Universidad de La Coruña

Informe de ejercicios de Python

Software Profesional en Finazas

Miriam Gutiérrez Serrano

Índice

1. Exponencial

Teniendo en cuenta que

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Implementar un programa que obtenga el número de términos necesarios para calcular e^1 con el menor error posible.

Archivo: `Ej2_56.py`

- Primero calculamos la serie de la exponencial $e^1 = \sum_{n=0}^{\infty} \frac{1}{n!}$.
- Iteramos hasta que el siguiente término sea menor que la precisión de máquina (`sys.float_info.epsilon`).
- Se acumulan los términos en la variable `suma` y calculamos el error absoluto frente al valor real obtenido con `math.exp(1)`.

Resultados:

```
Valor aproximado de e^1: 2.71828182845904553
Error respecto valor real de e^1: 4.44089209850062616e-16
```

2. Método de Bisección

Se nos pedía implementar una función que calcule la solución exacta de la ecuación $f(x) = x^2 - e^{-x} - 1,5$ en el intervalo $[-1, 1]$ por el método de bisección.Archivo: `biseccion.py`

- Primero implementamos la función $f(x) = x^2 - e^{-x} - 1,5$.
- Aplicamos el método de bisección en un intervalo $[a, b]$ hasta alcanzar una tolerancia o un número máximo de iteraciones.
- Finalmente, se informa de la solución aproximada x o de la imposibilidad de hallar raíz si el signo en los extremos no cambia.

Resultados:

```
Solución: x = -0.4740962982
```

3. Módulo para la realización de cálculos sobre hipotecas

Sabiendo que la cuota mensual m que hemos de pagar para amortizar una hipoteca de h euros a los largo de n años a un interés compuesto de i por cien anual se calcula con la fórmula:

$$m = \frac{hr}{1 - (1 + r)^{-12n}},$$

donde $r = \dfrac{i}{100 \cdot 12}$, se nos pedía diseñar un mósulo que contenga las funciones descritas después.

Archivo: `Hipoteca_funciones.py`

Este Script contiene las funciones de cálculo matemático para hipotecas pedidas.

Función `cuota(h, n, i)`

- Calcula la cuota mensual m para un préstamo de capital h, plazo n años y tipo de interés anual $i\,\%$.
- Calculo
$$r = \frac{i}{100 \times 12}, \qquad m = \frac{h\,r}{1 - (1+r)^{-12\,n}}.$$
- Salida: Un `float` con la cuota mensual.

Función `total_pagado(h, n, i)`

- Tenemos que obtener el importe total abonado tras las $12n$ cuotas.
- Cálculamos
$$\text{Total} = m \times 12 \times n,$$
donde $m = \texttt{cuota}(h, n, i)$.
- Salida: Un `float` con el total pagado.

Función `intereses(h, n, i)`

- calcula la cantidad de intereses pagados a lo largo de la hipoteca.
- calculo:
$$\text{Intereses} = \text{Total pagado} - h.$$
- Salida: Un `float` con los intereses totales.

Función `intereses_porcentaje(h, n, i)`

- Expresa los intereses como porcentaje del capital inicial.
- Calculo
$$\frac{\text{Intereses}}{h} \times 100\,\%.$$
- Salida: Un `float` con el porcentaje de sobrecoste.

Archivo: `Hipoteca_resultados.py`

Además, una vez diseñado el módulo, teníamos que implementar un programa que pida al usuario el capital h, el tipo de interés anual i, y el número de años n, y muestre por pantalla la cuota mensual, la cantidad de intereses que habremos pagado al banco y con qué tanto por cien del capital inicial se corresponden.

Reultados:

```
        ---RESULTADOS---
Cuota mensual: 1166.75
Total pagado:  210015.000€
Total intereses pagados al banco: 60015.000€
Porcentaje pagado en intereses: 40.01%
```

4. Programación Orientada a Objetos

Archivo: `POO.py`

- Definimos la clase `Punto` con atributos `x`, `y` y métodos para mover y sumar puntos.
- Define la clase `Linea` con dos `Punto` (inicio y fin), métodos para imprimir y sumar líneas.
- Mostramos el uso creando puntos y líneas, y mostrando resultados de suma e impresión de extremos.

Resultados:

```
 Linea(Punto(0,0), Punto(1,1))
Linea(Punto(0,1), Punto(1,0))
Resultado suma de l1 y l2:
Linea(Punto(0,1), Punto(2,1))
Extremos de la linea l1:
Inicio = Punto(0,0)
Fin = Punto(1,1)
```

5. Valoración de opciones

5.1. Ejercicio 1

Primero se nos pedía Implementar el método de Monte Carlo en una función que permitiera valorar tanto opciones vainilla europeas de compra como de venta.
Archivo: `opcion_vainilla_mc.py`

Los parámetros se encuentran en el Script explicados.

- Discretizavión temporal: $dt = T/N$
- Simulación de las trayectorias: creamos una matriz S de tamaño $(N+1) \times I$, fijamos $S[0]$ y para cada paso genera $z = N(0,1)$ para simular el avance logonormal

$$S_t = S_{t-1} \exp\left(\left(r - \frac{1}{2}\sigma^2\right) dt + \sigma\sqrt{dt} z\right)$$

- Se calcula el Payoff
- Y por último el descuento y promedio:

$$V_0 = e^{-rT} \frac{1}{I} \sum_{i=1}^{\infty} h_T(i)$$

5.2. Ejercicio 2

Después, teníamos que implementar la fórmula analítica de Black-Scholes en una función que nos permitiera valorar tanto opciones vainilla europeas de compra como de venta.
Archivo: `opcion_vainilla_bs.py`

- Cálculo de d_1 y d_2:

$$d_1 = \frac{\ln\big(S_0/K\big) + \big(r + \frac{1}{2}\sigma^2\big)T}{\sigma\sqrt{T}}, \qquad d_2 = \frac{\ln\big(S_0/K\big) + \big(r - \frac{1}{2}\sigma^2\big)T}{\sigma\sqrt{T}},$$

- Fórmulas de precio:

$$\begin{aligned} C &= S_0\,\Phi\big(d_1\big) - K\,e^{-rT}\,\Phi\big(d_2\big), && \text{(call)} \\ P &= K\,e^{-rT}\,\Phi\big(-d_2\big) - S_0\,\Phi\big(-d_1\big), && \text{(put)} \end{aligned}$$

  donde Φ es la función de distribución acumulada de la normal estándar.

- Finalmente, un `if` para considerar si la opción es de venta (put) o de compra (call).

5.3. Ejercicio 3

Archivo: `Ejercicio3.py`

Por último, había que valorar con ambas funciones una opción europea de compra y otra de venta con vencimiento 3 meses, precio de ejercicio 15.00, precio del activo hoy 17.00, tipo de interés 3 % y volatilidad del activo 25 %.
Resultados:

```
El precio de la opcion vainilla europea de compra (Black-Scholes) es 2.259120
El precio de la opcion vainilla europea de venta (Black-Scholes) es 0.147040
El precio de la opcion vainilla europea de compra (Monte Carlo) es 2.259370
El precio de la opcion vainilla europea de venta (Monte Carlo) es 0.149070
```

5.4. Ejercicio 4

Archivo: `Ejercicio4.py`

En el ejercicio 4, se nos pedía implementar en Python dos funciones que permitan calcular el valor de opciones vainilla europeas de compra y de venta con dividendos continuos. En una se calcula el precio con el método de Monte Carlo (más el intervalo de confianza al 95 %) y en la otra con la fórmula de Black-Scholes.

Son funciones muy parecidas a las desarrolladas en los apartados anteriores, pero adaptanto los dividendos D_0: en Monte Carlo a las trayectorias y en Black-Scholes a d_1, d_2 y $C(t,S)$ y $P(t,S)$.

Para concluir, se utilizan estas funciones para valorar una opción vainilla europea de compra y otra de venta (en el caso de monte Carlo también se muestra el intervalo de confianza al 95 %) con los siguientes datos: vencimiento 3 meses, precio de ejercicio 15.00, precio del activo en el instante inicial 17.00, tipo de interés 3 %, volatilidad del activo 25 % y tasa de dividendos 1,5 %.
Resultados:

```
---Valoración de una opción vainilla de compra con dividendos continuos---

El precio de la opción por el Método de Monte Carlo es 2.208658
Intervalo de confianza al 95%: (2.208658, 2.208658)
```

```
El precio de la opción por la fórmula analítica de B{S es 2.204000
------------------------------------------------------------

---Valoración de una opción vainilla de venta con dividendos continuos---

El precio de la opción por el Método de Monte Carlo es 0.155493
Intervalo de confianza al 95%: (0.155493, 0.155493)

El precio de la opción por la fórmula analítica de B{S es 0.155552
```

6. Valoración de Opciones con QuantLib

6.1. Ejercicio 6: Opción Europea con Dividendos Continuos

Archivo: `Ejercicio6.py`

- Creamos cotizaciones y curvas de tipo libre de riesgo y dividendos (`FlatForward`) con day-count `Actual365Fixed`.
- Superficie de volatilidad constante (`BlackConstantVol`).
- Se define el proceso estocástico con `GeneralizedBlackScholesProcess`, se usa este porque es con Dividendos continuos.
- Valoramos la opción europea `Call` usando:
  1. Método de diferencias finitas.
  2. Monte Carlo y 100 000 muestras.
  3. Fórmula analítica de Black–Scholes.
- Se muestran los precios de cada uno de ellos respectivamente y el error de Monte Carlo.

Resultados:

```
--Valoración de una opción vainilla europea de compra con dividendos continuos--

Precio de la opcion europea de compra por Diferenciass Finitas: 7.069977
Precio opción europea compra por Monte Carlo: 7.081710
Error cometido con el método de Monte Carlo: 0.024155
Precio opción europea compra por fórmula Black Scholes: 7.069926
```

Los archivos correspondientes a cada sección se adjuntan junto con este informe para su consulta y ejecución.