DM n°2 : Polynômes - Vecteurs

Exercice 1 (Recherche des racines évidentes (***))

Nous voulons mettre à profit cette remarque du cours *si un polynôme à coefficients entiers a une racine entière, ce ne peut être qu'un diviseur du terme constant*, pour écrire un programme qui recherche systématiquement toutes les racines entières d'un polynôme P de degré quelconque et qui affiche la factorisation qui en résulte. Dans le cas d'un polynôme résiduel du 2e degré, si le discriminant est négatif, le programme pourra annoncer que la factorisation est ultime.

1. Écrire le programme
2. Tester ce programme sur les quelques polynômes suivants :
   - $P_1(x) = 3x^3 + x^2 - 12x - 4$
   - $P_2(x) = 6x^3 + 3x^2 - 27x + 12$
   - $P_3(x) = -5x^3 + 45x^2 - 130x + 120$
   - $P_4(x) = x^4 - 6x^3 - 7x^2 + 48x - 36$
   - $P_5(x) = 2x^4 - 13x^3 - 7x^2 + x - 7$
   - $P_6(x) = x^4 - 4x^3 - 6x^2 + 28x - 16$
   - $P_7(x) = x^4 - 7x^3 + 13x^2 + 3x - 18$
   - $P_8(x) = x^5 + 4x^4 + 7x^3 + 7x^2 + 4x + 1$
   - $P_9(x) = x^6 + 3x^5 + 2x^4 - x^2 - 3x - 2$
   - $P_{10}(x) = x^7 - 3x^6 + 3x^5 - x^4 + x^3 - 3x^2 + 3x - 1$

Exercice 2 (Le coin du chercheur (*** polynômes))

Soit E_n l'ensemble des polynômes de degré 2 à coefficients entiers, inférieurs ou égaux en valeur absolue à un entier n. Le polynôme $x^2 - 3x + 1$ par exemple appartient à E_3, et aussi à E_4, E_5,..., mais pas à E_2 ni à E_1.

⇨ Quelle est la proportion $F(1, n)$ des polynômes de E_n qui n'ont pas de racine ?

Répondre à cette question pour les premières valeurs de n en vous aidant d'un programme.

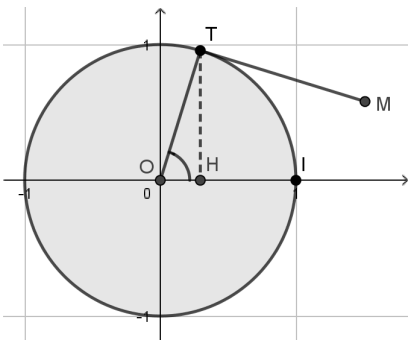
En utilisant ce même programme, répondre aux questions suivantes :

⇨ Quelle est la proportion $F(2, n)$ des polynômes de E_n qui ont des racines rationnelles (sans tenir compte de la multiplicité) ? Quelle est la proportion $F(3, n)$ des polynômes de E_n qui ont des racines irrationnelles positives ? À partir de quelle valeur de n, parmi les équations de E_n ayant 2 racines positives, y a t-il davantage de racines irrationnelles que rationnelles ?

Exercice 3 (Courbe 1 (*))

Une ficelle de longueur 10 est enroulée autour d'un disque de rayon 1 dans le sens horaire. Son extrémité libre est notée M. On représentera cette situation par un cercle trigonométrique. Lorsque la ficelle est complètement enroulée, le point M coïncide avec le point $I(1; 0)$ du cercle trigonométrique. On déroule progressivement la ficelle en veillant à toujours la maintenir tendue. Nommons T le point où la ficelle perd le contact avec le disque et notons λ l'abscisse curviligne de T (voir la figure). Dans cette situation, on s'interroge sur la courbe que va décrire le point M.

1. Faire une figure à la main sur laquelle seront placés les positions de M pour les valeurs suivantes de l'abscisse curviligne λ de T : $\frac{\pi}{2}$, π, $\frac{3\pi}{2}$ et 2π.
2. H étant le projeté orthogonal de T sur (OI), expliquer pourquoi $\widehat{MTH} = \widehat{TOH}$, puis déterminer les coordonnées du point M en fonction de λ.
3. Déterminer la position du point M lorsque la ficelle est complètement déroulée, puis tracer la courbe complète avec Geogebra (afficher la trace de M).



DM n°2 : Polynômes - Vecteurs

Exercice 4 (Quadrilatère dans parallélogramme (**))

Soit $ABDC$ un parallélogramme quelconque. On place E, F, G et H respectivement sur $[AC]$, $[AB]$, $[CD]$ et $[BD]$ de manière à inscrire le quadrilatère $EFHG$ dans le parallélogramme.

Partie 1 : $EFHG$ forme un trapèze de bases $[EF]$ et $[GH]$. Faire une figure sur Geogebra qui permette de déplacer les bases du trapèze ; observer sur quel lieu se déplace le point d'intersection de ses diagonales.

L'objectif des questions qui suivent est de prouver cette propriété.

1. Montrer que l'on traduit l'énoncé en choisissant des nombres e, f, g et h dans l'intervalle $[0;1]$ tels que, dans le repère $(A, \overrightarrow{AB}, \overrightarrow{AC})$, les sommets du trapèze aient pour coordonnées $E(0;e)$, $F(f;0)$, $G(g;1)$ et $H(1;h)$, avec $f(h-1) = e(g-1)$
2. Déterminer les équations des droites (AD), (EH) et (FG).
3. Déterminer les points d'intersection $M_1 = (EH) \cap (AD)$ et $M_2 = (FG) \cap (AD)$
4. Prouver alors que M_1 et M_2, quand ils existent, sont confondus

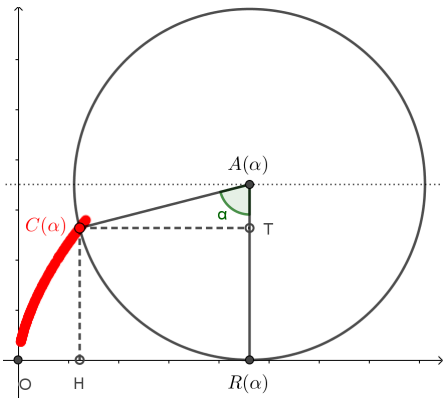
Partie 2 : Les diagonales (EH) et (FG) du quadrilatère sont parallèles aux côtés du parallélogramme. Faire une figure sur Geogebra, puis montrer que les côtés (EG) et (FH) du quadrilatère sont

- soit parallèles entre eux et, dans ce cas, parallèles à la diagonale (AD)
- soit sécants et, dans ce cas, leur point d'intersection se trouve sur la diagonale (AD)

Exercice 5 (Courbe 2 (***))

Un chewing-gum s'est collé sur la roue avant de mon vélo. Cette roue ayant un diamètre extérieur de $700mm$, on aimerait déterminer l'expression analytique des coordonnées du point C (comme chewing-gum) lorsque l'axe A de la roue se déplace d'une longueur x, la roue restant en contact avec la route en un point noté R.

On considèrera que le chewing-gum est en contact avec le sol (R et C sont alors confondus en un point noté O) lorsque $x = 0$ et on mesurera la rotation de la roue par un angle, noté α, égal à 0 lorsque R est en O.



1. Faire une figure avec Geogebra (afficher la trace de C) pour $x \in [0, 700\pi]$ où le point mobile est A (ou R).
2. H étant le projeté orthogonal de C sur (OR), et T celui de C sur (AR), déterminer les coordonnées du point C en fonction de α.
3. Déterminer la longueur approximative (à $1mm$ près) de la trajectoire parcourue par le chewing-gum pendant 1 tour de roue.

Utiliser pour cela un programme qui subdivise l'intervalle $[0, 2\pi]$ en n parties égales et détermine la longueur de la trajectoire entre deux positions voisines de C en estimant que celle-ci peut être assimilée au segment de droite reliant ces deux points. Lorsque n est grand, la somme de ces longueurs s'approche de la longueur cherchée. Jusqu'à quelle valeur de n faut-il aller pour obtenir une estimation précise à $1mm$ près ?

Correction de l'exercice 1 (Recherche des racines évidentes (***))

Le programme détermine les diviseurs de la valeur absolue du coefficient constant et les place, avec leurs opposés, dans une liste. Ensuite, le polynôme est évalué pour chacune des valeurs de la liste : lorsque l'image calculée est nulle — cela signifie que la valeur est une racine — les coefficients du polynômes sont recalculés. En effet, si α est une racine de P alors $P(x) = (x-\alpha)Q(x)$ et les coefficients de Q se calculent facilement en identifiant les coefficients du développement de $(x-\alpha)Q(x)$ avec ceux de P. Cela explique la fonction `recalcule(x,a)` qui réalise cela.

Montrons ce point pour le degré 3, la formule obtenue étant valable pour un degré n quelconque.

Si $P(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ et si α est une racine de P, alors $P(x) = (x-\alpha)Q(x)$ avec $Q(x) = b_2x^2 + b_1x + b_0$ (le degré de Q est inférieur de 1 au degré de P).

Développons $(x-\alpha)Q(x)$. On obtient alors :

$P(x) = x(b_2x^2 + b_1x + b_0) - \alpha(b_2x^2 + b_1x + b_0) = b_2x^3 + (b_1 - \alpha b_2)x^2 + (b_0 - \alpha b_1)x - \alpha b_0$

L'identification des termes de même degré conduit au système suivant :

$$\begin{cases} b_2 = a_3 \\ b_1 - \alpha b_2 = a_2 \\ b_0 - \alpha b_1 = a_1 \\ -\alpha b_0 = a_0 \end{cases} \iff \begin{cases} b_2 = a_3 = \frac{b_1 - a_2}{\alpha} \\ b_1 = \frac{b_0 - a_1}{\alpha} \\ b_0 = \frac{-a_0}{\alpha} \end{cases}$$

Ainsi, en partant de $b_0 = \frac{-a_0}{\alpha}$ (division qui aboutit toujours sur un entier puisque α est un diviseur de a_0), on calcule ainsi facilement b_1, puis b_2. Pour les degrés supérieurs à 3, on utilise le même principe pour calculer les coefficients de Q : $n \geqslant 1, b_n = \frac{b_{n-1} - a_n}{\alpha}$.

La fonction `polynome()`, quant à elle, évalue le polynôme pour une valeur de la variable.

Le plus compliqué est, finalement, de faire afficher correctement le polynôme résiduel, car les conventions d'écritures imposent d'afficher, par exemple, `(-2x+1)` ou `x-1`, et non `(-2x^1+1)`ou `1x+-1`... Ceci est sans doute un détail et peut très bien ne pas être envisagé (on peut se contenter, vous en conviendrez, d'un affichage moins formaté comme $[1,-2][1,2][3,1]$ à la place du plus conventionnel $(x-2)(x+2)(3x+1)$).

P_1	$3x^3 + x^2 - 12x - 4$	$(x-2)(x+2)(3x+1)$	ultime
P_2	$6x^3 + 3x^2 - 27x + 12$	$6x^3 + 3x^2 - 27x + 12$	
P_3	$-5x^3 + 45x^2 - 130x + 120$	$-5(x-2)(x-3)(x-4)$	ultime
P_4	$x^4 - 6x^3 - 7x^2 + 48x - 36$	$(x-1)(x-2)(x+3)(x-6)$	ultime
P_5	$2x^4 - 13x^3 - 7x^2 + x - 7$	$(x+1)(x-7)(2x^2 - x + 1)$	ultime
P_6	$x^4 - 4x^3 - 6x^2 + 28x - 16$	$(x-2)(x-4)(x^2 + 2x - 2)$	
P_7	$x^4 - 7x^3 + 13x^2 + 3x - 18$	$(x+1)(x-2)(x-3)(x-3)$	ultime
P_8	$x^5 + 4x^4 + 7x^3 + 7x^2 + 4x + 1$	$(x+1)(x+1)(x+1)(x^2 + x + 1)$	ultime
P_9	$x^6 + 3x^5 + 2x^4 - x^2 - 3x - 2$	$(x-1)(x+1)(x+1)(x+2)(x^2+1)$	ultime
P_{10}	$x^7 - 3x^6 + 3x^5 - x^4 + x^3 - 3x^2 + 3x - 1$	$(x-1)(x-1)(x-1)(x^4+1)$	

```
quel est le degré du polynôme 5
coefficient de x^0 (0 si absent)1
coefficient de x^1 (0 si absent)4
coefficient de x^2 (0 si absent)7
coefficient de x^3 (0 si absent)7
coefficient de x^4 (0 si absent)4
coefficient de x^5 (0 si absent)1
La factorisation trouvée du polynôme :(x+1)(x+1)(x+1)(x^2+x+1)
La factorisation trouvée est ultime
```

```
quel est le degré du polynôme 3
coefficient de x^0 (0 si absent)12
coefficient de x^1 (0 si absent)-27
coefficient de x^2 (0 si absent)3
coefficient de x^3 (0 si absent)6
La factorisation trouvée du polynôme :(6x^3+3x^2-27x+12)
```

```
quel est le degré du polynôme 7
coefficient de x^0 (0 si absent)-1
coefficient de x^1 (0 si absent)3
coefficient de x^2 (0 si absent)-3
coefficient de x^3 (0 si absent)1
coefficient de x^4 (0 si absent)-1
coefficient de x^5 (0 si absent)3
coefficient de x^6 (0 si absent)-3
coefficient de x^7 (0 si absent)1
La factorisation trouvée du polynôme :(x-1)(x-1)(x-1)(x^4+1)
```

```
quel est le degré du polynôme 4
coefficient de x^0 (0 si absent)-16
coefficient de x^1 (0 si absent)28
coefficient de x^2 (0 si absent)-6
coefficient de x^3 (0 si absent)-4
coefficient de x^4 (0 si absent)1
La factorisation trouvée du polynôme :(x-2)(x-4)(x^2+2x-2)
```

Voilà donc une proposition de programme qui fonctionne mais qui peut ne pas répondre tout-à-fait à toutes les situations possibles. Je ne l'ai testé que sur les exemples proposés. La 3e colonne du tableau qui précède présente les factorisations qu'il trouve tandis que la 4e indique si la factorisation est ultime, ce que le programme détecte.
Commentaires : Le programme affiche les bons résultats, sauf quand il ne trouve pas de racines évidentes auquel cas il ne le signale pas (ce qui peut être un défaut). Le polynôme P_2 par exemple se factorise en $3(2x-1)(x^2+x-4)$ mais il ne trouve pas la racine $\frac{1}{2}$. Un autre défaut qui pourrait facilement être corrigé : lorsqu'une racine est multiple, il n'affiche pas la multiplicité. Par exemple pour P_{10} il devrait afficher $(x-1)^3(x^4+1)$ puisque 1 y est une racine triple (de même pour les polynômes P_7, P_8 et P_9 qui ont chacun une racine multiple). Á la place de cela, mon programme affiche un inélégant $(x-1)(x-1)(x-1)(x^4+1)$. Le choix qui prime ici est la simplicité du programme : pour gagner en sophistiquation, on doit accepter de perdre en clarté.
Signalons enfin que certains logiciels savent factoriser. Geogebra, par exemple, a un volet « calcul formel » qui effectue cela très bien : quand on tape `Factoriser(6x^3+3x^2-27x+12)`, on obtient le résultat `3(2x-1)(x^2+x-4)` sans problème. De même, ce logiciel nous donne $(x-1)^3(x^4+1)$ pour la factorisation de P_{10}.

```
def polynome(x,a):
    valeur=0
    for e in range(len(a)): valeur+=a[e]*x**e
    return valeur

def recalcule(x,a):
    b=[-a[0]/x]
    for i in range(1,len(a)-1): b.append((b[i-1]-a[i])/x)
    return b

def ecriture(a):
    pol="("
    for e in range(len(a)-1,-1,-1):
        if                          a[e]!=0:
            if                         e==0:
                if   len(a)-1==0 or a[e]<0:pol+=str(int(a[e]))
                elif                a[e]>0:pol+="+"+str(int(a[e]))
            elif                       e==1:
                if             abs(a[e])==1:
                    if             a[e]==1:pol+="+x"
                    else                   :pol+="-x"
                elif len(a)-1==1 or a[e]<0:pol+=str(int(a[e]))+"x"
                else                       :pol+="+"+str(int(a[e]))+"x"
            else                           :
                if             abs(a[e])==1:
                    if             a[e]==1:pol+="x^"+str(e)
                    else                   :pol+="-x^"+str(e)
                elif e==len(a)-1 or a[e]<0:pol+=str(int(a[e]))+"x^"+str(e)
                else                       :pol+="+"+str(int(a[e]))+"x^"+str(e)
    return pol+")"
```

```
n=int(input("quel est le degré du polynôme "))
factorisation,ultime="",False
coeff,liste_diviseurs=[],[]
for i in range(n+1):
    coeff.append(int(input("coefficient de x^{} (0 si absent)".format(i))))
for i in range(1,abs(coeff[0])+1):
    if abs(coeff[0])%i==0:
        liste_diviseurs.append(i)
        liste_diviseurs.append(-i)
for i in liste_diviseurs:
    while polynome(i,coeff)==0:
        coeff=recalcule(i,coeff)
        if i>0:factorisation+="(x-{})".format(i)
        else  :factorisation+="(x+{})".format(-i)
if len(coeff)==1:
    if  coeff[0]==-1:factorisation="-"+factorisation
    elif coeff[0]!=1:factorisation=str(int(coeff[0]))+factorisation
else            :factorisation+=ecriture(coeff)
print("La factorisation trouvée du polynôme :{} ".format(factorisation))
if len(coeff)==3:
    delta=coeff[1]**2-4*coeff[2]*coeff[0]
    if delta<0: ultime=True
elif len(coeff)<=2: ultime=True
if ultime==True: print("La factorisation trouvée est ultime ")
```

Correction de l'exercice 2 (Le coin du chercheur (*** polynômes))

Une remarque préliminaire : d'une façon évidente, les équations $x^2-3x+1=0$ et $-x^2+3x-1=0$ ont les mêmes solutions. Pour éviter de compter deux fois les polynômes qui ont les mêmes racines, je peux me limiter aux équations dont le coefficient a est positif. Mais comme $x^2-3x+1=0$ et $2x^2-6x+2=0$ ont aussi les mêmes solutions, je peux encore limiter mon étude aux trinômes dont les coefficients sont premiers entre eux (n'ont pas de diviseur en commun autre que 1). De cette façon, j'examine une liste plus restreinte dont chaque élément a des racines particulières.

Avant de programmer cette recherche, je me reporte à la partie du cours « étude du signe des racines » qui donne un algorithme pour détecter les racines de même signe.

Pour détecter les racines rationnelles, j'utiliserai juste un test qui compare $\sqrt{\Delta}$ et sa partie entière : s'il y a égalité, j'en déduirai que les racines sont rationnelles.

Je tiens donc là l'essentiel du contenu mathématique, il ne me reste plus qu'à ajouter le traitement informatique : je vais tester toutes les valeurs possibles et construire des listes contenant les coefficients des trinômes appartenant aux 7 catégories à différencier :

- $F0$: trinômes sans racine
- $F1i$: trinômes avec racines irrationnelles et de signes contraires
- $F1r$: trinômes avec racines rationnelles et de signes contraires
- $F2ip$: trinômes avec racines irrationnelles et de signes positifs
- $F2rp$: trinômes avec racines rationnelles et de signes positifs
- $F2in$: trinômes avec racines irrationnelles et de signes négatifs
- $F2rn$: trinômes avec racines rationnelles et de signes négatifs

Les dénombrements de ces différentes catégories seront simplement affichés dans la console pour une valeur de n donnée. Je reprends ensuite ces nombres dans une feuille de tableur (Calc d'OpenOffice) pour les stocker, effectuer les traitements statistiques éventuels et aussi pour présenter les résultats de cette étude.

Voici donc le programme utilisé : il est écrit en Python pour un ordinateur, mais sur une calculatrice Numworks le principe resterait le même.

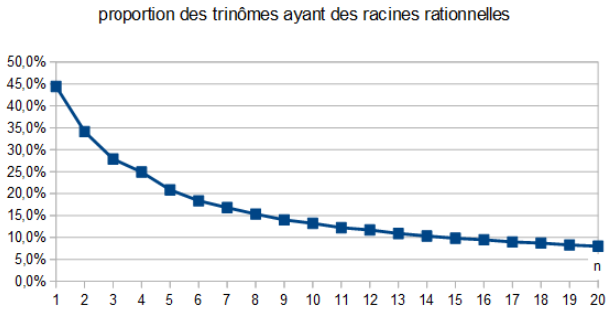
```
from math import sqrt
def diviseur(a,b,c):
	for i in range(2,n+1):
		if a%i==0 and b%i==0 and c%i==0:return i
	return 1

n=int(input("n="))
La=[]#liste des valeurs possibles pour a (a>0)
Lbc=[0]#liste des valeurs possibles pour b et c
F0,F1i,F1r,F2ip,F2rp,F2in,F2rn=[],[],[],[],[],[],[]
for i in range(n):
	La.append(i+1)
	Lbc.append(i+1)
	Lbc.append(-(i+1))
for a in La:
	for b in Lbc:
		for c in Lbc:
			if diviseur(a,b,c)==1:
				d=b**2-4*a*c
				if d<0:F0.append((a,b,c)) #pas de racine
				else:
					rat=False
					if int(sqrt(d))==sqrt(d):rat=True #racines rationnelles
					if a*c<0: #racines de signes contraires
						if rat==True:F1r.append((a,b,c))
						else:F1i.append((a,b,c))
					else:#racines de même signe
						if a*b<0:#racines positives
							if rat==True:F2rp.append((a,b,c))
							else:F2ip.append((a,b,c))
						else:#racines négatives
							if rat==True:F2rn.append((a,b,c))
							else:F2in.append((a,b,c))
print("résultats:",len(F0),len(F1i),len(F1r),len(F2ip),len(F2rp),len(F2in),len(F2rn))
```

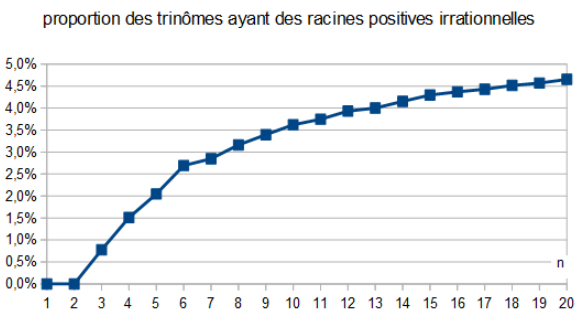
Passons maintenant aux résultats ; en voici les effectifs détaillés jusqu'à $n = 20$:

n	pas de solution	deux de signes contraires		deux de même signe				total	rationnelles	irrationnelles
				positives		négatives				
		irrationnelles	rationnelles	irrationnelles	rationnelles	irrationnelles	rationnelles			
1	3	2	1	0	1	0	2	9	4	2
2	15	12	5	0	4	0	5	41	14	12
3	49	42	15	1	10	1	11	129	36	44
4	99	92	29	4	18	4	19	265	66	100
5	203	200	49	11	31	11	32	537	112	222
6	309	314	71	22	39	22	40	817	150	358
7	519	542	107	39	61	39	62	1369	230	620
8	731	780	141	61	77	61	78	1929	296	902
9	1049	1144	185	94	101	94	102	2769	388	1332
10	1369	1508	237	131	120	131	121	3617	478	1770
11	1899	2128	297	188	158	188	159	5017	614	2504
12	2323	2626	359	242	180	242	181	6153	720	3110
13	3071	3514	431	325	227	325	228	8121	886	4164
14	3713	4282	503	408	255	408	256	9825	1014	5098
15	4565	5304	601	520	293	520	294	12097	1188	6344
16	5421	6330	695	628	333	628	334	14369	1362	7586
17	6701	7894	795	787	398	787	399	17761	1592	9468
18	7663	9058	903	918	434	918	435	20329	1772	10894
19	9265	11030	1019	1123	508	1123	509	24577	2036	13276
20	10549	12608	1137	1303	550	1303	551	28001	2238	15214
Total	59516	69410	7580	6805	3798	6805	3818	157732	15196	83020
Fréquence	38%	44%	5%	4%	2%	4%	2%	100%	10%	53%
		49%		13%						
	F(1,n)			F(3,n)					F(2,n)	

➪ La proportion $F(1,n)$ des polynômes de E_n qui n'ont pas de racine semble se stabiliser rapidement sur une valeur de $37,7\%$, soit plus d'un tiers.
Les premiers de ces trinômes sans racine sont :
x^2+1, x^2+x+1 et x^2-x+1 (pour les 3 de E_1),
x^2+2, x^2+x+2, x^2-x+2, x^2+2x+2, x^2-2x+2, $2x^2+1$, $2x^2+x+1$, $2x^2+x+2$, $2x^2-x+1$, $2x^2-x+2$, $2x^2+2x+1$ et $2x^2-2x+1$ (pour les 12 de E_2 qui ne sont pas dans E_1), etc.

➪ La proportion $F(2,n)$ des polynômes de E_n qui ont des racines rationnelles décroit régulièrement (courbe de gauche) de 45% environ pour $n = 1$ à 8% environ pour $n = 20$. Comme les trinômes n'ayant pas de racine sont en proportion stable, cette baisse est compensée par l'augmentation des trinômes ayant des racines irrationnelles, ce qui n'est pas étonnant compte tenu de la prépondérance des nombres irrationnels. Un examen plus détaillé de la répartition montre que les trinômes ayant des racines rationnelles se répartissent à peu près équitablement entre ceux qui ont des racines de mêmes signes et ceux qui ont des racines de signes contraires.
Les premiers de ces trinômes ayant des racines rationnelles sont :
x^2-1, x^2-x, x^2+x et x^2 (pour les 4 de E_1),
x^2+x-2, x^2-x-2, $2x^2+x-1$, $2x^2+x-1$, x^2-2x, x^2-2x+1, $2x^2-x$, x^2+2x, x^2+2x+1, $2x^2+x$ (pour les 10 de E_2 qui ne sont pas dans E_1), etc.





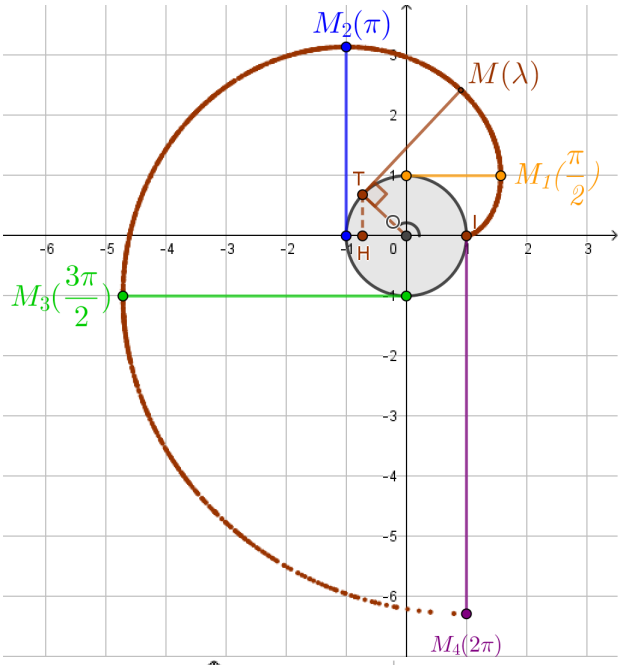
➪ La proportion $F(3,n)$ des polynômes de E_n qui ont des racines irrationnelles positives croit régulièrement (courbe de droite) de 0 à près de 5% pour $n = 20$. Il faut noter l'égalité parfaite entre les trinômes ayant des racines irrationnelles positives et négatives. Ceci est du au fait que les numérateurs des racines $-b+\sqrt{\Delta}$ et $-b-\sqrt{\Delta}$ ne font que changer de signe quand on remplace b par $-b$: $-(-b)+\sqrt{\Delta} = -(b-\sqrt{\Delta})$ et $-(-b)-\sqrt{\Delta} = -(b+\sqrt{\Delta})$. Le même phénomène a lieu avec les racines rationnelles, excepté pour les équations de type $x^n = 0$ qui ne sont présente qu'en un exemplaire que mon programme a placé dans les racines négatives, ce qui n'est pas fondamentalement gênant.

Les premiers de ces trinômes ayant des racines positives et irrationnelles sont :
x^2-3x+1 (pour le premier qui apparait, dans E_3),
x^2-4x+1, x^2-4x+2, $2x^2-4x+1$ (pour les 3 de E_4 qui ne sont pas dans E_3),
x^2-5x+1, x^2-5x+2, x^2-5x+3, x^2-5x+5, $2x^2-4x+1$, $2x^2-5x+1$, $3x^2-5x+1$, $5x^2-5x+1$ (pour les 8 de E_5 qui ne sont pas dans E_4), etc.

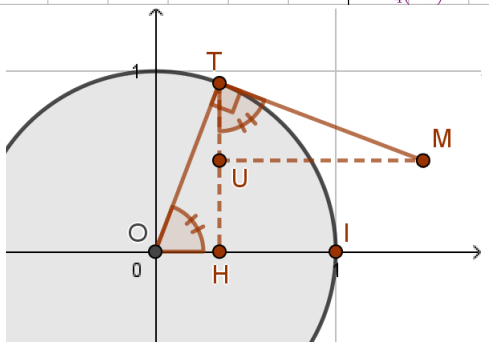
➪ C'est à partir de $n=10$ que, parmi les équations de E_n ayant 2 racines positives, il y a davantage de racines irrationnelles que rationnelles. Pour $n=9$, en effet, il y a 94 trinomes ayant des racines positives irrationnelles contre 101 rationnelles alors que pour $n=10$, en effet, il y a 131 trinomes ayant des racines positives irrationnelles contre 120 rationnelles.

Correction de l'exercice 3 (Courbe 1 (*))

Voici la figure demandée où sont placées les positions de M pour $\lambda \in \{\frac{\pi}{2};\pi;\frac{3\pi}{2};2\pi\}$.



J'ai fait la figure sur Geogebra plutôt que réellement à la main, ce qui permet d'afficher la trace du point M pour d'autres positions de T. Cette trace apparait comme une courbe, celle-la même que nous cherchons à exprimer de façon analytique.
Pour réaliser cette figure, il suffit de constater que la longueur de la ficelle déroulée est égale à la longueur de l'arc $\overset{\frown}{IT}$, soit à λ directement puisque λ est exprimé en radians et que le rayon du cercle vaut 1.
Déterminons maintenant les coordonnées de M dans le repère orthonormé (O,I,J), naturellement associé au cercle trigonométrique. La figure de l'énoncé suggérait que $\widehat{OTM}$ est un angle droit, (MT) étant la tangente au cercle en T. Cette caractéristique de la situation fait que, H étant le projeté orthogonal de T sur (OI), on a $\widehat{MTH}=\widehat{TOH}$. En effet, dans le triangle TOH rectangle en H, l'angle $\widehat{OTH}$ est le complémentaire de $\widehat{TOH}$. Or $\widehat{OTM}$ étant droit, $\widehat{OTH}$ est également le complémentaire de $\widehat{MTH}$. Par conséquent $\widehat{MTH}$ et $\widehat{TOH}$ ont même mesure, notée λ.
Considérons alors le point U, projeté orthogonal de M sur (TH).

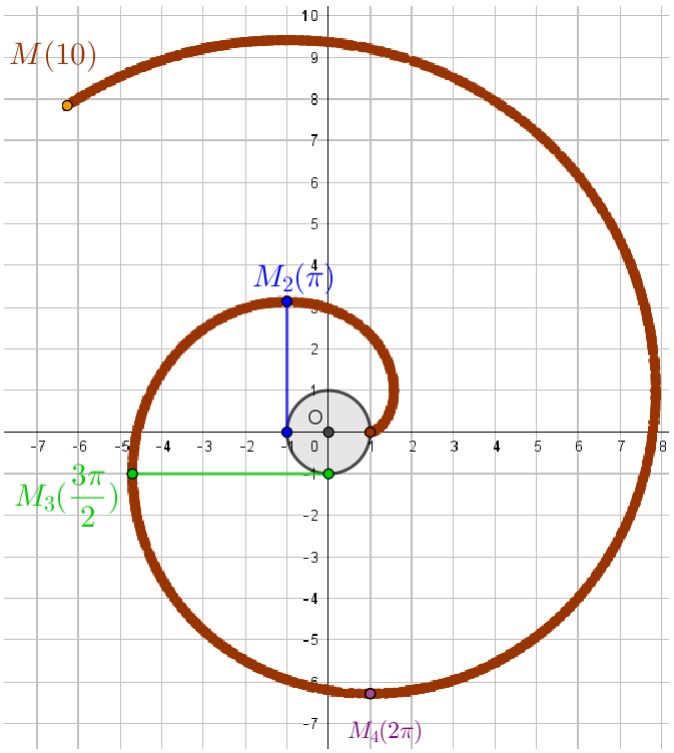


L'abscisse de M est égale à la somme $OH+UM$ (les longueurs sont ici considérées comme toujours positives, ce qui n'arrive réellement que lorsque $\lambda \in [0,\frac{\pi}{2}]$, mais en réalité elles pourront être négatives, comme OH dans la figure du haut où $\lambda \in [\frac{\pi}{2},\pi]$). Or $OH=\cos(\lambda)$ et $UM=\sin(\lambda)\times TM$ et on a vu que $TM=\lambda$, d'où $x_M=OH+UM=\cos(\lambda)+\lambda\sin(\lambda)$.
Pour l'ordonnée de M, ce n'est pas très différent : $y_M=TH-TU$ or $TH=\sin(\lambda)$ et $TU=\lambda\cos(\lambda)$, d'où $\boxed{M(\lambda)\begin{cases}x=\cos(\lambda)+\lambda\sin(\lambda)\\ y=\sin(\lambda)-\lambda\cos(\lambda)\end{cases}}$

Lorsque la ficelle est complètement déroulée (sa longueur est égale à 10), le point T se retrouve à l'abscisse curviligne $10 = 10 - 2\pi \approx 3,7168[2\pi]$. Les coordonnées de M sont alors calculées à partir de la formule

$$M(10)\begin{cases} x = \cos(10) + 10\sin(10) \approx -6,2793 \\ y = \sin(10) - 10\cos(10) \approx 7,8467 \end{cases}$$

Cette courbe est appelée « développante » du cercle trigonométrique ou aussi, plus savamment, « anti-clothoïde ». Elle a tendance à s'approcher d'une spirale d'Archimède au fur et à mesure que l'on déroule la ficelle. Une spirale d'Archimède est la trajectoire décrite par un point animé d'un mouvement rectiligne uniforme sur une demi-droite tournant elle-même de façon uniforme autour de l'origine de cette demi-droite (la courbe réalisée par un diamant qui parcourt les sillons d'un disque « vinyle » en est un exemple).



Voir les propriétés, développements et autres applications (engrenages, etc.) sur le site Mathcurve [1]

Correction de l'exercice 4 (Quadrilatère dans parallélogramme (**))

Partie 1 :

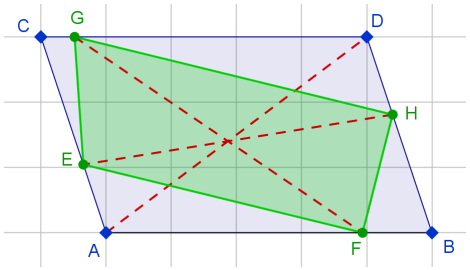
En manipulant la figure créée sur GeoGebra, on remarque que le point d'intersection des diagonales du trapèze « semble » se déplacer sur la diagonale (AD) du parallélogramme (je mets des guillemets parce qu'il s'agit du terme consacré pour une conjecture : on voit que le point est « toujours » sur la diagonale).

1- Dans le repère $(A, \overrightarrow{AB}, \overrightarrow{AC})$, on a :

$$A(0\ ;\ 0),\ B(1\ ;\ 0),\ C(0\ ;\ 1)$$

De plus, $ABDC$ étant un parallélogramme, on a $\overrightarrow{AD} = \overrightarrow{AB} + \overrightarrow{AC}$ et donc $D(1\ ;\ 1)$.

Pour traduire $E \in [AC]$, on doit écrire qu'il existe un nombre $e \in [0;1]$ tel que $\overrightarrow{AE} = e\overrightarrow{AC}$ et donc $E(0;e)$. De même pour les trois autres points, en particulier comme $H \in [BD]$ alors $\overrightarrow{AH} = \overrightarrow{AB} + h\overrightarrow{AC}$ avec $h \in [0;1]$ et donc $H(1;h)$.



Avec ces coordonnées de points, calculons les composantes des vecteurs $\overrightarrow{EF}$ et $\overrightarrow{GH}$ dans la base $(\overrightarrow{AB}, \overrightarrow{AC})$: $\overrightarrow{EF}(f;-e)$ et $\overrightarrow{GH}(1-g;h-1)$.

Écrivons maintenant que $\overrightarrow{EF}$ et $\overrightarrow{GH}$ sont colinéaires pour traduire le parallélisme des bases du trapèze :

$$f \times (h-1) - (-e) \times (1-g) = 0 \iff f \times (h-1) = e(g-1)$$

C'est la relation de l'énoncé qu'il nous restait à justifier.

2- L'équation de la droite (AD) est évidemment $y = x$ ($A(0\ ;\ 0)$ et $D(1\ ;\ 1)$ vérifient cette égalité). Déterminons l'équation de la droite (EH) : on a $\overrightarrow{EH}(1;h-e)$ et, pour un point $M(x;y)$ quelconque $\overrightarrow{EM}(x;y-e)$. D'où $M \in (EH) \iff y - e - x(h-e) = 0 \iff y = x(h-e) + e$.

Déterminons, de même, l'équation de la droite (FG) : on a $\overrightarrow{FG}(g-f;h-e)$ et, pour un point $M(x;y)$ quelconque $\overrightarrow{FM}(x-f;y)$. D'où $M \in (FG) \iff x + y(f-g) - f = 0$.

3- Les coordonnées du point d'intersection $M_1 = (EH) \cap (AD)$ vérifient $y = x$ et $y = x(h-e) + e$ ce qui conduit à $x = x(h-e) + e \iff x(1+e-h) = e$.

1. https://www.mathcurve.com/courbes2d/developpantedecercle/developpantedecercle.shtml

Si $1+e-h \neq 0$, c'est-à-dire si $1+e \neq h$ ($1+e=h$ n'arrive jamais quand $e \in]0;1]$ et $h \in [0;1[$; ce cas correspond à $E=A$ et $H=D$), alors on a $\boxed{x_{M_1}=y_{M_1}=\frac{e}{1+e-h}}$

De même, les coordonnées du point d'intersection $M_2=(FG)\cap(AD)$ vérifient les équations $y=x$ et $x+y(f-g)-f=0$ ce qui conduit à $x+x(f-g)-f=0 \iff x(1+f-g)=f$.

Si $1+f-g \neq 0$, c'est-à-dire si $1+f \neq g$ ($1+f=g$ n'arrive jamais quand $f \in]0;1]$ et $g \in [0;1[$; ce cas correspond à $F=A$ et $G=D$), alors on a $\boxed{x_{M_2}=y_{M_2}=\frac{f}{1+f-g}}$

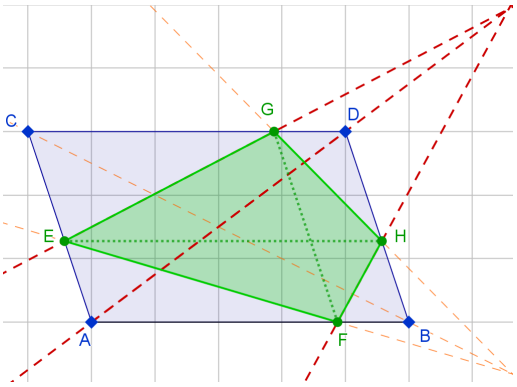
4- Les points M_1 et M_2 sont confondus si et seulement si ils ont mêmes coordonnées.

Or $\frac{e}{1+e-h}=\frac{f}{1+f-g}$, dans le cas général où E et F diffèrent de A et G et H diffèrent de D, est équivalent à $e(1+f-g)=f(1+e-h) \iff e(1-g)=f(1-h)$, ce qui est l'égalité de la question 1 traduisant le parallélisme des bases du trapèze.

Conclusion : *un trapèze inscrit dans un parallélogramme a des diagonales qui se coupent sur la diagonale du parallélogramme que coupent les bases parallèles du trapèze.*

Partie 2 :

Sur notre figure, les côtés (EG) et (FH) du quadrilatère sont sécants et leur point d'intersection semble se trouver sur la diagonale (AD) du parallélogramme et à l'extérieur du parallélogramme. Selon la position des points, ce point d'intersection est d'un côté ou de l'autre du parallélogramme, la position intermédiaire correspondant au parallélisme $(EG)//(FH)$. On remarque que lorsque les côtés (GH) et (EF) sont sécants, leur point d'intersection semblent se trouver sur la diagonale (BC).



Dans le repère $(A,\overrightarrow{AB},\overrightarrow{AC})$, on a toujours $A(0\ ;\ 0)$, $B(1\ ;\ 0)$, $C(0\ ;\ 1)$ et $D(1\ ;\ 1)$.

Comme précédemment, avec $e \in [0;1]$ et $f \in [0;1]$, on a $E(0;e)$ et $F(f;0)$ qui traduit $E \in [AC]$ et $F \in [AB]$.

Comme, de plus, $(EH)//(AB)$ et $(FG)//(AC)$, on a nécessairement $H(1;e)$ et $G(f;1)$.

Les composantes des vecteurs $\overrightarrow{EG}$ et $\overrightarrow{FH}$ dans la base $(\overrightarrow{AB},\overrightarrow{AC})$ sont : $\overrightarrow{EG}(f;1-e)$ et $\overrightarrow{FH}(1-f;e)$.

$(EG)//(FH) \iff det(\overrightarrow{EG},\overrightarrow{FH})=0 \iff ef-(1-e)(1-f)=0 \iff e+f=1$.

Si $e+f \neq 1$, les droites (EG) et (FH) sont donc sécantes.

Cherchons les coordonnées de $M=(EG)\cap(FH)$.

Pour cela déterminons les équations de ces deux droites :

- Équation de (EG) : on a $\overrightarrow{EG}(f;1-e)$ et, pour un point $M(x;y)$ quelconque $\overrightarrow{EM}(x;y-e)$. D'où $M \in (EG) \iff f(y-e)-x(1-e)=0 \iff y=x\frac{1-e}{f}+e$
- Équation de (FH) : on a $\overrightarrow{FH}(1-f;e)$ et, pour un point $M(x;y)$ quelconque $\overrightarrow{FM}(x-f;y)$. D'où $M \in (FH) \iff y(1-f)-e(x-f)=0 \iff y=e\frac{x-f}{1-f}$.

Les coordonnées de M vérifient, $y=x\frac{1-e}{f}+e$ et $y=e\frac{x-f}{1-f}$ ce qui conduit à $x\frac{1-e}{f}+e=e\frac{x-f}{1-f}$.

Je passe sur les détails du calcul, mais si $f \neq 1$ et $e+f \neq 1$, je trouve $x=\frac{ef}{e+f-1}$ et $y=\frac{ef}{e+f-1}$, soit $x_M=y_M$: le point M est sur (AD), la diagonale du parallélogramme.

Remarques : on aurait pu éviter de calculer les coordonnées de M puisqu'il s'agissait seulement de montrer que $x_M=y_M$. On aurait juste pu additionner membre à membre les équations de (EG) et (FH). Cela donne $f(y-e)-x(1-e)+y(1-f)-e(x-f)=0$ et se réduit en $-x+y=0 \iff x=y$.

Par ailleurs, ce que nous venons de faire pour les côtés opposés (EG) et (FH) pourrait être refait pour l'autre paire de côtés opposés ((EF) et (GH)). Il suffit, pour cela, d'échanger les noms de points. La propriété est vraie pour les deux paires de côtés opposés.

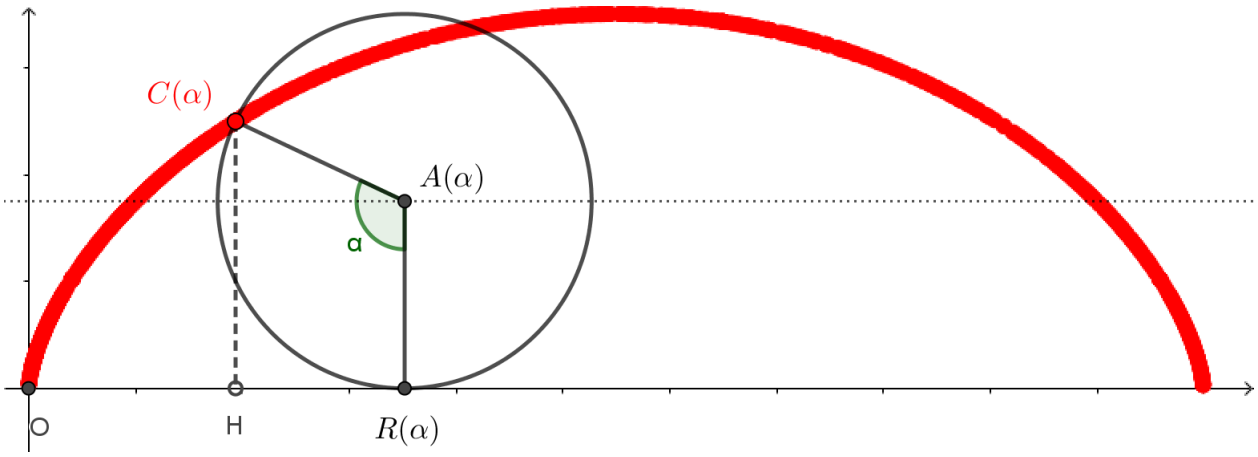
Conclusion : *un quadrilatère inscrit dans un parallélogramme dont les diagonales sont parallèles aux côtés du parallélogramme a ses côtés opposés parallèles ou sinon sécants, le point d'intersection étant alors toujours situé sur la diagonale correspondante du parallélogramme.*

Correction de l'exercice 5 (Courbe 2 (***))

1- Voici la figure demandée, réalisée avec Geogebra, pour $\alpha \in [0, 2\pi]$. Pour cet intervalle de valeurs, la roue effectue un tour complet. Le périmètre de la roue étant égal à πd où $d = 2r$ est le diamètre, égal à $700mm$ selon l'énoncé, la valeur de x parcourra ainsi tout l'intervalle $[0, 700\pi]$ comme il était demandé.
Pour placer correctement le point C sur la roue, il faut déjà réaliser que la longueur $x = OR$ doit être égale à l'arc $\overset{\frown}{RC}$. Or, la longueur de cet arc est égale à $r\alpha$. On en déduit la valeur de l'angle à donner à Geogebra pour placer C à partir de r :

$$\alpha = \frac{OR}{AR} = \frac{x}{r} = \frac{x}{700}, \text{ si } x \text{ est exprimé en } mm \text{ comme } r$$

Sur ma figure, j'ai pris $r = 3,5cm = 35mm$, soit une échelle de $\frac{1}{10}$. La courbe est réalisée en activant la trace du point C et en déplaçant le point R mobile sur la droite horizontale passant par O.



2- L'abscisse de C est égale à la différence $OR - RH$. Or $OR = x = r\alpha$ et $RH = r\sin\alpha$ (on doit imaginer un cercle trigonométrique de centre A avec le point $I(1, 0)$ sur la demi-droite $[AR)$ et orienté dans le sens horaire). On en déduit que $x_C = OR - RH = r\alpha - r\sin(\alpha) = r(\alpha - \sin\alpha)$.
Pour l'ordonnée de M, ce n'est pas très différent : $y_C = RA - AT$ or $RA = r$ et $AT = r\cos\alpha$ (je n'ai pas indiqué le point T sur cette nouvelle figure pour ne pas la surcharger - se reporter à la figure de l'énoncé), d'où

$$C(\alpha)\begin{cases} x = r(\alpha - \sin\alpha) \\ y = r(1 - \cos\alpha) \end{cases}$$

Cette courbe est appelée « roulette » ou plutôt cycloïde. Voir à son propos dans Mathcurve [2].

3- Le programme qui va nous permettre d'estimer la longueur de cette arche de cycloïde utilise la formule que nous venons de mettre au point pour les coordonnées d'un point de cette courbe. L'intervalle $[0, 2\pi]$ est découpé en n tranches avec des angles de la forme $\alpha_i = \frac{2i\pi}{n}$, l'entier i décrivant l'ensemble $\{0, 1, 2, \cdots, n\}$.

2. https://www.mathcurve.com/courbes2d/cycloid/cycloid.shtml

```
from math import *
n=int(input("Combien de subdivisions? "))
diam=700
r=diam/2
long=0
alpha=0
x1,y1=0,0
for i in range(1,n+1):
    alpha+=2*pi/n
    x2=r*(alpha-sin(alpha))
    y2=r*( 1   -cos(alpha))
    long+=sqrt((x1-x2)**2+(y1-y2)**2)
    x1,y1=x2,y2
print("Longueur estimée = {}mm".format(long))
print("Rapport avec diam = {}".format(long/diam))
```

```
Combien de subdivisions? 100
Longueur estimée = 2799.884908283351mm
Rapport avec diam = 3.99983558326193

Combien de subdivisions? 1000000
Longueur estimée = 2799.9999999988186mm
Rapport avec diam = 3.999999999998312
```

Les exécutions du 1er programme (en bleu ci-dessus) montrent qu'avec $n = 100$ subdivisions, on s'approche déjà beaucoup d'une valeur qui parait la limite atteignable $2800mm$, soit 4 fois le diamètre. Avec $n = 1\,000\,000$ subdivisions, la valeur obtenue ne s'écarte de cette limite hypothétique que d'un milliardième. La preuve mathématique de ce résultat (la longueur d'une branche de cycloïde vaut 4 fois le diamètre du cercle qui l'a engendrée) a été donnée au XVIIe siècle par Christopher Wren, l'architecte qui conçut la cathédrale Saint-Paul de Londres.

```
from math import *
n=9 #nombre de subdivisions
diam=700
r=diam/2
long_total=0
diff=1
while long_total<4*diam-diff/2:
    n+=1
    long,alpha=0,0
    x1,y1=0,0
    for i in range(1,n+1):
        alpha+=2*pi/n
        x2=r*(alpha-sin(alpha))
        y2=r*( 1   -cos(alpha))
        long+=sqrt((x1-x2)**2+(y1-y2)**2)
        x1,y1=x2,y2
    long_total=long
print("à partir de n={}, la longueur est à moins de {}mm près".format(n,diff/2))
```

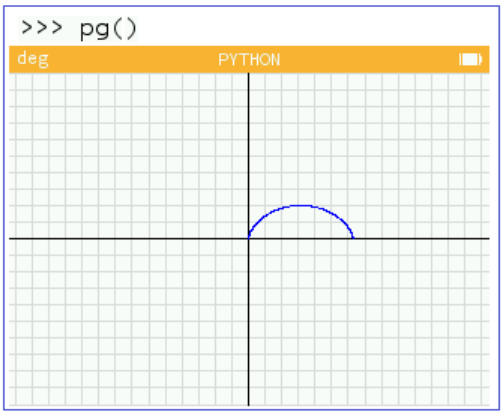
```
à partir de n=48, la longueur est à moins de 0.5mm près
```

Pour savoir à partir de quelle valeur de n la longueur estimée dépasse $2799,5mm$ (l'arrondi au mm le plus proche doit donner la valeur limite), on peut tâtonner. J'ai préféré écrire un 2e programme, assez proche du 1er (voir ci-dessus), qui détermine directement ce nombre : il faut au moins 48 subdivisions pour obtenir cette précision du millimètre.

Un petit prolongement pour ceux qui, munis d'une calculatrice Numworks, se demandent si on ne pourrait pas tracer directement cette courbe au moyen d'un programme en Python. La question est intéressante : on dispose d'une formule pour cette courbe (les coordonnées x et y du point C en fonction du paramètre $\alpha \in [0, 2\pi]$) et le module Kandinsky permet de changer la couleur d'un pixel de l'écran (composé de 320 pixels sur sa longueur et de 222 pixels sur sa hauteur). La question est comment faire ?
La 1re approche que l'on peut faire de ce problème est de s'inspirer (essayer de comprendre) d'un programme existant. Dans le Workshop de Numworks, je trouve le programme `parametric_graphing.py` créé par `ferr0fluidmann`, avec cette description explicite (pour initiés seulement)
`Plot parametric graphs with parametric_graph(x_function, y_function, tMin, tMax)`
Le moins qu'on puisse dire est qu'il n'est pas évident de s'y retrouver dans ce programme ! J'ai cependant adapté les fonctions `x_function` et `y_function` pour qu'elles traduisent notre formule et j'ai modifié le nom (`pg` à la place de `parametric_graph`). Le résultat se voit dans la fenêtre : en tapant `pg()`, on obtient bien un arc de cycloïde.

```
from math import *
from kandinsky import *
tMin = 0
tMax = 2*pi
def x_function(x):
  return x-sin(x)

def y_function(x):
  return 1-cos(x)

def pg(it=1000,fx=x_function,fy=y_function,tmax=tMax,tmin=tMin,xmin=-14.4,xmax=14.4,ymin=-10,ymax=10):
  col=color(0,0,255)
  blk=color(0,0,0)
  gry=color(220, 220, 220)
  y_0=int(ymax*221/(ymax-ymin))
  x_0=int(xmin*319/(xmin-xmax))
  gridx = int(xmin)
  while gridx <= xmax:
    x=int(319.0*(gridx-xmin)/(xmax-xmin))
    for y in range(222):
      set_pixel(x,y,gry)
    gridx+=1
  gridy = int(ymin)
  while gridy <= ymax:
    y=int(221.0*(gridy-ymin)/(ymax-ymin))
    for x in range(320):
      set_pixel(x,y,gry)
    gridy+=1
  for x in range(320):
    set_pixel(x,y_0,blk)
  for y in range(222):
    set_pixel(x_0,y,blk)
  for t in range(it):
    T = tmin+t*(tmax-tmin)/float(it)
    try:
      X=fx(T)
      Y=fy(T)
      pixel_x=int((X-xmin)*319/(xmax-xmin))
      pixel_y=int((Y-ymax)*221/(ymin-ymax))
      set_pixel(pixel_x,pixel_y,col)
    except:
      X=0
      Y=0
```

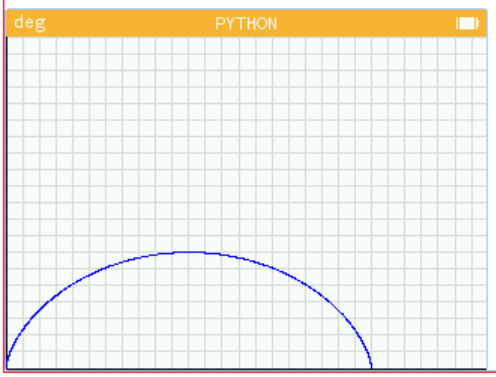
```
>>> pg()
```



modification

```
def x_function(x):
  return 3.5*(x-sin(x))
def y_function(x):
  return 3.5*(1-cos(x))
```

```
>>> pg(xmin=0,xmax=28.8,ymin=0,ymax=20)
```



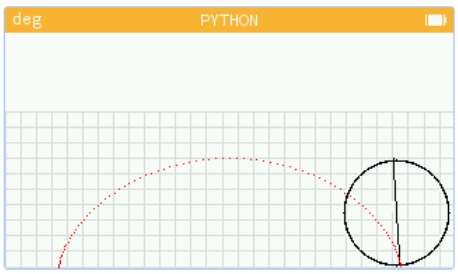
Les deux premières boucles `While` servent à tracer le quadrillage (en gris, `gry`). Les deux boucles `For` qui suivent servent à tracer les axes de coordonnées (en noir, `blk`). La dernière boucle trace la courbe (en bleu, `col`) : l'intervalle $[0, 2\pi]$ est subdivisé en `it`$= 1000$ points et, pour chaque point, le programme calcule ses coordonnées et les attribue à un pixel de l'écran.

Pour voir si j'ai bien compris le fonctionnement du programme, j'adapte les paramètres envoyés à cette fonction pour que soit affichée une courbe correspondant à un diamètre de $700mm$ (ici il y aura 7 carreaux) : Je modifie légèrement les fonctions pour que le diamètre fasse 7 et non 2. Je double les valeurs maximales par défaut et annule les valeurs minimales pour supprimer les cadrans où on a $x < 0$ ou $y < 0$. Pour cela, je lance le programme avec la commande `pg(xmin=0,xmax=28.8,ymin=0,ymax=20)`. Le résultat est encadré en rouge.

On pourrait faire mieux et tracer la roue qui se déplace avec la trajectoire du point C qui se dessine progressivement. Il faut fabriquer une fonction qui fera office de *timer*, pour ralentir le tracé et donner l'illusion d'une roue qui roule (un exemple de timer est la fonction `temporise` qui se trouve dans mon programme `horloge.py`, disponible dans le Workshop). Il faut aussi une fonction qui trace un cercle de centre et de rayon donné (il y a une telle fonction dans `horloge.py`). Bien que ces perfectionnements sortent des limites de ce corrigé, je me suis amusé à programmer une telle animation.

Je n'ai pas eu besoin du timer car, en fait, les tracés demandés prennent suffisamment de temps pour que la roue avance à vitesse lente (même très lente). Ce qui est contraignant dans ce module Kandinsky, c'est qu'une figure tracée doit être effacée (donc tracée à nouveau mais avec un stylo de couleur blanche) avant d'être retracée ailleurs. Sinon, les figures se superposent. Sans ces effacements, après avoir tracé le cercle dans ses 1000 positions intermédiaires, on ne verrai qu'une grosse tâche noire. Il faut aussi retracer le quadrillage si on veut le garder intact tout le temps de l'animation.
Comme ce programme n'est pas destiné à être adapté à d'autres situations, j'ai simplifié le programme initial de `ferr0fluidmann`. J'ai ajouté des fonctions pour tracer un cercle (pour la roue) et un segment (je voulais'« voir » la roue tourner, il fallait donc au moins 1 rayon, j'ai tracé un diamètre). J'ai relégué à de mini-fonctions le rôle de transformer les coordonnées réelles en coordonnées pour l'écran : par exemple, le centre du cercle a, au début, pour coordonnées réelles $(0;3.5)$ et pour l'écran $(0;183)$. Les 222 pixels de hauteur de l'écran étant divisé en 20 unités, comme $222-\frac{3,5\times222}{20}=183,15$, le pixel est arrondi à 183.
Une dernière chose : j'écris le programme dans le Workshop de Numworks car c'est beaucoup plus facile (copié/collés autorisés, utilisation du clavier de l'ordinateur) mais l'exécution de cette animation n'est visible à chaque étape que sur la calculatrice. Dans le Workshop, on ne voit que la phase finale (celle que j'ai mis sur l'illustration ci-dessous).

```
from math import *
from kandinsky import *
def cercle(x0,y0,r,c,e):
  for i in range(2*e):
    xd=x0-int((r-i*0.5)/sqrt(2))
    xf=x0+int((r-i*0.5)/sqrt(2))
    for x in range(xd,xf+1):
      x1=x
      y1=y0+int(sqrt((r-i*0.5)**2-(x-x0)**2))
      set_pixel(x,y1,c)
      for j in range(3):
        x2=x0+y1-y0
        y2=y0+x0-x1
        set_pixel(x2,y2,c)
        x1,y1=x2,y2

  def seg(xa,ya,xb,yb,c):
    if abs(yb-ya)<abs(xb-xa):
      if xb<xa:
        xa,xb=xb,xa
        ya,yb=yb,ya
      m=(yb-ya)/(xb-xa)
      p=ya-m*xa
      for i in range(xb-xa):
        set_pixel(int(xa+i),int(m*(xa+i)+p),c)
    else:
      if yb<ya:
        ya,yb=yb,ya
        xa,xb=xb,xa
      m=(xb-xa)/(yb-ya)
      p=xa-m*ya
      for i in range(yb-ya):
        set_pixel(int(m*(ya+i)+p),int(ya+i),c)
```




```
xmax,ymax=28.8,20

def px(x):
  return int(x*319/xmax)

def py(y):
  return int((ymax-y)*221/ymax)

def pg():
  col=color(255,0,0)
  blk=color(0,0,0)
  wht=color(255,255,255)
  gry=color(220, 220, 220)
  r=3.5
  X1,Y1,X2,Y2=0,0,1,1
  for x in range(320):
    set_pixel(x,221,blk)
  for i in range(100):
    gridx,gridy=0,10
    while gridx<=xmax:
      x=int(319.0*gridx/xmax)
      for y in range(111,222):
        set_pixel(x,y,gry)
      gridx+=1
    while gridy<=ymax:
      y=int(221.0*gridy/ymax)
      for x in range(320):
        set_pixel(x,y,gry)
      gridy+=1
    seg(px(X1),py(Y1),px(X2),py(Y2),wht)
    cercle(px((X1+X2)/2),py(3.5),px(r),wht,1)
    T=i*pi/50
    X1=r*(1+T-sin(T))
    Y1=r*(1-cos(T))
    X2=r*(1+T+sin(T))
    Y2=r*(1+cos(T))
    seg(px(X1),py(Y1),px(X2),py(Y2),blk)
    cercle(px((X1+X2)/2),py(3.5),px(r),blk,1)
    for t in range(i):
      T=t*pi/50
      X=r*(1+T-sin(T))
      Y=r*(1-cos(T))
      set_pixel(px(X),py(Y),col)
```