus bibliotecas científicas para realizar cálculos, generar simulaciones y presentar resultados mediante gráficas visualmente comprensibles.

3. Desarrollo

3.1. Planteamiento del Problema

El objetivo del proyecto consistió en desarrollar un programa que permitiera calcular y simular el valor final del ahorro, considerando los aportes regulares y las ganancias por

intereses, además de ser flexible para soportar diferentes intervalos de aportes (mensuales, trimestrales, bimestrales, etc.).

Durante la definición del proyecto, los técnicos del banco indicaron que el programa debía incluir las siguientes características:

1. **Flexibilidad en los intervalos de aportes:** La calculadora debía soportar configuraciones para realizar aportes en períodos distintos al semanal (por ejemplo, mensual o trimestral).
2. **Escenarios simulados:** El sistema debía permitir la variación de parámetros como el depósito inicial, la frecuencia de los aportes, el monto de los aportes y la tasa de interés.
3. **Interfaz comprensible:** Se solicitó una salida de datos que presentara los cálculos de manera estructurada, similar a la tabla de ejemplo proporcionada.

Este proyecto buscó dotar al banco de una herramienta que no solo facilitara la evaluación de escenarios de ahorro, sino que también permitiera ajustar los parámetros clave para evaluar estrategias de ahorro más atractivas para sus clientes.

3.2. Demostración Matemática de la Solución

3.2.1. Cálculo Básico del Interés Compuesto

La fórmula inicial para calcular el valor final V_f de un capital inicial V_0, sometido a una tasa de interés i durante n períodos, está dada por:

$$V_f = V_0(1+i)^n$$

Sin embargo, esta fórmula no considera los aportes regulares al capital.

3.2.2. Incorporación de los Aportes Regulares

Los aportes periódicos A también generan intereses, y su efecto acumulado se agrega al capital inicial. Esto se expresa como:

$$V_f = V_0(1+i)^n + A(1+i)^{n-1} + A(1+i)^{n-2} + \cdots + A(1+i)^1$$

Reescribiendo esta suma, se representa como una progresión geométrica:

$$V_f = V_0(1+i)^n + \sum_{k=1}^{n-1} A(1+i)^k$$

3.2.3. Progresión Geométrica

En esta progresión geométrica:

- El primer término es $a_0 = A(1+i)$,
- La razón común es $r = 1+i$,
- El número de términos es $m = n-1$.

La suma de la progresión geométrica se calcula como:

$$S_n = a_0 \frac{r^m - 1}{r - 1}$$

Sustituyendo los valores correspondientes:

$$S_n = A(1+i)\frac{(1+i)^{n-1} - 1}{(1+i) - 1}$$

Simplificando, se obtiene:

$$S_n = A\frac{(1+i)^n - (1+i)}{i}$$

3.2.4. Fórmula General para el Valor Final

Integrando la suma de los aportes S_n en la fórmula original, se deriva:

$$V_f = V_0(1+i)^n + A\frac{(1+i)^n - (1+i)}{i}$$

3.2.5. Determinación de la Tasa de Interés mediante el Método de la Secante

Para encontrar la tasa de interés i que satisface el sistema, se empleó el método de la secante. Se partió de la ecuación:

$$V_f - V_0(1+i)^n - A\frac{(1+i)^n - (1+i)}{i} = 0$$

El método de la secante resuelve esta ecuación iterativamente mediante aproximaciones iniciales para i. Este enfoque numérico garantiza convergencia bajo las condiciones iniciales definidas.

3.3. Implementación en Python

El código implementado tiene como objetivo calcular el interés compuesto acumulado de una cuenta de ahorros con depósitos periódicos, así como generar una gráfica para visualizar la relación entre el tiempo (representado por los períodos) y el saldo total acumulado, que incluye los intereses generados.

3.3.1. Back-End

El propósito de este código es realizar cálculos relacionados con el interés compuesto, aportes periódicos, y graficar su comportamiento, así como determinar la tasa de interés que satisface ciertos valores usando el método de la secante. A continuación, se explica cada función y cómo funciona dentro del programa:

Generación de la Fórmula de Cálculo

```
def f(v0, interes, a):
    def funcion(x):
        return v0 * (1 + interes) ** x + a * (((1 + interes) ** x - (1 + interes)) / i
    return funcion
```

Esta función genera otra función matemática basada en los parámetros de entrada:

- `v0`: Capital inicial.
- `interes`: Tasa de interés.
- `a`: Aporte periódico.

Devuelve una función anidada (`funcion`) que calcula el saldo total acumulado para un período x. Utiliza la fórmula de interés compuesto sumada a los efectos de los aportes periódicos expresados en forma de progresión geométrica.

Gráfica del Crecimiento del Capital

```
def graficar(capitalInicial, aporte, interes, frequency):
    ...
```

Esta función se encarga de crear una gráfica que muestre la relación entre el tiempo (períodos) y el saldo acumulado (incluidos intereses y aportes).

- Utiliza la función generada por `f` para calcular los valores del saldo acumulado para una serie de períodos.
- Genera un rango de valores para los períodos en función de la frecuencia de los aportes (diario, mensual, etc.), obtenida con la función `periodo`.
- Calcula los valores de saldo acumulado para esos períodos y los grafica con `matplotlib`.
- Guarda la gráfica como un archivo PNG en la carpeta `static`, verificando si el archivo se generó correctamente.

Determinación de Períodos por Frecuencia La función `periodo` calcula el número de períodos en función de la frecuencia de los aportes.

```
def periodo(frequency):
    ...
```

Devuelve un número fijo de períodos asociado a una frecuencia de aportes:

- Diario: 365.
- Mensual: 12.
- Semanal: 52.
- Trimestral: 4.
- Semestral: 2.
- Anual: 1.

Generación de una Tabla de Resultados La función `table_data` crea una tabla con datos detallados sobre el saldo acumulado, ganancia obtenida, capital inicial, y contribuciones realizadas para cada período.

```
def table_data(initial_capital, num_periods, periodic_contribution,
interest, start_period=1):
    ...
```

- Calcula el saldo total, la ganancia y las contribuciones para cada período usando la función generada por `f`.
- Almacena los resultados en una lista de diccionarios, donde cada diccionario representa un período y contiene:
  - `period`: El número de período.
  - `contribution`: La cantidad aportada en ese período.
  - `capital`: El saldo acumulado previo.
  - `gain`: La ganancia obtenida en ese período.
  - `total`: El saldo acumulado total.
- Devuelve la lista de datos como resultado.

Función para Determinar el Interés La función `f`, utilizada en el método de la secante, encuentra el interés desconocido que satisface una ecuación basada en los parámetros.

```
def f(v0, vf, n, a):
    def funcion(x):
        if abs(x) < 1e-10:
            return vf - v0 * (1 + x) ** n
        else:
            return vf - v0 * (1 + x) ** n - a * (((1 + x) ** n - (1 + x)) / x)
    return funcion
```

- V_0: Capital inicial.
- V_f: Capital final deseado.
- n: Número de períodos.
- a: Aporte periódico.

La función calcula la diferencia entre el capital final esperado y el capital acumulado calculado usando una tasa de interés tentativa (x). Si x es muy cercano a cero, la fórmula se simplifica porque el denominador tendería a infinito.

Cálculo del Interés Usando el Método de la Secante La función `obtener_interes` tiene como objetivo encontrar la tasa de interés (x) que hace que la fórmula del capital acumulado sea igual al capital final deseado.

```
def obtener_interes(capital_inicial, capital_final, numero_periodos, aporte):
    return optimize.newton(f(capital_inicial,
    capital_final, numero_periodos, aporte), 1e-5)
```

- Utiliza el método de la secante implementado en `scipy.optimize.newton`, que encuentra el valor de interés iterativamente.
- La función `f` generada previamente actúa como entrada para este método.
- El cálculo comienza con un valor inicial aproximado (1×10^{-5}).

3.3.2. Front-End

Para llevar a cabo este proyecto de manera efectiva, fue necesario combinar diferentes tecnologías, cada una aportando elementos clave para su funcionalidad y presentación. Desde la creación de una interfaz amigable hasta la realización de cálculos y generación de gráficos, estas herramientas trabajaron en conjunto para garantizar que la calculadora de interés compuesto ofreciera una experiencia intuitiva y confiable. A continuación, se describe cómo se implementaron estas tecnologías y su rol dentro del sistema:

1. **Flask:**
   Flask fue el corazón del backend del proyecto. Este framework nos permitió manejar las operaciones principales de la calculadora, como los cálculos de interés compuesto y la generación de gráficos. Por ejemplo, en el archivo `app.py`, se definieron rutas como `/api/calculate`, encargadas de procesar los datos ingresados por el usuario, aplicar la fórmula correspondiente y devolver el resultado en formato JSON. También se utilizó Flask para generar un gráfico dinámico del comportamiento del interés, que se guarda automáticamente en el directorio `static` y se muestra al usuario de manera visual.

2. **HTML:**
   HTML se utilizó para dar forma a la estructura de la aplicación. Con archivos como `index.html` y `details.html`, se construyeron formularios que permiten a los usuarios ingresar datos clave, como el capital inicial, los aportes periódicos y la frecuencia de los mismos. Además, las tablas y gráficos que muestran los resultados se integraron dentro de estas páginas, facilitando una presentación clara y organizada de la información.

3. **CSS:**
   Para garantizar que la calculadora no solo fuera funcional, sino también agradable a la vista, se empleó CSS para estilizar la interfaz. El archivo `styles.css` contiene los estilos que aportan un diseño moderno y profesional. Entre los detalles más destacados están los gradientes de colores, las esquinas redondeadas, y las animaciones que hacen que incluso la pantalla de carga sea visualmente atractiva. Todo esto contribuyó a que la experiencia de uso fuera más cómoda y fluida para el usuario.

4. **JavaScript y JSON:**
   JavaScript, en conjunto con JSON, fue esencial para dotar al sistema de dinamismo y fluidez. A través de JavaScript, se implementaron funciones que gestionan la interacción entre el usuario y la aplicación, como el envío de datos al backend y la actualización de resultados en tiempo real. Por ejemplo, en los archivos `script.js` y `details.js`, funciones como `fetchInterest` envían datos clave ingresados por el usuario al backend utilizando el formato JSON. Este formato permitió estructurar y enviar la información de manera eficiente, mientras que las respuestas del backend, también en JSON, se procesaron para actualizar dinámicamente los resultados en la interfaz, como tablas y gráficos.

5. **Gunicorn:** Gunicorn (Green Unicorn) es un servidor HTTP WSGI que actúa como intermediario entre las solicitudes del cliente y aplicaciones web Python. Su modelo de workers permite gestionar múltiples solicitudes concurrentes de manera eficiente. Compatible con frameworks como Flask, Django y FastAPI, Gunicorn es ideal para despliegues de producción. Aunque no maneja archivos estáticos ni es óptimo para aplicaciones completamente asíncronas, su integración con servidores proxy como Nginx facilita el balanceo de carga y la entrega de archivos estáticos. Su ligereza y configurabilidad lo convierten en una opción popular en el ecosistema Python.

6. **SciPy:** SciPy es una biblioteca de código abierto que extiende las capacidades de NumPy, ofreciendo herramientas avanzadas para computación científica. Sus módulos abarcan optimización, integración, álgebra lineal, estadística y procesamiento de señales. Es ampliamente utilizada en física, ingeniería y aprendizaje automático, y su integración con bibliotecas como Matplotlib y Pandas la posiciona como parte esencial del ecosistema científico de Python. Su documentación extensa y su eficiencia la hacen indispensable tanto para principiantes como para expertos.

7. **NumPy:** NumPy es una biblioteca fundamental para la computación científica en Python. Su objeto principal, `ndarray`, permite manejar arreglos multidimensionales y realizar operaciones matemáticas vectorizadas de manera eficiente. Incluye funciones avanzadas como álgebra lineal, transformadas de Fourier y generación de números aleatorios. Altamente optimizada y escrita mayoritariamente en C, NumPy se integra perfectamente con otras bibliotecas como SciPy, Matplotlib y Pandas, siendo esencial en campos como simulaciones científicas y aprendizaje automático.

8. **Matplotlib:** Matplotlib es una biblioteca para crear visualizaciones gráficas en Python, como gráficos de líneas, barras, histogramas y dispersión. Su módulo `pyplot` ofrece una interfaz similar a MATLAB, ideal para usuarios de todos los niveles. Altamente personalizable, permite ajustar estilos, colores, etiquetas y leyendas, y se integra con bibliotecas como NumPy y Pandas para trabajar con datos estructurados. Por su capacidad de generar gráficos de alta calidad en formatos como PNG, PDF y SVG, Matplotlib es ampliamente utilizada en ciencia de datos, investigación y análisis de negocios.

4. Casos de Prueba

4.1. Prueba 1

- **Capital Inicial:** 1000

- **Aporte Periódico:** 100
- **Frecuencia del Aporte:** Mensual
- **Capital Final:** 2,268.91
- **Número de Períodos:** 12 meses

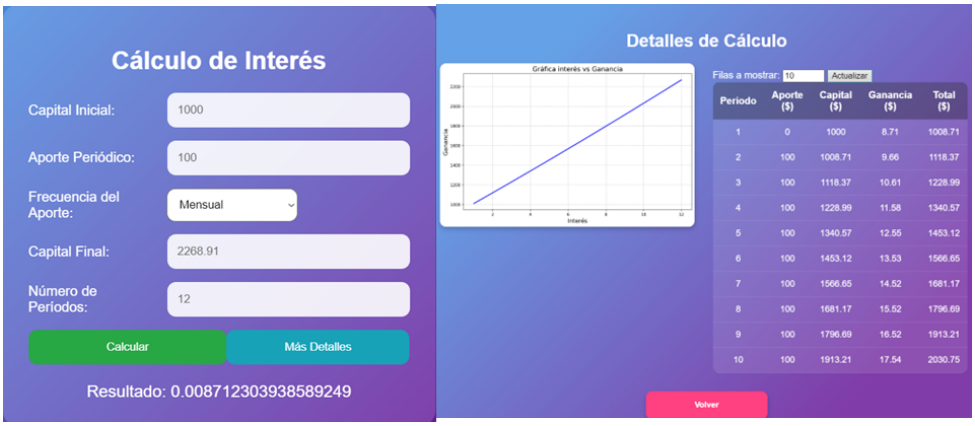

Figura 1: Prueba 1 - Mensual.

4.2. Prueba 2

- **Capital Inicial:** 5000
- **Aporte Periódico:** 500
- **Frecuencia del Aporte:** Trimestral
- **Capital Final:** 9,901.60
- **Número de Períodos:** 4 trimestres

Figura 2: Prueba 2 - Trimestral.

4.3. Prueba 3

- **Capital Inicial:** 10000
- **Aporte Periódico:** 520
- **Frecuencia del Aporte:** Anual
- **Capital Final:** 15,762.82
- **Número de Períodos:** 1 año

Figura 3: Prueba 3 - Anual.

4.4. Prueba 4

- **Capital Inicial:** 2500
- **Aporte Periódico:** 200
- **Frecuencia del Aporte:** Semanal
- **Capital Final:** 10,098.77
- **Número de Períodos:** 24 semanas

Figura 4: Prueba 1 - Semanal.

4.5. Prueba 5

- **Capital Inicial:** 0
- **Aporte Periódico:** 50
- **Frecuencia del Aporte:** Diario
- **Capital Final:** 18,491.55
- **Número de Períodos:** 365 días

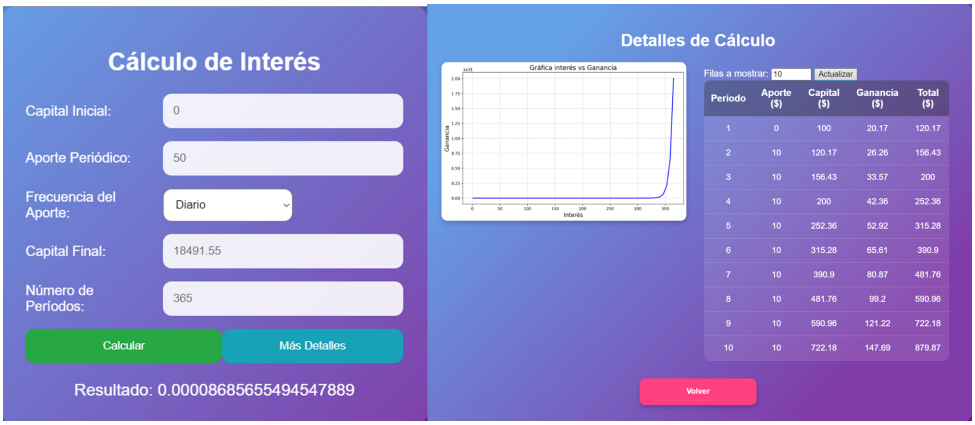

Figura 5: Prueba 5 - Diario.

4.6. Prueba 6

- **Capital Inicial:** 3000
- **Aporte Periódico:** 500
- **Frecuencia del Aporte:** Semestral
- **Capital Final:** 10,442.04
- **Número de Períodos:** 2 semestres

Figura 6: Prueba 6 - Semestral.

Las pruebas realizadas demostraron que la calculadora de interés compuesto funciona correctamente en diversos escenarios. Cada prueba se diseñó para evaluar distintas combinaciones de capital inicial, aportes periódicos y frecuencias, asegurando que el sistema pudiera adaptarse a diferentes necesidades y casos de uso.

Los resultados obtenidos fueron consistentes con los cálculos esperados, lo que refleja la precisión en la implementación tanto del backend como del frontend. Además, las gráficas generadas y las tablas dinámicas mostraron los datos de manera clara y ordenada, lo que permite al usuario interpretar fácilmente los resultados.

Estas pruebas confirmaron que la calculadora no solo es funcional, sino también confiable y robusta, incluso frente a escenarios con grandes volúmenes de datos o frecuencias de aporte variadas.

Conclusiones

- La implementación de fórmulas basadas en la progresión geométrica y el interés compuesto ha demostrado ser una herramienta eficaz para calcular la acumulación de capital a lo largo del tiempo. El uso del método de la secante para determinar tasas de interés desconocidas ha añadido precisión a cálculos que, de otra forma, serían complejos y propensos a errores si se realizaran manualmente. Esto refleja la importancia de los métodos numéricos en la resolución de problemas financieros no lineales.

- La integración de gráficos que representan la evolución del capital acumulado y la ganancia en función del tiempo proporciona una visión clara e intuitiva del impacto de las decisiones financieras. Esta representación visual no solo mejora la comprensión de los resultados, sino que también permite identificar tendencias clave, como el efecto exponencial del interés compuesto y la influencia de los aportes periódicos en el crecimiento del capital.

- La capacidad de ajustar parámetros como el capital inicial, la frecuencia de aportes, el interés y el número de períodos permite modelar una amplia variedad de escenarios financieros. Esto hace que la herramienta sea altamente adaptable a las necesidades de diferentes usuarios, desde individuos que buscan optimizar su ahorro hasta instituciones que requieren proyectar el comportamiento de inversiones a largo plazo.

5. Anexo

En este anexo se incluye un diagrama de flujo que ilustra el proceso descrito en el documento. Este diagrama ofrece una representación visual de los pasos clave y las decisiones involucradas en el desarrollo del sistema.

Figura Anexo 1: Diagrama de flujo del proceso