

Corrigé Geipi Polytech 2023 : QCM, Mathématiques et NSI

Proposition de corrigé — non officielle. Concours Geipi Polytech du 3 mai 2023, trois heures. QCM de mathématiques obligatoire et choix de deux spécialités : cette proposition traite Mathématiques et NSI, avec les feuilles réponses officielles.

[Sujet officiel](#) · [Annales de ce concours ou examen](#)

Périmètre : QCM commun, Mathématiques et NSI

Le concours du **3 mai 2023** demande le QCM de mathématiques et **deux spécialités au choix**. Cette proposition traite le parcours **Mathématiques–NSI**, avec les deux exercices de chacune de ces spécialités. Les options Physique-Chimie, SVT et Sciences de l'ingénieur ne sont pas corrigées ici. Utilisez aussi les [documents réponses officiels](#), nécessaires pour les cases et les programmes à compléter.

1. QCM de mathématiques : les neuf exercices

V signifie vrai ; **F**, faux. Plusieurs affirmations d'un même exercice peuvent être vraies. Une erreur est pénalisée ; une absence de réponse ne retire pas de point. Les explications suivantes permettent de comprendre les choix, même si la grille ne demande pas de justification.

Exercice I — logarithme

Pour $f(x) = \ln(x^2 + 1)$, **A : V ; B : F ; C : F ; D : V**.

- **A** : $x^2 + 1$ est strictement positif pour tout réel, donc le domaine est $\mathbb{R}$.
- **B** : $f'(x) = 2x/(x^2 + 1)$, d'où $f'(0) = 0$, et non 1.
- **C** : $x^2 + 1 > 1$ si $x < 0$; son logarithme est donc positif, pas négatif.
- **D** : lorsque x tend vers $-\infty$, $x^2 + 1$ tend vers $+\infty$, puis son logarithme aussi.

Exercice II — courbe et tangente

A : F ; B : V ; C : F.

- **A** : $g(1) = 0$ place le point $(1 ; 0)$ sur l'axe des **abscisses**. Un point de l'axe des ordonnées doit avoir une abscisse nulle.
- **B** : en $x = 1$, la tangente est $y = g'(1)(x - 1) + g(1) = 3(x - 1) + 2 = 3x - 1$.
- **C** : si $g'' \geq 0$, la fonction est convexe ; sa courbe se situe **au-dessus** de ses tangentes. Une fonction comme $g(x) = x^2$ donne immédiatement un contre-exemple à l'affirmation « en dessous ».

Exercice III — identités et équation

A : F ; B : F ; C : F ; D : F.

- **A** : $e^{(3x+1)} = e^{(3x)} \times e$, et non $e^{(3x)} + e$. À $x = 0$, les deux expressions proposées valent e et $1 + e$.
- **B** : une différence de logarithmes peut devenir le logarithme d'un quotient ; un **quotient** de logarithmes ne suit pas cette règle. À $x = 1$, $\ln(x^2)/\ln(x^2 + 4)$ vaut 0, tandis que $\ln[x^2/(x^2 + 4)]$ vaut $\ln(1/5)$.
- **C** : $2 \ln(e^{\sqrt{x}}) = 2\sqrt{x}$. À $x = 1$, cette expression vaut 2 et non 1. L'égalité demandée n'est donc pas vraie pour tout x positif.
- **D** : poser $t = e^x > 0$. L'équation devient $t^2 - 3t + 2 = (t - 1)(t - 2) = 0$. Ses solutions sont **$x = 0$ et $x = \ln 2$** ; la proposition oublie la seconde.

Exercice IV — quotient d'exponentielles

Pour $h(x) = (e^{(2x)} + 1)/(e^x + 1)$, A : F ; B : V ; C : V ; D : F.

- **A** : en $+\infty$, $h(x)$ est équivalent à e^x , donc tend vers $+\infty$.
- **B** : $h(0) = 2/2 = 1$.
- **C** : en $-\infty$, les exponentielles tendent vers 0 ; la limite vaut 1.
- **D** : la règle du quotient donne $h'(x) = [e^{(3x)} + 2e^{(2x)} - e^x]/(e^x + 1)^2$. Le dénominateur proposé, $e^{(2x)} + 1$, omet le terme $2e^x$.

Exercice V — suite géométrique

A : V ; B : F ; C : V.

La raison vaut $1/2$ et $u_2 = 1$: **$u_0 = 4$** , donc **$u_n = 4(1/2)^n$** . La suite converge vers 0. L'affirmation B oublie le facteur 4. Pour $n \geq 1$, la somme géométrique donne **$u_1 + \dots + u_n = 4[1 - (1/2)^n]$** .

Exercice VI — somme télescopique

A : F ; B : F ; C : F ; D : V.

Le premier terme calculé est $v_1 = 0 + 1/(1 \times 2) = 1/2$, pas $1/6$. Chaque incrément est positif : la suite est croissante. Comme $1/[(k+1)(k+2)] = 1/(k+1) - 1/(k+2)$, la somme se simplifie : **$v_n = 1 - 1/(n+1) = n/(n+1)$** . Elle tend vers 1, et non 0.

Exercice VII — probabilités conditionnelles

A : V ; B : V ; C : F ; D : F.

- **A** : $P_B(A)P(B)$ et $P_A(B)P(A)$ valent tous deux $P(A \cap B)$.
- **B** : $P_A(A) = P(A \cap A)/P(A) = 1$, puisque $P(A) > 0$.
- **C** : changer la condition de A à son complément ne remplace pas automatiquement une probabilité par son complément. Si A et B sont indépendants et $P(B) = 1/4$, les deux probabilités conditionnelles valent $1/4$, alors que $1 - P_A(B)$ vaut $3/4$.

• **D** : la formule correcte est $P(B) = P_A(B)P(A) + P_{\bar{A}}(B)P(\bar{A})$. Les poids $P(A)$ et $P(\bar{A})$ manquent dans l'expression proposée.

Exercice VIII — loi binomiale

A : V ; B : V ; C : F.

Pour X de loi $B(10 ; 0,2)$, X est entier. Ainsi $P(1 \leq X < 3) = P(X \leq 2) - P(X = 0)$. La probabilité $P(X > 1)$ est positive, puisque $P(X = 2) = 45 \times 0,2^2 \times 0,8^8 > 0$. Enfin, $P(X = 0) = 0,8^{10}$, et non $0,2^{10}$: les dix essais doivent être des échecs.

Exercice IX — géométrie plane

A : F ; B : V ; C : V ; D : V.

$A(-1 ; 1)$, $B(3 ; 4)$, $C(8 ; 3/2)$ donnent $AB = \sqrt{4^2 + 3^2} = 5$. Les coordonnées de A et B satisfont $3x - 4y + 7 = 0$. Le milieu de $[AB]$ est $M(1 ; 5/2)$; la droite $8x + 6y - 25 = 0$ passe par M et a pour vecteur normal $(4 ; 3)$, parallèle à AB : c'est la médiatrice. Enfin, $D(5 ; 11/2)$ appartient à (AB) et $CD = (-3 ; 4)$ est orthogonal à AB , car $-12 + 12 = 0$. D est bien le projeté orthogonal de C .

2. Mathématiques spécialité — exercice I : bons d'achat

I-1 à I-4 — arbre et récurrence

$a_1 = 1/5$. Sur l'arbre, les premières branches portent a_n et $1 - a_n$. Après A_n , les branches A_{n+1} et son complément portent **3/10 et 7/10**. Après $\bar{A}_n$, elles portent **1/10 et 9/10**.

$$P(A_{n+1} \cap A_n) = 3a_n/10 \text{ et } P(A_{n+1} \cap \bar{A}_n) = (1 - a_n)/10.$$

La partition par A_n et $\bar{A}_n$ donne $a_{n+1} = 3a_n/10 + (1 - a_n)/10 = a_n/5 + 1/10$.

I-5 et I-6 — suite auxiliaire

Pour $u_n = a_n - 1/8$, $u_1 = 1/5 - 1/8 = 3/40$. En substituant la récurrence, $u_{n+1} = (a_n - 1/8)/5 = u_n/5$. La suite est géométrique de raison **1/5**.

Pour $n \geq 1$, $u_n = (3/40)(1/5)^{n-1} = 3/(8 \times 5^n)$, puis $a_n = 1/8 + 3/(8 \times 5^n)$.

I-7 et I-8 — convergence et rang

Comme 5^n tend vers $+\infty$, a_n tend vers **1/8**. Le terme ajouté est toujours positif, donc $a_n > 1/8$.

Chercher $a_n - 1/8 \leq 10^{-5}$ revient à demander $5^n \geq 37\,500$, puis $n \geq \ln(37\,500)/\ln 5 \approx 6,54$. Le premier rang entier est donc $n_0 = 7$. À $n = 6$, l'écart vaut $2,4 \times 10^{-5}$; à $n = 7$, il vaut $4,8 \times 10^{-6}$. Le rang 6 ne suffit pas.

3. Mathématiques spécialité — exercice II : géométrie dans l'espace

II-1 à II-3 — racines, milieux et paramètre

L'équation $X^2 - 4X + 2 = 0$ a pour discriminant 8 et pour solutions **$2 - \sqrt{2}$ et $2 + \sqrt{2}$** .

Les milieux sont $J(2 ; 1 ; -\sqrt{3})$ et $L(2 ; 1 ; \sqrt{3})$. Comme $AI = (a ; 0 ; 0)$ et $AE = (4 ; 0 ; 0)$, $\lambda = a/4$. Lorsque a parcourt $[0 ; 4]$, I décrit le segment $[AE]$: choix A.

II-4 et II-5 — distances et orthogonalité

Les vecteurs $IJ = (2 - a ; 1 ; -\sqrt{3})$ et $IL = (2 - a ; 1 ; \sqrt{3})$ donnent $IJ^2 = IL^2 = (2 - a)^2 + 4$. Leurs longueurs sont donc égales.

Le produit scalaire vaut $(2 - a)^2 + 1 - 3 = a^2 - 4a + 2$: $m = 1$, $n = -4$, $p = 2$. Les vecteurs sont orthogonaux si et seulement si $a = 2 - \sqrt{2}$ ou $2 + \sqrt{2}$, deux valeurs situées dans $[0 ; 4]$.

II-6 — équation du plan

On prend désormais $a = 2 + \sqrt{2}$. Alors $IJ = (-\sqrt{2} ; 1 ; -\sqrt{3})$ et $IL = (-\sqrt{2} ; 1 ; \sqrt{3})$. Ces vecteurs ne sont pas colinéaires : leurs deuxièmes composantes imposeraient un rapport 1, incompatible avec leurs troisièmes composantes. Les trois points définissent donc un plan.

Le vecteur $n = (1 ; \sqrt{2} ; 0)$, non nul, est orthogonal à IJ et à IL : les deux produits scalaires valent $-\sqrt{2} + \sqrt{2} = 0$. Une équation du plan passant par I est $x + \sqrt{2}y - (2 + \sqrt{2}) = 0$.

II-7 à II-9 — intersection et carré

La droite (CG) admet la représentation $x = t$, $y = 2$, $z = 0$, t réel. Remplacer dans l'équation du plan donne $t = 2 - \sqrt{2}$, donc $K(2 - \sqrt{2} ; 2 ; 0)$.

Le quadrilatère **IJKL est un carré**. En effet, $JK = IL$ et $KL = -IJ$: c'est un parallélogramme. Les côtés IJ et IL sont de même longueur $\sqrt{6}$ et orthogonaux ; il a donc quatre côtés égaux et quatre angles droits.

4. NSI — exercice I : réseau de transports

I-1 — représentation des deux lignes

```
ligne1 = {"num": 1, "arrets": ["gare", "piscine", "lycée"],
          "circulaire": True}
ligne2 = {"num": 2, "arrets": ["mairie", "parc", "stade", "piscine"],
          "circulaire": False}
```

L'ordre des arrêts reproduit le schéma. Une ligne circulaire dessert tous ses arrêts sans demi-tour ; les lignes non circulaires font l'aller-retour.

I-2 — trajet direct

```
def direct(reseau, depart, arrivee):
    for ligne in reseau:
        nb = 0
        for station in ligne["arrets"]:
            if (station == depart) or (station == arrivee):
                nb = nb + (station == depart) + (station == arrivee)
        if nb == 2:
            return True
    return False
```

Une même ligne doit contenir les deux extrémités. Les noms d'arrêts ne sont pas répétés dans une ligne. Pour deux stations distinctes, l'incrément usuel $nb + 1$ convient. Le code proposé compte séparément les deux conditions : en Python, un booléen vrai vaut 1 dans cette addition. Il traite aussi

depart == arrivee de façon cohérente avec la fonction suivante : une station présente est reliée à elle-même sans changement de ligne. Si l'on impose dans le contrat deux stations distinctes, la version nb + 1 suffit.

I-3 — dictionnaire des lignes par station

```
def lister_lignes(reseau):
    dico = {}
    for ligne in reseau:
        for station in ligne["arrets"]:
            if station in dico:
                dico[station].append(ligne["num"])
            else:
                dico[station] = [ligne["num"]]
    return dico
```

Pour les deux lignes ci-dessus, piscine est associée à [1, 2], gare et lycée à [1], mairie, parc et stade à [2]. Chaque station rassemble les numéros des lignes qui la desservent.

I-4 — trajet direct avec le dictionnaire

```
def direct_bis(dico, depart, arrivee):
    for ligne in dico.get(depart, []):
        if ligne in dico.get(arrivee, []):
            return True
    return False
```

Le test recherche un numéro de ligne commun. Le sujet prévoit des stations du réseau, pour lesquelles dico[depart] et dico[arrivee] conviennent aussi. get(..., []) renvoie ici une liste vide pour un nom inconnu et évite une erreur de clé.

5. NSI — exercice II : regrouper des images

II-1 — similarité cosinus

Les quatre remplacements sont ① produit_scalaire ; ② $x[i] * y[i]$; ③ $x[i] ** 2$; ④ $y[i] ** 2$.

```
from math import sqrt

def simcos(x, y):
    produit_scalaire = 0
    sumX2 = 0
    sumY2 = 0
    for i in range(0, len(x)):
        produit_scalaire = produit_scalaire + x[i] * y[i]
        sumX2 = sumX2 + x[i] ** 2
        sumY2 = sumY2 + y[i] ** 2
    return produit_scalaire / sqrt(sumX2 * sumY2)
```

Le numérateur est le produit scalaire ; le dénominateur est le produit des normes. La formule exige **deux vecteurs non nuls**, de même longueur. La simple indication « valeurs dans [0 ; 1] » n'exclut pas le vecteur nul : dans ce cas limite, le cosinus est indéfini et le code divise par zéro. Une version destinée à des données quelconques doit refuser explicitement ce cas, par exemple en levant ValueError si une des deux sommes de carrés est nulle ; on ne lui attribue pas arbitrairement une similarité 0.

II-2 — groupes gloutons

Les remplacements sont ① `base[0]` ; ② `base[1:]` ; ③ `simcos(x, y) >=` ; ④ `(Fi, ti)` ; ⑤ `tous_les_groupes.append(groupe)`.

```
def glouton(base, seuil):
    tous_les_groupes = []
    while len(base) > 0:
        (F0, t0) = base[0]
        groupe = [(F0, t0)]
        x = GetFeatures(F0)
        for Fi, ti in base[1:]:
            y = GetFeatures(Fi)
            if simcos(x, y) >= seuil:
                groupe.append((Fi, ti))
        for g in groupe:
            base.remove(g)
        tous_les_groupes.append(groupe)
    return tous_les_groupes
```

La tranche `base[1:]` exclut l'image initiale, déjà placée dans le groupe. Parcourir toute la base la ferait ajouter deux fois, puis supprimer deux fois. Les retraits interviennent après la sélection des membres. Cette fonction **modifie la liste base** et finit par la vider : utiliser une copie si l'on veut conserver la liste d'origine. Le groupe rassemble des images similaires au premier fichier ; la similarité n'étant pas transitive, tous ses membres ne sont pas forcément similaires deux à deux.

II-3 — conserver le fichier le plus volumineux

Les remplacements sont ① `range(1, len(groupe))` ; ② `i` ; ③ `1` ; ④ `> taille` ; ⑤ `groupe[i]` ; ⑥ `fichier` ; ⑦ `fichier` ; ⑧ `groupe[i][0]`.

```
def plusGrosse(groupe):
    fichier, taille = groupe[0]
    for i in range(1, len(groupe)):
        if groupe[i][1] > taille:
            fichier, taille = groupe[i]
    return fichier
```

```
def plusPetites(groupe):
    liste = []
    fichier, taille = groupe[0]
    for i in range(1, len(groupe)):
        if groupe[i][1] > taille:
            liste.append(fichier)
            fichier, taille = groupe[i]
        else:
            liste.append(groupe[i][0])
    return liste
```

Les fonctions supposent un groupe non vide. À chaque nouveau maximum, l'ancien maximum rejoint les fichiers écartés ; les autres fichiers les rejoignent directement. Avec `[('a', 10), ('b', 30), ('c', 20)]`, on conserve b et on écarte a et c. En cas d'égalité, le premier maximum est conservé. La taille du fichier est le critère choisi par le sujet, sans être une mesure universelle de qualité visuelle.

II-4 — regrouper et choisir les représentants

Les remplacements sont ① `plusGrosse(g)` ; ② `plusPetites(g)` ; ③ `in glouton(base, seuil)`.

```
def regrouper(base, seuil):  
    return [(plusGrosse(g), plusPetites(g))  
            for g in glouton(base, seuil)]
```

On renvoie un couple par groupe : nom du fichier conservé, puis liste des noms écartés. Ce programme **renvoie des noms**, il ne supprime aucun fichier du disque.

II-5 — mémoriser les caractéristiques

Les remplacements sont ① not in ; ② featCache[fichier] ; ③ featCache[fichier].

```
featCache = {}
```

```
def GetFeaturesCache(fichier):  
    if fichier not in featCache.keys():  
        featCache[fichier] = GetFeatures(fichier)  
    return featCache[fichier]
```

Dans glouton, remplacer les deux appels à GetFeatures par GetFeaturesCache. Une description déjà calculée est réutilisée ; l'analyse par réseau de neurones n'est faite qu'au premier appel pour ce fichier. Les opérations sur les booléens, listes et dictionnaires sont décrites dans la [documentation Python](#).