Applications numériques aux oraux de l'agrégation interne de mathématiques

Sébastien Peronno

www.joliesmaths.fr/sup/num-agreg/

27 mars 2026

Plan

Outils et enjeux : rapport de jury

« L'enseignement des mathématiques nécessite l'utilisation d'outils informatiques, qu'il s'agisse de logiciels prêts à l'emploi ou d'algorithmes résolvant des problèmes de manière effective. (...)

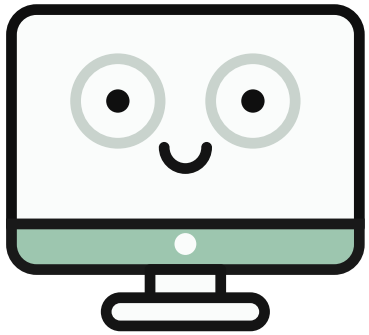
« L'usage pertinent des outils numériques pour illustrer ou clarifier un thème est à favoriser et est quasiment inévitable pour le traitement de certains sujets qui énoncent des aspects algorithmiques ou des calculs approchées d'intégrales (...) Dans les thèmes de probabilités, il serait agréable de voir des propositions de simulation informatique de variables aléatoires réelles, des calculs approchés de probabilités ou d'espérances. »

Rapport du jury 2024

Outils et enjeux : AgregOS

- Un système d'exploitation avec LibreOffice, Geogebra, Python (Pyzo), Scilab, wxMaxima, etc.
- Documents disponibles : livres numériques, rapports de jury, etc.

Outils et enjeux : apport du numérique



- L'impasse sur l'informatique ?
- Meilleure compréhension d'algorithmes
- Clarté de l'exposé, curiosité, compétences
- Gains éventuels de fractions de points

Outils et enjeux : points de vigilance

- Ce n'est pas une épreuve de code
- Être capable d'expliquer l'algorithme
- Code compliqué : perte de temps à déboguer
- Conseil : finir par l'informatique

Développement : étude de la fonction gamma

Pour tout $x \in \mathbb{R}_+^*$, on définit : $\Gamma(x) \overset{\text{def}}{=} \int_0^{+\infty} t^{x-1} e^{-t} \, \mathrm{d}t$

Alors :

(a) Γ est bien définie

(b) Γ est C^∞ sur $\mathbb{R}_+^*$ et $\forall k \in \mathbb{N}, \Gamma^{(k)}(x) = \int_0^{+\infty} \ln(t)^k t^{x-1} e^{-t} \, \mathrm{d}t$

(c) Γ est log-convexe (et donc convexe)

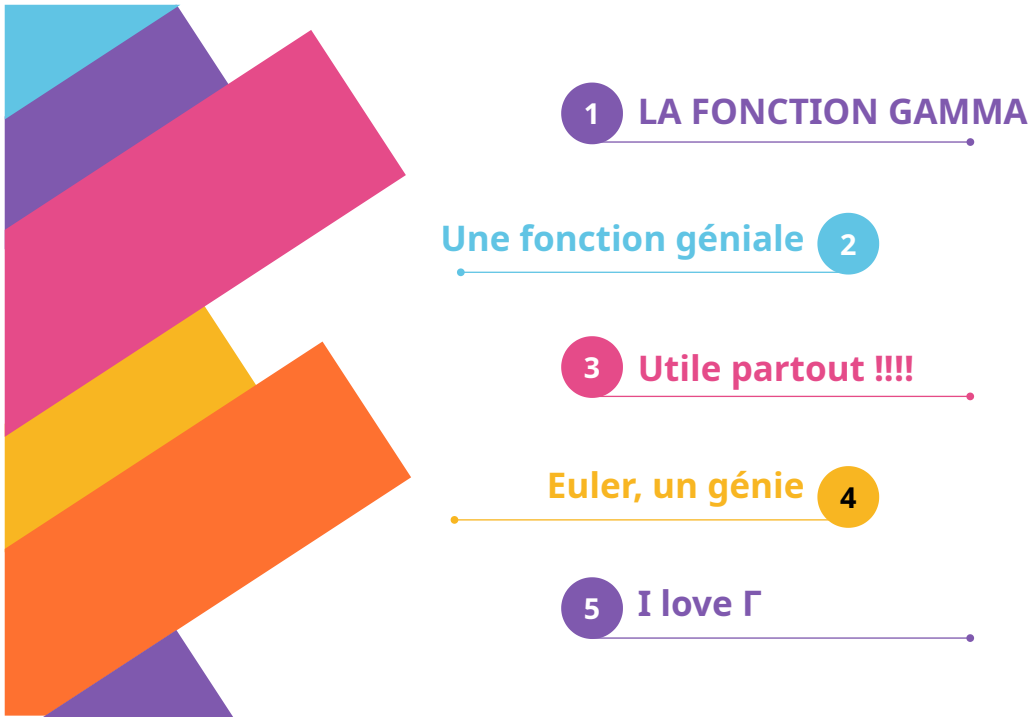
(d) $\forall k \in \mathbb{N}, \Gamma(n+1) = n!$

(e) Γ admet un minimum entre 1 et 2

1

Réaliser une diapo pour :

- Écrire l'énoncé du développement
- Présenter une illustration
- Afficher le résultat d'un code



- Choisir un design sobre

Suites et séries

- Définies explicitement
- Définies par récurrence
- Accélération de convergence
- Représentations graphiques

Définition explicite

	A	B	C
1	1	2	=2^A1*SIN(PI()/2^A1)
2	2	2,828427124746	
3	3	3,061467458921	
4	4	3,121445152258	
5	5	3,136548490546	
6	6	3,140331156955	
7	7	3,141277250933	
8	8	3,141513801144	
9	9	3,141572940367	
10	10	3,141587725277	

```python
from math import *

def u(n):
    return 2**n * sin(pi/2**n)

for k in range(1, 10):
    print(u(k))
```

- Exemple : $u_n = 2^n \sin\left(\frac{\pi}{2^n}\right)$

Définition par récurrence (racine de deux)

	A	B	C
1	u0	2	
2	u1	1,5	=(B1+2/B1)/2
3	u2	1,416666667	
4	u3	1,414215686	
5	u4	1,414213562	
6	u5	1,414213562	
7	u6	1,414213562	
8	u7	1,414213562	

```
u = 2

for _ in range(7):
    print(u)
    u = (u + 2/u) /2
```

- Exemple : $u_0 = 2$, $u_{n+1} = \frac{1}{2}\left(u_n + \frac{2}{u_n}\right)$ (Héron)

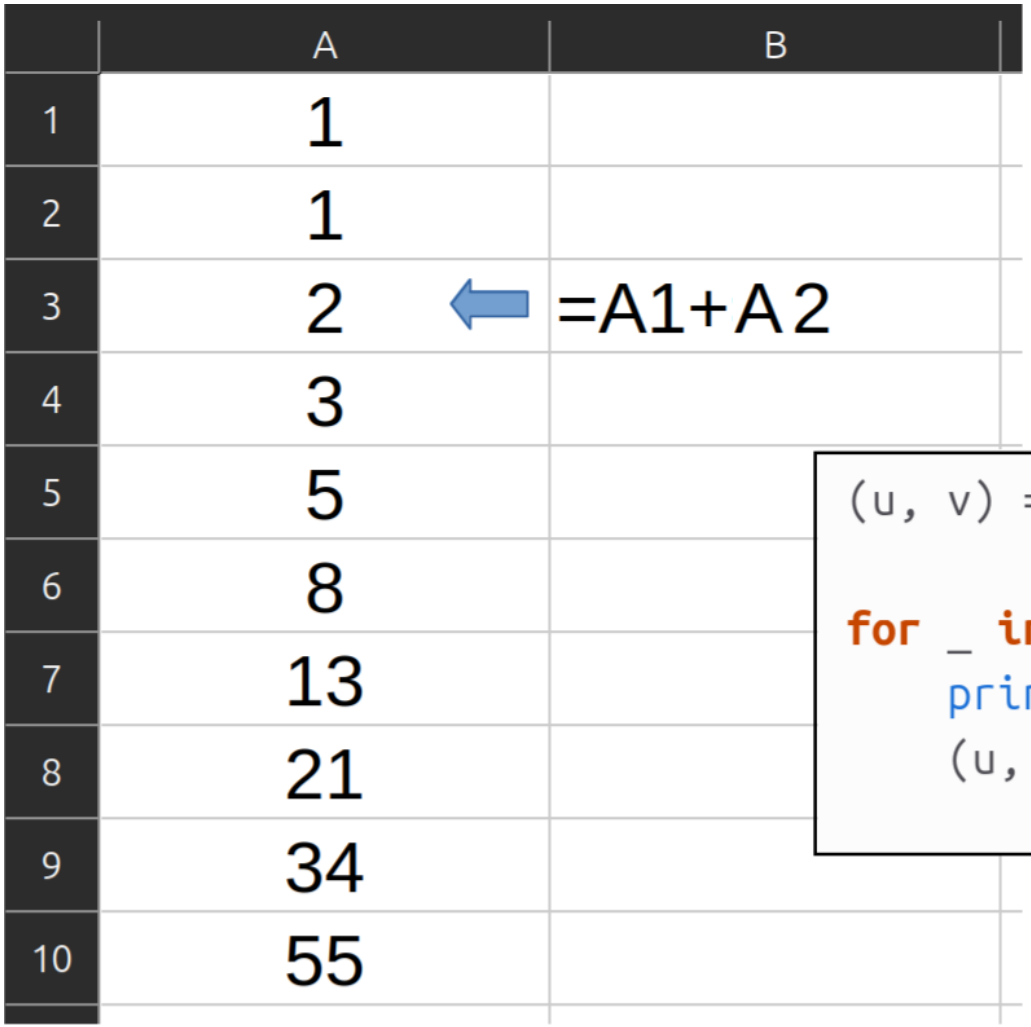
Définition par récurrence (suites de Héron)

	A	B	C	D
1	u0	5	=D2/2	Racine de :
2	u1	3,5	=(B1+D2/B1)/2	10
3	u2	3,178571429		
4	u3	3,162319422		
5	u4	3,16227766		
6	u5	3,16227766		
7	u6	3,16227766		
8	u7	3,16227766		
9				

```
a = 10
u = a/2

for _ in range(7):
    print(u)
    u = (u + a/u) /2
```

- Exemple : $a \in \mathbb{R}_+^*$, $u_0 = \frac{a}{2}$, $u_{n+1} = \frac{1}{2}\left(u_n + \frac{a}{u_n}\right)$ (Héron)

Définition par récurrence (Fibonacci)




```
(u, v) = (1, 1)

for _ in range(10):
    print(u)
    (u, v) = (v, u+v)
```

- Exemple : suite de Fibonacci

Accélération de convergence : par équivalent

Accélération de $u_n = \sum_{k=1}^{n} \frac{1}{k^2}$ grâce à l'équivalent $v_n = u_n + \frac{1}{n}$

```
1 from math import *
2
3 u = sum([1/k**2 for k in range(1, 151)])
4
5 print("u_150 = ", u)
6 print("v_150 = ", u+1/150)
7 print("pi^2/6 =", pi**2/6)
```

```
u_150 =  1.6382895730215048
v_150 =  1.6449562396881714
pi^2/6 = 1.6449340668482264
```

Accélération de convergence : méthode d'Aitken (Delta-2)

Théorème (Aitken)

Soit $(u_n)_{n\in\mathbb{N}}$ une suite qui converge vers L avec un rapport $|\lambda| \in]0;1[$ (vitesse géométrique). Alors la suite $(v_n)_{n\in\mathbb{N}}$ définie par $v_n := u_n - \dfrac{(u_{n+1}-u_n)^2}{u_{n+2}-2u_{n+1}+u_n}$ converge vers L plus rapidement.

Remarque. Avec la convergence lente $|\lambda| = 1$, l'accélération fonctionne tant que l'on a $\left|\frac{u_{n+1}-L}{u_n-L}\right| < 1$

Exemple. Accélération de $u_n = \sum_{k=0}^{n} \dfrac{(-1)^k}{2k+1}$ $\quad (\lim u_n = \frac{\pi}{4})$

Accélération de convergence : méthode d'Aitken (Delta-2)

```python
from math import *

(N, u, v) = (50, [1], [])

for k in range(1, N+3):
    a = (-1)**k / (2*k + 1)
    u.append(u[k-1] + a)

for k in range(N+1):
    b = u[k] - (u[k+1]-u[k])**2 / (u[k]+u[k+2]-2*u[k+1])
    v.append(b)

print(f"u{N}  =", u[-3])
print(f"v{N}  =", v[-1])
print("pi/4 =", pi/4)
```

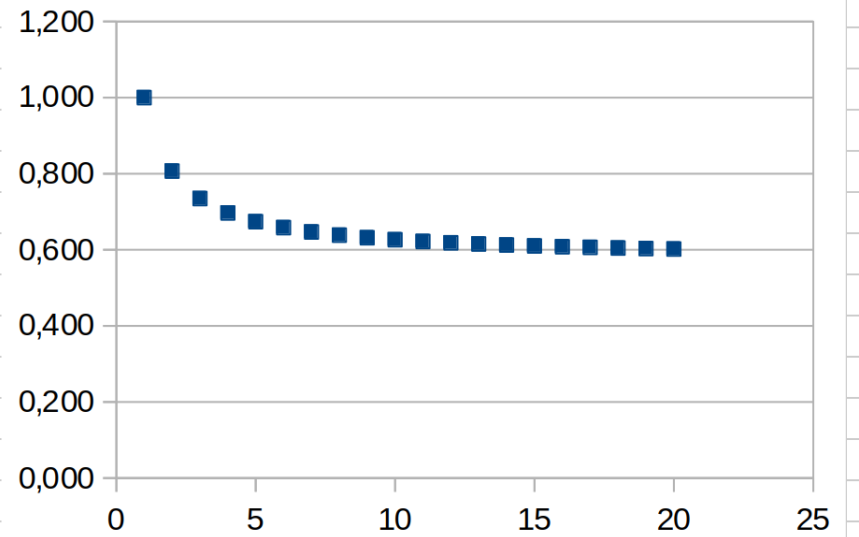
```
u50  = 0.7902996532467627
v50  = 0.7853986076903773
pi/4 = 0.7853981633974483
```

Représentations graphiques : avec tableur

	A	B	C	D	E	F	G
1	1	1,000	1,000		=B1-LN(A1)		
2	2	1,500	0,807		=1/A2+B1		
3	3	1,833	0,735				
4	4	2,083	0,697				
5	5	2,283	0,674				
6	6	2,450	0,658				
7	7	2,593	0,647				
8	8	2,718	0,638				
9	9	2,829	0,632				
10	10	2,929	0,626				
11	11	3,020	0,622				
12	12	3,103	0,618				
13	13	3,180	0,615				
14	14	3,252	0,613				
15	15	3,318	0,610				
16	16	3,381	0,608				
17	17	3,440	0,606				
18	18	3,495	0,605				
19	19	3,548	0,603				
20	20	3,598	0,602				

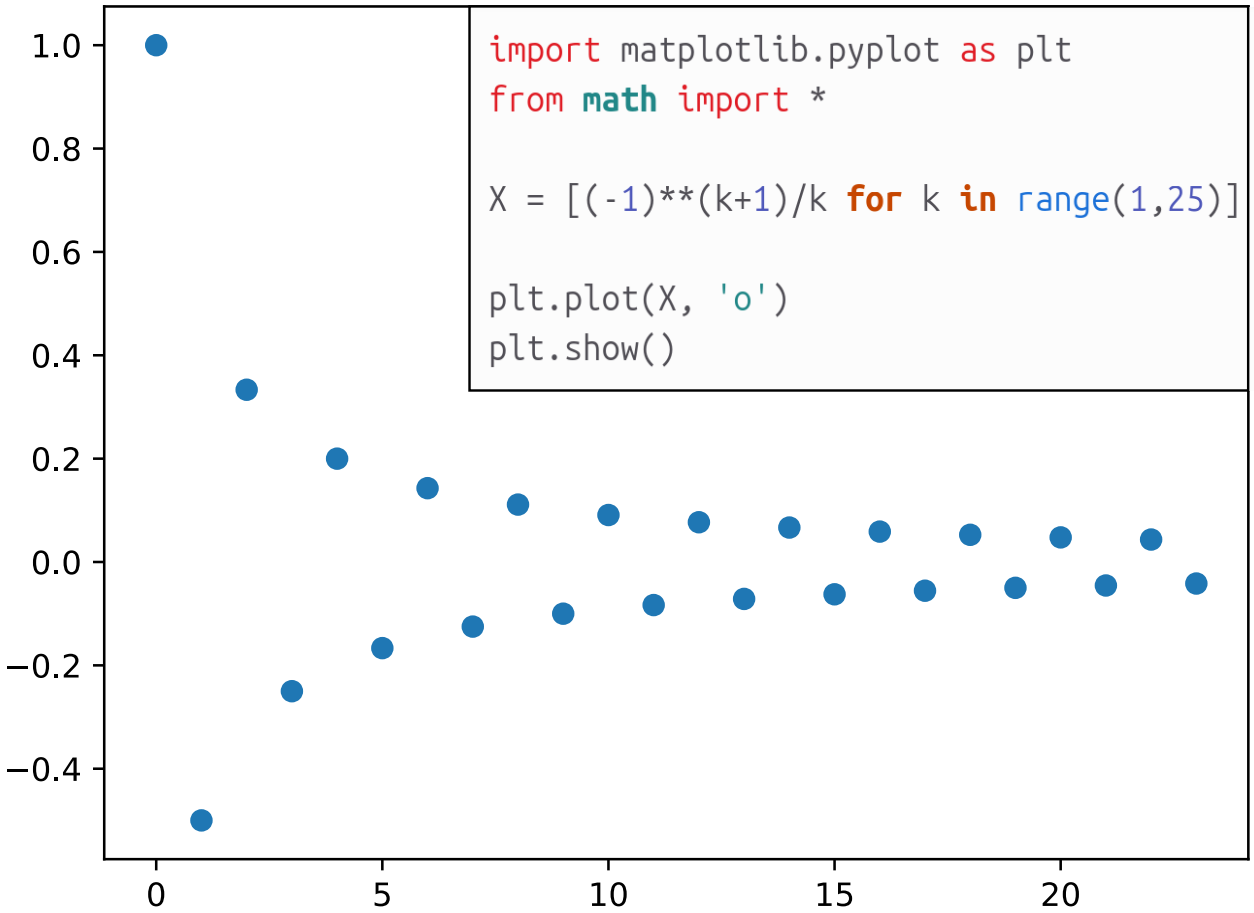
Format : Cellules > Nombres > Code de Format : 0,000

Insertion > Diagramme > XY Dispersion > Série de Données en colonnes



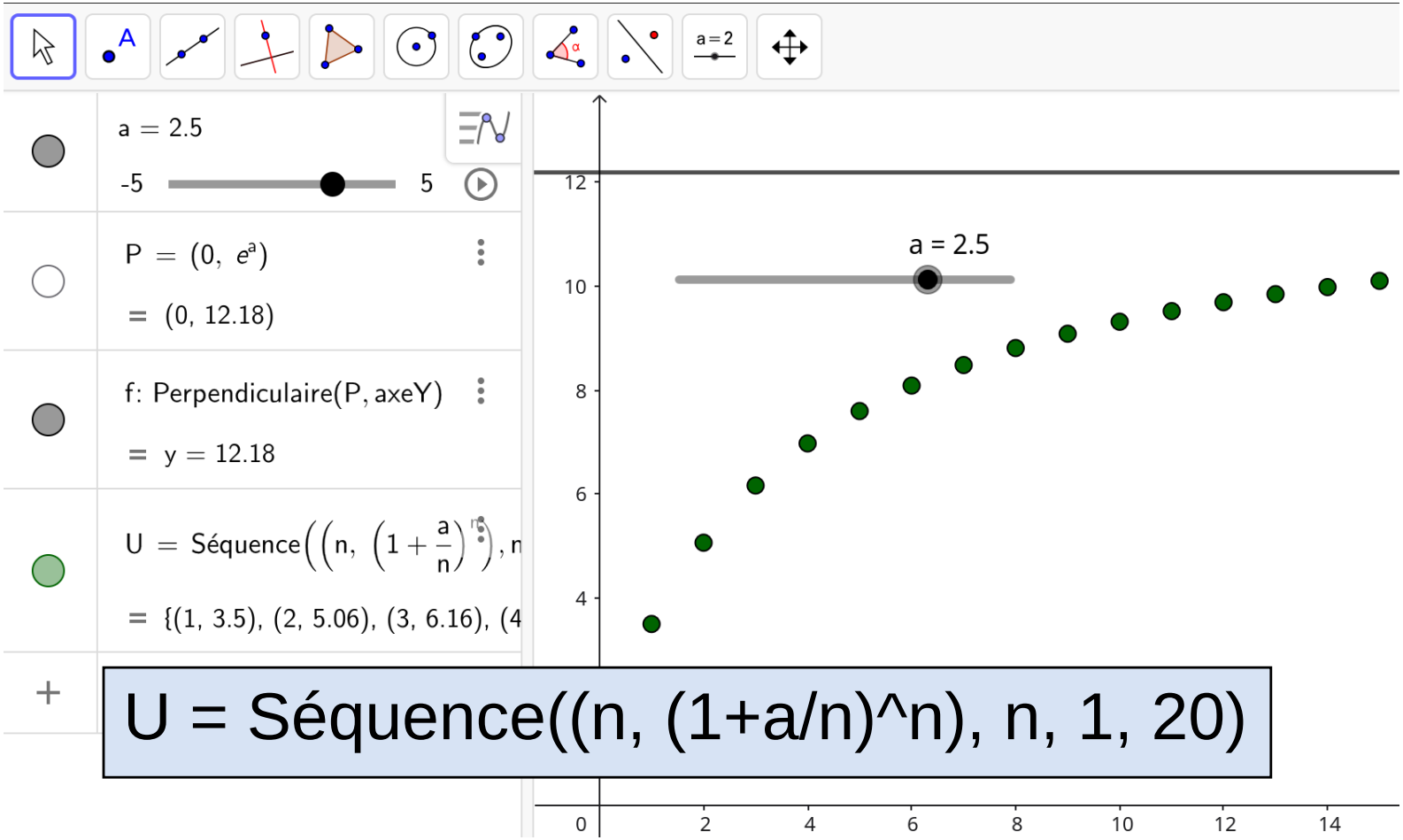
- Convergence de $\sum_{k=1}^{n} \frac{1}{k} - \ln n$ vers $\gamma \approx 0{,}577$ (Euler-Mascheroni)
- Tableur : rapide, petit nombre de termes

Représentations graphiques : avec Python



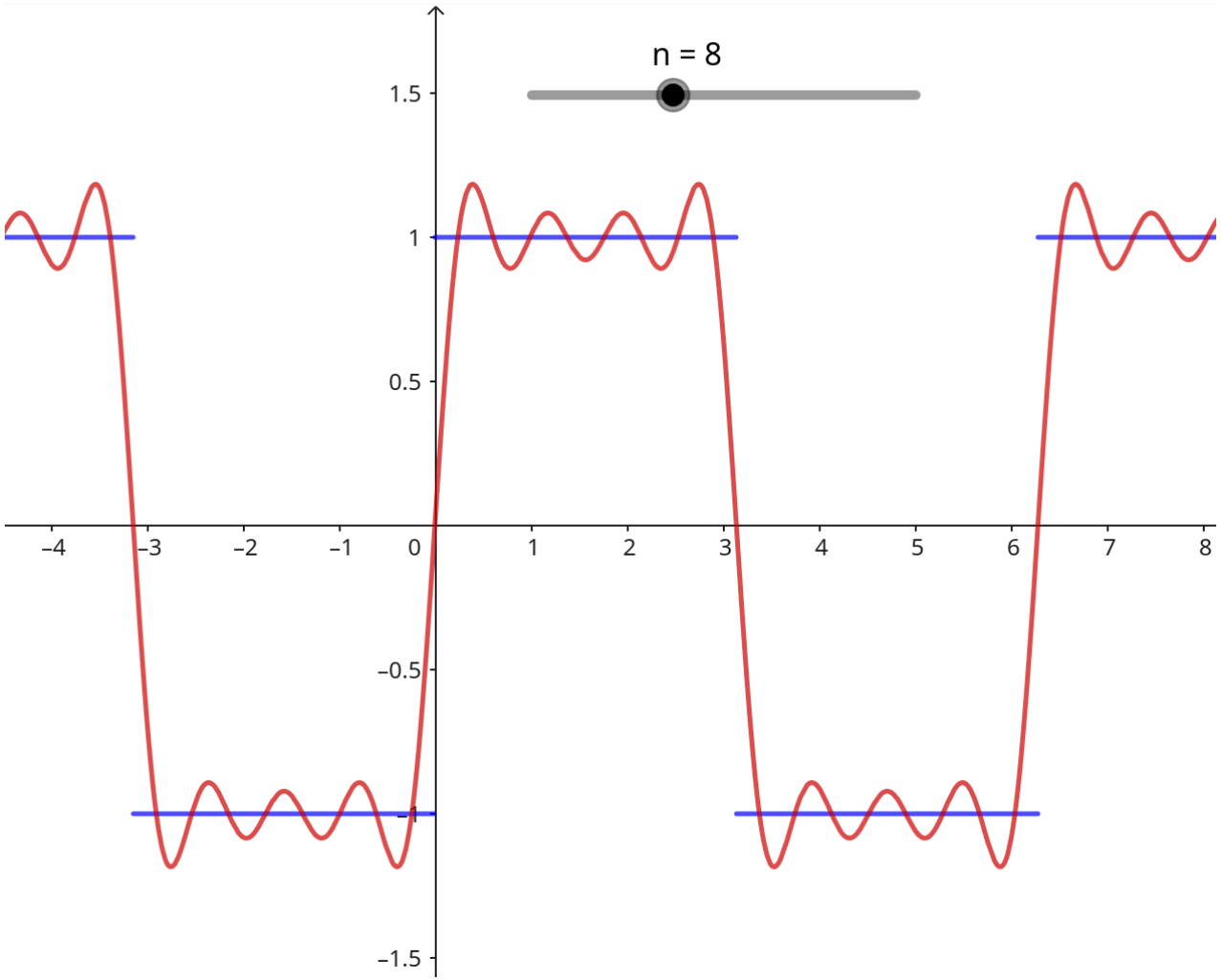
- Représentation de la suite $u_k = \dfrac{(-1)^{k+1}}{k}$
- Python : définition complexe, grand nombre de termes

Représentations graphiques : avec Geogebra



- Convergence de la suite $u_n = \left(1 + \frac{a}{n}\right)^n$ vers e^a
- Geogebra : paramètre variable

Représentations graphiques : avec Geogebra



- Signal carré pair et sa série de Fourier d'ordre 8

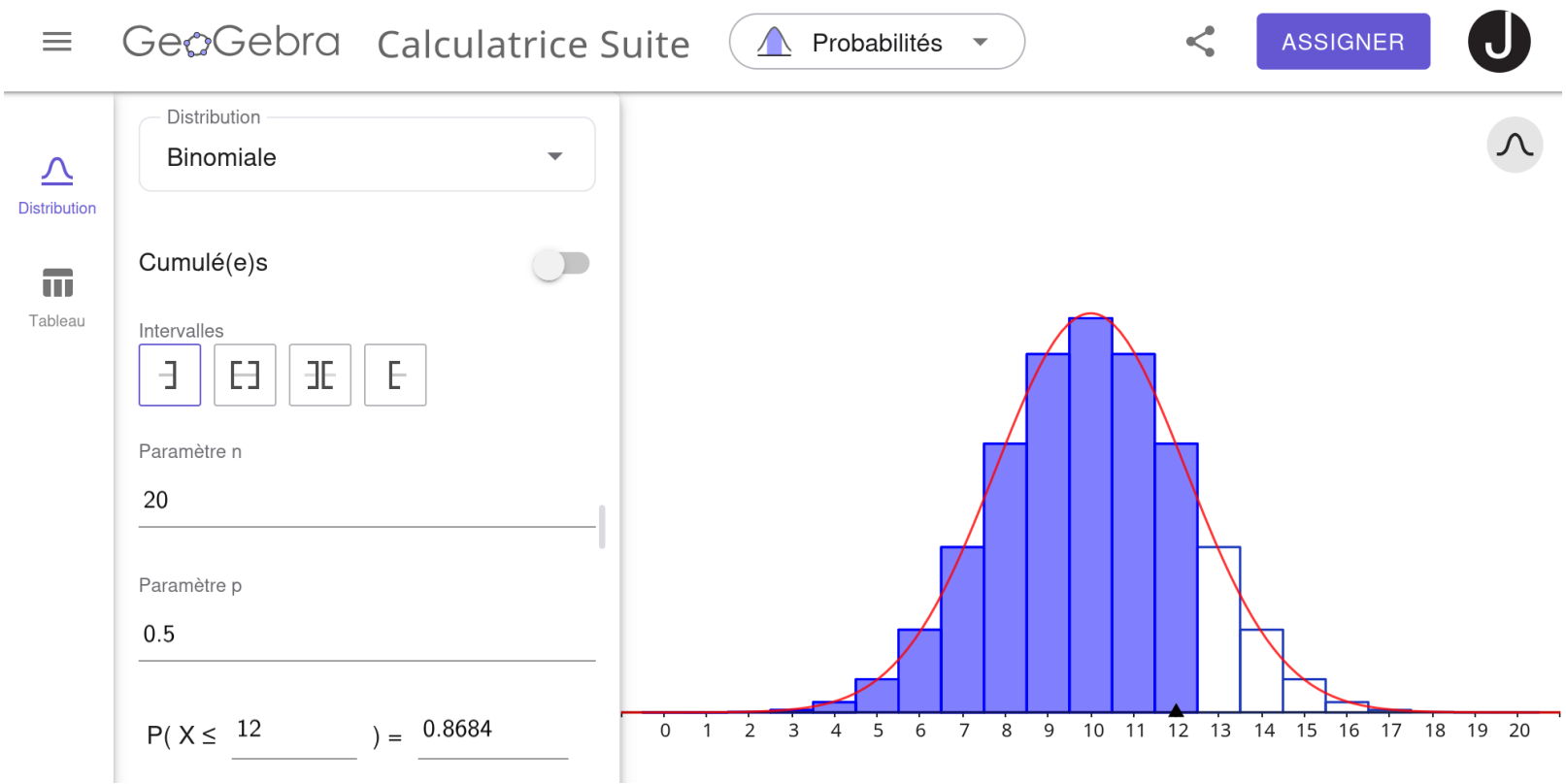
Tracé du signal carré pair et sa série de Fourier d'ordre 8

- Créer un curseur n=8, entier variant de 1 à 20
- Saisir une fonction permettant un calcul modulaire sur les réels : `md(x,y) = y*((x/y) - floor(x/y))`
- Tracer `f(x) = Si(-pi < md(x,2*pi) < pi, 1, -1)`
- Saisir `s(x) = Somme(4 / ((2m+1)*pi) * sin((2m+1)*x), m, 0, floor((n-1)/2))`
- Clic droit sur plan : `axeX:axeY > 3:1`, zoomer et ajuster la fenêtre

Probabilités et statistiques : plan

- Représentations des lois
- Simulations

Représentations des lois



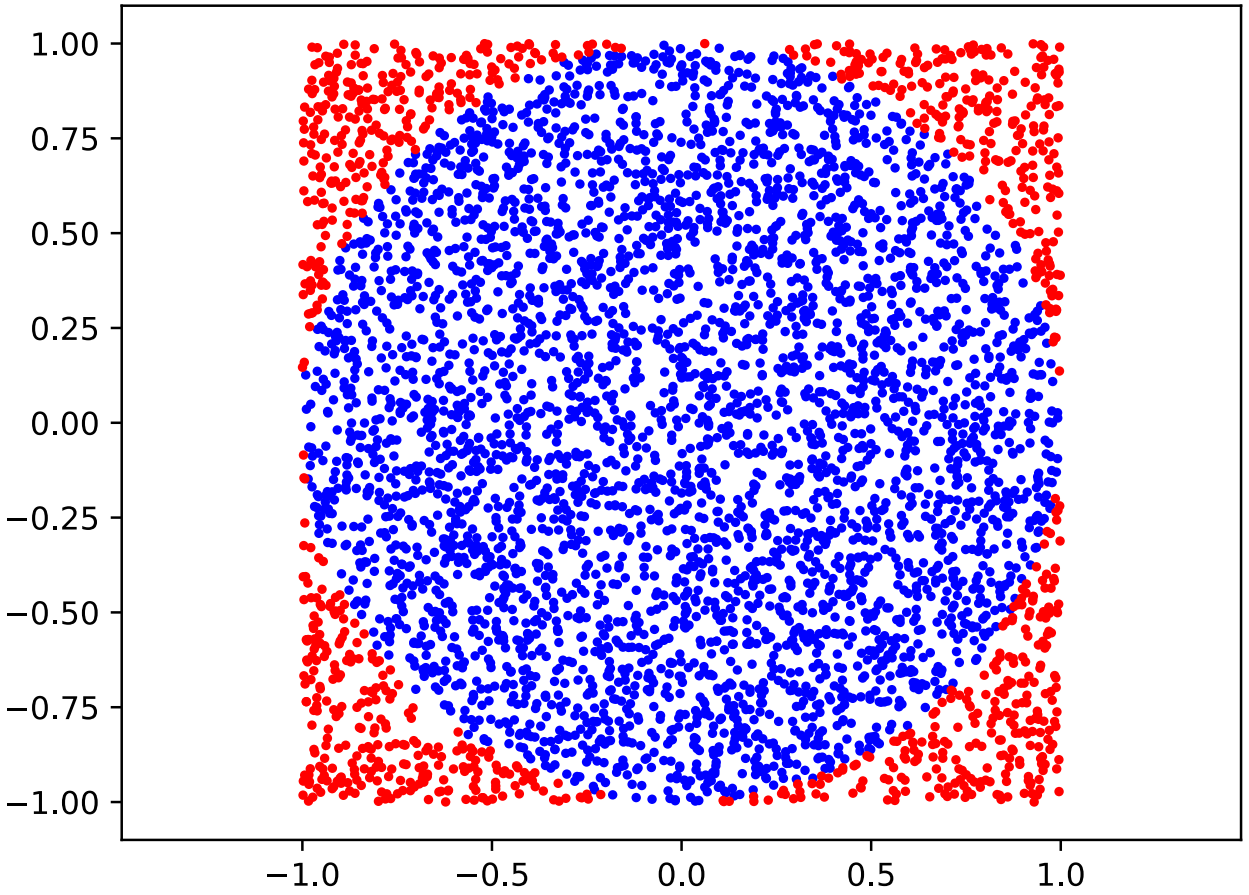
- Le module « Probabilités » de Geogebra permet de représenter facilement toutes les lois usuelles.

Simulations : avec tableur

	A	B	C	D	E	F	G
1	Dé 1	Dé 2	S				
2	3	3	6	=ALEA.ENTRE.BORNES(1;6)			
3	6	1	7		=A3+B3		
4	4	4	8			S	Sorties
5	5	5	10	=NB.SI(C$2:C$100;F5)		2	1
6	6	2	8			3	4
7	2	6	8			4	7
8	3	1	4			5	10
9	2	6	8			6	13
10	6	5	11			7	17
11	5	4	9			8	14
12	2	1	3			9	13

- Les fonctions `ALEA.ENTRE.BORNES` et `NB.SI` sont utiles pour simuler avec un tableur.

Simulations : avec Python

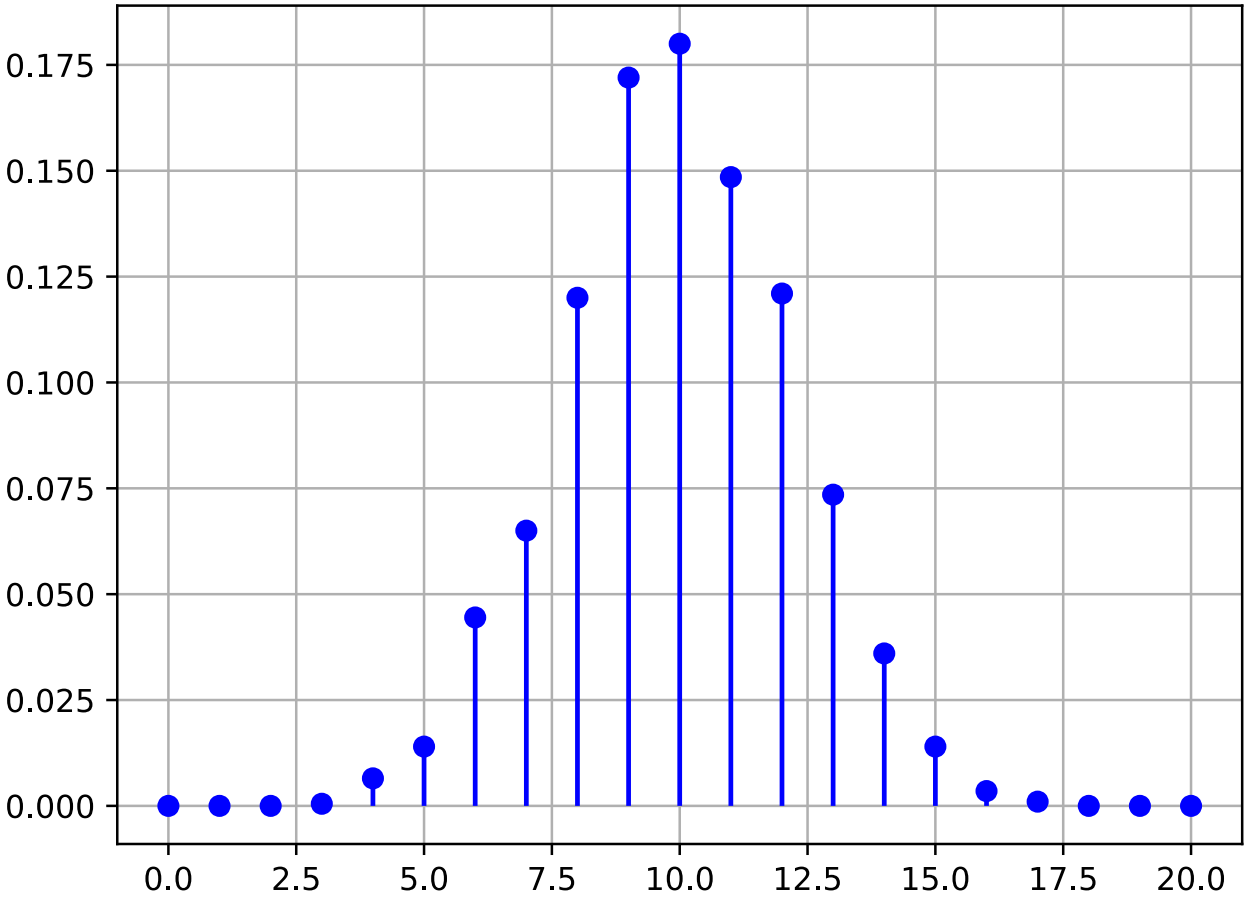


- Méthode de Monte Carlo pour approximer π

Simulations : avec Python

```python
from math import *
import random as rd
import matplotlib.pyplot as plt

N, Xi, Yi, Xe, Ye = 5000, [], [], [], []

for k in range(N):
    x = 1-2*rd.random()
    y = 1-2*rd.random()
    if(x**2+y**2 < 1):
        Xi.append(x)
        Yi.append(y)
    else:
        Xe.append(x)
        Ye.append(y)

plt.plot(Xi, Yi, 'o', c='blue', ms="2")
plt.plot(Xe, Ye, 'o', c='red', ms="2")
plt.axis('equal')
plt.show()
```

Simulations : avec Python



- Planche de Galton (lois binomiale et normale)

Simulations : avec Python

```python
from math import *
import numpy as np
import matplotlib.pyplot as plt
import random as rd

n, b = 20, 2000   # clous, billes
C = np.zeros(n+1, dtype=int) # n+1 compartimts vides

for i in range(b):
    k = 0
    for i in range(n):
        k += rd.randint(0,1) # +1 = a droite
    C[k] += 1

X = range(0, n+1)
plt.stem(X, C/b, 'b-', 'bo', " ")

plt.grid()
plt.show()
```

Algèbre et géométrie

- Arithmétique
- Algèbre linéaire
- Figures

Arithmétique : algorithme d'Euclide

L'algorithme d'Euclide pour trouver le PGCD : la base !

```
a, b = 36, 20

while(b!=0):
    q = a//b
    r = a%b
    print(a, "=", q, "*", b, "+", r)
    (a, b) = (b, r)

print("Le PGCD est :", a)
```

```
36 = 1 * 20 + 16
20 = 1 * 16 + 4
16 = 4 * 4 + 0
Le PGCD est : 4
```

Arithmétique : algorithme d'Euclide étendu

L'algorithme d'Euclide étendu : nécessaire pour l'identité de Bézout.

Exemple. Trouver $\{(u, v) \in \mathbb{Z} \times \mathbb{Z},\ 112u + 85v = 1\}$

$$
\begin{aligned}
112 &= 1 \times 85 + 27 \\
85 &= 3 \times 27 + 4 \\
27 &= 6 \times 4 + 3 \\
4 &= 1 \times 3 + 1 \\
3 &= 3 \times 1 + 0
\end{aligned}
$$

On obtient une solution particulière $(u_0\,, v_0) = (-22\,; 29)$

r	u	v	q
112	1	0	$\times$
85	0	1	$\times$
27	1	-1	1
4	-3	4	3
3	19	-25	6
1	-22	29	1

L'ensemble solution est $\{(112(85k - 22), 85(29 + 112k)), k \in \mathbb{Z}\}$

Arithmétique : algorithme d'Euclide étendu

Version récursive de l'algorithme

```python
def bezout(a,b):
    if(b==0):
        return a, 1, 0
    d, u, v = bezout(b,a%b)
    return d, v, u-(a//b)*v

print(bezout(112, 85))
```

```
(1, -22, 29)
```

version "détaillée" téléchargeable sur www.joliesmaths.fr/sup/num-agreg/

Théorème des restes chinois

Soient $n_1, \ldots, n_k$ des entiers deux à deux premiers entre eux et $n = \prod_{i=1}^{k} n_i$. Alors, pour tous entiers $a_1, \ldots, a_k$, il existe un entier x, unique modulo n, tel que :

$$\left\{ \begin{array}{lll} x & \equiv & a_1 \quad [n_1] \\ \ldots & & \\ x & \equiv & a_k \quad [n_k] \end{array} \right. \qquad (S)$$

Exemple. Trouver un x tel que
$x \equiv 4\ [5]\ ;\ x \equiv 6\ [8]\ ;\ x \equiv 3\ [9]$

Solution.
$x \equiv 4 \times 72 \times (-2) + 6 \times 45 \times (-3) + 3 \times 40 \times (-2) \equiv 174\ [360]$

Arithmétique : équations diophantiennes

```python
import numpy as np

def bezout(a,b):
    if(b==0):
        return a, 1, 0
    d, u, v = bezout(b, a%b)
    return (d, v, u-(a//b)*v)

A, N = [4, 6, 3], [5, 8, 9]
r, n = len(A), np.prod(N)
Q = [n//N[i] for i in range(r)]
Y = [bezout(Q[i],N[i])[1] for i in range(r)]
x, eg = 0, "x = "

for i in range(r):
    x += (A[i]*Q[i]*Y[i])
    eg += f"{A[i]}*{Q[i]}*{Y[i]} + "

x = x%n
print(f"{eg[0:-3]} = {x} [{n}]")
```

RSA : transmission du nombre secret "1901"

```
(p, q) = (101, 113)
n = p*q
(c, d) = (7927, 3463)
x = 1901

print("x =", x)
print("n =", n)
print("x^c mod n =", x**c%n)
print("(x^c)^d mod n =", ((x**c%n)**d)%n)
```

```
x = 1901
n = 11413
x^c mod n = 2028
(x^c)^d mod n = 1901
```

Définition

Soit $A \in \mathcal{M}_n(\mathbb{K})$. On dit que A admet une décomposition LU s'il existe deux matrices L et U de $\mathcal{M}_n(\mathbb{K})$, telles que $A = LU$, où L est triangulaire inférieure à diagonale unité et U est triangulaire supérieure.

A s'écrit alors : $A = LU = \begin{pmatrix} 1 & 0 & \dots & 0 \\ * & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ * & \dots & * & 1 \end{pmatrix} \begin{pmatrix} * & * & \dots & * \\ 0 & * & \ddots & * \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & * \end{pmatrix}$

où $(*)$ sont des éléments à déterminer.

Exemple. $A \begin{pmatrix} 4 & -6 & -1 & 3 \\ -8 & 18 & 0 & -5 \\ 20 & -48 & 2 & 8 \\ -12 & 54 & -13 & 14 \end{pmatrix}$ se décompose :

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ 5 & -3 & 1 & 0 \\ -3 & 6 & -4 & 1 \end{pmatrix} \times \begin{pmatrix} 4 & -6 & -1 & 3 \\ 0 & 6 & -2 & 1 \\ 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

On peut utiliser l'algorithme de Crout.

Algèbre linéaire : décomposition LU

```
1  import numpy as np
2
3  A = np.array([[4,-6,-1,3], [-8,18,0,-5], [20,-48,2,8],
       [-12,54,-13,14]])
4
5  def somme(i,j):
6      return sum ([L[i,k]*U[k,j] for k in range(N)])
7
8  N = len(A)
9  L, U = np.identity(N), np.zeros((N,N))
10
11 for r in range(0,N):
12     for j in range(r,N):
13         U[r,j] = A[r,j] - somme(r,j)
14     for i in range(r+1,N):
15         L[i,r] = 1/U[r,r] * (A[i,r] - somme(i,r))
16     print("\nETAPE", r+1, "\nU =\n", U, "\n\nL =\n", L)
17
18 print("\nA = LU =\n", np.dot(L,U))
```

Figure : ellipse de Steiner

Théorème (ellipse de Steiner)

Soit ABC un triangle non aplati de l'espace affine euclidien. Alors il existe une unique ellipse $(\mathcal{E})$ tangente aux milieux des côtés du triangle.

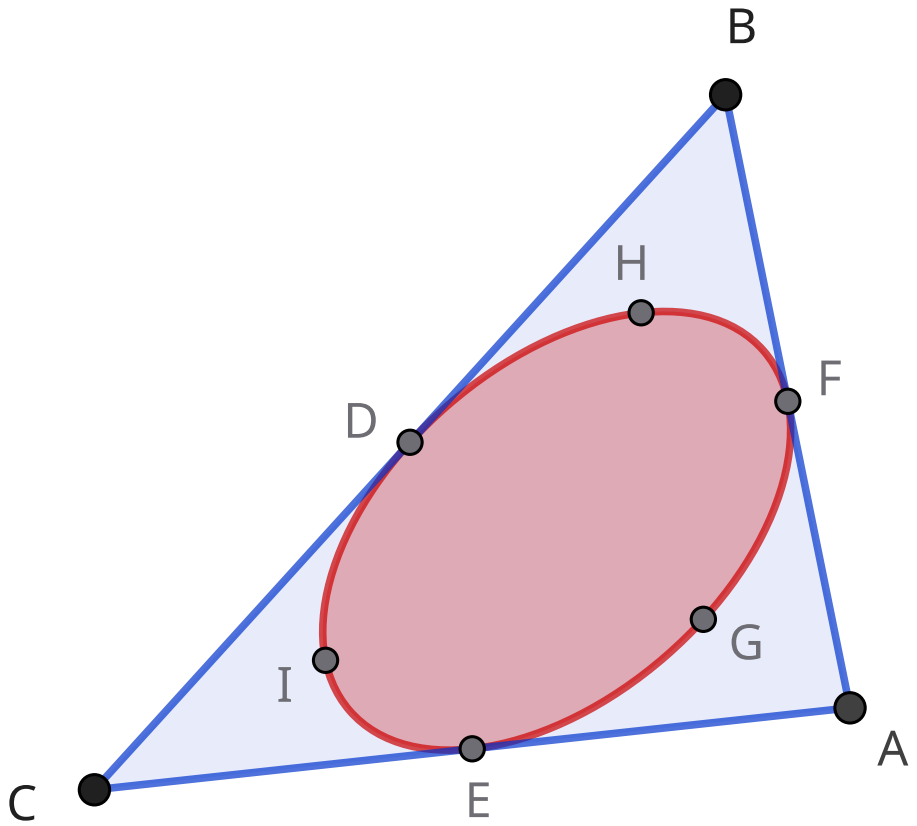
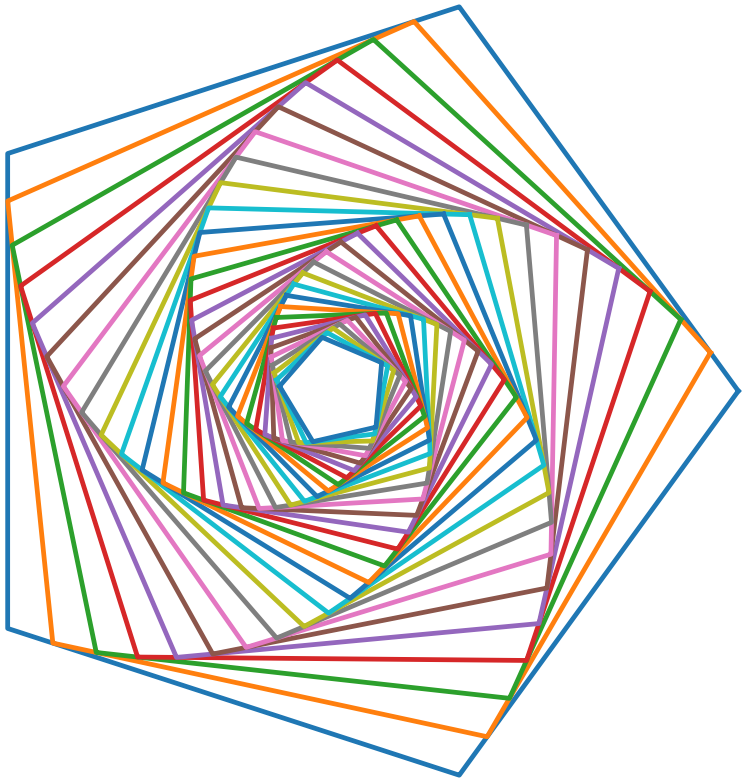


Figure : ellipse de Steiner

1. Cacher la grille et les axes.
2. Avec l'outil `Polygone`, placer 3 points A, B et C non alignés.
3. Cacher les étiquettes des segments a, b, c et du triangle `t1`.
4. Avec l'outil `Milieu ou centre`, placer D, E et F milieux respectifs de $[BC]$, $[AC]$ et $[AB]$.
5. Placer G tel que $\overrightarrow{AG} = \frac{1}{3}\overrightarrow{AD}$. Pour cela, écrire dans le champ de saisie : `G = (2/3 x(A) + 1/3 x(D), 2/3 y(A) + 1/3 y(D))`
   Attention aux notations, G n'est pas l'isobarycentre ici.
6. Placer H en saisissant `H = (2/3 x(B) + 1/3 x(E), 2/3 y(B) + 1/3 y(E))` (on pourra utiliser ⋮ `Dupliquer la saisie` dans le champ de `G`).
7. En utilisant l'outil `Conique passant par cinq points`, tracer l'ellipse.
8. Pour finaliser, cacher l'étiquette de l'ellipse et changer son opacité. Éventuellement, placer le centre O de l'ellipse et placer I tel que $\overrightarrow{CI} = \frac{1}{3}\overrightarrow{CF}$, ou, au contraire cacher G et H.



- « Suite de polygones », *Carnet de voyage en Algébrie* – Philippe Caldero et Marie Peronnier

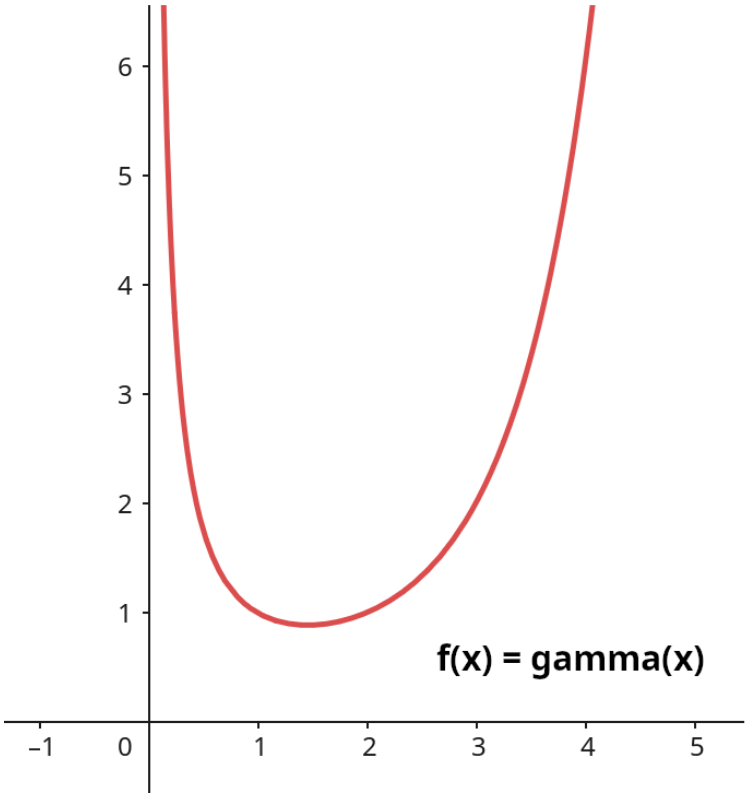
Figure : suite de polygones

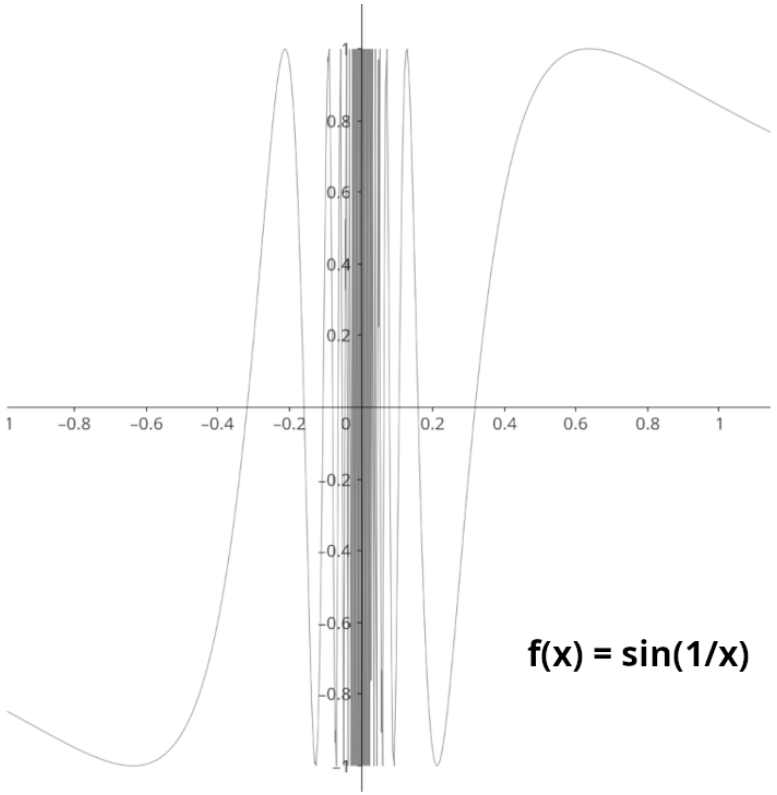
```
1  import matplotlib.pyplot as plt
2  from math import *
3
4  def f(M): # fermer la polyligne
5      return M + [M[0]]
6
7  X = [cos(k*2*pi/5) for k in range(5)]
8  Y = [sin(k*2*pi/5) for k in range(5)]
9  (a, b, n) = (.9, .1, len(X))
10 plt.plot(f(X), f(Y))
11
12 for l in range(30):
13     (Xp, Yp) = ([], [])
14     for k in range(5):
15         Xp = Xp + [a*X[k] + b*X[(k+1)%n]]
16         Yp = Yp + [a*Y[k] + b*Y[(k+1)%n]]
17     plt.plot(f(Xp),f(Yp))
18     (X, Y) = (Xp, Yp)
19
20 plt.axis("equal"); plt.axis("off"); plt.show()
```

Fonctions

- Tracés
- Zéros

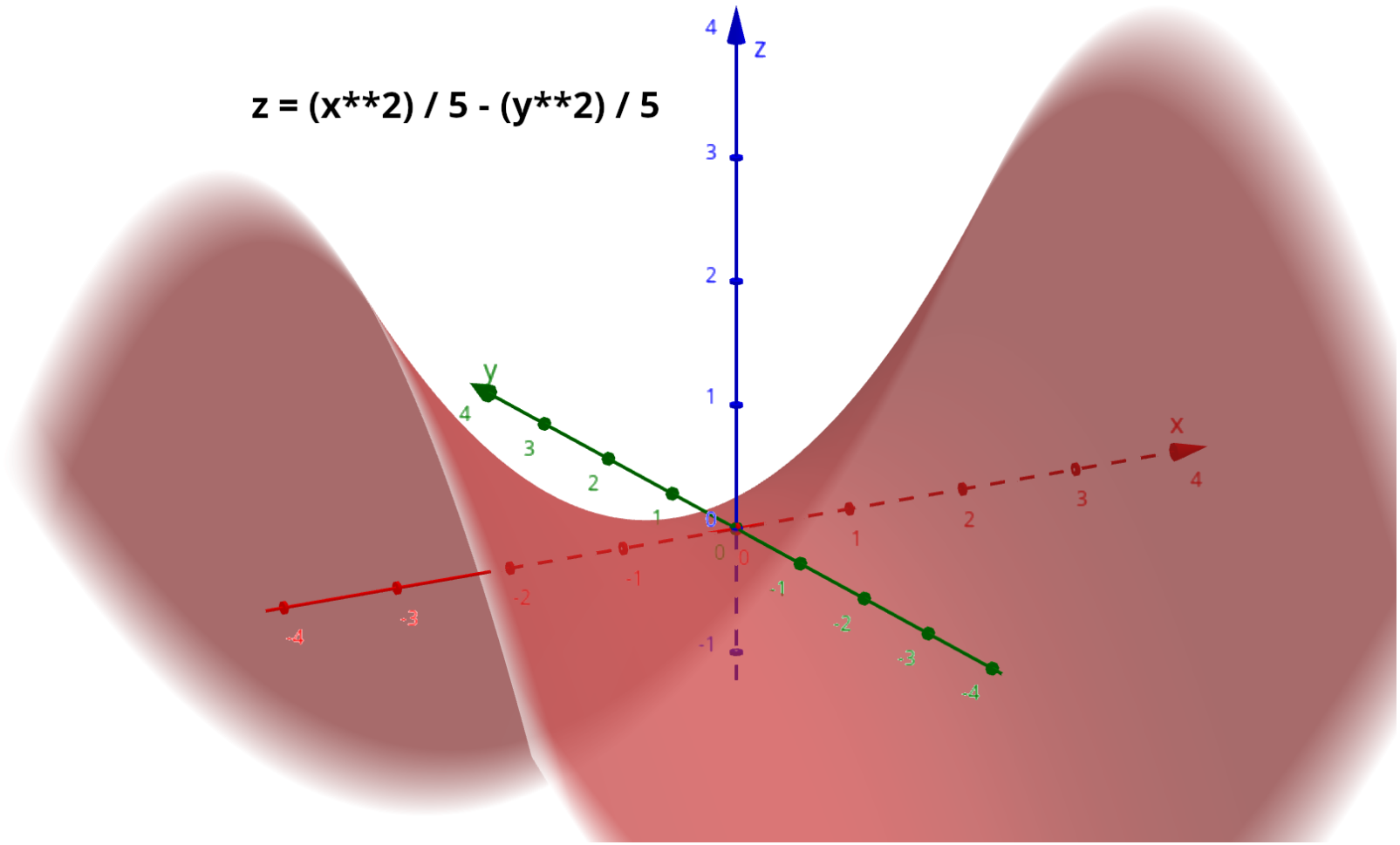
Geogebra : fonctions de $\mathbb{R}$ dans $\mathbb{R}$





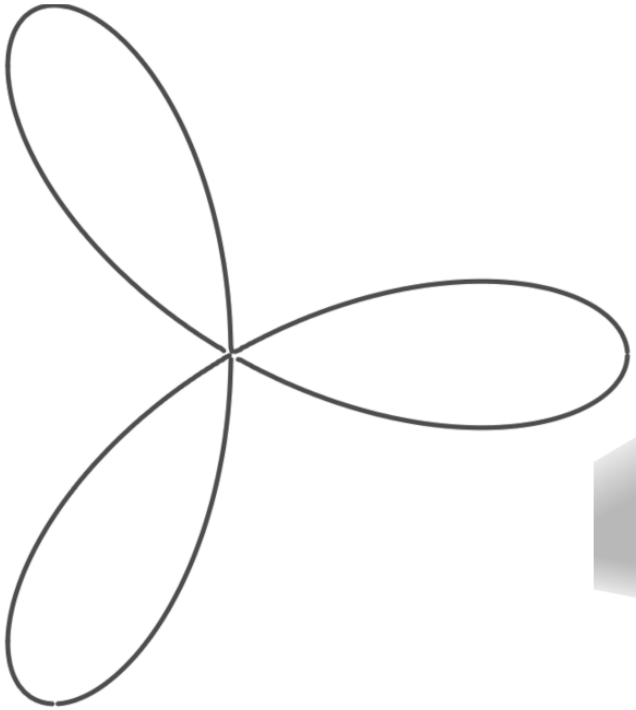
- Fonctions élémentaires et fonctions spéciales (ex : gamma)

Geogebra : fonctions de $\mathbb{R}^2$ dans $\mathbb{R}$

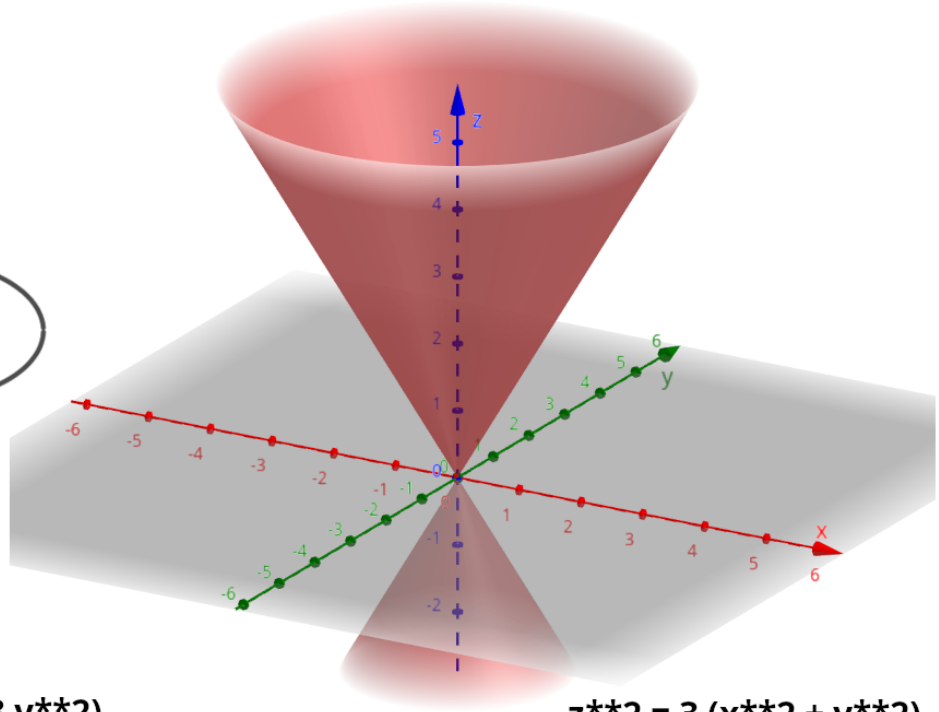


- Surfaces pour des fonctions de type $z = f(x, y)$

Geogebra : équations implicites



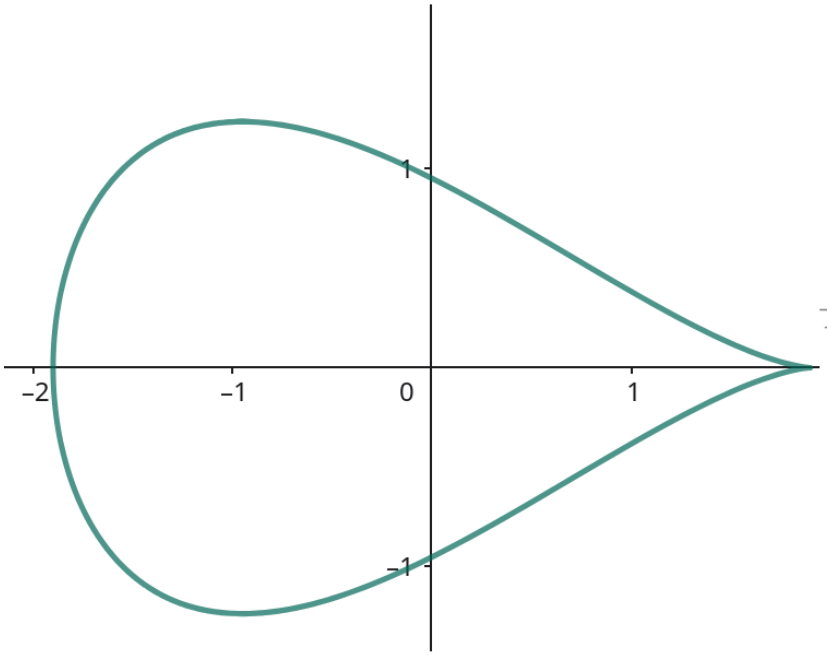
(x2 + y**2)**2 = 4 x (x**2 - 3 y**2)**



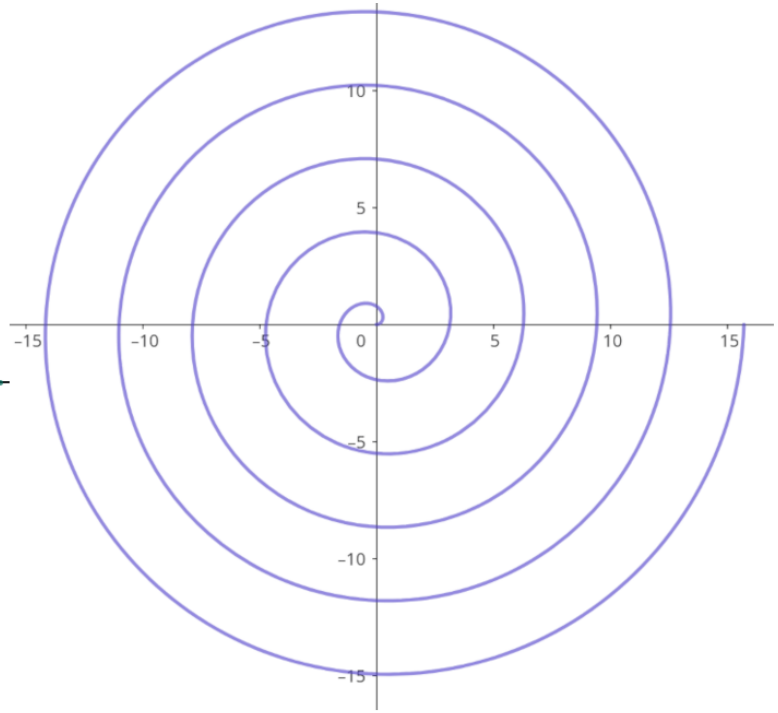
z2 = 3 (x**2 + y**2)**

- Courbes et surfaces définies implicitement

Geogebra : courbes paramétriques



Courbe((a cos(t), a sin(t) sin(t/2)^n), t, 0, 2 pi)



Courbe((t/2; t), t, 0, 10 pi)

- Définition paramétrique : séparateur « , » pour $(x(t), y(t))$
- Définition en polaire : séparateur « ; » pour $(\rho(\theta)\,;\theta)$

Théorème (des valeurs intermédiaires)

Soient I un intervalle de $\mathbb{R}$ et f une fonction continue de I dans $\mathbb{R}$. Alors $f(I)$ est un intervalle.

Démonstration en trois étapes :

- Si $f(a)f(b) \leqslant 0$ alors il existe $c \in [a,b]$ tel que $f(c) = 0$ (théorème de Bolzano)
- Soit y compris entre $f(a)$ et $f(b)$, alors il existe $c \in [a,b]$ tel que $f(c) = y$
- $f(I)$ est intervalle.

Dichotomie

```python
(a, b) = (2, 4)

def f(x):
    return x**2 - 10

if(f(a)*f(b)>0):
    exit("Fonction inutilisable")

for _ in range(20):
    x = (a+b) / 2
    if (f(a)*f(x)<0):
        (a, b) = (a, x)
    else:
        (a, b) = (x, b)

print("Connue   : ", 10**(1/2))
print("Calculée : ", x, "(dichotomie)")
```

```
Connue   :  3.1622776601683795
Calculée :  3.162279129028320 3 (dichotomie)
```

Théorème (Newton)

Soit $(a,b) \in \mathbb{R}^2$, $a < b$. Soit f une fonction $\mathcal{C}^2$ définie sur $[a,b]$ telle que $f(a)f(b) < 0$ et telle que les dérivées première et seconde ne s'annulent pas sur $[a,b]$.
Alors l'équation $f(x) = 0$ admet une unique solution α dans $[a,b]$

De plus, α est la limite de la suite $(x_k)_{k\in\mathbb{N}}$ définie par :

$$\begin{cases} x_0 = a \text{ si } f'f'' < 0 \\ \quad = b \text{ si } f'f'' > 0 \\ x_{k+1} = x_k - \dfrac{f(x_k)}{f'(x_k)} \end{cases}$$

et la convergence de cette suite est quadratique.

Méthode de Newton

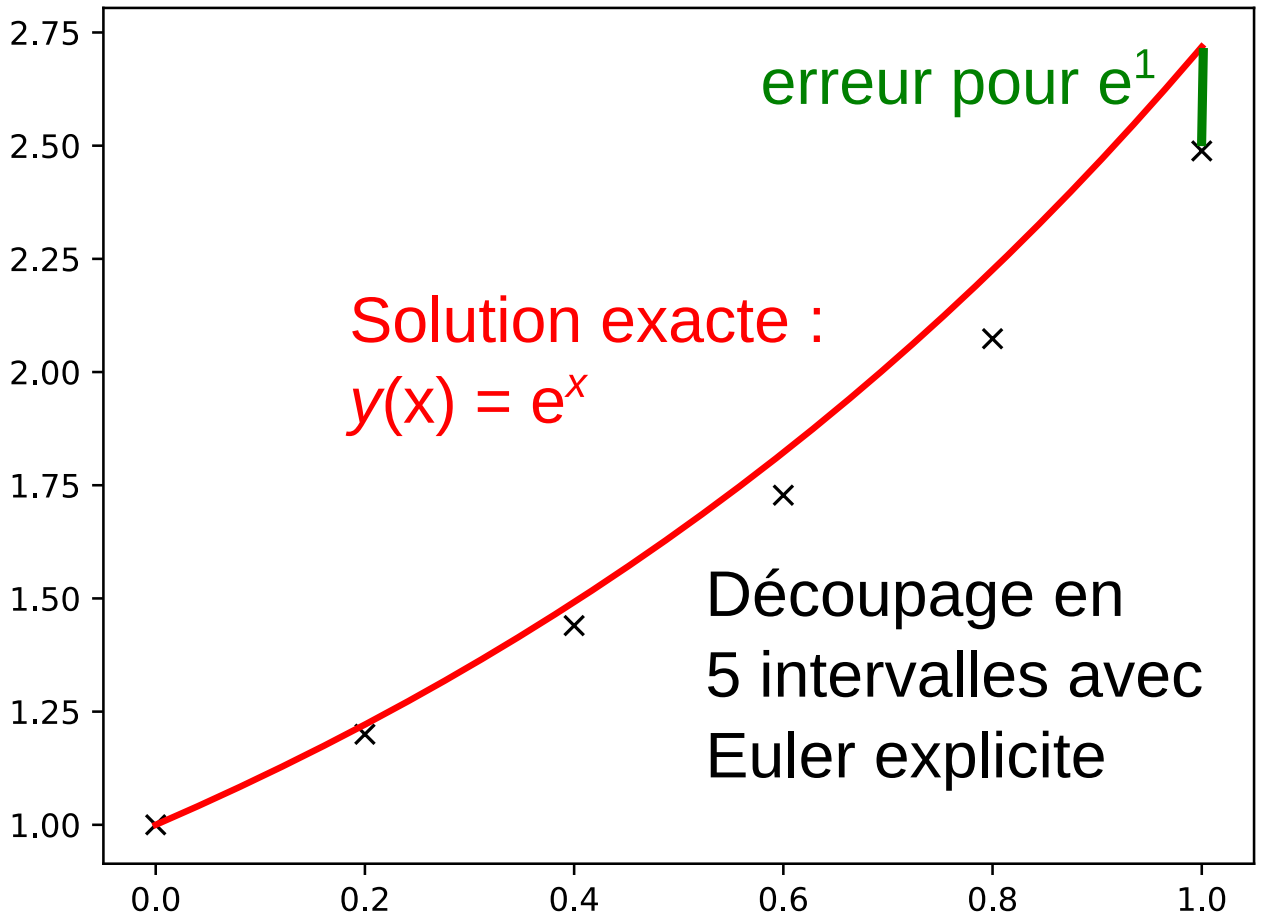
```python
(a, b, x) = (2, 4, 3)

def derf(x):
    return 2*x

def f(x):
    return x**2 - 10

for _ in range(4):
    x = x - f(x)/derf(x)

print("Connue    : ", 10**(1/2))
print("Calculée : ", x, "(Newton)")
```

```
Connue    :  3.1622776601683795
Calculée :  3.1622776601683795 (Newton)
```

Calcul différentiel et intégral

- Méthode d'Euler explicite
- Méthode de Runge-Kutta 4
- Méthode des rectangles

Méthode d'Euler explicite



- Résolution approchée de $y' = y,\ y(0) = 1$ par la méthode d'Euler explicite

Méthode d'Euler explicite

```python
from math import *
import numpy as np

(a, b, N) = (0, 1, 20)  # intervalle, nombre
u = 1   # condition initiale y0
T = np.linspace(a, b, N+1) ;  h = (b-a)/N

def f(t, y):          # y' = f(t, y)
    return y

for k in range(0,N):
    u = u + h*f(T[k], u)

print("Valeur connue :  ", e)
print("Valeur calculée :", u, "avec", N, "intervalles")
print("Erreur :", e-u)
```

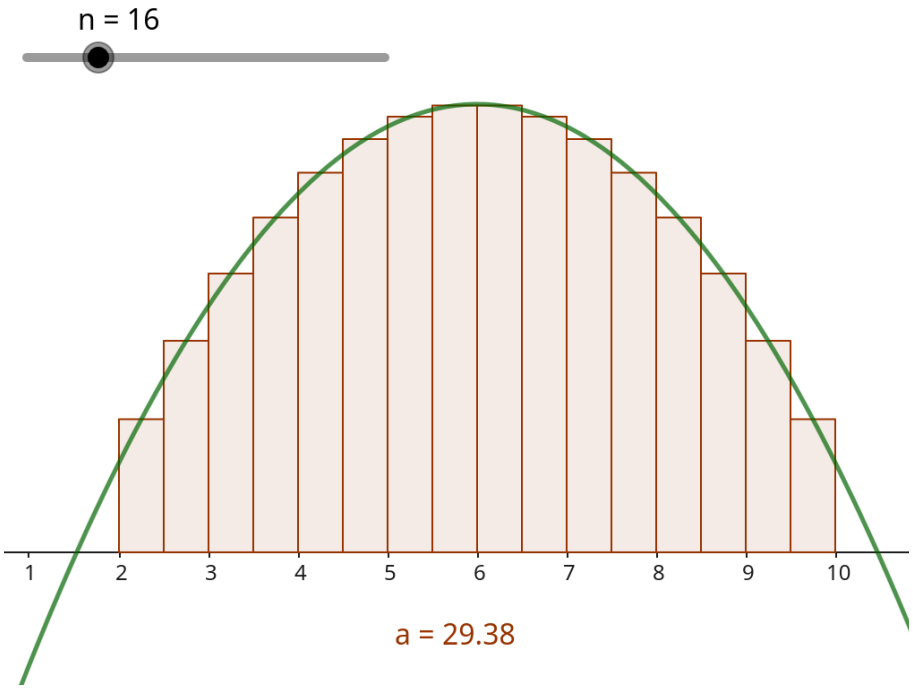
```
Valeur connue :   2.718281828459045
Valeur calculée : 2.6532977051444213 avec 20 intervalles
Erreur : 0.06498412331462378
```

Méthode de Runge-Kutta 4

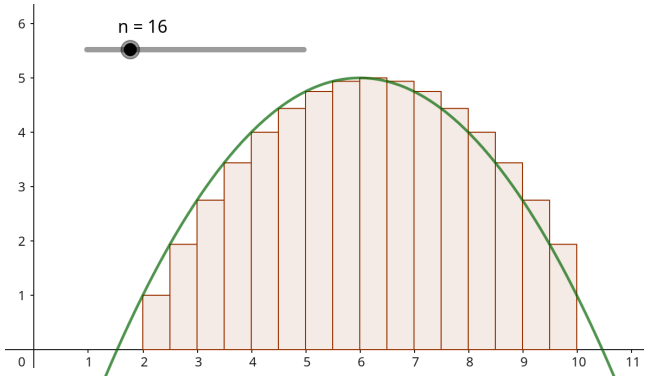
```
11 def RK(t, u):          # Runge-Kutta 4
12     k1 = f(t, u)
13     k2 = f(t+h/2, u+h*k1/2)
14     k3 = f(t+h/2, u+h*k2/2)
15     k4 = f(t+h, u+h*k3)
16     return h/6*(k1 + 2*k2 + 2*k3 + k4)
17
18 for k in range(0,N):
19     u = u + RK(T[k], u)
20
21 (...)
```

```
Valeur connue :   2.718281828459045
Valeur calculée : 2.718281692656335 avec 20 intervalles
Erreur : 1.3580270996627064e-07
```

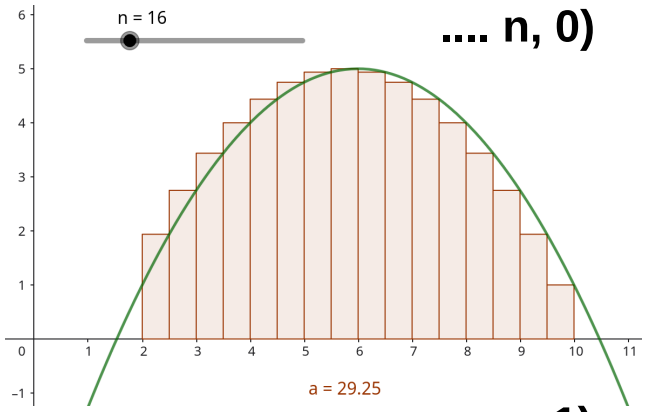
Méthode des rectangles



SommeRectangles(f(x), 2, 10, n, 1/2)



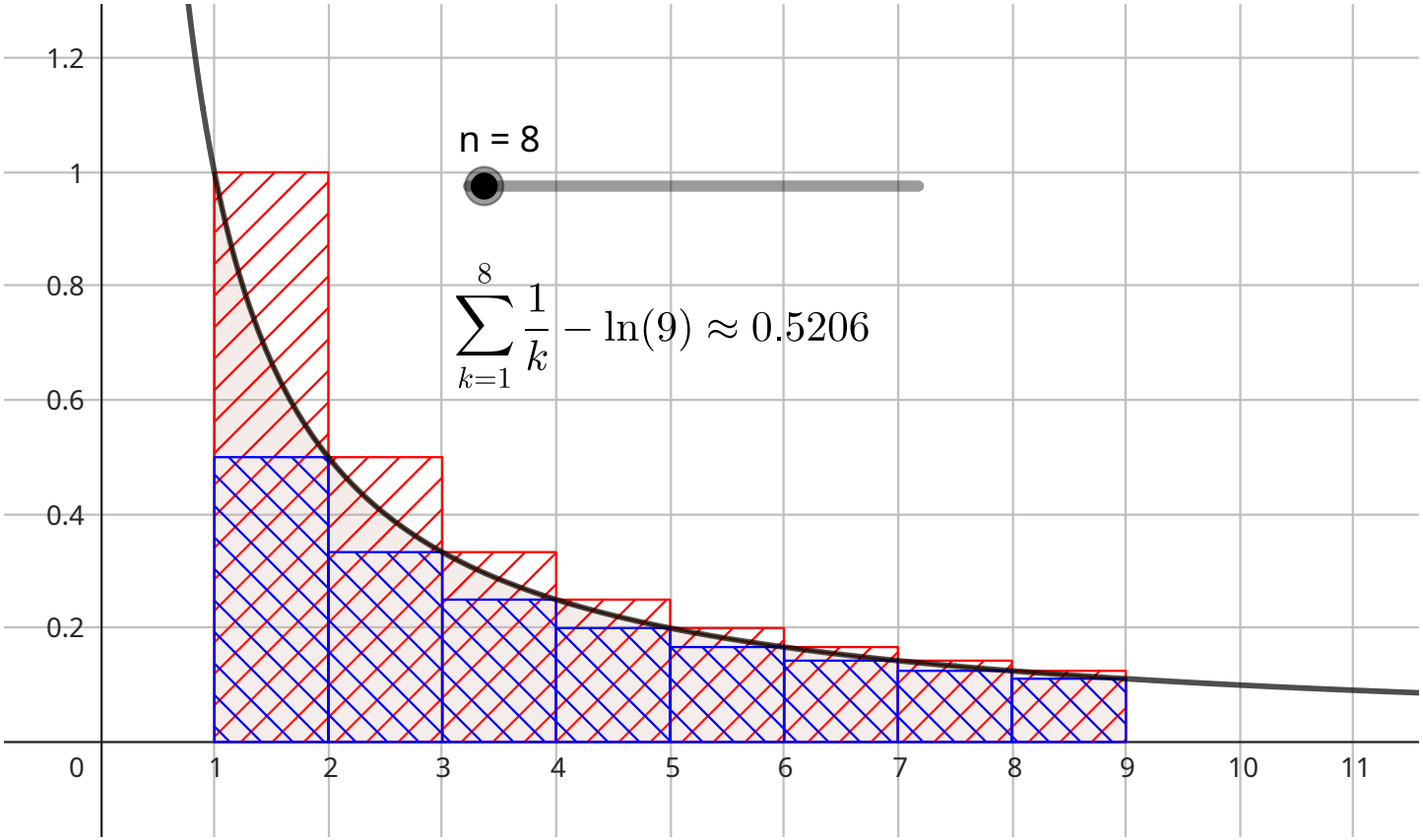
.... n, 0)



.... n, 1)

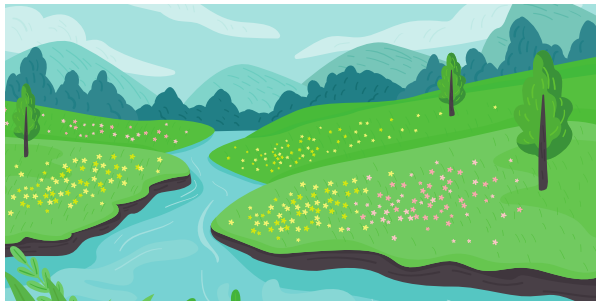
- Calcul approchée d'une intégrale par la méthode des rectangles (milieu, gauche et droite)

Méthode des rectangles



- Illustration de l'encadrement $\displaystyle\sum_{k=1}^{n} \frac{1}{k+1} \leqslant \ln(n+1) \leqslant \sum_{k=1}^{n} \frac{1}{k}$ pour la constante γ d'Euler-Mascheroni

Sources



- Alain Busser, *Python pour les (futurs) professeurs*, Ellipses, 2025
- Philippe Caldero et Marie Peronnier, *Carnet de voyage en Algébrie*, Calvage et Mounet, 2022
- Grégoire Dupont, *Probabilités et statistiques pour l'enseignement (2e édition)*, Dunod, 2020
- Francis Filbet, *Analyse numérique. Algorithme et étude mathématique (2e édition)*, Dunod, 2024
- Bastien Lartigue, *30 développements pour l'agrégation interne de mathématiques – 2e édition*, Cépaduès, 2024
- Sébastien Peronno, *40 thèmes illustrés par le numérique pour l'agrégation interne de mathématiques*, Ellipses, 2026