

Applications financières en C# (2024 – 2025)

L. Davoust

SOMMAIRE

Exercices d'application.....	2
Exercice 1 : Manipulation de la console et des types élémentaires.....	2
Exercice 2 : Création d'une table de valeur actuelle de 1€ payable dans n périodes.....	3
Exercice 3 : Fonction factorielle.....	3
Exercice 4 : Nombre de Neper	3
Exercice 5 : Suite de Fibonacci	3
Exercice 6 : Fonction factorielle, Nombre de Neper, Suite de Fibonacci.....	4
Exercice 7 : Distance euclidienne.....	4
Exercice 8 : Création d'une application bancaire	4
Exercice 9 : Manipulation des taux d'intérêt	4
Exercice 10 : Récupération de données financières depuis YahooFinance	5
Projets d'application.....	7
Projet 1 : Simulation de prix de sous-jacent.....	7
Projet 2 : Pricing d'option et de stratégie d'option.....	9
Projet Final d'Evaluation	Erreur ! Signet non défini.
Sujet.....	Erreur ! Signet non défini.
Etapes du projet	Erreur ! Signet non défini.
Cahier des charges	Erreur ! Signet non défini.
Exemples de projet.....	Erreur ! Signet non défini.

Applications financières en C# (2024 – 2025)

L. Davoust

EXERCICES D'APPLICATION

Exercice 1 : Manipulation de la console et des types élémentaires

1. Tester les éléments suivants pour une variable de type int :
 - Addition (+) : $5 + 3 = 8$
 - Soustraction (-) : $5 - 3 = 2$
 - Multiplication (*) : $5 * 3 = 15$
 - Division (/) : $5 / 3 = 1$ si int
 - Modulo (%) : $5 \% 3 = 2$
2. Tester les éléments suivants pour une variable de type int :
 - Égalité (==) : $5 == 3$ donne False
 - Inégalité (!=) : $5 != 3$ donne True
 - Supérieur (>) : $5 > 3$ donne True
 - Inférieur (<) : $5 < 3$ donne False
 - Supérieur ou égal (>=) : $5 >= 3$ donne True
 - Inférieur ou égal (<=) : $5 <= 3$ donne False
3. Tester les éléments suivants pour une variable de type int :
 - ET (&) : $5 \& 3$ donne 1 (en considérant les représentations binaires de 5 (101) et 3 (011), le résultat est 001)
 - OU (|) : $5 | 3$ donne 7 (en considérant les représentations binaires de 5 (101) et 3 (011), le résultat est 111)
4. Tester les éléments suivants pour une variable de type char :
 - Égalité (==) : $'a' == 'b'$ donne False
 - Inégalité (!=) : $'a' != 'b'$ donne True
 - Supérieur (>) : $'b' > 'a'$ donne True
 - Inférieur (<) : $'a' < 'b'$ donne True
 - Supérieur ou égal (>=) : $'b' >= 'a'$ donne True
 - Inférieur ou égal (<=) : $'a' <= 'b'$ donne True
5. Tester les éléments suivants pour une variable de type char :
 - Conversion en majuscule : $'a'.ToUpper()$ donne 'A'
 - Conversion en minuscule : $'A'.ToLower()$ donne 'a'
 - Vérifier si le caractère est une lettre : $'a'.IsLetter$ donne True
 - Vérifier si le caractère est un chiffre : $'1'.IsDigit$ donne True
6. Tester les éléments suivants pour une variable de type string :
 - Concatène « Hello » et « world ! » avec le + et le String.Concat

Applications financières en C# (2024 – 2025)

L. Davoust

- Compare « Apple » et « apple »
- Recherche « world » dans « Hello world »
- Remplace « world » par « C# » dans « Hello world ! »
- Découpe la chaîne de caractères au niveau des virgules « apple,orange,banana »

Exercice 2 : Création d'une table de valeur actuelle de 1€ payable dans n périodes

On va créer une table qui aura les périodes en ligne et les taux d'intérêt en colonne.

La valeur associée sera donc $(1 + r)^{-T}$.

Ecrire les résultats dans un fichier CSV / Excel.

Exercice 3 : Fonction factorielle

La fonction factorielle (dit $n!$) se calcule comme suit :

$$n! = \begin{cases} 1 \times 2 \times \dots \times (n-1) \times (n), & \text{si } n \geq 1 \\ 1, & \text{si } n = 0 \end{cases}$$

1. Calculez le résultat pour $n=10$ puis $n=15$.
2. Packagez cette procédure dans un fonction Factorielle
3. Bonus : Rendre cette fonction récursive

Exercice 4 : Nombre de Neper

On peut approximer le nombre de Neper par convergence de la suite suivante :

$$e \approx \sum_{i=0}^{+\infty} \frac{1}{i!}$$

1. Créer la fonction qui répond à cet algorithme.
2. Calculez pour $n=10$, puis $n=15$, puis $n=20$.
3. Etudiez la convergence de cette suite par rapport à la valeur théorique.

Exercice 5 : Suite de Fibonacci

La suite de Fibonacci s'écrit comme suit :

$$u(0) = 0, u(1) = 1,$$

$$u(n) = u(n-1) + u(n-2) \quad \forall n \geq 2$$

1. Créer la fonction de Fibonacci.
2. Bonus : Rendre cette fonction récursive
3. Calculez pour $n=10$, pour $n=20$ et pour $n=30$

Applications financières en C# (2024 – 2025)

L. Davoust

Exercice 6 : Fonction factorielle, Nombre de Neper, Suite de Fibonacci

1. Packagez ces fonctions dans des classes.
2. Optimisez leur fonctionnement à l'aide de dictionnaire temporaire.

Exercice 7 : Distance euclidienne

La distance euclidienne entre deux vecteurs se calcule comme suit :

$$\text{distance} = \sqrt{\sum_i (x_i^A - x_i^B)^2}$$

1. Créer une fonction associée à ce calcul.
2. Créer une classe dérivée où l'on utilise la valeur absolue à la place du carré.

Exercice 8 : Création d'une application bancaire

Cette application bancaire (de type console) permet de gérer les comptes d'un utilisateur.

1. Une classe Utilisateur va exister et contenir les identifiants d'un utilisateur (id, prénom, nom, ville de naissance, adresse).
2. Une classe abstraite Compte va exister et contenir les informations suivantes Solde, Déposer, Retirer. On va auditer les différents mouvements.
3. Une classe CompteCourante va reprendre les éléments essentiels de Compte.
4. Une classe CompteEpargne va ajouter une notion d'intérêt quotidien calculé par solde * taux d'intérêt
5. Extension du dépôt et du retrait par devise.
6. Extension du dépôt et du retrait en ajoutant des moyens de paiement (Cash, Chèque, CB).

Exercice 9 : Manipulation des taux d'intérêt

Soit $P(t, T)$ le prix d'une obligation payant 1 EUR en $t = T$, vu de $t = t_0$

Mettez en place le pricing en fonction des conventions suivantes :

- (ICA) Intérêt composé annuel (aka Yield convention)

$$P(t_0, T) = \frac{1}{(1 + r(t_0, T))^{T-t_0}}$$

- (ICS) Intérêt composé avec k subdivisions de l'année

$$P(t_0, T) = \frac{1}{\left(1 + \frac{r(t_0, T)}{k}\right)^{(T-t_0) \times k}}$$

Applications financières en C# (2024 – 2025)

L. Davoust

- (CC) Composition en temps continue :

$$P(t_0, T) = \exp(r(t_0, T) \times (T - t_0))$$

- (CL) Composition linéaire :

$$P(t_0, T) = \frac{1}{1 + r(y_0, T) \times (T - t_0)}$$

Sur la base de l'exercice précédent, ajoutez une définition dynamique de la maturité en années à partir de date. Les conventions à utiliser sont les suivantes :

- (MM) Money Market : $ACT/360$ où ACT = nombre de jours exactes entre les deux dates
- (BB) Bond Basis :
- $30/360 = \frac{\max(30-d_1, 0) + \min(d_2, 30) + 30 \times (m_2 - m_1 - 1) + 360 \times (\text{annee}_2 - \text{annee}_1)}{360}$
- (DF) Discount Factors : $ACT/365$

Exercice 10 : Récupération de données financières depuis YahooFinance

1. A partir du site Yahoo Finance, récupérer la série de prix du Cac40.
2. Automatiser ce processus en fonction du code fourni par l'utilisateur.
3. Optimiser cette fonction en utilisant async/await pour lancer plusieurs requêtes simultanément.
- 4.

Exercice 11 : Trading algorithmique avec moyennes mobiles

1. Télécharger des données de prix sur Yahoo Finance. Utilisons ces données sous format CSV dans un premier temps simplifier les étapes.
2. Créer une classe qui reflète les champs de ce fichier et créer un processus d'importation grâce à la librairie CSVHelper.
3. Mettre en place la stratégie de trading suivante :
 - a. Calcul de la moyenne mobile de taille n . Permettre le calcul de cette moyenne avec deux méthodes :
 - i. Classique : $\bar{x} = \frac{\sum_i X_i}{n}$
 - ii. Exponentielle : soit N le nombre de période qui représente 86% du poids total, $\alpha = 1 - (1 - 0.86)^{1/N}$ la constante de lissage associée, $\bar{x}_i = \sum_i \alpha(1 - \alpha)^n x_{t-i} = \alpha(x_i + (1 - \alpha)x_{i-1} + (1 - \alpha)^2 x_{i-2} + \dots) = \bar{x}_{i-1}(1 - \alpha) + x_i \alpha$

Applications financières en C# (2024 – 2025)

L. Davoust

- b. Soit deux moyennes mobiles : 1 dite de court terme (par exemple $n=50$) et 1 dite de long terme (par exemple $n=200$). Si la court terme croise à la hausse la long terme, alors c'est un signal d'achat. Si la court terme croise à la baisse la long terme, alors c'est un signal de vente.
4. Backtester cette stratégie : vous calculerez les indicateurs suivants : la volatilité, la performance totale, annualisée et la performance mois à mois.
5. Vous comparerez cette stratégie par rapport à une stratégie de Buy & Hold.
6. Vous connecterez les chroniques de prix via l'API de Yahoo Finance pour pouvoir tester plusieurs chroniques de prix (vous stockerez les données en temporaire pour éviter de relancer les requêtes)
7. Optimisez cette stratégie entre deux dates en ex-ante (par exemple avant 1 janvier 2020) puis testez la stratégie sur la période restante.
8. Exportez les résultats dans Excel (ce fichier sera un fichier « *template* » où l'on pourra avoir des graphiques etc.)

Applications financières en C# (2024 – 2025)

L. Davoust

PROJETS D'APPLICATION

Ces projets sont des projets que l'on verra pendant les séances de TD.

Projet 1 : Simulation de prix de sous-jacent

Objectif :

Obtenir un simulateur de prix pour les prix d'un actif financier ou d'un panier d'actifs. Ce simulateur permettra de générer de longues séries chronologiques destinées aux gestionnaires de risques et gérants pour anticiper les différents contextes de marché. Dans le cadre de ce projet, le simulateur ne porte que sur des produits linéaires (et non dérivés). Cependant une évolution sur les dérivés pourra être envisagé en cas de succès sur ce projet.

Fonctionnalités :

1. Génération d'un échantillon de taille N de nombres aléatoires uniforme compris en 0 et 1. Ils serviront de base pour les échantillons suivants. Une vérification des résultats doit être possible (comme la moyenne, l'écart-type, le minimum, le maximum, un graphique de distribution, les quantiles 10/25/50/75/90).
2. Transformation d'un échantillon de taille N de nombres aléatoires uniforme en un échantillon de nombres aléatoires Gaussiens centrés réduits (moyenne 0 et variance 1). A partir de 2 nombres u_1 et u_2 uniformes, on peut simuler un point avec la formule suivante $z_1 = \sqrt{-2 \times \log(u_1)} \times \cos(2\pi \times u_2)$ et $z_2 = \sqrt{-2 \times \log(u_1)} \times \sin(2\pi u_2)$. Comme pour 1., faire un audit des résultats. (*Transformation de Box Muller*)
3. Simulation d'un mouvement brownien qui est la base de calcul pour la génération de prix d'actif financier. Un mouvement brownien $W(t)$ peut être simulé en utilisant la relation $W(t_{i+1}) = \sqrt{\Delta t} \cdot Z_i$ où est la taille du pas de temps et Z_i est une variable aléatoire normalement distribuée avec une moyenne de 0 et une variance de 1.
4. Simulation d'un mouvement brownien géométrique qui est souvent utilisé pour simuler le prix des actifs financiers. La formule de ce modèle est $S(t) = S(0) \times \exp\left(\left(r - \frac{\sigma^2}{2}\right)t + \sigma W(t)\right)$ où $S(0)$ est le prix initial, r est le taux de rendement, σ est la volatilité et $W(t)$ est le mouvement brownien simulé.
5. Mise en place d'une décomposition de Cholesky qui permet de corrélérer deux séries indépendantes. La décomposition de Cholesky est une méthode qui décompose une

Applications financières en C# (2024 – 2025)

L. Davoust

matrice symétrique définie positive A en un produit de deux matrices triangulaires : une matrice triangulaire inférieure L et sa transposée L^T de sorte que $A = L \cdot L^T$.

Algorithme de la décomposition :

- La matrice A doit être symétrique ($A = A^T$), à vérifier, et définie positive ($x^T \cdot A \cdot x > 0$ pour tout vecteur non nul x).
- Initialisation : Créer une matrice L de la taille de A ($n \times n$) remplie de 0.
- Pour $i = 0$ à $n - 1$ (lignes de L):
 - Pour $j = 0$ à i (parcours des colonnes de L jusqu'à la diagonale principale) :
 - Si $i = j$ (diagonale) alors $L_{ii} = \sqrt{A_{ii} - \sum_{k=0}^{i-1} L_{ik}^2}$.
 - Si $i \neq j$ alors $L_{ij} = \frac{1}{L_{jj}} (A_{ij} - \sum_{k=0}^{j-1} L_{ik} L_{jk})$

6. Simulation de chroniques de prix corrélés. On va utiliser la décomposition de Cholesky sur la matrice de variance. Soit deux chroniques $W_A(t)$ et $W_B(t)$ suivant une loi normale centrée réduite indépendante. On applique la transformation suivante : $Z = L \cdot W$. Les deux processus sont donc définis par : $S_i(t) = S_i(0) \exp \left((r_i - 0.5\sigma_i^2)\Delta t + \sigma_i Z_i(t)\sqrt{\Delta t} \right)$, $i = \{A; B\}$

7. Utilisation d'une notion en quantité – prix pour simuler le prix d'un portefeuille financier

Tests à mettre en place (non exhaustif) :

- Vérification des lois de probabilités (convergence en moyenne et écart type)
- Possibilité de répliquer des simulations si on met un seed
- Vérification du résultat de la décomposition de Cholesky
- Vérification que le résultat de deux variables parfaitement corrélées = 1 seule variable

Livrable attendu :

Il est attendu une console dans laquelle on peut saisir l'ensemble de ces informations (nombre d'actifs, volatilité, taux sans-risque, etc.). Le paramétrage peut aussi prendre sa source dans un fichier json.

Les résultats devront être stockés dans un fichier CSV et JSON.

Applications financières en C# (2024 – 2025)

L. Davoust

Projet 2 : Pricing d'option et de stratégie d'option

Objectif :

Création d'un pricer pour option vanille de type européenne (call et put) sans dividende, sans repo et avec un taux sans risque unique quel que soit la maturité. Cet outil servira aux traders et aux équipes de risque. Différentes méthodes de calcul pourront être implémenté.

Fonctionnalités :

- Utilisation de la méthode par formule de Black Scholes pour pricer un call et put européen (cf. les formules usuelles). Avec cette fonctionnalité, il est attendu de pouvoir extraire les différentes valeurs intermédiaires comme d_1 , d_2 , et les inputs.

$$c = S_0 \times N(d_1) - K \times \exp(-rT) \times N(d_2)$$

$$p = K \times \exp(-rT) \times (1 - N(d_2)) - S_0 \times (1 - N(d_1))$$

$$d_1 = \frac{\ln(S/K) + (r + (0.5\sigma^2)T)}{\sigma\sqrt{T}}, d_2 = d_1 - \sigma\sqrt{T}$$

$$N(.) = \text{cumulative normal probability}$$
- Utilisation de la méthode par arbre binomiaux pour pricer un call et un put. Dans cette approche-là uniquement, permettre le pricing d'option américaine. Ici, il est attendu de pouvoir exporter l'arbre binomial pour le prix du sous-jacent mais aussi du payoff.
- Utilisation de la méthode de Monte Carlo pour pricer un call et un put européen. Pour rappel, $S(t) = S(0) \times \exp\left(\left(r - \frac{\sigma^2}{2}\right)t + \sigma W(t)\right)$ où $S(0)$ est le prix initial, r est le taux de rendement, σ est la volatilité et $W(t)$ est le mouvement brownien simulé. On souhaite avoir la possibilité de fixer le seed pour pouvoir reproduire les calculs.
- Implémentation des sensibilités d'ordre 1 à l'aide de l'approche des différences finies :

$$\frac{dc}{dS} = \frac{P(S+\epsilon) - P(S-\epsilon)}{2\epsilon}$$
 pour le prix du sous-jacent, le temps qui passe et la volatilité.
- Pour les méthodes de Monte Carlo et par arbre binomial, indiquez la probabilité d'exécution du produit (ie quand le produit est dans la monnaie)
- Optimisation des performances : éviter de relancer plusieurs fois les mêmes calculs si les mêmes sont demandés.

Tests à mettre en place (non exhaustif) :

- Vérification des lois de probabilités des simulations de Monte Carlo (convergence en moyenne et écart type)

Applications financières en C# (2024 – 2025)

L. Davoust

- Vérification des résultats en fonction des 3 méthodes
- Vérification du résultat de la décomposition de Cholesky
- Vérification que le résultat de deux variables parfaitement corrélées = 1 seule variable
- Calcul suivant : $S_0 = K = 10$, $\sigma = 30\%$, $r = 3\%$, $T = 1 \rightarrow c = 1.328$, $p = 1.033$
- Vérifier la relation call - put : $c + K \times \exp(-rT) = p + S_0$

Livrable attendu :

1. Il est attendu une console dans laquelle on peut saisir l'ensemble de ces informations et avoir les résultats directement dans la console.
2. Il est attendu aussi de pouvoir calculer en « batch » ces calculs à l'aide d'un fichier input en CSV ou JSON. Les résultats seront alors exportés en CSV ou JSON.
3. Facultatif : Avoir la possibilité de lancer les calculs directement depuis Excel.

Applications financières en C# (2024 – 2025)

L. Davoust