MASTER 272
INGÉNIERIE ÉCONOMIQUE ET FINANCIÈRE
UNIVERSITÉ PARIS DAUPHINE

Applications financières en C# (2025 – 2026)

L. Davoust

SOMMAIRE

MASTER 272
INGÉNIERIE ÉCONOMIQUE ET FINANCIÈRE
UNIVERSITÉ PARIS DAUPHINE

Applications financières en C# (2025 – 2026)

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

- 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

Ecrire dans une console les résultats de la valeur actualisée pour 1€ pour les maturités allant de 1% à 10%, pour les maturités allant de 1 an à 20 ans.
La valeur associée sera donc $(1 + r)^{-T}$.

1. On s'attend à un format du type :

Taux en % ; Maturité en année ; Valeur actualisée

2. On s'attend maintenant à ce que le résultat soit restitué dans des dictionnaires imbriqués de taux -> maturité -> valeur

Exercice 3 : Suite de Fibonacci

La suite de Fibonnacci s'ecrit comme suit :
$u(0) = 0,\ u(1) = 1,$
$u(n) = u(n-1) + u(n-2)\ \forall n \geq 2$

1. A l'aide d'une boucle for, calculez u(30).
2. A refaire avec une fonction récursive.

Exercice 4 : Applications LINQ

1. A partir de la liste suivante [10, 9, 8, 7, 6], récupérez une nouvelle liste en utilisant LINQ renvoyant uniquement pour les multiples de 2, leur valeur au carré.
2. A partir de cette même liste [10, 9, 8, 7, 6], regroupez par si élément pair ou non, et renvoyez sous la forme d'un dictionnaire : true/false → moyenne(x*x)

Exercice 5 : 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). *Pas de contrainte sur l'aspect Id.*
2. Une classe abstraite Compte va exister et contenir les informations suivantes
   - Utilisateur,
   - Solde,
   - Déposer : ajout d'un montant dans le solde,

- Retirer : soustraction d'un montant dans le solde.
- Mettre en place un audit (Date + Mouvement + Type : enum) sur le compte : Liste d'une nouvelle classe ?

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 : notamment ajout d'une méthode « CalculInteret » qui ajoute le montant d'intérêt (ne pas oublier d'ajouter un élément dans l'audit …).
5. Extension du dépôt et du retrait par devise : CompteCourantDevise qui étend le CompteCourant avec la notion de devise, la valeur en devise et le taux de change appliqué.
6. Optionnel : extension du dépôt et du retrait en ajoutant des moyens de paiement (Cash, Chèque, CB) : utilisation du enum et d'une notion de frais bancaire en cas de réception avec la création d'au moins une classe MoyenPaiement
7. Mise en place de virement entre Compte.
8. Mise en place de tests unitaires pour vérifier le fonctionnement des différents processus (retrait, dépôt, virement, paiement)
9. Utilisation du pattern Manager pour gérer les comptes bancaires et les utilisateurs avec une sauvegarde des données en JSON (et un chargement en JSON).

Exercice 6 : Récupération de données financières depuis YahooFinance & Trading algorithmique

Partie 1 :

1. A partir du site Yahoo Finance, récupérer la série de prix de Apple : https://query2.finance.yahoo.com/v8/finance/chart/AAPL?period1=1699549200&period2=1731319200&interval=1d
2. Automatisez ce processus en fonction du code fourni par l'utilisateur.
3. Optimisez cette fonction en utilisant async/await pour lancer plusieurs requêtes simultanément.

Partie 2 :

1. 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, $\bar{x_t} = \sum_i \alpha(1-\alpha)^n x_{t-i} = \alpha(x_i + (1-\alpha)x_{i-1} + (1-\alpha)^2 x_{i-2} + \cdots) = \bar{x}_{i-1}(1-\alpha) + x_i\alpha$
   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.

2. Backtester cette stratégie : vous calculerez les indicateurs suivants : la volatilité, la performance totale, annualisée et la performance mois à mois.
3. Vous comparerez cette stratégie par rapport à une stratégie de Buy & Hold.
4. 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)
5. Optimisez cette stratégie entre deux dates en ex-ante (par exemple avant 1 janvier 2020) puis testez la stratégie sur la période restante.
6. Exportez les résultats dans Excel (ce fichier sera un fichier « *template* » où l'on pourra avoir des graphiques etc.)

Exercice (Optionnel) : Fonction factorielle

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

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

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 (Optionnel) : 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=3, puis n=5, puis n=7.

Exercice (Optionnel) : Distance euclidienne

La distance euclidienne entre deux vecteurs se calcule comme suit :

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

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 (Optionnel) : Manipulation des taux d'intérêt

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

Applications financières en C# (2025 – 2026)

L. Davoust

Mettre en place le pricing en fonction des conventions suivantes :

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

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

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

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

- (CC) Composition en temps continue :

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

- (CL) Composition linéaire :

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

Ajoutez une définition dynamique de la maturité en années à partir de date. Les conventions à utiliser sont les suivantes (comparable à la fonction YearFrac en Excel):

- (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

Applications financières en C# (2025 – 2026)

L. Davoust

PROJETS D'APPLICATION

Projet : 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 :

1. 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 d1, d2, et les inputs.
   $c = S_0 \times N(d_1) - K \times \exp(-rT) \times N(d_2))$
   $p = K \times \exp(-rT) \times \big(1 - N(d_2)\big) - S_0 \times \big(1 - N(d_1)\big)$
   $d_1 = \dfrac{\ln\left({}^{S}/_{K}\right) + (r + (0.5\sigma^2)T}{\sigma\sqrt{T}}$, $d_2 = d_1 - \sigma\sqrt{T}$
   $N(.) =$ cumulative normal probability

2. 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.

3. 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.

4. 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.

5. 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)

6. 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)
- 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. Mettre en place des tests unitaires.

Projet (Optionnel) : 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 u1 et u2 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

$W(t_{i+1}) = \sqrt{\Delta t}.Z_i$ 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 corréler deux séries indépendantes. La décomposition de Cholesky est une méthode qui décompose une 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 $A = L.L^T$.

   Algorithme de la décomposition :
   - La matrice A doit être symétrique ($A = A^T$), à vérifier, et définie positive ($x^T.A.x > 0$ 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 $L_{ij} = \frac{1}{L_{jj}}\left(A_{ij} - \sum_{k=0}^{j-1} L_{ik}L_{jk}\right)$

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 : $Z = L.W$. Les deux processus sont donc définis par : $S_i(t) = S_i(0)\exp\left(\left(r_i - 0.5\sigma_i^2\right)\Delta t + \sigma_i Z_i(t)\sqrt{\Delta t}\right)$, $i = \{A; B\}$

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

Applications financières en C# (2025 – 2026)

L. Davoust

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.

PROJET FINAL D'EVALUATION

Sujet

- Travail par groupe de 2 à 3 étudiants maximum (le nombre de personne impacte la notation).
- Le choix du sujet est libre. Cependant, il doit porter sur un sujet de la finance de marché (Buy Side ou Sell Side) et doit aboutir à un développement 100% C# en appliquant les principes de **la Programmation Orienté Objet**.
- Le plagiat est interdit dans le cadre de ce projet, ainsi tout projet copié / collé (même partiellement) depuis un GitHub / Internet sera sanctionné par un 0 et reporté à l'administration.
- Le livrable doit à minima être constitué
    - code utilisé pour le projet
    - cahier des charges
    - note explicative fonctionnelle et technique
- Le formateur est joignable soit par LinkedIn soit par mail : laurent.davoust@phit-formation.com
- Le projet peut utiliser n'importe quelle librairie C# à partir du moment que cette librairie profite à un problème plus global. (Par exemple, on peut utiliser CSVHelper pour la gestion des fichiers CSV).

Etapes du projet

1. Constitution du groupe et proposition du sujet.

2. Constitution du cahier des charges et présentation orale des projets souhaités (5 minutes maximum).
   Date limite : 5ème séance.

3. Projet à rendre.
   Date limite : 3 semaines après la dernière séance.

Cahier des charges

Le cahier des charges doit comprendre à minima les parties suivantes :
- Introduction au problème posé : description, objectifs, contexte, limites générales du projet, utilisateurs et sources théoriques.
- Expression fonctionnelle du besoin : liste de toutes les fonctionnalités (nom, description, objectifs, critère, contraintes pour chaque fonctionnalité)

- Solution proposée pour répondre au besoin (donnée, modèle mathématique, structure de développement). Cette partie peut être directement présent dans une documentation technique.
- Eléments annexes : budget/délais (en jour de travail théorique vs effectif), livrables attendus, documentation fonctionnelle pour l'utilisation de l'outil, documentation technique des modèles utilisés et de la structure de code, gestion de projet (et répartition des différentes tâches), documents d'analyse des résultats.

Exemples de projet

Cette partie a pour objectif de proposer des sujets types (mais non exhaustif) :

- Calculs et backtests
- Value at Risk selon différentes méthodologies,
- Calibration et implémentation de stress scénario à un portefeuille,
- Création d'un outil de valorisation de produits structurés sur mesure,
- Création d'un outil d'optimisation de portefeuille,
- Création d'un outil de trading algorithmique (pair trading avec test de cointégration, stratégie d'analyse technique avec backtest et optimisation des paramètres),
- Création d'un outil qui optimise le choix du portefeuille pour un produit dérivé sur Worst Of,
- Calculateur du ratio de couverture d'un portefeuille de dérivés (ou non) par rapport à un indice de marché ou de dérivés (réplication statique ou dynamique),
- Logiciel de passage d'ordre
- Logiciel de gestion de portefeuille