Plaquette 5 – Problèmes de synthèse : algorithmes du programme et débogage

Algorithmique et Python — Première spé maths

Corrections détaillées

Boîte à outils

Les algorithmes à connaître en première.

Algorithme	Principe	Coût
seuil	boucle *while* jusqu'au dépassement	$O(n)$
dichotomie	couper l'intervalle en deux	$O(\log n)$
somme de termes	accumulateur dans une boucle *for*	$O(n)$
recherche d'extremum	balayage avec mémorisation	$O(n)$
simulation	*random* répété, puis fréquence	$O(N)$
méthode d'Euler	$y_{k+1} = y_k + hf(x_k, y_k)$	$O(n)$

Terminaison. On exhibe une quantité qui **décroît strictement** vers une borne (l'amplitude en dichotomie, l'écart au seuil pour une suite monotone), ou l'on invoque la finitude d'une boucle *for*.

Correction. On énonce un **invariant** : une propriété vraie avant la boucle, préservée à chaque tour, et qui donne le résultat à la sortie.

Exemple d'invariant — pour la recherche du maximum : « après k tours, m est le maximum des k premiers éléments ».

Méthode d'Euler pour $y' = f(x, y)$ avec $y(x_0) = y_0$: on avance de h en suivant la tangente,

$$y_{k+1} = y_k + h\,f(x_k, y_k)$$

```
def euler(x0, y0, h, n):
    x, y = x0, y0
    for k in range(n):
        y = y + h * f(x, y)
        x = x + h
    return y
```

L'erreur est d'ordre h : diviser le pas par 10 divise l'erreur par 10.

Débogage : les cinq suspects.

1. Accumulateur initialisé **dans** la boucle. 2. Bornes de *range* décalées d'une unité. 3. Inégalité stricte au lieu de large (ou l'inverse). 4. `//` et `/` confondus. 5. Boucle *while* dont la condition ne peut jamais devenir fausse.

Correction de l'exercice 1 – Méthode d'Euler

La méthode d'Euler remplace la courbe par sa tangente sur chaque petit pas : $y(x+h) \approx y(x) + h\,y'(x)$. Ici $y' = 2y$, donc

$$y_{k+1} = y_k + 0{,}25 \times 2y_k = 1{,}5\,y_k$$

En partant de $y_0 = 1$, on obtient 1,5, 2,25, 3,375, puis $y_4 = 1{,}5^4 = 5{,}0625$ au bout de

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

quatre pas, c'est-à-dire en $x = 1$.

La solution exacte est $y(x) = e^{2x}$, donc $y(1) = e^2 \approx 7{,}389$. L'erreur vaut environ 2,33, soit 31,5 % : le pas est trop grand. La méthode **sous-estime** ici, car la courbe est au-dessus de ses tangentes.

En réduisant le pas, l'approximation s'améliore : avec $h = 0{,}1$, on obtient $1{,}2^{10} \approx 6{,}19$, et avec $h = 0{,}01$, $1{,}02^{100} \approx 7{,}24$. La méthode d'Euler donne $y(1) \approx 5{,}0625$, contre $e^2 \approx 7{,}389$.

Réponse. $y(1) \approx 5{,}0625$ contre $e^2 \approx 7{,}389$ — erreur de 31,5 % avec $h = 0{,}25$.

Correction de l'exercice 2 – Déboguer un calcul de somme

Première erreur : *range(n)* commence à $k = 0$, et le premier tour calcule *1 / 0* : Python s'arrête sur une division par zéro.

Seconde erreur : *range(n)* s'arrête à $n - 1$, donc le terme $\frac{1}{n}$ manquerait.

```
def harmonique(n):
    s = 0
    for k in range(1, n + 1):
        s = s + 1 / k
    return s
```

On teste sur les plus petits cas, dont on connaît la valeur :

$$H_1 = 1 \qquad H_2 = 1 + \frac{1}{2} = 1{,}5 \qquad H_3 = 1 + \frac{1}{2} + \frac{1}{3} \approx 1{,}833$$

Tester d'abord $n = 1$ aurait révélé l'erreur immédiatement. Pour $n = 10$, la fonction corrigée renvoie environ 2,929. Il faut donc écrire **range(1, n + 1)**.

Réponse. *range(n)* provoque une division par zéro et s'arrête trop tôt : il faut *range(1, n + 1)*.

Correction de l'exercice 3 – Dichotomie avec compteur

On part de $[2\,;3]$, qui contient $\sqrt{7}$, car $2^2 = 4 < 7$ et $3^2 = 9 > 7$. À chaque étape, on garde la moitié qui contient encore la solution, et l'on compte les étapes.

```
a, b = 2.0, 3.0
```

```
n = 0
while b - a > 1e-4:
    m = (a + b) / 2
    if m*m < 7:
        a = m
    else:
        b = m
    n = n + 1
print(n, a, b)
```

Les premières étapes donnent $m = 2{,}5$ (car $6{,}25 < 7$, on garde $[2{,}5\,;3]$), puis $m = 2{,}75$ (car $7{,}5625 > 7$, on garde $[2{,}5\,;2{,}75]$). À chaque étape, on a toujours $a^2 < 7 \leq b^2$, donc $\sqrt{7}$ reste dans $[a\,;b]$. L'amplitude, divisée par 2 à chaque étape, vaut $\frac{1}{2^k}$ après k étapes :

$$\frac{1}{2^k} \leq 10^{-4} \Leftrightarrow 2^k \geq 10\,000$$

Comme $2^{13} = 8192$ et $2^{14} = 16\,384$, il faut 14 **étapes** ; l'encadrement final contient $\sqrt{7} \approx 2{,}64575$.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Réponse. 14 étapes, encadrant $\sqrt{7} \approx 2{,}64575$.

Correction de l'exercice 4 – Algorithme de seuil sur un modèle

Chaque année, le volume est multiplié par 0,92 puis augmenté de 600 m³. On boucle tant que le volume reste au moins égal à 9000 m³ :

```
v = 12000.0
```

```
n = 0
while v >= 9000:
    v = 0.92 * v + 600
    n = n + 1
print(n)
```

Terminaison. Le point fixe vérifie $0{,}92x + 600 = x$, soit $x = 7500$. Avec $w_n = v_n - 7500$, on a $w_{n+1} = 0{,}92\, w_n$ et $w_0 = 4500$, donc

$$v_n = 7500 + 4500 \times 0{,}92^n$$

La suite décroît vers $7500 < 9000$: elle passe forcément sous 9000 et la boucle s'arrête. Un seuil de 7000 ne serait au contraire jamais atteint.

On encadre : $v_{13} \approx 9022{,}1 \geq 9000$ et $v_{14} \approx 8900{,}4 < 9000$. Le volume passe sous 9000 m³ la 14^{e} **année**.

Réponse. La 14^{e} année ; terminaison car $v_n \to 7500 < 9000$ en décroissant.

Correction de l'exercice 5 – Invariant de boucle

Un invariant est une propriété vraie à chaque tour de boucle. Ici : **après le tour d'indice k, $p = k!$ et $s = 1! + 2! + \cdots + k!$.**

— Au premier tour, p devient $1 \times 1 = 1 = 1!$ et s devient 1 : la propriété est vraie pour $k = 1$.

— Si elle est vraie après le tour $k - 1$, alors au tour k, p devient $(k-1)! \times k = k!$, puis s reçoit ce nouveau terme : elle reste vraie après le tour k.

À la sortie, après $k = 5$, on a donc $s = 1! + 2! + 3! + 4! + 5!$. Les valeurs de p sont 1, 2, 6, 24, 120, et

$$s = 1 + 2 + 6 + 24 + 120 = 153$$

L'ordre des instructions compte : en ajoutant p à s avant de le multiplier par k, on calculerait $0! + 1! + \cdots + 4! = 34$. Le script affiche 153.

Réponse. Invariant : $p = k!$ et $s = \sum_{i=1}^{k} i!$; le script affiche 153.

Correction de l'exercice 6 – Déboguer une simulation

Première erreur : le test *random() < 0.6* réussit avec la probabilité 0,6, alors que le paramètre est 0,4. Il faut *random() < 0.4*.

Seconde erreur : *s > 2* compte les échantillons où $X \geq 3$. Pour $P(X \geq 2)$, il faut *s >= 2*.

La valeur cherchée se calcule par l'événement contraire :

$$P(X \geq 2) = 1 - P(X = 0) - P(X = 1) = 1 - 0{,}6^5 - 5 \times 0{,}4 \times 0{,}6^4$$

soit $1 - 0{,}07776 - 0{,}2592 = 0{,}66304$.

Le script erroné estime $P(Y \geq 3)$ pour Y de paramètres 5 et 0,6, qui vaut $0{,}3456 + 0{,}2592 + 0{,}07776 = 0{,}68256$. Les deux valeurs sont proches : les deux erreurs se masquent presque. D'où la règle de débogage : tester chaque partie séparément, par exemple vérifier que la fréquence d'un succès vaut bien 0,4. La fréquence corrigée doit

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

être proche de 0,663.

Réponse. *0.6* au lieu de *0.4*, et *s > 2* au lieu de *s >= 2* ; la vraie valeur est 0,66304.

Correction de l’exercice 7 – Somme de termes d’une suite

On ajoute chaque terme **avant** de calculer le suivant, pour sommer u_0 à u_{20}, soit 21 termes :

```
s = 0
```

```
u = 3.0
for k in range(21):
    s = s + u
    u = 0.8 * u + 1
print(s)
```

Pour vérifier par le calcul, on cherche une forme explicite. Le point fixe vérifie $0{,}8x + 1 = x$, soit $x = 5$. Avec $v_n = u_n - 5$, on a $v_{n+1} = 0{,}8\,v_n$ et $v_0 = -2$, donc $u_n = 5 - 2 \times 0{,}8^n$. La somme se sépare en une somme constante et une somme géométrique :

$$S = 21 \times 5 - 2 \times \frac{1 - 0{,}8^{21}}{1 - 0{,}8} \approx 105 - 9{,}908 = 95{,}092$$

Contrôle sur les cinq premiers termes : $3 + 3{,}4 + 3{,}72 + 3{,}976 + 4{,}1808 = 18{,}2768$, et la formule donne $25 - 2 \times \dfrac{1 - 0{,}8^5}{0{,}2} = 18{,}2768$. On obtient $S \approx 95{,}092$.

Réponse. $S \approx 95{,}092$ (avec $u_n = 5 - 2 \times 0{,}8^n$).

Correction de l’exercice 8 – Balayage pour un extremum

On évalue f en $x = 0$, 0,01, …, 2, soit 201 points, et l’on garde l’abscisse qui donne la plus grande valeur :

```
def f(x):
```

```
    return x * (4 - x*x)

xm = 0
for k in range(201):
    x = k * 0.01
    if f(x) > f(xm):
        xm = x
print(xm, f(xm))
```

Le balayage donne $x = 1{,}15$ et $f(1{,}15) \approx 3{,}07913$.

Calcul exact : $f(x) = 4x - x^3$, donc $f'(x) = 4 - 3x^2$, qui s’annule sur $[0\,;2]$ en $x = \dfrac{2}{\sqrt{3}} \approx 1{,}1547$, positive avant et négative après. Le maximum vaut

$$f\left(\frac{2}{\sqrt{3}}\right) = \frac{2}{\sqrt{3}} \times \frac{8}{3} = \frac{16}{3\sqrt{3}} \approx 3{,}07920$$

L’abscisse trouvée est à 0,005 près, mais la valeur du maximum est presque exacte : près d’un maximum, la courbe est presque horizontale. Résultats : $x \approx 1{,}15$ et un maximum d’environ 3,0792.

Réponse. Balayage : $x = 1{,}15$ et $f \approx 3{,}07913$; exact : $x = \dfrac{2}{\sqrt{3}} \approx 1{,}1547$ et $f = \dfrac{16}{3\sqrt{3}} \approx 3{,}07920$.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s’expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Correction de l’exercice 9 – Algorithme de dichotomie générique

La fonction ne suppose pas que f est croissante : elle teste le **signe** du produit $f(a) \times f(m)$. S’il est négatif ou nul, f change de signe entre a et m, et la racine est dans la moitié gauche ; sinon, elle est dans la moitié droite.

```
def dicho(f, a, b, eps):
    while b - a > eps:
        m = (a + b) / 2
        if f(a) * f(m) <= 0:
            b = m
        else:
            a = m
    return (a + b) / 2
```

Pour $f(x) = x^3 + x - 3$ sur $[1\,;2]$: $f(1) = -1 < 0$ et $f(2) = 7 > 0$, et $f'(x) = 3x^2 + 1 > 0$, donc f est strictement croissante et s’annule une seule fois. L’appel *dicho(f, 1, 2, 1e-6)* renvoie

$$\alpha \approx 1{,}21341$$

Il faut que $f(a)$ et $f(b)$ soient de signes contraires ; sinon la fonction renvoie une valeur sans signification. **Contrôle** : $1{,}21341^3 + 1{,}21341 \approx 3{,}0000$.

Réponse. Dichotomie générique par le test de signe ; racine $\alpha \approx 1{,}21341$.

Correction de l’exercice 10 – Coût comparé de deux algorithmes

Par une boucle, on multiplie par 1,02 tant que le terme ne dépasse pas 100 :

```
u = 1.0
n = 0
while u <= 100:
    u = 1.02 * u
    n = n + 1
print(n)
```

Elle fait 233 tours, autant que le résultat.

Par dichotomie sur n, on encadre le rang cherché entre $a = 0$, où $1{,}02^0 = 1 \le 100$, et $b = 1000$, où $1{,}02^{1000} > 100$. À chaque étape, on calcule $1{,}02^m$ pour l’entier m du milieu et l’on garde la moitié qui contient le changement. L’intervalle de 1000 entiers est divisé par 2 à chaque étape ; il suffit d’environ 10 étapes, car

$$2^{10} = 1024 > 1000$$

On vérifie le résultat : $1{,}02^{232} \approx 98{,}91 \le 100$ et $1{,}02^{233} \approx 100{,}89 > 100$. Les deux méthodes donnent $n = 233$; la dichotomie demande bien moins d’étapes, et l’écart grandit avec le seuil.

Réponse. $n = 233$; la dichotomie demande une dizaine d’étapes, la boucle 233 tours.

Correction de l’exercice 11 – Déboguer une méthode d’Euler

L’équation est $y' = -y$: la pente vaut $-y$, et la méthode d’Euler s’écrit $y_{k+1} = y_k + h \times (-y_k)$. Le script ajoute $h \times y$ au lieu de le retrancher : il faut écrire `y = y - h * y`.

Le script erroné affiche $1{,}5^4 = 5{,}0625$: une valeur qui **augmente**, alors que la solution de $y' = -y$ partant de 1 est positive et décroissante. Ce résultat absurde aurait dû alerter.

Corrigé, le script calcule $y_{k+1} = 0{,}5\,y_k$: on obtient 0,5, 0,25, 0,125, puis

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s’expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

$$y(2) \approx 0{,}5^4 = 0{,}0625$$

La solution exacte est e^{-x}, et $e^{-2} \approx 0{,}135$: avec un pas aussi grand, l'erreur dépasse la moitié de la valeur. Avec $h = 0{,}1$, on obtiendrait $0{,}9^{20} \approx 0{,}122$, bien plus proche. La valeur correcte pour $h = 0{,}5$ est 0,0625.

Réponse. Il faut *y = y - h * y* ; on obtient alors $y(2) \approx 0{,}0625$ contre $e^{-2} \approx 0{,}1353$.

Correction de l'exercice 12 – Simulation et intervalle de fluctuation

Pour chaque échantillon, on compte les piles avec *randint(0, 1)*, qui vaut 1 pour pile, puis on teste si la fréquence sort de [0,4 ; 0,6].

```
from random import randint

c = 0
for k in range(200):
    s = 0
    for i in range(100):
        s = s + randint(0, 1)
    f = s / 100
    if f < 0.4 or f > 0.6:
        c = c + 1
print(c)
```

La fréquence sort de l'intervalle quand le nombre de piles X vaut au plus 39 ou au moins 61. X suit la loi binomiale de paramètres 100 et 0,5 ; à la calculatrice, $P(X \leq 39) \approx 0{,}0176$, et autant pour $P(X \geq 61)$ par symétrie. La probabilité de sortie vaut environ 0,035, d'où

$$200 \times 0{,}035 \approx 7$$

On attend environ 7 échantillons hors de l'intervalle, en général entre 2 et 12 : l'intervalle [0,4 ; 0,6] contient donc un peu plus de 95 % des fréquences.

Réponse. Environ 7 échantillons (et non 10), typiquement entre 2 et 12.

Correction de l'exercice 13 – Calcul d'une somme double

Deux boucles imbriquées parcourent les $5 \times 5 = 25$ couples (i, j) et accumulent les produits :

```
s = 0
for i in range(1, 6):
    for j in range(1, 6):
        s = s + i*j
print(s)
```

Pour une valeur de i fixée, la somme intérieure vaut $i \times (1 + 2 + 3 + 4 + 5) = 15i$. On peut donc factoriser :

$$S = 15 \times (1 + 2 + 3 + 4 + 5) = 15 \times 15 = 225$$

Contrôle : les sommes intérieures valent 15, 30, 45, 60 et 75, de total 225. Cette valeur est aussi la somme des cubes, $1 + 8 + 27 + 64 + 125 = 225$: un exemple de l'identité $1^3 + \cdots + n^3 = (1 + \cdots + n)^2$. Le script affiche 225.

Réponse. $225 = 15^2$ (la somme double se factorise).

Correction de l'exercice 14 – Recherche dans une liste triée

Dans une liste triée, on compare x à l'élément du milieu : s'il est plus grand, x ne peut être que dans la moitié droite ; sinon, dans la moitié gauche. On élimine ainsi la moitié

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

des candidats à chaque étape.

```
def recherche(L, x):
    a, b = 0, len(L) - 1
    while a <= b:
        m = (a + b) // 2
        if L[m] == x:
            return m
        if L[m] < x:
            a = m + 1
        else:
            b = m - 1
    return -1
```

L'indice du milieu doit être entier, d'où la division entière *//*. Les mises à jour *a = m + 1* et *b = m - 1* font rétrécir l'intervalle à chaque tour, ce qui garantit l'arrêt ; si x est absent, la fonction renvoie -1.

Pour $n = 10^6$, il suffit d'environ 20 comparaisons, car $2^{20} = 1\,048\,576 > 10^6$. Une recherche linéaire en demande jusqu'à 10^6, soit

$$\frac{10^6}{20} = 50\,000$$

fois plus.

La dichotomie n'est valable que si la liste est **triée** ; sinon elle peut conclure à tort que x est absent.

Réponse. 20 comparaisons contre 10^6 — un gain d'un facteur 50 000 (liste triée obligatoire).

Correction de l'exercice 15 – Détecter une boucle infinie

Une boucle *while* se termine quand la suite calculée finit par rendre la condition fausse : on regarde donc vers quoi tend la suite.

a) **Oui** : $u_n = 1{,}1^n$ dépasse tout seuil, et $1{,}1^{72} \approx 956 < 1000 < 1{,}1^{73} \approx 1051$: 73 tours.

b) **Oui** : u diminue de 0,0001 à chaque tour, il passe sous 0,001 après $\dfrac{1 - 0{,}001}{0{,}0001} = 9990$ tours.

c) **Oui** : le point fixe vérifie $\dfrac{x}{2} + 0{,}4 = x$, soit $x = 0{,}8 < 1$. La suite décroît vers 0,8 et passe sous 1 ; on trouve $u_5 \approx 1{,}0875$ et $u_6 \approx 0{,}944$, donc 6 tours.

d) Mathématiquement, $\dfrac{1}{2^n}$ n'est **jamais** nul : la boucle devrait tourner indéfiniment. En pratique, elle s'arrête, car les nombres en machine ont une précision limitée et finissent par être arrondis à 0, après environ

1075 divisions

Règle : ne jamais tester l'égalité exacte d'un nombre décimal avec *!=* ; on écrit plutôt *while u > 1e-15*. Les quatre boucles se terminent, mais d) seulement grâce à une limite de la machine.

Réponse. Les quatre se terminent, mais d) seulement par sous-dépassement des flottants — à éviter.

Correction de l'exercice 16 – Synthèse : étude algorithmique complète

a) On applique la relation : $u_1 = 5 + 0{,}3 = 5{,}3$, $u_2 \approx 3{,}216$, $u_3 \approx 2{,}541$ et $u_4 \approx 2{,}451$.

b) Si la suite converge vers $\ell > 0$, alors $\ell = \dfrac{\ell}{2} + \dfrac{3}{\ell}$, soit $\dfrac{\ell}{2} = \dfrac{3}{\ell}$ et $\ell^2 = 6$:

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

$$\ell = \sqrt{6} \approx 2{,}4494897$$

On peut montrer que $u_{n+1} - \sqrt{6} = \dfrac{\left(u_n - \sqrt{6}\right)^2}{2u_n} \geq 0$: à partir de u_1, les termes restent au-dessus de $\sqrt{6}$ en décroissant. C'est la méthode de Newton appliquée à $x^2 - 6 = 0$.

c) On répète la relation tant que $\left|u_n^2 - 6\right| \geq 10^{-8}$:

```
u = 10.0
```

```
n = 0
while abs(u*u - 6) >= 1e-8:
    u = u/2 + 3/u
    n = n + 1
print(n, u)
```

Les écarts $\left|u_n^2 - 6\right|$ valent environ 22,1 ; 4,34 ; 0,456 ; 0,0080 ; $2{,}7 \times 10^{-6}$, puis $3{,}0 \times 10^{-13} < 10^{-8}$. Le script effectue **six** tours et affiche $u \approx 2{,}4494897$.

Réponse. a) 5,3 ; 3,216 ; 2,541 ; 2,451 · b) limite $\sqrt{6}$ (méthode de Newton) · c) six tours.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).