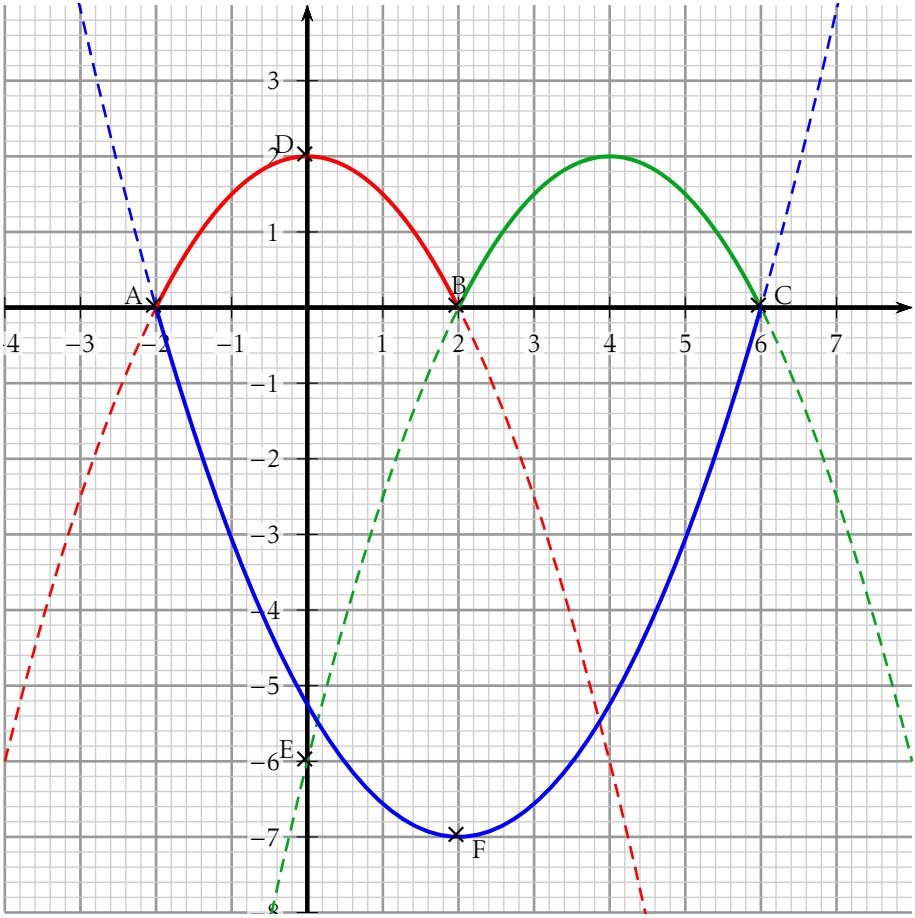
♥ MATHS ♥



1. Déterminer, en détaillant la méthode, une équation de chacune des paraboles (utiliser uniquement les points indiqués : leurs coordonnées sont des entiers relatifs.)
2. Les arcs de paraboles délimitent un domaine qui admet la droite d'équation $x = 2$ comme axe de symétrie.
   Un mathématicien amoureux y voit un cœur.

L'objectif est de modifier le programme python monte_carlo1.py ou monte_carlo2.py (sur mon site), afin de déterminer une approximation de l'aire du cœur par la méthode de Monte-Carlo.

- Expliquer votre démarche.
- Modifier le progamme, le sauver sous le nom : `nom1_nom2_coeur.py`, puis l'envoyer en respectant les consignes à appliquer aux travaux en distanciel.
- Donner une approximation de l'aire du cœur.

Aides

- la commande python `plt.axis('equal')` permet d'obtenir un repère orthonormé (à écrire à la ligne précédent `plt.show()`)
- le mot clé python `and` permet de tester plusieurs conditions :
  par exemple « si $x \in [-2;2]$ » pourra s'écrire : `if (x >= -2) and (x <= 2)`.
- il n'est pas obligatoire que le programme affiche le cœur en entier pour connaître l'aire...

Correction

Pour ceux qui ont raisonné en découpant le cœur en trois parties avec un calcul de la forme :

`k1 = points_sous_vert(n)` et aire du rectangle associé : 8

`k2 = points_sous_rouge(n)` et aire du rectangle associé : 8

`k3 = points_sur_bleu(n)` et aire du rectangle associé : 56

Le raisonnement doit être le suivant :

on cherche à estimer l'aire totale $\mathscr{A}$ comme somme des estimations de l'aire de chaque domaine.

donc $\mathscr{A} = \frac{k_1}{n} \times 8 + \frac{k_2}{n} \times 8 + \frac{k_3}{n} \times 56 = \frac{8k_1 + 8k_2 + 56k_3}{n}$

et non pas, comme je le trouve souvent :

$\mathscr{A} = \frac{k_1 + k_2 + k_3}{3n} \times 72$

AM.Yo - BE.Pa : 20/20 : Excellent travail ! Félicitations. Envoyez le fichier LibO plutôt qu'un .pdf (c'est moins lourd).

ligne 11, 26, 4 : ne changez pas l'écriture, la fonction f n'est pas la fonction F.

ligne 11 : attention : `y=uniform(0, -7)` ordre des bornes.

ligne 14 : c'est la partie *gauche* du cœur.

BA.Am : 7/20 : Objet du mail ? C'est la dernière fois que je corrige une évaluation qui ne respecte pas les consignes du distanciel. Essaye de grouper toutes tes feuilles en un seul fichier .pdf. Programme Python ?

- Parabole rouge : $\frac{2}{4}$: simplifie ! Tu oublies le carré...
- Parabole verte : $\frac{2}{4}$: simplifie ! Tu oublies le carré...
- Parabole bleue : $4^2 = 16$

BE.Im : 16/20 : Très bon travail de recherche. Très jolie mise en page, mais c'est très dommage que tu n'aies pas utilisé l'éditeur d'équations pour écrire les formules…

- revoir raisonnement pour ton programme : tu places $3n$ points (et non n) et la répartition n'est pas uniforme (beaucoup plus de points dans la partie positive du repère). Un même point est compté plusieurs fois.
- revoir le test `if y >= 7/16*(x+2)*(x-6)`
- la méthode des moyennes des rectangles est fausse (et 72 devient 56).

BE.so : 17/20 : Très bon travail. C'est la dernière fois que j'accepte une évaluation avec une PJ de ce poids. Attention orthographe.

- Revoir la rédaction ! Si C_1 est le nom de la parabole, tu ne peux pas écrire $C_1 = a(x - x_1)(x - x_2)$; de même si A est un point, tu ne peux pas écrire $A = x_1$ avec x_1 réel. Idem pour la suite.
- Détaille la fin du raisonnement : comment trouver l'aire de la moitié du cœur ?
- Ton programme te donne une aire de 50 ?

CH.Pe - LE.Ti : 17/20 : Très bien pour MarkDown ! Très bien pour la recherche mathématique ; revoir le programme Python. (Si vous donnez des explications : les écrire dans le fichier MarkDown ou dans un simple fichier nommé nom1_nom2.txt, sinon je dois rechercher les informations dans le mail et ce n'est pas pratique)

- revoir le test `if y >= 7/16*x**2 - 7/4*x - 21/4`
- Attention : votre programme ne peut fonctionner qu'avec certains interpréteurs Python : vous nommez les fonctions par les mêmes noms : la dernière fonction écrite `aire_sous` fait appel à la dernière fonction écrite `points_sous` et ne prend pas en compte la première. Avec PyScripter cela fonctionne comme vous le voulez.

CO.Ma : 16/20. TBon travail. Bel effort de MarkDown. As-tu testé tes fonctions ?

- pour écrire × : `\times`, et pour les fractions : `\dfrac{num}{den}`
- tu n'es pas obligée de développer les expressions.
- fonction `aire_sur_bleu`, revoir le test `if y >= 7/16*x**2 - 7/4*x - 21/4`
- tes fonctions ne donnent pas le résultat attendu car la commande `return` n'est pas indentée correctement. La boucle n'est effectuée qu'une seule fois.

```
for i in range(n):
    x = uniform(-2, 2)
    y = uniform(0, 2)
    if y <= -1/2*x**2 + 2 :
        point_rouge = point_
        plot(x, y, '.b')
        plot(x, y, '.r')
return point_rouge
```

DI.Dj : 11/20 : Attention erreurs de clacul. Revoir raisonnements pour le programme. C'est la dernière fois que j'accepte une évaluation avec des PJ de ce poids ! (Si tu ne peux travailler que sur smartphone : demande moi, on devrait trouver une méthode pour alléger le poids des PJ.). Je ne peux pas tester ton programme. Merci de scanner les feuilles dans le bon sens.

- Python, ligne 26 : si tu écris 0,1 cela ne va donner le résultat voulu ; il faut écrire 0.1 : le séparateur décimal en Python est le point.
- Python, ligne 44 : même remarque.
- Python, linge 61 : `y=uniform(0, -7)` : ordre des bornes.
- Python, ligne 63 : équation fausse, test faux : *au dessus*, donc >=
- Python, ligne 79 : raisonnement faux.
- Copie, lignes 5, 20 et 30 : remplace ensuite α et β par leur valeur. C'est plus facile pour comprendre ce que tu fais ensuite.
- Copie, lignes 23 et 34 : Attention calculs ! !

DO.Ao - SR.Ph - WO.Ya : 17/20 : Très bon travail. Félicitations. TBien pour Markdown ! Vérifier les graphiques obtenus répondent à la demande !

- Courbe bleue : le point à utiliser est le point F, elle ne passe pas exactement par $(5;-3)$.
- Méthode correcte pour trouver l'aire, résultats faux.
- `X_sous, Y_sous, X_sur, Y_sur = [-2], [2], [0], [2]`, il faut laisser les listes vides : `X_sous, Y_sous, X_sur, Y_sur = [], [], [], []`
- fonction `points_sous3`, le test est : `if y >= 3/7 * (x + 2) * (x - 6)`
- attention : `y=uniform(0, -7)` !?!

GA.Te : 9/20 : Pas de programme Python.

- Explique comment tu obtiens la première expression de chaque parabole.
- parabole bleue : $(6-2)^2$: commence par calculer *avant* de développer (cela évitera des erreurs de calcul) !
- parabole rouge / verte : tu oublies le carré.

GO.Em - TA.Ki : 14/20 : Abon travail. Correct pour la partie mathématique. Revoir certains points dans la partie informatique : vous n'expliquez pas comment obtenir l'aire du cœur ?

ligne 22 : que signifie S[4;2] ?

ligne 32 : que signifie S[2;−7] ?

ligne 39 : calcul faux

ligne 45 : nom de fonction incohérent : `points_sur1` : vous comptez les points qui sont sous la courbe.

ligne 71 : idem

ligne 103 : test incohérent avec votre expression de h.

KI.In - MA.Ga : 17/20 : Très bon travail. Revoir une fonction du programme python.

- fonction `points_sous_blue`, le test est : `if y >= 7/16 * (x - 2)**2 - 7`
- erreur de raisonnement masquée par le manque de cohérence dans vos notations...

```
if y <= 7/16 * (x - 2)**2 - 7 : # x^
        point_blue = point_blue + 1
        # au-dessus de la courbe
        plot(x, y, '.r') # prépare l
```

GO.Em - TA.Ki :

- calcul : fonction h
- Python : nom des fonctions incohérents avec le programme (`points_sur1` calcule le nombre de points *sous* la courbe !

LE.Ke : Ojet du mail ? Nom du fichier ? Ce devoir est considéré comme non rendu.

PH.Ji - NG.Da : 13/20 : ABien pour la partie mathématique. Revoir la partie informatique.

- la recherche incluse dans le fichier python, pourquoi pas... mais il faut maîtriser l'encodage :

```
#S [0;2]
#f(x) = a(x-alpha)Ã‚Â²+beta
#      = a(x-0)Ã‚Â²+2
#f(2)=0
#f(2) = -a(2-0)Ã‚Â²+2
#f(2) = -4a+2
```

- Fonction h : attention calcul.
- Python : nom des fonctions incohérents avec le programme (`points_sur1` calcule le nombre de points *sous* la courbe !
- Erreur de raisonnement : (`points_sous` calcule les points sous la courbe, mais il faut calculer le nombre de points au-dessus !

PR.Vi - RO.Io : 13/20 : Très bien pour la partie mathématique. Programme Python ?

RO.Ki : 14/20 : bien pour la partie mathématique, j'ai un doute pour la partie informatique.

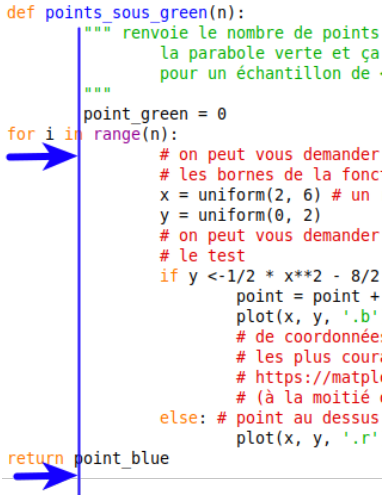
- parabole rouge : simplifie $x - 0 = x$...
- parabole bleue : tu oublies les carrés.
- Explique ta méthode pour calculer l'aire du cœur. Comment fonctionne ton programme ?
- `X_sous, Y_sous, X_sur, Y_sur = [-2], [2], [0], [2]`, il faut laisser les listes vides : `X_sous, Y_sous, X_sur, Y_sur = [], [], [], []`
- fonction `points_sous3`, le test est : `if y >= 7/16 * (x - 2)**2 - 7`
- `y=uniform(0, -7)` ordre des bornes.

SO.Da : 16/20 : Bon travail, mais le programme ne correspond pas au fichier d'explications...

- la parabole bleue est orientée « vers le haut » , donc le coefficient de x^2 est ?
- l'image de ton document ne correspond pas à ce que donne ton programme...
- attention : `y=uniform(0, -7)` !?!
- ton explication ne correspond pas au programme envoyé ?!

TO.Aï : 15/20 : Assez bon travail. C'est la dernière fois que j'accepte une évaluation avec des PJ de ce poids !

- courbe verte : si tu développes (ce n'est pas obligé pour cet exercice), simplifie les fractions !
- fonction `points_sur_blue`, le test est : `if y >= 7/16 * x**2 + 7/4 * x - 21/4`
- Tu as testé ton programme ? Il ne peut pas donner le résultat attendu car l'indentation n'est pas respectée, la logique de programmation n'est pas comprise.



```
def points_sous_green(n):
        """ renvoie le nombre de points
                la parabole verte et ça
                pour un échantillon de
        """
        point_green = 0
for i in range(n):
                # on peut vous demander
                # les bornes de la fonc
                x = uniform(2, 6) # un
                y = uniform(0, 2)
                # on peut vous demander
                # le test
                if y <-1/2 * x**2 - 8/2
                        point = point +
                        plot(x, y, '.b'
                        # de coordonnée
                        # les plus cour
                        # https://matpl
                        # (à la moitié
                else: # point au dessus
                        plot(x, y, '.r'
return point_blue
```