Cours de mathématiques
Première année

Compléments

Exo7

Sommaire

Cours et exercices de maths
exo7.emath.fr

Exo7

1 Zéros des fonctions

Vidéo ■ partie 1. La dichotomie
Vidéo ■ partie 2. La méthode de la sécante
Vidéo ■ partie 3. La méthode de Newton

Dans ce chapitre nous allons appliquer toutes les notions précédentes sur les suites et les fonctions, à la recherche des zéros des fonctions. Plus précisément, nous allons voir trois méthodes afin de trouver des approximations des solutions d'une équation du type $(f(x)=0)$.

1. La dichotomie

1.1. Principe de la dichotomie

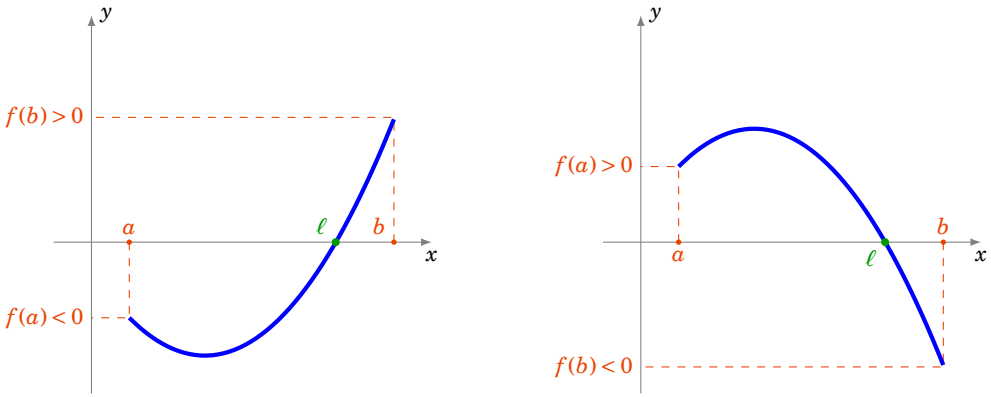
Le principe de dichotomie repose sur la version suivante du ***théorème des valeurs intermédiaires*** :

Théorème 1

Soit $f:[a,b]\to\mathbb{R}$ une fonction continue sur un segment.

Si $f(a)\cdot f(b)\leqslant 0$, alors il existe $\ell\in[a,b]$ tel que $f(\ell)=0$.

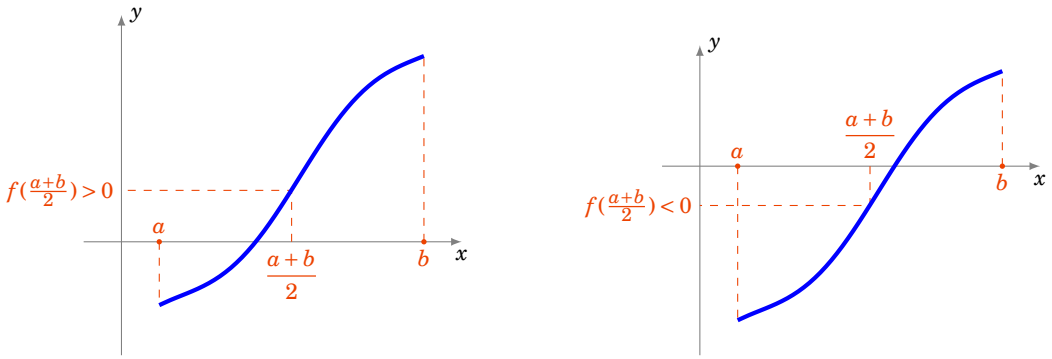
La condition $f(a)\cdot f(b)\leqslant 0$ signifie que $f(a)$ et $f(b)$ sont de signes opposés (ou que l'un des deux est nul). L'hypothèse de continuité est essentielle !



Ce théorème affirme qu'il existe au moins une solution de l'équation $(f(x)=0)$ dans l'intervalle $[a,b]$. Pour le rendre effectif, et trouver une solution (approchée) de l'équation $(f(x)=0)$, il s'agit maintenant de l'appliquer sur un intervalle suffisamment petit. On va voir que cela permet d'obtenir un ℓ solution de l'équation $(f(x)=0)$ comme la limite d'une suite.

Voici comment construire une suite d'intervalles emboîtés, dont la longueur tend vers 0, et contenant chacun une solution de l'équation $(f(x)=0)$.
On part d'une fonction $f:[a,b]\to\mathbb{R}$ continue, avec $a<b$, et $f(a)\cdot f(b)\leqslant 0$.
Voici la première étape de la construction : on regarde le signe de la valeur de la fonction f appliquée au point milieu $\frac{a+b}{2}$.

- Si $f(a)\cdot f(\frac{a+b}{2})\leqslant 0$, alors il existe $c\in[a,\frac{a+b}{2}]$ tel que $f(c)=0$.
- Si $f(a)\cdot f(\frac{a+b}{2})>0$, cela implique que $f(\frac{a+b}{2})\cdot f(b)\leqslant 0$, et alors il existe $c\in[\frac{a+b}{2},b]$ tel que $f(c)=0$.



Nous avons obtenu un intervalle de longueur moitié dans lequel l'équation $(f(x)=0)$ admet une solution. On itère alors le procédé pour diviser de nouveau l'intervalle en deux.
Voici le processus complet :

- **Au rang 0 :**
  On pose $a_0=a$, $b_0=b$. Il existe une solution x_0 de l'équation $(f(x)=0)$ dans l'intervalle $[a_0,b_0]$.
- **Au rang 1 :**
  - Si $f(a_0)\cdot f(\frac{a_0+b_0}{2})\leqslant 0$, alors on pose $a_1=a_0$ et $b_1=\frac{a_0+b_0}{2}$,
  - sinon on pose $a_1=\frac{a_0+b_0}{2}$ et $b_1=b$.
  - Dans les deux cas, il existe une solution x_1 de l'équation $(f(x)=0)$ dans l'intervalle $[a_1,b_1]$.
- ...
- **Au rang n :** supposons construit un intervalle $[a_n,b_n]$, de longueur $\frac{b-a}{2^n}$, et contenant une solution x_n de l'équation $(f(x)=0)$. Alors :
  - Si $f(a_n)\cdot f(\frac{a_n+b_n}{2})\leqslant 0$, alors on pose $a_{n+1}=a_n$ et $b_{n+1}=\frac{a_n+b_n}{2}$,
  - sinon on pose $a_{n+1}=\frac{a_n+b_n}{2}$ et $b_{n+1}=b_n$.
  - Dans les deux cas, il existe une solution x_{n+1} de l'équation $(f(x)=0)$ dans l'intervalle $[a_{n+1},b_{n+1}]$.

À chaque étape on a

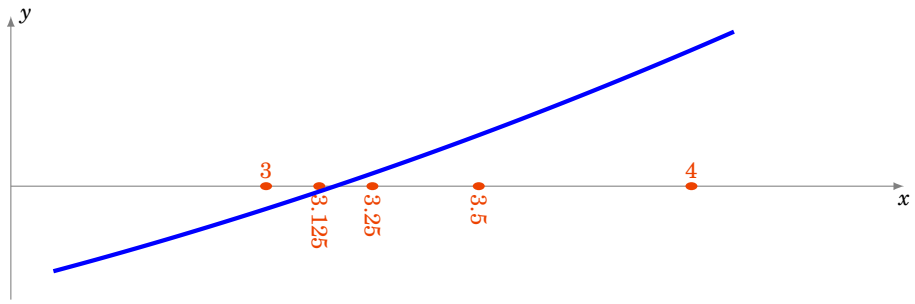
$$a_n\leqslant x_n\leqslant b_n.$$

On arrête le processus dès que $b_n-a_n=\frac{b-a}{2^n}$ est inférieur à la précision souhaitée.
Comme (a_n) est par construction une suite croissante, (b_n) une suite décroissante, et $(b_n-a_n)\to 0$ lorsque $n\to+\infty$, les suites (a_n) et (b_n) sont adjacentes et donc elles admettent une même limite. D'après le théorème des gendarmes, c'est aussi la limite disons ℓ de la suite (x_n). La continuité de f montre que $f(\ell)=\lim_{n\to+\infty}f(x_n)=\lim_{n\to+\infty}0=0$. Donc les suites (a_n) et (b_n) tendent toutes les deux vers ℓ, qui est une solution de l'équation $(f(x)=0)$.

1.2. Résultats numériques pour $\sqrt{10}$

Nous allons calculer une approximation de $\sqrt{10}$. Soit la fonction f définie par $f(x)=x^2-10$, c'est une fonction continue sur $\mathbb{R}$ qui s'annule en $\pm\sqrt{10}$. De plus $\sqrt{10}$ est l'unique solution positive de

l'équation $(f(x)=0)$. Nous pouvons restreindre la fonction f à l'intervalle $[3,4]$: en effet $3^2=9\leqslant 10$ donc $3\leqslant\sqrt{10}$ et $4^2=16\geqslant 10$ donc $4\geqslant\sqrt{10}$. En d'autre termes $f(3)\leqslant 0$ et $f(4)\geqslant 0$, donc l'équation $(f(x)=0)$ admet une solution dans l'intervalle $[3,4]$ d'après le théorème des valeurs intermédiaires, et par unicité c'est $\sqrt{10}$, donc $\sqrt{10}\in[3,4]$.
Notez que l'on ne choisit pas pour f la fonction $x\mapsto x-\sqrt{10}$ car on ne connaît pas la valeur de $\sqrt{10}$. C'est ce que l'on cherche à calculer !



Voici les toutes premières étapes :

1. On pose $a_0=3$ et $b_0=4$, on a bien $f(a_0)\leqslant 0$ et $f(b_0)\geqslant 0$. On calcule $\frac{a_0+b_0}{2}=3,5$ puis $f(\frac{a_0+b_0}{2})$: $f(3,5)=3,5^2-10=2,25\geqslant 0$. Donc $\sqrt{10}$ est dans l'intervalle $[3;3,5]$ et on pose $a_1=a_0=3$ et $b_1=\frac{a_0+b_0}{2}=3,5$.
2. On sait donc que $f(a_1)\leqslant 0$ et $f(b_1)\geqslant 0$. On calcule $f(\frac{a_1+b_1}{2})=f(3,25)=0,5625\geqslant 0$, on pose $a_2=3$ et $b_2=3,25$.
3. On calcule $f(\frac{a_2+b_2}{2})=f(3,125)=-0,23\ldots\leqslant 0$. Comme $f(b_2)\geqslant 0$ alors cette fois f s'annule sur le second intervalle $[\frac{a_2+b_2}{2},b_2]$ et on pose $a_3=\frac{a_2+b_2}{2}=3,125$ et $b_3=b_2=3,25$.

À ce stade, on a prouvé : $3,125\leqslant\sqrt{10}\leqslant 3,25$.
Voici la suite des étapes :

$a_0=3$	$b_0=4$
$a_1=3$	$b_1=3,5$
$a_2=3$	$b_2=3,25$
$a_3=3,125$	$b_3=3,25$
$a_4=3,125$	$b_4=3,1875$
$a_5=3,15625$	$b_5=3,1875$
$a_6=3,15625$	$b_6=3,171875$
$a_7=3,15625$	$b_7=3,164062\ldots$
$a_8=3,16015\ldots$	$b_8=3,164062\ldots$

Donc en 8 étapes on obtient l'encadrement :

$$3,160\leqslant\sqrt{10}\leqslant 3,165$$

En particulier, on vient d'obtenir les deux premières décimales : $\sqrt{10}=3,16\ldots$

1.3. Résultats numériques pour $(1,10)^{1/12}$

Nous cherchons maintenant une approximation de $(1,10)^{1/12}$. Soit $f(x)=x^{12}-1,10$. On pose $a_0=1$ et $b_0=1,1$. Alors $f(a_0)=-0,10\leqslant 0$ et $f(b_0)=2,038\ldots\geqslant 0$.

$$\begin{array}{ll} a_0 = 1 & b_0 = 1,10 \\ a_1 = 1 & b_1 = 1,05 \\ a_2 = 1 & b_2 = 1,025 \\ a_3 = 1 & b_3 = 1,0125 \\ a_4 = 1,00625 & b_4 = 1,0125 \\ a_5 = 1,00625 & b_5 = 1,00937\ldots \\ a_6 = 1,00781\ldots & b_6 = 1,00937\ldots \\ a_7 = 1,00781\ldots & b_7 = 1,00859\ldots \\ a_8 = 1,00781\ldots & b_8 = 1,00820\ldots \end{array}$$

Donc en 8 étapes on obtient l'encadrement :

$$1,00781 \leqslant (1,10)^{1/12} \leqslant 1,00821$$

1.4. Calcul de l'erreur

La méthode de dichotomie a l'énorme avantage de fournir un encadrement d'une solution ℓ de l'équation $(f(x) = 0)$. Il est donc facile d'avoir une majoration de l'erreur. En effet, à chaque étape, la taille l'intervalle contenant ℓ est divisée par 2. Au départ, on sait que $\ell \in [a,b]$ (de longueur $b-a$) ; puis $\ell \in [a_1,b_1]$ (de longueur $\frac{b-a}{2}$) ; puis $\ell \in [a_2,b_2]$ (de longueur $\frac{b-a}{4}$) ; ... ; $[a_n,b_n]$ étant de longueur $\frac{b-a}{2^n}$.
Si, par exemple, on souhaite obtenir une approximation de ℓ à 10^{-N} près, comme on sait que $a_n \leqslant \ell \leqslant b_n$, on obtient $|\ell - a_n| \leqslant |b_n - a_n| = \frac{b-a}{2^n}$. Donc pour avoir $|\ell - a_n| \leqslant 10^{-N}$, il suffit de choisir n tel que $\frac{b-a}{2^n} \leqslant 10^{-N}$.
Nous allons utiliser le logarithme décimal :

$$\begin{aligned} \frac{b-a}{2^n} \leqslant 10^{-N} &\iff (b-a)10^N \leqslant 2^n \\ &\iff \log(b-a) + \log(10^N) \leqslant \log(2^n) \\ &\iff \log(b-a) + N \leqslant n\log 2 \\ &\iff n \geqslant \frac{N + \log(b-a)}{\log 2} \end{aligned}$$

Sachant $\log 2 = 0,301\ldots$, si par exemple $b-a \leqslant 1$, voici le nombre d'itérations suffisantes pour avoir une précision de 10^{-N} (ce qui correspond, à peu près, à N chiffres exacts après la virgule).

10^{-10} ($\sim$ 10 décimales)	34 itérations
10^{-100} ($\sim$ 100 décimales)	333 itérations
10^{-1000} ($\sim$ 1000 décimales)	3322 itérations

Il faut entre 3 et 4 itérations supplémentaires pour obtenir une nouvelle décimale.

Remarque

En toute rigueur il ne faut pas confondre précision et nombre de décimales exactes, par exemple 0,999 est une approximation de 1,000 à 10^{-3} près, mais aucune décimale après la virgule n'est exacte. En pratique, c'est la précision qui est la plus importante, mais il est plus frappant de parler du nombre de décimales exactes.

1.5. Algorithmes

Voici comment implémenter la dichotomie dans le langage `Python`. Tout d'abord on définit une fonction f (ici par exemple $f(x) = x^2 - 10$) :

Algorithme . dichotomie.py (1)

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

Puis la dichotomie proprement dite : en entrée de la fonction, on a pour variables a, b et n le nombre d'étapes voulues.

Algorithme . dichotomie.py (2)

```
def dicho(a,b,n):
    for i in range(n):
        c = (a+b)/2
        if f(a)*f(c) <= 0:
            b = c
        else:
            a = c
    return a,b
```

Même algorithme, mais avec cette fois en entrée la précision souhaitée :

Algorithme . dichotomie.py (3)

```
def dichobis(a,b,prec):
    while b-a>prec:
        c = (a+b)/2
        if f(a)*f(c) <= 0:
            b = c
        else:
            a = c
    return a,b
```

Enfin, voici la version récursive de l'algorithme de dichotomie.

Algorithme . dichotomie.py (4)

```
def dichotomie(a,b,prec):
    if b-a<=prec:
        return a,b
    else:
        c = (a+b)/2
        if f(a)*f(c) <= 0:
            return dichotomie(a,c,prec)
        else:
```

```
        return dichotomie(c,b,prec)
```

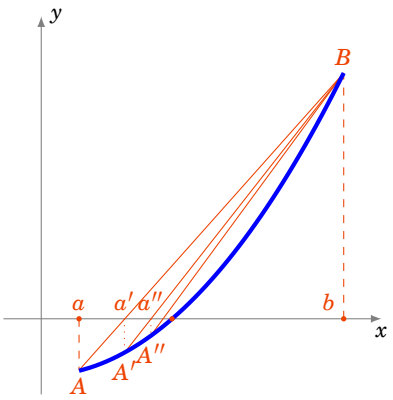
Mini-exercices

1. À la main, calculer un encadrement à 0,1 près de $\sqrt{3}$. Idem avec $\sqrt[3]{2}$.
2. Calculer une approximation des solutions de l'équation $x^3+1=3x$.
3. Est-il plus efficace de diviser l'intervalle en 4 au lieu d'en 2 ? (À chaque itération, la dichotomie classique nécessite l'évaluation de f en une nouvelle valeur $\frac{a+b}{2}$ pour une précision améliorée d'un facteur 2.)
4. Écrire un algorithme pour calculer plusieurs solutions de $(f(x)=0)$.
5. On se donne un tableau trié de taille N, rempli de nombres appartenant à $\{1,\dots,n\}$. Écrire un algorithme qui teste si une valeur k apparaît dans le tableau et en quelle position.

2. La méthode de la sécante

2.1. Principe de la sécante

L'idée de la méthode de la sécante est très simple : pour une fonction f continue sur un intervalle $[a,b]$, et vérifiant $f(a)\leqslant 0$, $f(b)>0$, on trace le segment $[AB]$ où $A=(a,f(a))$ et $B=(b,f(b))$. Si le segment reste au-dessus du graphe de f alors la fonction s'annule sur l'intervalle $[a',b]$ où $(a',0)$ est le point d'intersection de la droite (AB) avec l'axe des abscisses. La droite (AB) s'appelle la ***sécante***. On recommence en partant maintenant de l'intervalle $[a',b]$ pour obtenir une valeur a''.



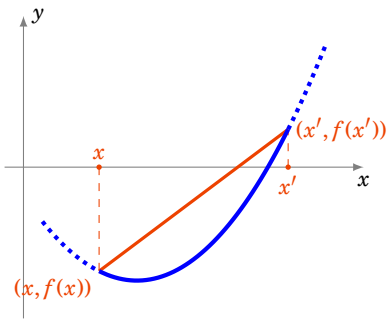
Proposition 1

Soit $f : [a,b] \to \mathbb{R}$ une fonction continue, strictement croissante et convexe telle que $f(a) \leqslant 0$, $f(b) > 0$. Alors la suite définie par

$$a_0 = a \qquad \text{et} \qquad a_{n+1} = a_n - \frac{b-a_n}{f(b)-f(a_n)} f(a_n)$$

est croissante et converge vers la solution ℓ de $(f(x) = 0)$.

L'hypothèse f ***convexe*** signifie exactement que pour tout x, x' dans $[a,b]$ la sécante (ou corde) entre $(x, f(x))$ et $(x', f(x'))$ est au-dessus du graphe de f.



Démonstration

1. Justifions d'abord la construction de la suite récurrente.
   L'équation de la droite passant par les deux points $(a, f(a))$ et $(b, f(b))$ est
   $$y = (x-a)\frac{f(b)-f(a)}{b-a} + f(a)$$
   Cette droite intersecte l'axe des abscisses en $(a', 0)$ qui vérifie donc $0 = (a'-a)\frac{f(b)-f(a)}{b-a} + f(a)$, donc $a' = a - \frac{b-a}{f(b)-f(a)} f(a)$.
2. Croissance de (a_n).
   Montrons par récurrence que $f(a_n) \leqslant 0$. C'est vrai au rang 0 car $f(a_0) = f(a) \leqslant 0$ par hypothèse. Supposons vraie l'hypothèse au rang n. Si $a_{n+1} < a_n$ (un cas qui s'avérera *a posteriori* jamais réalisé), alors comme f est strictement croissante, on a $f(a_{n+1}) < f(a_n)$, et en particulier $f(a_{n+1}) \leqslant 0$. Sinon $a_{n+1} \geqslant a_n$. Comme f est convexe : la sécante entre $(a_n, f(a_n))$ et $(b, f(b))$ est au-dessus du graphe de f. En particulier le point $(a_{n+1}, 0)$ (qui est sur cette sécante par définition a_{n+1}) est au-dessus du point $(a_{n+1}, f(a_{n+1}))$, et donc $f(a_{n+1}) \leqslant 0$ aussi dans ce cas, ce qui conclut la récurrence.
   Comme $f(a_n) \leqslant 0$ et f est croissante, alors par la formule $a_{n+1} = a_n - \frac{b-a_n}{f(b)-f(a_n)} f(a_n)$, on obtient que $a_{n+1} \geqslant a_n$.
3. Convergence de (a_n).
   La suite (a_n) est croissante et majorée par b, donc elle converge. Notons ℓ sa limite. Par continuité $f(a_n) \to f(\ell)$. Comme pour tout n, $f(a_n) \leqslant 0$, on en déduit que $f(\ell) \leqslant 0$. En particulier, comme on suppose $f(b) > 0$, on a $\ell < b$. Comme $a_n \to \ell$, $a_{n+1} \to \ell$, $f(a_n) \to f(\ell)$, l'égalité $a_{n+1} = a_n - \frac{b-a_n}{f(b)-f(a_n)} f(a_n)$ devient à la limite (lorsque $n \to +\infty$) : $\ell = \ell - \frac{b-\ell}{f(b)-f(\ell)} f(\ell)$, ce qui implique $f(\ell) = 0$.
   Conclusion : (a_n) converge vers la solution de $(f(x) = 0)$.

2.2. Résultats numériques pour $\sqrt{10}$

Pour $a=3$, $b=4$, $f(x)=x^2-10$ voici les résultats numériques, est aussi indiquée une majoration de l'erreur $\varepsilon_n=\sqrt{10}-a_n$ (voir ci-après).

$a_0=3$	$\varepsilon_0 \leqslant 0,1666\ldots$
$a_1=3,14285714285\ldots$	$\varepsilon_1 \leqslant 0,02040\ldots$
$a_2=3,16000000000\ldots$	$\varepsilon_2 \leqslant 0,00239\ldots$
$a_3=3,16201117318\ldots$	$\varepsilon_3 \leqslant 0,00028\ldots$
$a_4=3,16224648985\ldots$	$\varepsilon_4 \leqslant 3,28\ldots\cdot 10^{-5}$
$a_5=3,16227401437\ldots$	$\varepsilon_5 \leqslant 3,84\ldots\cdot 10^{-6}$
$a_6=3,16227723374\ldots$	$\varepsilon_6 \leqslant 4,49\ldots\cdot 10^{-7}$
$a_7=3,16227761029\ldots$	$\varepsilon_7 \leqslant 5,25\ldots\cdot 10^{-8}$
$a_8=3,16227765433\ldots$	$\varepsilon_8 \leqslant 6,14\ldots\cdot 10^{-9}$

2.3. Résultats numériques pour $(1,10)^{1/12}$

Voici les résultats numériques avec une majoration de l'erreur $\varepsilon_n=(1,10)^{1/12}-a_n$, avec $f(x)=x^{12}-1,10$, $a=1$ et $b=1,1$

$a_0=1$	$\varepsilon_0 \leqslant 0,0083\ldots$
$a_1=1,00467633\ldots$	$\varepsilon_1 \leqslant 0,0035\ldots$
$a_2=1,00661950\ldots$	$\varepsilon_2 \leqslant 0,0014\ldots$
$a_3=1,00741927\ldots$	$\varepsilon_3 \leqslant 0,00060\ldots$
$a_4=1,00774712\ldots$	$\varepsilon_4 \leqslant 0,00024\ldots$
$a_5=1,00788130\ldots$	$\varepsilon_5 \leqslant 0,00010\ldots$
$a_6=1,00793618\ldots$	$\varepsilon_6 \leqslant 4,14\ldots\cdot 10^{-5}$
$a_7=1,00795862\ldots$	$\varepsilon_7 \leqslant 1,69\ldots\cdot 10^{-5}$
$a_8=1,00796779\ldots$	$\varepsilon_8 \leqslant 6,92\ldots\cdot 10^{-6}$

2.4. Calcul de l'erreur

La méthode de la sécante fournit l'encadrement $a_n \leqslant l \leqslant b$. Mais comme b est fixe cela ne donne pas d'information exploitable pour $|l-a_n|$. Voici une façon générale d'estimer l'erreur, à l'aide du théorème des accroissements finis.

Proposition 2

Soit $f : I \to \mathbb{R}$ une fonction dérivable et ℓ tel que $f(\ell)=0$. S'il existe une constante $m>0$ telle que pour tout $x \in I$, $|f'(x)| \geqslant m$ alors

$$|x-\ell| \leqslant \frac{|f(x)|}{m} \qquad \text{pour tout } x \in I.$$

Démonstration

Par l'inégalité des accroissement finis entre x et ℓ : $|f(x)-f(\ell)| \geqslant m|x-\ell|$ mais $f(\ell)=0$, d'où la majoration.

Exemple 1. Erreur pour $\sqrt{10}$

Soit $f(x)=x^2-10$ et l'intervalle $I=[3,4]$. Alors $f'(x)=2x$ donc $|f'(x)|\geqslant 6$ sur I. On pose donc $m=6$, $\ell=\sqrt{10}$, $x=a_n$. On obtient l'estimation de l'erreur :

$$\varepsilon_n=|\ell-a_n|\leqslant\frac{|f(a_n)|}{m}=\frac{|a_n^2-10|}{6}$$

Par exemple on a trouvé $a_2=3,16...\leqslant 3,17$ donc $\sqrt{10}-a_2\leqslant\frac{|3,17^2-10|}{6}=0,489$.
Pour a_8 on a trouvé $a_8=3,1622776543347473\ldots$ donc $\sqrt{10}-a_8\leqslant\frac{|a_8^2-10|}{6}=6,14\ldots\cdot 10^{-9}$. On a en fait 7 décimales exactes après la virgule.

Dans la pratique, voici le nombre d'itérations suffisantes pour avoir une précision de 10^{-n} pour cet exemple. Grosso-modo, une itération de plus donne une décimale supplémentaire.

10^{-10} ($\sim$ 10 décimales)	10 itérations
10^{-100} ($\sim$ 100 décimales)	107 itérations
10^{-1000} ($\sim$ 1000 décimales)	1073 itérations

Exemple 2. Erreur pour $(1,10)^{1/12}$

On pose $f(x)=x^{12}-1,10$, $I=[1;1,10]$ et $\ell=(1,10)^{1/12}$. Comme $f'(x)=12x^{11}$, si on pose de plus $m=12$, on a $|f'(x)|\geqslant m$ pour $x\in I$. On obtient

$$\varepsilon_n=|\ell-a_n|\leqslant\frac{|a_n^{12}-1,10|}{12}.$$

Par exemple $a_8=1.0079677973185432\ldots$ donc

$$|(1,10)^{1/12}-a_8|\leqslant\frac{|a_8^{12}-1,10|}{12}=6,92\ldots\cdot 10^{-6}.$$

2.5. Algorithme

Voici l'algorithme : c'est tout simplement la mise en œuvre de la suite récurrente (a_n).

Algorithme . secante.py

```
def secante(a,b,n):
    for i in range(n):
        a = a-f(a)*(b-a)/(f(b)-f(a))
    return a
```

Mini-exercices

1. À la main, calculer un encadrement à 0,1 près de $\sqrt{3}$. Idem avec $\sqrt[3]{2}$.
2. Calculer une approximation des solutions de l'équation $x^3+1=3x$.
3. Calculer une approximation de la solution de l'équation $(\cos x=0)$ sur $[0,\pi]$. Idem avec $(\cos x=2\sin x)$.

4. Étudier l'équation $(\exp(-x) = -\ln(x))$. Donner une approximation de la (ou des) solution(s) et une majoration de l'erreur correspondante.

3. La méthode de Newton

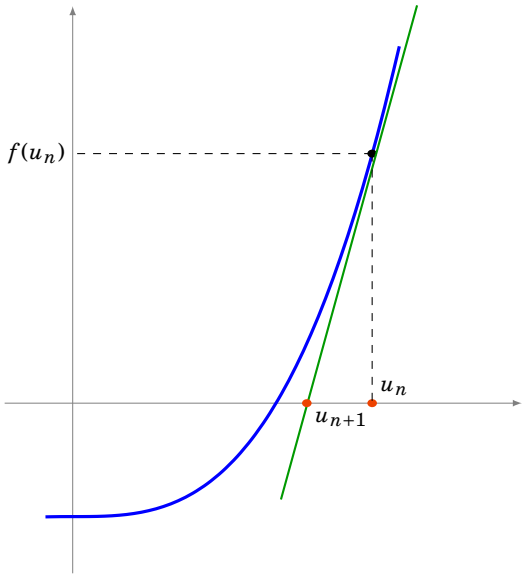
3.1. Méthode de Newton

La méthode de Newton consiste à remplacer la sécante de la méthode précédente par la tangente. Elle est d'une redoutable efficacité.
Partons d'une fonction dérivable $f : [a,b] \to \mathbb{R}$ et d'un point $u_0 \in [a,b]$. On appelle $(u_1, 0)$ l'intersection de la tangente au graphe de f en $(u_0, f(u_0))$ avec l'axe des abscisses. Si $u_1 \in [a,b]$ alors on recommence l'opération avec la tangente au point d'abscisse u_1. Ce processus conduit à la définition d'une suite récurrente :

$$u_0 \in [a,b] \qquad \text{et} \qquad u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)}.$$

Démonstration

En effet la tangente au point d'abscisse u_n a pour équation : $y = f'(u_n)(x-u_n)+f(u_n)$. Donc le point $(x,0)$ appartenant à la tangente (et à l'axe des abscisses) vérifie $0 = f'(u_n)(x-u_n)+f(u_n)$. D'où $x = u_n - \frac{f(u_n)}{f'(u_n)}$.



3.2. Résultats pour $\sqrt{10}$

Pour calculer $\sqrt{a}$, on pose $f(x) = x^2 - a$, avec $f'(x) = 2x$. La suite issue de la méthode de Newton est déterminée par $u_0 > 0$ et la relation de récurrence $u_{n+1} = u_n - \frac{u_n^2 - a}{2u_n}$. Suite qui pour cet exemple s'appelle ***suite de Héron*** et que l'on récrit souvent

$$u_0 > 0 \qquad \text{et} \qquad u_{n+1} = \frac{1}{2}\left(u_n + \frac{a}{u_n}\right).$$

Proposition 3

Cette suite (u_n) converge vers $\sqrt{a}$.

Pour le calcul de $\sqrt{10}$, on pose par exemple $u_0 = 4$, et on peut même commencer les calculs à la main :

$$\begin{aligned}
u_0 &= 4\\
u_1 &= \tfrac{1}{2}\left(u_0 + \tfrac{10}{u_0}\right) = \tfrac{1}{2}\left(4 + \tfrac{10}{4}\right) = \tfrac{13}{4} = 3{,}25\\
u_2 &= \tfrac{1}{2}\left(u_1 + \tfrac{10}{u_1}\right) = \tfrac{1}{2}\left(\tfrac{13}{4} + \tfrac{10}{\frac{13}{4}}\right) = \tfrac{329}{104} = 3{,}1634\ldots\\
u_3 &= \tfrac{1}{2}\left(u_2 + \tfrac{10}{u_2}\right) = \tfrac{216401}{68432} = 3{,}16227788\ldots\\
u_4 &= 3{,}162277660168387\ldots
\end{aligned}$$

Pour u_4 on obtient $\sqrt{10} = 3{,}1622776601683\ldots$ avec déjà 13 décimales exactes !
Voici la preuve de la convergence de la suite (u_n) vers $\sqrt{a}$.

Démonstration

$$u_0 > 0 \quad \text{et} \quad u_{n+1} = \frac{1}{2}\left(u_n + \frac{a}{u_n}\right).$$

1. Montrons que $u_n \geqslant \sqrt{a}$ pour $n \geqslant 1$.
   Tout d'abord
   $$u_{n+1}^2 - a = \frac{1}{4}\left(\frac{u_n^2 + a}{u_n}\right)^2 - a = \frac{1}{4u_n^2}(u_n^4 - 2au_n^2 + a^2) = \frac{1}{4}\frac{(u_n^2 - a)^2}{u_n^2}$$
   Donc $u_{n+1}^2 - a \geqslant 0$. Comme il est clair que pour tout $n \geqslant 0$, $u_n \geqslant 0$, on en déduit que pour tout $n \geqslant 0$, $u_{n+1} \geqslant \sqrt{a}$. (Notez que u_0 lui est quelconque.)
2. Montrons que $(u_n)_{n\geqslant 1}$ est une suite décroissante qui converge.
   Comme $\frac{u_{n+1}}{u_n} = \frac{1}{2}\left(1 + \frac{a}{u_n^2}\right)$, et que pour $n \geqslant 1$ on vient de voir que $u_n^2 \geqslant a$ (donc $\frac{a}{u_n^2} \leqslant 1$), alors $\frac{u_{n+1}}{u_n} \leqslant 1$, pour tout $n \leqslant 1$.
   Conséquence : la suite $(u_n)_{n\geqslant 1}$ est décroissante et minorée par 0 donc elle converge.
3. (u_n) converge vers $\sqrt{a}$.
   Notons ℓ la limite de (u_n). Alors $u_n \to \ell$ et $u_{n+1} \to \ell$. Lorsque $n \to +\infty$ dans la relation $u_{n+1} = \frac{1}{2}\left(u_n + \frac{a}{u_n}\right)$, on obtient $\ell = \frac{1}{2}\left(\ell + \frac{a}{\ell}\right)$. Ce qui conduit à la relation $\ell^2 = a$ et par positivité de la suite, $\ell = \sqrt{a}$.

3.3. Résultats numériques pour $(1{,}10)^{1/12}$

Pour calculer $(1{,}10)^{1/12}$, on pose $f(x) = x^{12} - a$ avec $a = 1{,}10$. On a $f'(x) = 12x^{11}$. On obtient $u_{n+1} = u_n - \frac{u_n^{12} - a}{12u_n^{11}}$. Ce que l'on reformule ainsi :

$$u_0 > 0 \quad \text{et} \quad u_{n+1} = \frac{1}{12}\left(11u_n + \frac{a}{u_n^{11}}\right).$$

Voici les résultats numériques pour $(1{,}10)^{1/12}$ en partant de $u_0 = 1$.

$$\begin{aligned}
u_0 &= 1\\
u_1 &= 1{,}0083333333333333\ldots\\
u_2 &= 1{,}0079748433368980\ldots\\
u_3 &= 1{,}0079741404315996\ldots\\
u_4 &= 1{,}0079741404289038\ldots
\end{aligned}$$

Toutes les décimales affichées pour u_4 sont exactes : $(1,10)^{1/12} = 1,0079741404289038\ldots$

3.4. Calcul de l'erreur pour $\sqrt{10}$

Proposition 4

1. Soit k tel que $u_1 - \sqrt{a} \leqslant k$. Alors pour tout $n \geqslant 1$:
$$u_n - \sqrt{a} \leqslant 2\sqrt{a}\left(\frac{k}{2\sqrt{a}}\right)^{2^{n-1}}$$
2. Pour $a = 10$, $u_0 = 4$, on a :
$$u_n - \sqrt{10} \leqslant 8\left(\frac{1}{24}\right)^{2^{n-1}}$$

Admirez la puissance de la méthode de Newton : 11 itérations donnent déjà 1000 décimales exactes après la virgule. Cette rapidité de convergence se justifie grâce au calcul de l'erreur : la précision est multipliée par 2 à chaque étape, donc à chaque itération le nombre de décimales exactes double !

10^{-10} ($\sim$ 10 décimales)	4 itérations
10^{-100} ($\sim$ 100 décimales)	8 itérations
10^{-1000} ($\sim$ 1000 décimales)	11 itérations

Démonstration

1. Dans la preuve de la proposition 3, nous avons vu l'égalité :
$$u_{n+1}^2 - a = \frac{(u_n^2 - a)^2}{4u_n^2} \text{ donc } (u_{n+1} - \sqrt{a})(u_{n+1} + \sqrt{a}) = \frac{(u_n - \sqrt{a})^2 (u_n + \sqrt{a})^2}{4u_n^2}$$
Ainsi comme $u_n \geqslant \sqrt{a}$ pour $n \geqslant 1$:
$$u_{n+1} - \sqrt{a} = (u_n - \sqrt{a})^2 \times \frac{1}{u_{n+1} + \sqrt{a}} \times \frac{1}{4}\left(1 + \frac{\sqrt{a}}{u_n}\right)^2 \leqslant (u_n - \sqrt{a})^2 \times \frac{1}{2\sqrt{a}} \times \frac{1}{4} \cdot (1+1)^2 = \frac{1}{2\sqrt{a}}(u_n - \sqrt{a})^2$$
Si k vérifie $u_1 - \sqrt{a} \leqslant k$, nous allons en déduire par récurrence, pour tout $n \geqslant 1$, la formule
$$u_n - \sqrt{a} \leqslant 2\sqrt{a}\left(\frac{k}{2\sqrt{a}}\right)^{2^{n-1}}$$
C'est vrai pour $n = 1$. Supposons la formule vraie au rang n, alors :
$$u_{n+1} - \sqrt{a} \leqslant \frac{1}{2\sqrt{a}}(u_n - \sqrt{a})^2 = \frac{1}{2\sqrt{a}}(2\sqrt{a})^2 \left(\left(\frac{k}{2\sqrt{a}}\right)^{2^{n-1}}\right)^2 = 2\sqrt{a}\left(\frac{k}{2\sqrt{a}}\right)^{2^n}$$
La formule est donc vrai au rang suivant.
2. Pour $a = 10$ avec $u_0 = 4$ on a $u_1 = 3,25$. Comme $3 \leqslant \sqrt{10} \leqslant 4$ alors $u_1 - \sqrt{10} \leqslant u_1 - 3 \leqslant \frac{1}{4}$. On fixe donc $k = \frac{1}{4}$. Toujours par l'encadrement $3 \leqslant \sqrt{10} \leqslant 4$, la formule obtenue précédemment devient
$$u_n - \sqrt{a} \leqslant 2 \cdot 4\left(\frac{\frac{1}{4}}{2 \cdot 3}\right)^{2^{n-1}} = 8\left(\frac{1}{24}\right)^{2^{n-1}}.$$

3.5. Algorithme

Voici l'algorithme pour le calcul de $\sqrt{a}$. On précise en entrée le réel $a \geqslant 0$ dont on veut calculer la racine et le nombre n d'itérations.

Algorithme . newton.py

```
def racine_carree(a,n):
    u=4                 # N'importe qu'elle valeur > 0
    for i in range(n):
        u = 0.5*(u+a/u)
    return u
```

En utilisant le module `decimal` le calcul de u_n pour $n = 11$ donne 1000 décimales de $\sqrt{10}$:

3,
16227766016837933199889354443271853371955513932521
68268575048527925944386392382213442481083793002951
87347284152840055148548856030453880014690519596700
15390334492165717925994065915015347411333948412408
53169295770904715764610443692578790620378086099418
28371711548406328552999118596824564203326961604691
31433612894979189026652954361267617878135006138818
62785804636831349524780311437693346719738195131856
78403231241795402218308045872844614600253577579702
82864402902440797789603454398916334922265261206779
26516760310484366977937569261557205003698949094694
21850007358348844643882731109289109042348054235653
40390727401978654372593964172600130699000095578446
31096267906944183361301813028945417033158077316263
86395193793704654765220632063686587197822049312426
05345411160935697982813245229700079888352375958532
85792513629646865114976752171234595592380393756251
25369855194955325099947038843990336466165470647234
99979613234340302185705218783667634578951073298287
51579452157716521396263244383990184845609357626020

Mini-exercices

1. À la calculette, calculer les trois premières étapes pour une approximation de $\sqrt{3}$, sous forme de nombres rationnels. Idem avec $\sqrt[3]{2}$.
2. Implémenter la méthode de Newton, étant données une fonction f et sa dérivée f'.
3. Calculer une approximation des solutions de l'équation $x^3 + 1 = 3x$.
4. Soit $a > 0$. Comment calculer $\frac{1}{a}$ par une méthode de Newton ?
5. Calculer n de sorte que $u_n - \sqrt{10} \leqslant 10^{-\ell}$ (avec $u_0 = 4$, $u_{n+1} = \frac{1}{2}\left(u_n + \frac{a}{u_n}\right)$, $a = 10$).

Auteurs

Auteurs : Arnaud Bodin, Niels Borne, Laura Desideri
Dessins : Benjamin Boutin

2 Algorithmes et mathématiques

Vidéo ■ partie 1. Premiers pas avec Python
Vidéo ■ partie 2. Ecriture des entiers
Vidéo ■ partie 3. Calculs de sinus, cosinus, tangente
Vidéo ■ partie 4. Les réels
Vidéo ■ partie 5. Arithmétique - Algorithmes récursifs
Vidéo ■ partie 6. Polynômes - Complexité d'un algorithme

1. Premiers pas avec Python

Dans cette partie on vérifie d'abord que Python fonctionne, puis on introduira les boucles (`for` et `while`), le test `if ... else ...` et les fonctions.

1.1. Hello world !

Pour commencer testons si tout fonctionne !

Travaux pratiques 1

1. Définir deux variables prenant les valeurs 3 et 6.
2. Calculer leur somme et leur produit.

Voici à quoi cela ressemble :

Algorithme . hello-world.py

```
>>> a=3
>>> b=6
>>> somme = a+b
>>> print(somme)
9
>>> # Les résultats
>>> print("La somme est", somme)
La somme est 9
>>> produit = a*b
>>> print("Le produit est", produit)
Le produit est 18
```

On retient les choses suivantes :

- On affecte une valeur à une variable par le signe égal =.
- On affiche un message avec la fonction `print()`.
- Lorsque qu'une ligne contient un dièse #, tout ce qui suit est ignoré. Cela permet d'insérer des commentaires, ce qui est essentiel pour relire le code.

Dans la suite on omettra les symboles >>>. Voir plus de détails sur le fonctionnement en fin de section.

1.2. Somme des cubes

Travaux pratiques 2

1. Pour un entier n fixé, programmer le calcul de la somme $S_n = 1^3 + 2^3 + 3^3 + \cdots + n^3$.
2. Définir une fonction qui pour une valeur n renvoie la somme $\Sigma_n = 1 + 2 + 3 + \cdots + n$.
3. Définir une fonction qui pour une valeur n renvoie S_n.
4. Vérifier, pour les premiers entiers, que $S_n = (\Sigma_n)^2$.

1.

Algorithme . somme-cubes.py (1)

```
n = 10
somme = 0
for i in range(1,n+1):
    somme = somme + i*i*i
print(somme)
```

Voici ce que l'on fait pour calculer S_n avec $n = 10$.

- On affecte d'abord la valeur 0 à la variable `somme`, cela correspond à l'initialisation $S_0 = 0$.
- Nous avons défini une ***boucle*** avec l'instruction *for* qui fait varier i entre 1 et n.
- Nous calculons successivement S_1, S_2,... en utilisant la formule de récurrence $S_i = S_{i-1} + i^3$. Comme nous n'avons pas besoin de conserver toutes les valeurs des S_i alors on garde le même nom pour toutes les sommes, à chaque étape on affecte à `somme` l'ancienne valeur de la somme plus i^3 : `somme = somme + i*i*i`.
- `range(1,n+1)` est l'ensemble des entiers $\{1,2,\ldots,n\}$. C'est bien les entiers **strictement inférieurs à** $n+1$. La raison est que `range(n)` désigne $\{0,1,2,\ldots,n-1\}$ qui contient n éléments.

2. Nous savons que $\Sigma_n = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$ donc nous n'avons pas besoin de faire une boucle :

Algorithme . somme-cubes.py (2)

```
def somme_entiers(n):
    return n*(n+1)/2
```

Une ***fonction*** en informatique est similaire à une fonction mathématique, c'est un objet qui prend en entrée des variables (dites variables formelles ou variables muettes, ici n) et retourne une valeur (un entier, une liste, une chaîne de caractères,... ici $\frac{n(n+1)}{2}$).

3. Voici la fonction qui retourne la somme des cubes :

Algorithme . somme-cubes.py (3)

```
def somme_cubes(n):
    somme = 0
    for i in range(1,n+1):
        somme = somme + i**3
    return somme
```

4. Et enfin on vérifie que pour les premiers entiers $S_n = \left(\frac{n(n+1)}{2}\right)^2$, par exemple pour $n = 12$:

Algorithme . somme-cubes.py (4)

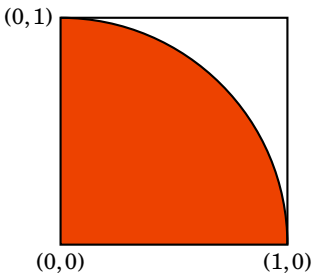
```
n = 12
if somme_cubes(n) == (somme_entiers(n)**2):
    print("Pour␣n=", n, "l'assertion␣est␣vraie.")
else:
    print("L'assertion␣est␣fausse␣!")
```

On retient :

- Les puissances se calculent aussi avec `**` : 5^2 s'écrit `5*5` ou `5**2`, 5^3 s'écrit `5*5*5` ou `5**3`,...
- Une fonction se définit par `def ma_fonction(variable):` et se termine par `return resultat`.
- *`if condition: ... else: ...`* exécute le premier bloc d'instructions si la condition est vraie ; si la condition est fausse cela exécute l'autre bloc.
- Exemple de conditions
  - `a < b` : $a < b$,
  - `a <= b` : $a \leqslant b$,
  - `a == b` : $a = b$,
  - `a != b` : $a \neq b$.
- Attention ! Il est important de comprendre que `a==b` vaut soit vraie ou faux (on compare a et b) alors qu'avec `a=b` on affecte dans a la valeur de b.
- Enfin en `Python` (contrairement aux autres langages) c'est l'indentation (les espaces en début de chaque ligne) qui détermine les blocs d'instructions.

1.3. Calcul de π au hasard

Nous allons voir qu'il est possible de calculer les premières décimales de π par la méthode de Monte-Carlo, c'est à dire avec l'aide du hasard. On considère le carré de coté 1, le cercle de rayon 1 centré à l'origine, d'équation $x^2 + y^2 = 1$, et la portion de disque dans le carré (voir la figure).



Travaux pratiques 3

1. Calculer l'aire du carré et de la portion de disque.
2. Pour un point (x,y) tiré au hasard dans le carré, quelle est la probabilité que le point soit en fait dans la portion de disque ?
3. Tirer un grand nombre de points au hasard, compter ceux qui sont dans la portion de disque.
4. En déduire les premières décimales de π.

Voici le code :

Algorithme . pi-hasard.py

```
import random             # Module qui génère des nombres aléatoires

Tir = 0                   # Numéro du tir
NbTirDansLeDisque = 0     # Nombre de tirs dans le disque

while (Tir < 1000):
    Tir = Tir + 1
    # On tire au hasard un point (x,y) dans [0,1] x [0,1]
    x = random.random()
    y = random.random()
    if (x*x+y*y <= 1):    # On est dans le disque
        NbTirDansLeDisque = NbTirDansLeDisque + 1

MonPi = 4*NbTirDansLeDisque / Tir
print("Valeur␣expérimentale␣de␣Pi␣:␣%0.3f" %MonPi)
```

Commentaires :

- Un petit calcul prouve que l'aire de la portion de disque est $\frac{\pi}{4}$, l'aire du carré est 1. Donc la probabilité de tomber dans le disque est $\frac{\pi}{4}$.
- Pour tirer un nombre au hasard on utilise une fonction `random()` qui renvoie un nombre réel de l'intervalle $[0,1[$. Bien sûr à chaque appel de la fonction `random()` le nombre obtenu est différent !
- Cette fonction n'est pas connue par défaut de `Python`, il faut lui indiquer le nom du ***module*** où elle se trouve. En début de fichier on ajoute `import random` pour le module qui gère les tirages au hasard. Et pour indiquer qu'une fonction vient d'un module il faut l'appeler

par `module.fonction()` donc ici `random.random()` (module et fonction portent ici le même nom !).

- La boucle est *`while condition: ...`* Tant que la condition est vérifiée les instructions de la boucle sont exécutées. Ici `Tir` est le compteur que l'on a initialisé à 0. Ensuite on commence à exécuter la boucle. Bien sûr la première chose que l'on fait dans la boucle est d'incrémenter le compteur `Tir`. On continue jusqu'à ce que l'on atteigne 999. Pour `Tir`= 1000 la condition n'est plus vraie et le bloc d'instructions du `while` n'est pas exécuté. On passe aux instructions suivantes pour afficher le résultat.
- À chaque tir on teste si on est dans la portion de disque ou pas à l'aide de l'inégalité $x^2+y^2 \leqslant 1$.
- Cette méthode n'est pas très efficace, il faut beaucoup de tirs pour obtenir le deux premières décimales de π.

1.4. Un peu plus sur `Python`

- Le plus surprenant avec `Python` c'est que c'est ***l'indentation*** qui détermine le début et la fin d'un bloc d'instructions. Cela oblige à présenter très soigneusement le code.
- Contrairement à d'autres langages on n'a pas besoin de déclarer le type de variable. Par exemple lorsque l'on initialise une variable par `x=0`, on n'a pas besoin de préciser si x est un entier ou un réel.
- Nous travaillerons avec la version 3 (ou plus) de `Python`, que l'on appelle par `python` ou `python3`. Pour savoir si vous avez la bonne version tester la commande `4/3`. Si la réponse est `1.3333...` alors tout est ok. Par contre avec les versions 1 et 2 de `Python` la réponse est `1` (car il considérait que c'est quotient de la division euclidienne de deux entiers).
- La première façon de lancer `Python` est en ligne de commande, on obtient alors l'invite `>>>` et on tape les commandes.
- Mais le plus pratique est de sauvegarder ses commandes dans un fichier et de faire un appel par `python monfichier.py`
- Vous trouverez sans problème de l'aide et des tutoriels sur internet !

Mini-exercices

1. Soit le produit $P_n = (1-\frac{1}{2}) \times (1-\frac{1}{3}) \times (1-\frac{1}{4}) \times \cdots \times (1-\frac{1}{n})$. Calculer une valeur approchée de P_n pour les premiers entiers n.
2. Que vaut la somme des entiers i qui apparaissent dans l'instruction `for i in range(1,10)`. Idem pour `for i in range(11)`. Idem pour `for i in range(1,10,2)`. Idem pour `for i in range(0,10,2)`. Idem pour `for i in range(10,0,-1)`.
3. On considère le cube $[0,1] \times [0,1] \times [0,1]$ et la portion de boule de rayon 1 centrée à l'origine incluse dans ce cube. Faire les calculs de probabilité pour un point tiré au hasard dans le cube d'être en fait dans la portion de boule. Faire une fonction pour le vérifier expérimentalement.
4. On lance deux dés. Expérimenter quelle est la probabilité que la somme soit 7, puis 6, puis 3 ? Quelle est la probabilité que l'un des deux dés soit un 6 ? d'avoir un double ? La fonction `randint(a, b)` du module `random` retourne un entier k au hasard, vérifiant $a \leqslant k \leqslant b$.
5. On lance un dé jusqu'à ce que l'on obtienne un 6. En moyenne au bout de combien de lancer s'arrête-t-on ?

2. Écriture des entiers

Nous allons faire un peu d'arithmétique : le quotient de la division euclidienne `//`, le reste `%` (modulo) et nous verrons l'écriture des entiers en base 10 et en base 2. Nous utiliserons aussi la notion de listes et le module `math`.

2.1. Division euclidienne et reste, calcul avec les modulo

La division euclidienne de a par b, avec $a \in \mathbb{Z}$ et $b \in \mathbb{Z}^*$ s'écrit :

$$a = bq + r \quad \text{et} \quad 0 \leqslant r < b$$

où $q \in \mathbb{Z}$ est le ***quotient*** et $r \in \mathbb{N}$ est le ***reste***.
En `Python` le quotient se calcule par : `a // b`. Le reste se calcule par `a % b`. Exemple : `14 // 3` retourne 4 alors que `14 % 3` (lire 14 modulo 3) retourne 2. On a bien $14 = 3 \times 4 + 2$.
Les calculs avec les modulos sont très pratiques. Par exemple si l'on souhaite tester si un entier est pair, ou impair cela revient à un test modulo 2. Le code est `if (n%2 == 0): ... else: ...`.
Si on besoin de calculer $\cos(n\frac{\pi}{2})$ alors il faut discuter suivant les valeurs de `n%4`.

Appliquons ceci au problème suivant :

Travaux pratiques 4

Combien y-a-t-il d'occurrences du chiffre 1 dans les nombres de 1 à 999 ? Par exemple le chiffre 1 apparaît une fois dans 51 mais deux fois dans 131.

Algorithme . nb-un.py

```
NbDeUn = 0
for N in range(1,999+1):
    ChiffreUnite = N % 10
    ChiffreDizaine = (N // 10) % 10
    ChiffreCentaine = (N // 100) % 10
    if (ChiffreUnite == 1):
        NbDeUn = NbDeUn + 1
    if (ChiffreDizaine == 1):
        NbDeUn = NbDeUn + 1
    if (ChiffreCentaine == 1):
        NbDeUn = NbDeUn + 1
print("Nombre␣d'occurences␣du␣chiffre␣'1'␣:", NbDeUn)
```

Commentaires :

- Comment obtient-on le chiffre des unités d'un entier N ? C'est le reste modulo 10, d'où l'instruction `ChiffreUnite = N % 10`.
- Comment obtient-on le chiffre des dizaines ? C'est plus délicat, on commence par effectuer la division euclidienne de N par 10 (cela revient à supprimer le chiffre des unités, par exemple si $N = 251$ alors `N // 10` retourne 25). Il ne reste plus qu'à calculer le reste modulo 10, (par exemple `(N // 10) % 10` retourne le chiffre des dizaines 5.
- Pour le chiffre des centaines on divise d'abord par 100.

2.2. Écriture des nombres en base 10

L'écriture décimale d'un nombre, c'est associer à un entier N la suite de ses chiffres $[a_0,a_1,\ldots,a_n]$ de sorte que a_i soit le i-ème chiffre de N. C'est-à-dire

$$N = a_n 10^n + a_{n-1}10^{n-1} + \cdots + a_2 10^2 + a_1 10 + a_0 \quad \text{et } a_i \in \{0,1,\ldots,9\}$$

a_0 est le chiffre des unités, a_1 celui des dizaines, a_2 celui des centaines,...

Travaux pratiques 5

1. Écrire une fonction qui à partir d'une liste $[a_0,a_1,\ldots,a_n]$ calcule l'entier N correspondant.
2. Pour un entier N fixé, combien a-t-il de chiffres ? On pourra s'aider d'une inégalité du type $10^n \leqslant N < 10^{n+1}$.
3. Écrire une fonction qui à partir de N calcule son écriture décimale $[a_0,a_1,\ldots,a_n]$.

Voici le premier algorithme :

Algorithme . decimale.py (1)

```
def chiffres_vers_entier(tab):
    N = 0
    for i in range(len(tab)):
        N = N + tab[i] * (10 ** i)
    return N
```

La formule mathématique est simplement $N = a_n 10^n + a_{n-1}10^{n-1} + \cdots + a_2 10^2 + a_1 10 + a_0$. Par exemple `chiffres_vers_entier([4,3,2,1])` renvoie l'entier 1234.

Expliquons les bases sur les ***listes*** (qui s'appelle aussi des ***tableaux***)

- En Python une liste est présentée entre des crochets. Par exemple pour `tab = [4,3,2,1]` alors on accède aux valeurs par `tab[i]` : `tab[0]` vaut 4, `tab[1]` vaut 3, `tab[2]` vaut 2, `tab[3]` vaut 1.
- Pour parcourir les éléments d'un tableau le code est simplement `for x in tab`, x vaut alors successivement 4, 3, 2, 1.
- La longueur du tableau s'obtient par `len(tab)`. Pour notre exemple `len([4,3,2,1])` vaut 4. Pour parcourir toutes les valeurs d'un tableau on peut donc aussi écrire `for i in range(len(tab))`, puis utiliser `tab[i]`, ici i variant ici de 0 à 3.
- La liste vide est seulement notée avec deux crochets : `[]`. Elle est utile pour initialiser une liste.
- Pour ajouter un élément à une liste `tab` existante on utilise la fonction `append`. Par exemple définissons la liste vide `tab=[]`, pour ajouter une valeur à la fin de la liste on saisit : `tab.append(4)`. Maintenant notre liste est [4], elle contient un seul élément. Si on continue avec `tab.append(3)`. Alors maintenant notre liste a deux éléments : [4,3].

Voici l'écriture d'un entier en base 10 :

Algorithme . decimale.py (2)

```
def entier_vers_chiffres(N):
    tab = []
    n = floor(log(N,10))   # le nombre de chiffres est n+1
    for i in range(0,n+1):
        tab.append((N // 10 ** i) % 10)
    return tab
```

Par exemple `entier_vers_chiffres(1234)` renvoie le tableau $[4,3,2,1]$. Nous avons expliqué tout ce dont nous avions besoin sur les listes au-dessus, expliquons les mathématiques.

- Décomposons $\mathbb{N}^*$ sous la forme $[1,10[\cup [10,100[\cup [100,1000[\cup [1\,000,10\,000[\cup \cdots$ Chaque intervalle est du type $[10^n, 10^{n+1}[$. Pour $N \in \mathbb{N}^*$ il existe donc $n \in \mathbb{N}$ tel que $10^n \leqslant N < 10^{n+1}$. Ce qui indique que le nombre de chiffres de N est $n+1$.
  Par exemple si $N = 1234$ alors $1\,000 = 10^3 \leqslant N < 10^4 = 10\,000$, ainsi $n = 3$ et le nombre de chiffres est 4.
- Comment calculer n à partir de N ? Nous allons utiliser le logarithme décimal $\log_{10}$ qui vérifie $\log_{10}(10) = 1$ et $\log_{10}(10^i) = i$. Le logarithme est une fonction croissante, donc l'inégalité $10^n \leqslant N < 10^{n+1}$ devient $\log_{10}(10^n) \leqslant \log_{10}(N) < \log_{10}(10^{n+1})$. Et donc $n \leqslant \log_{10}(N) < n+1$. Ce qui indique donc que $n = E(\log_{10}(N))$ où $E(x)$ désigne la partie entière d'un réel x.

2.3. Module `math`

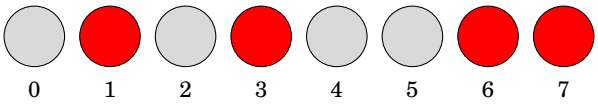
Quelques commentaires informatiques sur un module important pour nous. Les fonctions mathématiques ne sont pas définies par défaut dans `Python` (à part $|x|$ et x^n), il faut faire appel à une librairie spéciale : le module `math` contient les fonctions mathématiques principales.

`abs(x)`	$\lvert x\rvert$
`x ** n`	x^n
`sqrt(x)`	$\sqrt{x}$
`exp(x)`	$\exp x$
`log(x)`	$\ln x$ logarithme népérien
`log(x,10)`	$\log x$ logarithme décimal
`cos(x), sin(x), tan(x)`	$\cos x$, $\sin x$, $\tan x$ en radians
`acos(x), asin(x), atan(x)`	$\arccos x$, $\arcsin x$, $\arctan x$ en radians
`floor(x)`	partie entière $E(x)$:plus grand entier $n \leqslant x$ (*floor* = plancher)
`ceil(x)`	plus petit entier $n \geqslant x$ (*ceil* = plafond)

- Comme on aura souvent besoin de ce module on l'appelle par le code `from math import *`. Cela signifie que l'on importe toutes les fonctions de ce module et qu'en plus on n'a pas besoin de préciser que la fonction vient du module `math`. On peut écrire `cos(3.14)` au lieu `math.cos(3.14)`.
- Dans l'algorithme précédent nous avions utilisé le logarithme décimal `log(x,10)`, ainsi que la partie entière `floor(x)`.

2.4. Écriture des nombres en base 2

On dispose d'une rampe de lumière, chacune des 8 lampes pouvant être allumée (rouge) ou éteinte (gris).



On numérote les lampes de 0 à 7. On souhaite contrôler cette rampe : afficher toutes les combinaisons possibles, faire défiler une combinaison de la gauche à droite (la "chenille"), inverser l'état de toutes les lampes,... Voyons comment l'écriture binaire des nombres peut nous aider. L'***écriture binaire*** d'un nombre c'est son écriture en base 2.
Comment calculer un nombre qui est écrit en binaire ? Le chiffre des "dizaines" correspond à 2 (au lieu de 10), le chiffre des "centaines" à $4 = 2^2$ (au lieu de $100 = 10^2$), le chiffres des "milliers" à $8 = 2^3$ (au lieu de $1000 = 10^3$),... Pour le chiffre des unités cela correspond à $2^0 = 1$ (de même que $10^0 = 1$).
Par exemple 10011_b vaut le nombre 19. Car

$$10011_b = 1\cdot 2^4 + 0\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 16 + 2 + 1 = 19.$$

De façon générale tout entier $N \in \mathbb{N}$ s'écrit de manière unique sous la forme

$$N = a_n 2^n + a_{n-1}2^{n-1} + \cdots + a_2 2^2 + a_1 2 + a_0 \quad \text{et} \quad a_i \in \{0,1\}$$

On note alors $N = a_n a_{n-1} \ldots a_1 a_0\ _b$ (avec un indice b pour indiquer que c'est son écriture binaire).

Travaux pratiques 6

1. Écrire une fonction qui à partir d'une liste $[a_0, a_1, \ldots, a_n]$ calcule l'entier N correspondant à l'écriture binaire $a_n a_{n-1} \ldots a_1 a_0\ _b$.
2. Écrire une fonction qui à partir de N calcule son écriture binaire sous la forme $[a_0, a_1, \ldots, a_n]$.

La seule différence avec la base 10 c'est que l'on calcule avec des puissances de 2.

Algorithme . binaire.py (1)

```
def binaire_vers_entier(tab):
    N = 0
    for i in range(len(tab)):
        N = N + tab[i] * (2 ** i)
    return N
```

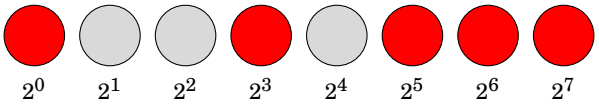
Idem pour le sens inverse où l'on a besoin du logarithme en base 2, qui vérifie $\log_2(2) = 1$ et $\log_2(2^i) = i$.

Algorithme . binaire.py (2)

```
def entier_vers_binaire(N):
    tab = []
    n = floor(log(N,2))    # le nombre de chiffres est n+1
```

```
    for i in range(0,n+1):
        tab.append((N // 2 ** i) % 2)
    return tab
```

Maintenant appliquons ceci à notre problème de lampes. Si une lampe est allumée on lui attribut 1, et si elle est éteinte 0. Pour une rampe de 8 lampes on code $[a_0, a_1, \ldots, a_7]$ l'état des lampes. Par exemple la configuration suivante :



est codé $[1,0,0,1,0,1,1,1]$ ce qui correspond au nombre binaire $11101001_b = 233$.

Travaux pratiques 7

1. Faire une boucle qui affiche toutes les combinaisons possibles (pour une taille de rampe donnée).
2. Quelle opération mathématique élémentaire transforme un nombre binaire $a_n \ldots a_1 a_0{}_b$ en $a_n \ldots a_1 a_0 0{}_b$ (décalage vers la gauche et ajout d'un 0 à la fin) ?
3. Soit $N' = a_n a_{n-1} \ldots a_1 a_0 0{}_b$ (une écriture avec $n+2$ chiffres). Quelle est l'écriture binaire de $N' \pmod{2^{n+1}}$? (C'est une écriture avec $n+1$ chiffres.)
4. En déduire un algorithme qui pour une configuration donnée de la rampe, fait permuter cycliquement (vers la droite) cette configuration. Par exemple $[1,0,1,0,1,1,1,0]$ devient $[0,1,0,1,0,1,1,1]$.
5. Quelle opération mathématique élémentaire permet de passer d'une configuration à son opposée (une lampe éteinte s'allume, et réciproquement). Par exemple si la configuration était $[1,0,1,0,1,1,1,0]$ alors on veut $[0,1,0,1,0,0,0,1]$. (Indication : sur cet exemple calculer les deux nombres correspondants et trouver la relation qui les lie.)

1. Il s'agit d'abord d'afficher les configurations. Par exemple si l'on a 4 lampes alors les configurations sont $[0,0,0,0]$, $[1,0,0,0]$, $[0,1,0,0]$, $[1,1,0,0]$,..., $[1,1,1,1]$. Pour chaque lampe nous avons deux choix (allumé ou éteint), il y a $n+1$ lampes donc un total de 2^{n+1} configurations. Si l'on considère ces configurations comme des nombres écrits en binaire alors l'énumération ci-dessus correspond à compter $0,1,2,3,\ldots,2^{n+1}-1$.
   D'où l'algorithme :

   Algorithme . binaire.py (3)

   ```
   def configurations(n):
       for N in range(2**(n+1)):
           print(entier_vers_binaire_bis(N,n))
   ```

   Où `entier_vers_binaire_bis(N,n)` est similaire à `entier_vers_binaire(N)`, mais en affichant aussi les zéros non significatifs, par exemple 7 en binaire s'écrit 111_b, mais codé sur 8 chiffres on ajoute devant des 0 non significatifs : 00000111_b.
2. En écriture décimale, multiplier par 10 revient à décaler le nombre initial et rajouter un zéro. Par exemple $10 \times 19 = 190$. C'est la même chose en binaire ! Multiplier un nombre par 2

revient sur l'écriture à un décalage vers la gauche et ajout d'un zéro sur le chiffre des unités. Exemple : $19 = 10011_b$ et $2 \times 19 = 38$ donc $2 \times 10011_b = 100110_b$.

3. Partant de $N = a_n a_{n-1} \dots a_1 a_0{}_b$. Notons $N' = 2N$, son écriture est $N' = a_n a_{n-1} \dots a_1 a_0 0{}_b$. Alors $N' \pmod{2^{n+1}}$ s'écrit exactement $a_{n-1} a_{n-2} \dots a_1 a_0 0{}_b$ et on ajoute a_n qui est le quotient de N' par 2^{n+1}.
   Preuve : $N' = a_n \cdot 2^{n+1} + a_{n-1} \cdot 2^n + \dots + a_0 \cdot 2$. Donc $N' \pmod{2^{n+1}} = a_{n-1} \cdot 2^n + \dots + a_0 \cdot 2$. Donc $N' \pmod{2^{n+1}} + a_n = a_{n-1} \cdot 2^n + \dots + a_0 \cdot 2 + a_n$.
4. Ainsi l'écriture en binaire de $N' \pmod{2^{n+1}} + a_n$ s'obtient comme permutation circulaire de celle de N. D'où l'algorithme :

Algorithme . binaire.py (4)

```
def decalage(tab):
    N = binaire_vers_entier(tab)
    n = len(tab)-1   # le nombre de chiffres est n+1
    NN = 2*N % 2**(n+1) + 2*N // 2**(n+1)
    return entier_vers_binaire_bis(NN,n)
```

5. On remarque que si l'on a deux configurations opposées alors leur somme vaut $2^{n+1} - 1$: par exemple avec $[1,0,0,1,0,1,1,1]$ et $[0,1,1,0,1,0,0,0]$, les deux nombres associés sont $N = 11101001_b$ et $N' = 00010110_b$ (il s'agit juste de les réécrire de droite à gauche). La somme est $N + N' = 11101001_b + 00010110_b = 11111111_b = 2^8 - 1$. L'addition en écriture binaire se fait de la même façon qu'en écriture décimale et ici il n'y a pas de retenue. Si M est un nombre avec $n+1$ fois le chiffres 1 alors $M + 1 = 2^{n+1}$. Exemple si $M = 11111_b$ alors $M + 1 = 100000_b = 2^5$; ainsi $M = 2^5 - 1$. Donc l'opposé de N est $N' = 2^{n+1} - 1 - N$ (remarquez que dans $\mathbb{Z}/(2^{n+1} - 1)\mathbb{Z}$ alors $N' \equiv -N$).
   Cela conduit à :

Algorithme . binaire.py (5)

```
def inversion(tab):
    N = binaire_vers_entier(tab)
    n = len(tab)-1   # le nombre de chiffres est n+1
    NN = 2**(n+1)-1 - N
    return entier_vers_binaire_bis(NN,n)
```

Mini-exercices

1. Pour un entier n fixé, combien y-a-t-il d'occurrences du chiffre 1 dans l'écriture des nombres de 1 à n ?
2. Écrire une fonction qui calcule l'écriture décimale d'un entier, sans recourir au log (une boucle `while` est la bienvenue).
3. Écrire un algorithme qui permute cycliquement une configuration de rampe vers la droite.
4. On dispose de $n+1$ lampes, chaque lampe peut s'éclairer de trois couleurs : vert, orange,

rouge (dans cet ordre). Trouver toutes les combinaisons possibles. Comment passer toutes les lampes à la couleur suivante ?

5. Générer toutes les matrices 4×4 n'ayant que des 0 et des 1 comme coefficients. On codera une matrice sous la forme de lignes $[[1,1,0,1],[0,0,1,0],[1,1,1,1],[0,1,0,1]]$.
6. On part du point $(0,0) \in \mathbb{Z}^2$. A chaque pas on choisit au hasard un direction Nord, Sud, Est, Ouest. Si on va au Nord alors on ajoute $(0,1)$ à sa position (pour Sud on ajoute $(0,-1)$; pour Est $(1,0)$; pour Ouest $(-1,0)$). Pour un chemin d'une longueur fixée de n pas, coder tous les chemins possibles. Caractériser les chemins qui repassent par l'origine. Calculer la probabilité p_n de repasser par l'origine. Que se passe-t-il lorsque $n \to +\infty$?
7. Écrire une fonction, qui pour un entier N, affiche son écriture en chiffres romains : $M = 1000$, $D = 500$, $C = 100$, $X = 10$, $V = 5$, $I = 1$. Il ne peut y avoir plus de trois symboles identiques à suivre.

3. Calculs de sinus, cosinus, tangente

Le but de cette section est le calcul des sinus, cosinus, et tangente d'un angle par nous même, avec une précision de 8 chiffres après la virgule.

3.1. Calcul de $\arctan x$

Nous aurons besoin de calculer une fois pour toute $\arctan(10^{-i})$, pour $i = 0,\dots,8$, c'est-à-dire que l'on cherche les angles $\theta_i \in]-\frac{\pi}{2},\frac{\pi}{2}[$ tels que $\tan\theta_i = 10^{-i}$. Nous allons utiliser la formule :

$$\arctan x = \sum_{k=0}^{+\infty} (-1)^k \frac{x^{2k+1}}{2k+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

Travaux pratiques 8

1. Calculer $\arctan 1$.
2. Calculer $\theta_i = \arctan 10^{-i}$ (avec 8 chiffres après la virgule) pour $i = 1,\dots,8$.
3. Pour quelles valeurs de i, l'approximation $\arctan x \simeq x$ était-elle suffisante ?

Algorithme . tangente.py (1)

```
def mon_arctan(x,n):
    somme = 0
    for k in range(0,n+1):
        if (k%2 == 0):  # si k est pair signe +
            somme = somme + 1/(2*k+1) * (x ** (2*k+1))
        else:           # si k est impair signe -
            somme = somme - 1/(2*k+1) * (x ** (2*k+1))
    return somme
```

– La série qui permet de calculer $\arctan x$ est une somme infinie, mais si x est petit alors chacun des termes $(-1)^k \frac{x^{2k+1}}{2k+1}$ est très très petit dès que k devient grand. Par exemple si $0 \leqslant x \leqslant \frac{1}{10}$

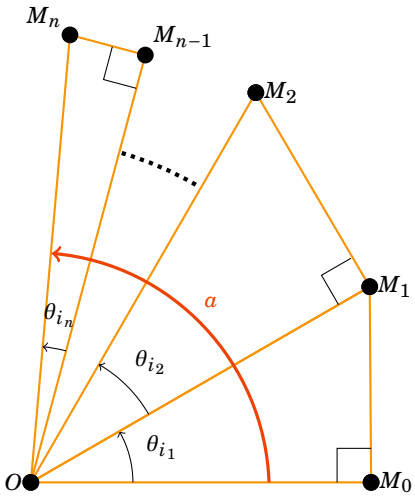
alors $x^{2k+1} \leqslant \frac{1}{10^{2k+1}}$ et donc pour $k \geqslant 4$ nous aurons $\left|(-1)^k \frac{x^{2k+1}}{2k+1}\right| < 10^{-9}$. Chacun des termes suivants ne contribue pas aux 8 premiers chiffres après la virgule. Attention : il se pourrait cependant que la somme de beaucoup de termes finissent par y contribuer, mais ce n'est pas le cas ici (c'est un bon exercice de le prouver).

- Dans la pratique on calcule la somme à un certain ordre $2k+1$ jusqu'à ce que les 8 chiffres après la virgules ne bougent plus. Et en fait on s'aperçoit que l'on a seulement besoin d'utiliser $\arctan x \simeq x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7}$.
- Pour $i \geqslant 4$, $\arctan x \simeq x$ donne déjà 8 chiffres exacts après la virgule !

On remplit les valeurs des angles θ_i obtenus dans une liste nommée `theta`.

3.2. Calcul de $\tan x$

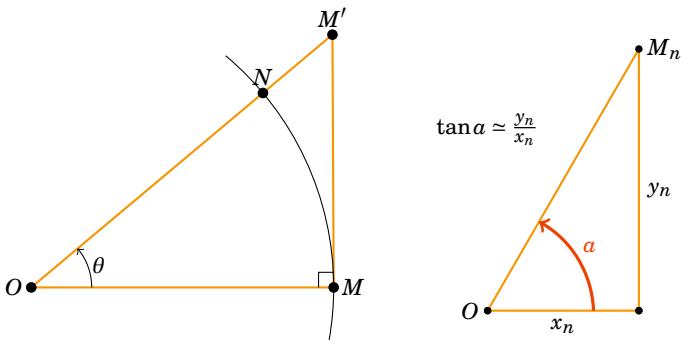
Le principe est le suivant : on connaît un certain nombre d'angles avec leur tangente : les angles θ_i (calculés ci-dessus) avec par définition $\tan\theta_i = 10^{-i}$. Fixons un angle $a \in [0, \frac{\pi}{2}]$. Partant du point $M_0 = (1,0)$, nous allons construire des points $M_1, M_2, \ldots, M_n$ jusqu'à ce que M_n soit (à peu près) sur la demi-droite correspondant à l'angle a. Si M_n a pour coordonnées (x_n, y_n) alors $\tan a = \frac{y_n}{x_n}$. L'angle pour passer d'un point M_k à M_{k+1} est l'un des angles θ_i.



Rappelons que si l'on a un point $M(x,y)$ alors la rotation centrée à l'origine et d'angle θ envoie $M(x,y)$ sur le point $N(x',y')$ avec

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad \text{c'est-à-dire} \quad \begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{cases}$$

Pour un point M, on note M' le point de la demi-droite $[ON)$ tel que les droites (OM) et (MM') soient perpendiculaires en M.



Travaux pratiques 9

1. (a) Calculer la longueur OM'.
   (b) En déduire les coordonnées de M'.
   (c) Exprimez-les uniquement en fonction de x, y et $\tan\theta$.
2. Faire une boucle qui décompose l'angle a en somme d'angles θ_i (à une précision de 10^{-8} ; avec un minimum d'angles, les angles pouvant se répéter).
3. Partant de $M_0 = (1,0)$ calculer les coordonnées des différents M_k, jusqu'au point $M_n(x_n, y_n)$ correspondant à l'approximation de l'angle a. Renvoyer la valeur $\frac{y_n}{x_n}$ comme approximation de $\tan a$.

Voici les préliminaires mathématiques :

- Dans le triangle rectangle OMM' on a $\cos\theta = \frac{OM}{OM'}$ donc $OM' = \frac{OM}{\cos\theta}$.
- D'autre part comme la rotation d'angle θ conserve les distances alors $OM = ON$. Si les coordonnées de M' sont (x'', y'') alors $x'' = \frac{1}{\cos\theta}x'$ et $y'' = \frac{1}{\cos\theta}y'$.
- Ainsi
$$\begin{cases} x'' = \frac{1}{\cos\theta}x' = \frac{1}{\cos\theta}\left(x\cos\theta - y\sin\theta\right) = x - y\tan\theta \\ y'' = \frac{1}{\cos\theta}y' = \frac{1}{\cos\theta}\left(x\sin\theta + y\cos\theta\right) = x\tan\theta + y \end{cases}$$
Autrement dit :
$$\begin{pmatrix} x'' \\ y'' \end{pmatrix} = \begin{pmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

Voici une boucle simple pour décomposer l'angle θ : on commence par retirer le plus grand angle θ_0 autant de fois que l'on peut, lorsque ce n'est plus possible on passe à l'angle θ_1,...

Algorithme . tangente.py (2)

```
i = 0
while (a > precision):      # boucle tant que la precision pas atteinte
    while (a < theta[i]):   # choix du bon angle theta_i à soustraire
        i = i+1
    a = a - theta[i]        # on retire l'angle theta_i et on recommence
```

Ici `precision` est la précision souhaité (pour nous 10^{-9}). Et le tableau `theta` contient les valeurs des angles θ_i.

Posons $x_0 = 1$, $y_0 = 0$ et $M_0 = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$. Alors on définit par récurrence $M_{k+1} = P(\theta_i)\cdot M_k$ où $P(\theta) = \begin{pmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{pmatrix}$. Les θ_i sont ceux apparaissant dans la décomposition de l'angle en somme de θ_i, donc on connaît $\tan\theta_i = 10^{-i}$. Ainsi si l'on passe d'un point M_k à M_{k+1} par un angle θ_i on a simplement :
$$\begin{cases} x_{k+1} = x_k - y_k\cdot 10^{-i} \\ y_{k+1} = x_k\cdot 10^{-i} + y_k \end{cases}$$

La valeur $\frac{y_n}{x_n}$ est la tangente de la somme des angles θ_i, donc une approximation de $\tan a$.
Le code est maintenant le suivant.

Algorithme . tangente.py (3)

```
def ma_tan(a):
    precision = 10**(-9)
    i = 0 ; x = 1 ; y = 0
    while (a > precision):
        while (a < theta[i]):
            i = i+1
        newa = a - theta[i]        # on retire l'angle theta_i
        newx = x - (10**(-i))*y    # on calcule le nouveau point
        newy = (10**(-i))*x + y
        x = newx
        y = newy
        a = newa
    return y/x                     # on renvoie la tangente
```

Commentaires pour conclure :

- En théorie il ne faut pas confondre «précision» et «nombre de chiffres exacts après la virgule». Par exemple 0.999 est une valeur approchée de 1 à 10^{-3} près, mais aucun chiffre après la virgule n'est exact. Dans la pratique c'est la précision qui importe plus que le nombre de chiffres exacts.
- Notez à quel point les opérations du calcul de $\tan x$ sont simples : il n'y a quasiment que des additions à effectuer. Par exemple l'opération $x_{k+1} = x_k - y_k \cdot 10^{-i}$ peut être fait à la main : multiplier par 10^{-i} c'est juste décaler la virgule à droite de i chiffres, puis on additionne. C'est cet algorithme «CORDIC» qui est implémenté dans les calculatrices, car il nécessite très peu de ressources. Bien sûr, si les nombres sont codés en binaire on remplace les 10^{-i} par 2^{-i} pour n'avoir qu'à faire des décalages à droite.

3.3. Calcul de $\sin x$ et $\cos x$

Travaux pratiques 10

Pour $0 \leqslant x \leqslant \frac{\pi}{2}$, calculer $\sin x$ et $\cos x$ en fonction de $\tan x$. En déduire comment calculer les sinus et cosinus de x.

Solution : On sait $\cos^2 + \sin^2 x = 1$, donc en divisant par $\cos^2 x$ on trouve $1 + \tan^2 x = \frac{1}{\cos^2 x}$. On en déduit que pour $0 \leqslant x \leqslant \frac{\pi}{2}$ $\cos x = \frac{1}{\sqrt{1+\tan^2 x}}$. On trouve de même $\sin x = \frac{\tan x}{\sqrt{1+\tan^2 x}}$.
Donc une fois que l'on a calculé $\tan x$ on en déduit $\sin x$ et $\cos x$ par un calcul de racine carrée. Attention c'est valide car x est compris entre 0 et $\frac{\pi}{2}$. Pour un x quelconque il faut se ramener par les formules trigonométriques à l'intervalle $[0, \frac{\pi}{2}]$.

Mini-exercices

1. On dispose de billets de 1, 5, 20 et 100 euros. Trouvez la façon de payer une somme de n euros avec le minimum de billets.
2. Faire un programme qui pour **n'importe quel** $x \in \mathbb{R}$, calcule $\sin x$, $\cos x$, $\tan x$.

3. Pour $t = \tan\frac{x}{2}$ montrer que $\tan x = \frac{2t}{1-t^2}$. En déduire une fonction qui calcule $\tan x$. (Utiliser que pour x assez petit $\tan x \simeq x$).
4. Modifier l'algorithme de la tangente pour qu'il calcule aussi directement le sinus et le cosinus.

4. Les réels

Dans cette partie nous allons voir différentes façons de calculer la constante γ d'Euler. C'est un nombre assez mystérieux car personne ne sait si γ est un nombre rationnel ou irrationnel. Notre objectif est d'avoir le plus de décimales possibles après la virgule en un minimum d'étapes. Nous verrons ensuite comment les ordinateurs stockent les réels et les problèmes que cela engendre.

4.1. Constante γ d'Euler

Considérons la ***suite harmonique*** :

$$H_n = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

et définissons

$$u_n = H_n - \ln n.$$

Cette suite (u_n) admet une limite lorsque $n \to +\infty$: c'est la ***constante γ d'Euler***.

Travaux pratiques 11

1. Calculer les premières décimales de γ. Sachant que $u_n - \gamma \sim \frac{1}{2n}$, combien de décimales exactes peut-on espérer avoir obtenues ?
2. On considère $v_n = H_n - \ln\left(n + \frac{1}{2} + \frac{1}{24n}\right)$. Sachant $v_n - \gamma \sim -\frac{1}{48n^3}$, calculer davantage de décimales.

Algorithme . euler.py (1)

```
def euler1(n):
    somme = 0
    for i in range(n,0,-1):
        somme = somme + 1/i
    return somme - log(n)
```

Algorithme . euler.py (2)

```
def euler2(n):
    somme = 0
    for i in range(n,0,-1):
        somme = somme + 1/i
    return somme - log(n+1/2+1/(24*n))
```

Vous remarquez que la somme est calculée à partir de la fin. Nous expliquerons pourquoi en fin de section.

4.2. 1000 décimales de la constante d'Euler

Il y a deux techniques pour obtenir plus de décimales : (i) pousser plus loin les itérations, mais pour avoir 1000 décimales de γ les méthodes précédentes sont insuffisantes ; (ii) trouver une méthode encore plus efficace. C'est ce que nous allons voir avec la méthode de Bessel modifiée.
Soit

$$w_n = \frac{A_n}{B_n} - \ln n \quad \text{avec} \quad A_n = \sum_{k=1}^{E(\alpha n)} \left(\frac{n^k}{k!}\right)^2 H_k \quad \text{et} \quad B_n = \sum_{k=0}^{E(\alpha n)} \left(\frac{n^k}{k!}\right)^2$$

où $\alpha = 3.59112147...$ est la solution de $\alpha(\ln\alpha - 1) = 1$ et $E(x)$ désigne la partie entière. Alors

$$|w_n - \gamma| \leqslant \frac{C}{e^{4n}}$$

où C est une constante (non connue).

Travaux pratiques 12

1. Programmer cette méthode.
2. Combien d'itérations faut-il pour obtenir 1000 décimales ?
3. Utiliser le module `decimal` pour les calculer.

Voici le code :

Algorithme . euler.py (3)

```
def euler3(n):
    alpha = 3.59112147
    N = floor(alpha*n)      # Borne des sommes
    A = 0 ; B = 0
    H = 0
    for k in range(1,N+1):
        c = ( (n**k)/factorial(k) ) ** 2   # Coefficient commun
        H = H + 1/k                        # Somme harmonique
        A = A + c*H
        B = B + c
    return A/B - log(n)
```

Pour obtenir N décimales il faut résoudre l'inéquation $\frac{C}{e^{4n}} \leqslant \frac{1}{10^N}$. On «passe au log» pour obtenir $n \geqslant \frac{N\ln(10)+\ln(C)}{4}$. On ne connaît pas C mais ce n'est pas si important. Moralement pour une itération de plus on obtient (à peu près) une décimale de plus (c'est-à-dire un facteur 10 sur la précision !). Pour $n \geqslant 800$ on obtient 1000 décimales exactes de la constante d'Euler :

0,
5772156649015328606065120900824024310421593359399235988057672348848677267776646709369470632917467495
1463144724980708248096050401448654283622417399764492353625350033374293733773767394279259525824709491
6008735203948165670853233151776611528621199501507984793745085705740029921354786146694029604325421519
0587755352673313992540129674205137541395491116851028079842348775872050384310939973613725530608893312

676001724795378367592713515772261027349291394079843010341777177808815495706610750101619166334015227893586796549725203621287922655595366962817638879272680132431010476505963703947394957638906572967929601009015125195950922243501409349871228247949747195646976318506676129063811051824197444867836380861749455169892792301877391072945781554316005002182844096053772434203285478367015177394398700302370339518328690001558193988042707411542227819716523011073565833967348717650491941812300040654693142999297779569303100503086303418569803231083691640025892970890985486825777364288253954925873629596133298574739302

Pour obtenir plus de décimales que la précision standard de `Python`, il faut utiliser le module `decimal` qui permet de travailler avec une précision arbitraire fixée.

4.3. Un peu de réalité

En mathématique un réel est un élément de $\mathbb{R}$ et son écriture décimale est souvent infinie après la virgule : par exemple $\pi = 3,14159265\ldots$ Mais bien sûr un ordinateur ne peut pas coder une infinité d'informations. Ce qui se rapproche d'un réel est un ***nombre flottant*** dont l'écriture est :

$$\underbrace{\pm 1,234567890123456789}_{\text{mantisse}}\ e\ \underbrace{\pm 123}_{\text{exposant}}$$

pour $\pm 1,234\ldots \times 10^{\pm 123}$. La ***mantisse*** est un nombre décimal (positif ou négatif) appartenant à $[1,10[$ et l'exposant est un entier (lui aussi positif ou négatif). En `Python` la mantisse à une précision de 16 chiffres après la virgule.
Cette réalité informatique fait que des erreurs de calculs peuvent apparaître même avec des opérations simples. Pour voir un exemple de problème faites ceci :

Travaux pratiques 13

Poser $x = 10^{-16}$, $y = x + 1$, $z = y - 1$. Que vaut z pour `Python` ?

Comme `Python` est très précis nous allons faire une routine qui permet de limiter drastiquement le nombre de chiffres et mettre en évidence les erreurs de calculs.

Travaux pratiques 14

1. Calculer l'exposant d'un nombre réel. Calculer la mantisse.
2. Faire une fonction qui ne conserve que 6 chiffres d'un nombre (6 chiffres en tout : avant + après la virgule, exemple $123,456789$ devient $123,456$).

Voici le code :

Algorithme . reels.py (1)

```
precision = 6                 # Nombre de décimales conservées
def tronquer(x):
    n = floor(log(x,10))                      # Exposant
    m = floor( x * 10 ** (precision-1 - n))   # Mantisse
```

```
    return m * 10 ** (-precision+1+n)        # Nombre tronqué
```

Comme on l'a déjà vu auparavant l'exposant se récupère à l'aide du logarithme en base 10. Et pour tronquer un nombre avec 6 chiffres, on commence par le décaler vers la gauche pour obtenir 6 chiffres avant la virgule (123,456789 devient 123456,789) il ne reste plus qu'à prendre la partie entière (123456) et le redécaler vers la droite (pour obtenir 123,456).

Absorption

Travaux pratiques 15

1. Calculer `tronquer(1234.56 + 0.007)`.
2. Expliquer.

Chacun des nombres 1234,56 et 0,007 est bien un nombre s'écrivant avec moins de 6 décimales mais leur somme 1234,567 a besoin d'une décimale de plus, l'ordinateur ne retient pas la 7-ème décimale et ainsi le résultat obtenu est 1234,56. Le 0,007 n'apparaît pas dans le résultat : il a été victime d'une ***absorption***.

Élimination

Travaux pratiques 16

1. Soient $x = 1234{,}8777$, $y = 1212{,}2222$. Calculer $x - y$ à la main. Comment se calcule la différence $x - y$ avec notre précision de 6 chiffres ?
2. Expliquer la différence.

Comme $x - y = 22{,}6555$ qui n'a que 6 chiffres alors on peut penser que l'ordinateur va obtenir ce résultat. Il n'en est rien, l'ordinateur ne stocke pas x mais `tronquer(x)`, idem pour y. Donc l'ordinateur effectue en fait le calcul suivant : `tronquer(tronquer(x)-tronquer(y))`, il calcule donc $1234{,}87 - 1212{,}22 = 22{,}65$. Quel est le problème ? C'est qu'ensuite l'utilisateur considère –à tort– que le résultat est calculé avec une précision de 6 chiffres. Donc on peut penser que le résultat est 22,6500 mais les 2 derniers chiffres sont une pure invention.
C'est un phénomène d'***élimination***. Lorsque l'on calcule la différence de deux nombres proches, le résultat a en fait une précision moindre. Cela peut être encore plus dramatique avec l'exemple $\delta = 1234{,}569 - 1234{,}55$ la différence est 0,01900 alors que l'ordinateur retournera 0,01000. Il y a presque un facteur deux, et on aura des problèmes si l'on a besoin de diviser par δ.

Signalons au passage une erreur d'interprétation fréquente : ne pas confondre la ***précision*** d'affichage (exemple : on calcule avec 10 chiffres après la virgule) avec l'***exactitude*** du résultat (combien de décimales sont vraiment exactes ?).

Conversion binaire – décimale

Enfin le problème le plus troublant est que les nombres flottants sont stockés en écriture binaire et pas en écriture décimale.

Travaux pratiques 17

Effectuer les commandes suivantes et constater !

1. `sum = 0` puis `for i in range(10): sum = sum + 0.1`. Que vaut `sum` ?
2. `0.1 + 0.1 == 0.2` et `0.1 + 0.1 + 0.1 == 0.3`
3. `x = 0.2 ; print("0.2␣en␣Python␣=␣%.25f" %x)`

La raison est simple mais néanmoins troublante. L'ordinateur ne stocke pas $0,1$, ni $0,2$ en mémoire mais le nombre en écriture binaire qui s'en rapproche le plus.
En écriture décimale, il est impossible de coder $1/3 = 0,3333\ldots$ avec un nombre fini de chiffres après la virgule. Il en va de même ici : l'ordinateur ne peut pas stocker exactement $0,2$. Il stocke un nombre en écriture binaire qui s'en rapproche le plus ; lorsqu'on lui demande d'afficher le nombre stocké, il retourne l'écriture décimale qui se rapproche le plus du nombre stocké, mais ce n'est plus $0,2$, mais un nombre très très proche :

$$0.20000000000000000111022302\ldots$$

4.4. Somme des inverses des carrés

Voyons une situation concrète où ces problèmes apparaissent.

Travaux pratiques 18

1. Faire une fonction qui calcule la somme $S_n = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{n^2}$.
2. Faire une fonction qui calcule cette somme mais en utilisant seulement une écriture décimale à 6 chiffres (à l'aide de la fonction `tronquer()` vue au-dessus).
3. Reprendre cette dernière fonction, mais en commençant la somme par les plus petits termes.
4. Comparez le deux dernières méthodes, justifier et conclure.

La première fonction ne pose aucun problème et utilise toute la précision de `Python`.
Dans la seconde on doit, à chaque calcul, limiter notre précision à 6 chiffres (ici 1 avant la virgule et 5 après).

Algorithme . reels.py (2)

```
def somme_inverse_carres_tronq(n):
  somme = 0
  for i in range(1,n+1):
      somme = tronquer(somme + tronquer(1/(i*i)))
  return somme
```

Il est préférable de commencer la somme par la fin :

Algorithme . reels.py (3)

```
def somme_inverse_carres_tronq_inv(n):
  somme = 0
  for i in range(n,0,-1):
      somme = tronquer(somme + tronquer(1/(i*i)))
  return somme
```

Par exemple pour $n = 100\,000$ l'algorithme `somme_inverse_carres_tronq()` (avec écriture tronquée, sommé dans l'ordre) retourne $1,64038$ alors que l'algorithme `somme_inverse_carres_tronq_inv()` (avec la somme dans l'ordre inverse) on obtient $1,64490$. Avec une précision maximale et n très grand on doit obtenir $1,64493\ldots$ (en fait c'est $\frac{\pi^2}{6}$).
Notez que faire grandir n pour l'algorithme `somme_inverse_carres_tronq()` n'y changera rien, il bloque à 2 décimales exactes après la virgule : $1,64038$! La raison est un phénomène d'absorption : on rajoute des termes très petits devant une somme qui vaut plus de 1. Alors que si l'on part des termes petits, on ajoute des termes petits à une somme petite, on garde donc un maximum de décimales valides avant de terminer par les plus hautes valeurs.

Mini-exercices

1. Écrire une fonction qui approxime la constante α qui vérifie $\alpha(\ln\alpha - 1) = 1$. Pour cela poser $f(x) = x(\ln x - 1) - 1$ et appliquer la méthode de Newton : fixer u_0 (par exemple ici $u_0 = 4$) et $u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)}$.
2. Pour chacune des trois méthodes, calculer le nombre approximatif d'itérations nécessaires pour obtenir 100 décimales de la constante γ d'Euler.
3. Notons $C_n = \frac{1}{4n}\sum_{k=0}^{2n} \frac{[(2k)!]^3}{(k!)^4(16n)^{2k}}$. La formule de Brent-McMillan affirme $\gamma = \frac{A_n}{B_n} - \frac{C_n}{B_n^2} - \ln n + O(\frac{1}{e^{8n}})$ où cette fois les sommations pour A_n et B_n vont jusqu'à $E(\beta n)$ avec $\beta = 4,970625759\ldots$ la solution de $\beta(\ln\beta - 1) = 3$. La notation $O(\frac{1}{e^{8n}})$ indique que l'erreur est $\leqslant \frac{C}{e^{8n}}$ pour une certaine constante C. Mettre en œuvre cette formule. En 1999 cette formule a permis de calculer 100 millions de décimales. Combien a-t-il fallu d'itérations ?
4. Faire une fonction qui renvoie le terme u_n de la suite définie par $u_0 = \frac{1}{3}$ et $u_{n+1} = 4u_n - 1$. Que vaut u_{100} ? Faire l'étude mathématique et commenter.

5. Arithmétique – Algorithmes récursifs

Nous allons présenter quelques algorithmes élémentaires en lien avec l'arithmétique. Nous en profitons pour présenter une façon complètement différente d'écrire des algorithmes : les fonctions récursives.

5.1. Algorithmes récursifs

Voici un algorithme très classique :

Algorithme . recursif.py (1)

```
def factorielle_classique(n):
    produit = 1
    for i in range(1,n+1):
        produit = i * produit
    return produit
```

Voyons comment fonctionne cette boucle. On initialise la variable `produit` à 1, on fait varier un indice i de 1 à n. À chaque étape on multiplie `produit` par i et on affecte le résultat dans `produit`. Par exemple si $n = 5$ alors la variable `produit` s'initialise à 1, puis lorsque i varie la variable produit devient $1 \times 1 = 1$, $2 \times 1 = 2$, $3 \times 2 = 6$, $4 \times 6 = 24$, $5 \times 24 = 120$. Vous avez bien sûr reconnus le calcul de 5!

Étudions un autre algorithme.

Algorithme . recursif.py (2)

```
def factorielle(n):
    if (n==1):
        return 1
    else:
        return n * factorielle(n-1)
```

Que fait cet algorithme ? Voyons cela pour $n = 5$. Pour $n = 5$ la condition du «si» (if) n'est pas vérifiée donc on passe directement au «sinon» (else). Donc `factorielle(5)` renvoie comme résultat : `5 * factorielle(4)`. On a plus ou moins progressé : le calcul n'est pas fini car on ne connaît pas encore `factorielle(4)` mais on s'est ramené à un calcul au rang précédent, et on itère : `factorielle(5) = 5 * factorielle(4) = 5 * 4 * factorielle(3) = 5 * 4 * 3 * factorielle(2)` et enfin `factorielle(5) = 5 * 4 * 3 * 2 * factorielle(1)`. Pour `factorielle(1)` la condition du `if (n==1)` est vérifiée et alors `factorielle(1)=1`. Le bilan est donc que `factorielle(5) = 5 * 4 * 3 *` c'est bien 5!
Une fonction qui lorsque elle s'exécute s'appelle elle-même est une ***fonction récursive***. Il y a une analogie très forte avec la récurrence. Par exemple on peut définir la suite des factorielles ainsi :

$$u_1 = 1 \quad \text{et} \quad u_n = n \times u_{n-1} \text{ si } n \geqslant 2.$$

Nous avons ici $u_n = n!$ pour tout $n \geqslant 1$.
Comme pour la récurrence une fonction récursive comporte une étape d'***initialisation*** (ici `if (n==1): return 1` correspondant à $u_1 = 1$) et une étape d'***hérédité*** (ici `return n * factorielle(n-1)` correspondant à $u_n = n \times u_{n-1}$).

On peut même faire deux appels à la fonction :

Algorithme . recursif.py (3)

```
def fibonacci(n):
    if (n==0) or (n==1):
        return 1
    else:
        return fibonacci(n-1)+fibonacci(n-2)
```

Faites-le calcul de `fibonacci(5)`. Voici la version mathématique des nombres de Fibonacci.

$$F_0 = 1,\ F_1 = 1 \quad \text{et} \quad F_n = F_{n-1} + F_{n-2} \quad \text{si } n \geqslant 2.$$

On obtient un nombre en additionnant les deux nombres des rangs précédents :

$$1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 8 \quad 13 \quad 21 \quad 34 \quad \ldots$$

5.2. L'algorithme d'Euclide

L'algorithme d'Euclide est basé sur le principe suivant

$$\text{si } b|a \text{ alors } \operatorname{pgcd}(a,b) = b \qquad \text{sinon } \operatorname{pgcd}(a,b) = \operatorname{pgcd}(b, a \bmod b)$$

Travaux pratiques 19

1. Créer une fonction récursive `pgcd(a,b)` qui calcule le pgcd.
2. On note p_n la probabilité que deux entiers a,b tirés au hasard dans $1,2,\ldots,n$ soient premiers entre eux. Faire une fonction qui approxime p_n. Lorsque n devient grand, comparer p_n et $\frac{6}{\pi^2}$.

Voici le code pour l'algorithme d'Euclide récursif. Notez à quel point le code est succinct et épuré !

Algorithme . arith.py (1)

```
def pgcd(a,b):
    if a%b == 0:
        return b
    else:
        return pgcd(b, a%b)
```

Deux entiers a,b sont premiers entre eux ssi $\operatorname{pgcd}(a,b) = 1$, donc voici l'algorithme :

Algorithme . arith.py (2)

```
def nb_premiers_entre_eux(n,nbtirages):
    i = 1
    nbpremiers = 0
    while i <= nbtirages:
        i = i+1
        a = random.randint(1,n)
        b = random.randint(1,n)
```

```
        if pgcd(a,b)==1:
            nbpremiers = nbpremiers + 1
    return nbpremiers
```

On tire au hasard deux entiers a et b entre 1 et n et on effectue cette opération `nbtirages` fois. Par exemple entre 1 et 1000 si l'on effectue 10000 tirage on trouve une probabilité mesurée par `nbpremiers/nbtirages` de $0,60\ldots$ (les décimales d'après dépendent des tirages).
Lorsque n tend vers $+\infty$ alors $p_n \to \frac{6}{\pi^2} = 0.607927\ldots$ et on dit souvent que : «la probabilité que deux entiers tirés au hasard soient premiers entre eux est $\frac{6}{\pi^2}$.»

Commentaires sur les algorithmes récursifs :

- Les algorithmes récursifs ont souvent un code très court, et proche de la formulation mathématique lorsque l'on a une relation de récurrence.
- Selon le langage ou la fonction programmée il peut y avoir des problèmes de mémoire (si par exemple pour calculer 5! l'ordinateur a besoin de stocker 4! pour lequel il a besoin de stocker 3!...).
- Il est important de bien réfléchir à la condition initiale (qui est en fait celle qui termine l'algorithme) et à la récurrence sous peine d'avoir une fonction qui boucle indéfiniment !
- Il n'existe pas des algorithmes récursifs pour tout (voir par exemple les nombres premiers) mais ils apparaissent beaucoup dans les algorithmes de tris. Autre exemple : la dichotomie se programme très bien par une fonction récursive.

5.3. Nombres premiers

Les nombres premiers offrent peu de place aux algorithmes récursifs car il n'y a pas de lien de récurrence entre les nombres premiers.

Travaux pratiques 20

1. Écrire une fonction qui détecte si un nombre n est premier ou pas en testant s'il existe des entiers k qui divise n. (On se limitera aux entiers $2 \leqslant k \leqslant \sqrt{n}$, pourquoi ?).
2. Faire un algorithme pour le crible d'Eratosthène : écrire tous les entiers de 2 à n, conserver 2 (qui est premier) et barrer tous les multiples suivants de 2. Le premier nombre non barré (c'est 3) est premier. Barrer tous les multiples suivants de 3,...
3. Dessiner la spirale d'Ulam : on place les nombres entiers en spirale, et on colorie en rouge les nombres premiers.

$$\begin{array}{cccc} & & \cdots & \cdots \\ 5 & 4 & 3 & \vdots \\ 6 & 1 & 2 & 11 \\ 7 & 8 & 9 & 10 \end{array}$$

1. Si n n'est pas premier alors $n = a \times b$ avec $a, b \geqslant 2$. Il est clair que soit $a \leqslant \sqrt{n}$ ou bien $b \leqslant \sqrt{n}$ (sinon $n = a \times b > n$). Donc il suffit de tester les diviseurs $2 \leqslant k \leqslant \sqrt{n}$. D'où l'algorithme :

Algorithme . arith.py (3)

```
def est_premier(n):
    if (n<=1): return False
    k = 2
    while k*k <= n:
        if n%k==0:
            return False
        else:
            k = k +1
    return True
```

Notez qu'il vaut mieux écrire la condition `k*k <= n` plutôt que `k <= sqrt(n)` : il est beaucoup plus rapide de calculer le carré d'un entier plutôt qu'extraire une racine carrée.

Nous avons utilisé un nouveau type de variable : un ***booléen*** est une variable qui ne peut prendre que deux états Vrai ou Faux (ici *True* or *False*, souvent codé 1 et 0). Ainsi `est_premier(13)` renvoie `True`, alors que `est_premier(14)` renvoie `False`.

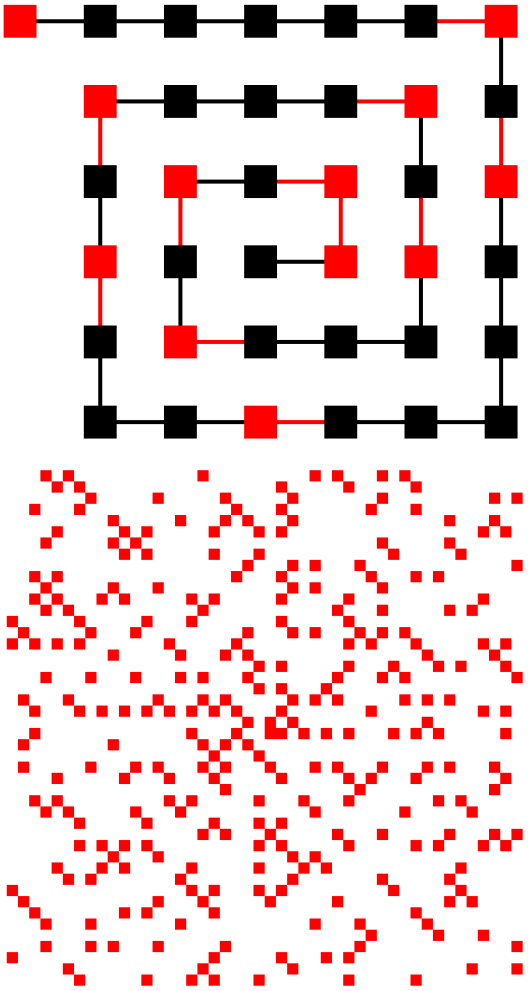
2. Pour le crible d'Eratosthène le plus dur est de trouver le bon codage de l'information.

Algorithme . arith.py (4)

```
def eratosthene(n):
    liste_entiers = list(range(n+1))  # tous les entiers
    liste_entiers[1] = 0              # 1 n'est pas premier
    k = 2                             # on commence par les multiples de 2
    while k*k <= n:
        if liste_entiers[k] != 0: # si le nombre k n'est pas barré
            i = k                 # les i sont les multiples de k
            while i <= n-k:
                i = i+k
                liste_entiers[i] = 0  # multiples de k : pas premiers
        k = k +1
    liste_premiers = [k for k in liste_entiers if k !=0]  # efface les 0
    return liste_premiers
```

Ici on commence par faire un tableau contenant les entiers `[0,1,2,3,4,5,6,7,8,9,10,11,12,13,...]`. Pour signifier qu'un nombre n'est pas premier ou remplace l'entier par `0`. Comme 1 n'est pas un nombre premier : on le remplace par 0. Puis on fait une boucle, on part de 2 et on remplace tous les autres multiples de 2 par 0 : la liste est maintenant : `[0,0,2,3,0,5,0,7,0,9,0,11,0,13,...]`. Le premiers nombre après 2 est 3 c'est donc un nombre premier. (car s'il n'a aucun diviseur autre que 1 et lui-même car sinon il aurait été rayé). On garde 3 et remplace tous les autres multiples de 3 par 0. La liste est maintenant : `[0,0,2,3,0,5,0,7,0,0,0,11,0,13,...]`. On itère ainsi, à la fin on efface les zéros pour obtenir : `[2,3,5,7,11,13,...]`.

3. Pour la spirale d'Ulam la seule difficulté est de placer les entiers sur une spirale, voici le résultat.



À gauche le début de la spirale (de $n = 1$ à 37) en rouge les nombres premiers (en noir les nombres non premiers) ; à droite le motif obtenu jusqu'à de grandes valeurs (en blanc les nombres non premiers).

Mini-exercices

1. Écrire une version itérative et une version récursive pour les fonctions suivantes : (a) la somme des carrés des entiers de 1 à n ; (b) 2^n (sans utiliser d'exposant) ; (c) la partie entière d'un réel $x \geqslant 0$; (d) le quotient de la division euclidienne de a par b (avec $a \in \mathbb{N}$, $b \in \mathbb{N}^*$) ; (e) le reste de cette division euclidienne (sans utiliser les commandes % ni //).
2. Écrire une version itérative de la suite de Fibonacci.
3. Écrire une version itérative de l'algorithme d'Euclide. Faire une version qui calcule les coefficients de Bézout.
4. Écrire une fonction itérative, puis récursive, qui pour un entier n renvoie la liste de ses diviseurs. Dessiner une spirale d'Ulam, dont l'intensité de la couleur dépend du nombre de diviseurs.
5. Une suite de Syracuse est définie ainsi : partant d'un entier s'il est pair on le divise par deux, s'il est impair on le multiplie par 3 et on ajoute 1. On itère ce processus. Quelle conjecture peut-on faire sur cette suite ?

6. Dessiner le triangle de Pascal $\begin{array}{ccccccc} & & & 1 & & & \\ & & 1 & & 1 & & \\ & 1 & & 2 & & 1 & \\ & & \cdots & & & & \end{array}$ Ensuite effacer tous les coefficients pairs (ou mieux : remplacer les coefficients pairs par un carré blanc et les coefficients impairs par un carré rouge). Quelle figure reconnaissez-vous ?

6. Polynômes – Complexité d'un algorithme

Nous allons étudier la complexité des algorithmes à travers l'exemple des polynômes.

6.1. Qu'est-ce qu'un algorithme ?

Qu'est ce qu'un algorithme ? Un algorithme est une succession d'instructions qui renvoie un résultat. Pour être vraiment un algorithme on doit justifier que le résultat retourné est ***exact*** (le programme fait bien ce que l'on souhaite) et ceci en un ***nombre fini d'étapes*** (cela renvoie le résultat en temps fini).
Maintenant certains algorithmes peuvent être plus rapides que d'autres. C'est souvent le temps de calcul qui est le principal critère, mais cela dépend du langage et de la machine utilisée. Il existe une manière plus mathématique de faire : la ***complexité*** d'un algorithme c'est le nombre d'opérations élémentaires à effectuer.
Ces opérations peuvent être le nombre d'opérations au niveau du processeur, mais pour nous ce sera le nombre d'additions +, le nombre de multiplications × à effectuer. Pour certains algorithmes la vitesse d'exécution n'est pas le seul paramètre mais aussi la taille de la mémoire occupée.

6.2. Polynômes

Travaux pratiques 21

On code un polynôme $a_0 + a_1X + \cdots + a_nX^n$ sous la forme d'une liste $[a_0, a_1, \ldots, a_n]$.

1. Écrire une fonction correspondant à la somme de deux polynômes. Calculer la complexité de cet algorithme (en terme du nombre d'additions sur les coefficients, en fonctions du degré des polynômes).
2. Écrire une fonction correspondant au produit de deux polynômes. Calculer la complexité de cet algorithme (en terme du nombre d'additions et de multiplications sur les coefficients).
3. Écrire une fonction correspondant au quotient et au reste de la division euclidienne de A par B où B est un polynôme unitaire (son coefficient de plus haut degré est 1). Majorer la complexité de cet algorithme (en terme du nombre d'additions et de multiplications sur les coefficients).

1. La seule difficulté est de gérer les indices, en particulier on ne peut appeler un élément d'une liste en dehors des indices où elle est définie. Une bonne idée consiste à commencer par définir une fonction `degre(poly)`, qui renvoie le degré du polynôme (attention au 0 non significatifs).

   Voici le code dans le cas simple où $\deg A = \deg B$:

Algorithme . polynome.py (1)

```
def somme(A,B):            # si deg(A)=deg(B)
    C = []
    for i in range(0,degre(A)+1):
        s = A[i]+B[i]
        C.append(s)
```

Calculons sa complexité, on suppose $\deg A \leqslant n$ et $\deg B \leqslant n$: il faut faire l'addition des coefficients $a_i + b_i$, pour i variant de 0 à n : donc la complexité est de $n+1$ additions (dans $\mathbb{Z}$ ou $\mathbb{R}$).

2. Pour le produit il faut se rappeler que si $A(X) = \sum_{i=0}^{m} a_i X^i$, $B(X) = \sum_{j=0}^{n} b_j X^j$ et $C = A \times B = \sum_{k=0}^{m+n} c_k X^k$ alors le k-ème coefficient de C est $c_k = \sum_{i+j=k} a_i \times b_j$. Dans la pratique on fait attention de ne pas accéder à des coefficients qui n'ont pas été définis.

Algorithme . polynome.py (2)

```
def produit(A,B):
    C = []
    for k in range(degre(A)+degre(B)+1):
        s = 0
        for i in range(k+1):
            if (i <= degre(A)) and (k-i <= degre(B)):
                s = s + A[i]*B[k-i]
        C.append(s)
    return C
```

Pour la complexité on commence par compter le nombre de multiplications (dans $\mathbb{Z}$ ou $\mathbb{R}$). Notons $m = \deg A$ et $n = \deg B$. Alors il faut multiplier les $m+1$ coefficients de A par les $n+1$ coefficients de B : il y a donc $(m+1)(n+1)$ multiplications.

Comptons maintenant les additions : les coefficients de $A \times B$ sont : $c_0 = a_0 b_0$, $c_1 = a_0 b_1 + a_1 b_0$, $c_2 = a_2 b_0 + a_1 b_1 + a_2 b_0$,...

Nous utilisons l'astuce suivante : nous savons que le produit $A \times B$ est de degré $m+n$ donc a (au plus) $m+n+1$ coefficients. Partant de $(m+1)(n+1)$ produits, chaque addition regroupe deux termes, et nous devons arriver à $m+n+1$ coefficients. Il y a donc $(m+1)(n+1) - (m+n+1) = mn$ additions.

3. Pour la division euclidienne, le principe est de poser une division de polynôme. Par exemple pour $A = 2X^4 - X^3 - 2X^2 + 3X - 1$ et $B = X^2 - X + 1$.

$$
\begin{array}{rl|l}
 & 2X^4-X^3-2X^2+3X-1 & X^2-X+1 \\
- & 2X^4-2X^3+2X^2 & \hline \\
\hline
 & X^3-4X^2+3X-1 & 2X^2+X-3 \\
- & X^3-X^2+X & \\
\hline
 & -3X^2+2X-1 & \\
- & -3X^2+3X-3 & \\
\hline
 & -X+2 &
\end{array}
$$

Alors on cherche quel monôme P_1 fait diminuer le degré de $A-P_1B$, c'est $2X^2$ (le coefficient 2 est le coefficient dominant de A). On pose ensuite $R_1 = A-P_1B = X^3-4X^2+3X-1$, $Q_1 = 2X^2$, on recommence avec R_1 divisé par B, $R_2 = R_1 - P_2B$ avec $P_2 = X$, $Q_2 = Q_1 + P_2$,... On arrête lorsque $\deg R_i < \deg B$.

Algorithme . polynome.py (3)

```
def division(A,B):
    Q = [0]      # Quotient
    R = A        # Reste
    while (degre(R) >= degre(B)):
        P = monome(R[degre(R)],degre(R)-degre(B))
        R = somme(R,produit(-P,B))
        Q = somme(Q,P)
    return Q,R
```

C'est une version un peu simplifiée du code : où $P = r_n X^{\deg R - \deg B}$ et où il faut remplacer $-P$ par $[-a_0,-a_1,...]$. Si $A,B \in \mathbb{Z}[X]$ alors le fait que B soit unitaire implique que Q et R sont aussi à coefficients entiers.

Quelle est la complexité de la division euclidienne ? À chaque étape on effectue une multiplication de polynômes ($P_i \times B$) puis une addition de polynôme ($R_i - P_iB$) ; à chaque étape le degré de R_i diminue (au moins) de 1. Donc il y a au plus $\deg A - \deg B + 1$ étapes.

Mais dans la pratique c'est plus simple que cela. La multiplication $P_i \times B$ est très simple : car P_i est un monôme $P_i = p_i X^i$. Multiplier par X^i c'est juste un décalage d'indice (comme multiplier par 10^i en écriture décimale) c'est donc une opération négligeable. Il reste donc à multiplier les coefficients de B par p_i : il y a donc $\deg B + 1$ multiplications de coefficients. La soustraction aussi est assez simple on retire à R_i un multiple de B, donc on a au plus $\deg B + 1$ coefficients à soustraire : il y a à chaque étape $\deg B + 1$ additions de coefficients.

Bilan : si $m = \deg A$ et $n = \deg B$ alors la division euclidienne s'effectue en au plus $(m-n+1)(m+1)$ multiplications et le même nombre d'additions (dans $\mathbb{Z}$ ou $\mathbb{R}$).

6.3. Algorithme de Karatsuba

Pour diminuer la complexité de la multiplication de polynômes, on va utiliser un paradigme très classique de programmation : « diviser pour régner ». Pour cela, on va décomposer les polynômes à

multiplier P et Q de degrés strictement inférieurs à $2n$ en

$$P = P_1 + P_2 \cdot X^n \quad \text{et} \quad Q = Q_1 + Q_2 \cdot X^n$$

avec les degrés de P_1, P_2, Q_1 et Q_2 strictement inférieurs à n.

Travaux pratiques 22

1. Écrire une formule qui réduit la multiplication des polynômes P et Q de degrés strictement inférieurs à $2n$ en multiplications de polynômes de degrés strictement inférieurs à n.
2. Programmer un algorithme récursif de multiplication qui utilise la formule précédente. Quelle est sa complexité ?
3. On peut raffiner cette méthode avec la remarque suivante de Karatsuba : le terme intermédiaire de $P \cdot Q$ s'écrit

   $$P_1 \cdot Q_2 + P_2 \cdot Q_1 = (P_1 + P_2) \cdot (Q_1 + Q_2) - P_1 Q_1 - P_2 Q_2$$

   Comme on a déjà calculé $P_1 Q_1$ et $P_2 Q_2$, on échange deux multiplications et une addition (à gauche) contre une multiplication et quatre additions (à droite). Écrire une fonction qui réalise la multiplication de polynômes à la Karatsuba.
4. Trouver la formule de récurrence qui définit la complexité de la multiplication de Karatsuba. Quelle est sa solution ?

1. Il suffit de développer le produit $(P_1 + X^n P_2) \cdot (Q_1 + X^n Q_2)$:

   $$(P_1 + X^n P_2) \cdot (Q_1 + X^n Q_2) = P_1 Q_1 + X^n \cdot (P_1 Q_2 + P_2 Q_1) + X^{2n} \cdot P_2 Q_2$$

   On se ramène ainsi aux quatre multiplications $P_1 Q_1$, $P_1 Q_2$, $P_2 Q_1$ et $P_2 Q_2$ entre polynômes de degrés strictement inférieurs à n, plus deux multiplications par X^n et X^{2n} qui ne sont que des ajouts de zéros en tête de liste.
2. On sépare les deux étapes de l'algorithme : d'abord la découpe des polynômes (dans laquelle il ne faut pas oublier de donner n en argument car ce n'est pas forcément le milieu du polynôme, n doit être le même pour P et Q). Le découpage `P1,P2 = decoupe(P,n)` correspond à l'écriture $P = P_1 + X^n P_2$.

   Algorithme . polynome.py (4)

   ```
   def decoupe(P,n):
       if (degre(P)<n): return P, [0]
       else: return P[0:n], P[n:]
   ```

   On a aussi besoin d'une fonction `produit_monome(P,n)` qui renvoie le polynôme $X^n \cdot P$ par un décalage. Voici la multiplication proprement dite avec les appels récursifs et leur combinaison.

Algorithme . polynome.py (5)

```
def produit_assez_rapide(P,Q):
    p = degre(P) ; q = degre(Q)
    if (p == 0): return [P[0]*k for k in Q]  # Condition initiale: P=cst
    if (q == 0): return [Q[0]*k for k in P]  # Condition initiale: Q=cst
    n = (max(p,q)+1)//2                      # demi-degré
    P1,P2 = decoupe(P,n)                     # decoupages
    Q1,Q2 = decoupe(Q,n)
    P1Q1 = produit_assez_rapide(P1,Q1)       # produits en petits degrés
    P2Q2 = produit_assez_rapide(P2,Q2)
    P1Q2 = produit_assez_rapide(P1,Q2)
    P2Q1 = produit_assez_rapide(P2,Q1)
    R1 = produit_monome(somme(P1Q2,P2Q1),n)  # décalages
    R2 = produit_monome(P2Q2,2*n)
    return somme(P1Q1,somme(R1,R2))          # sommes
```

La relation de récurrence qui exprime la complexité de cet algorithme est $C(n) = 4C(n/2) + O(n)$ et elle se résout en $C(n) = O(n^2)$. Voir la question suivante pour une méthode de résolution.

3.

Algorithme . polynome.py (6)

```
def produit_rapide(P,Q):
    p = degre(P) ; q = degre(Q)
    if (p == 0): return [P[0]*k for k in Q]  # Condition initiale: P=cst
    if (q == 0): return [Q[0]*k for k in P]  # Condition initiale: Q=cst
    n = (max(p,q)+1)//2                      # demi-degré
    P1,P2 = decoupe(P,n)                     # decoupages
    Q1,Q2 = decoupe(Q,n)
    P1Q1 = produit_rapide(P1,Q1)             # produits en petits degrés
    P2Q2 = produit_rapide(P2,Q2)
    PQ = produit_rapide(somme(P1,P2),somme(Q1,Q2))
    R1 = somme(PQ,somme([-k for k in P1Q1],[-k for k in P2Q2]))
    R1 = produit_monome(R1,n)                # décalages
    R2 = produit_monome(P2Q2,2*n)
    return somme(P1Q1,somme(R1,R2))          # sommes
```

4. Notons $C(n)$ la complexité de la multiplication entre deux polynômes de degrés strictement inférieurs à n. En plus des trois appels récursifs, il y a des opérations linéaires : deux calculs de degrés, deux découpes en $n/2$ puis des additions : deux de taille $n/2$, une de taille n, une de taille $3n/2$ et une de taille $2n$. On obtient donc la relation de récurrence suivante :

$$C(n) = 3 \cdot C(n/2) + \gamma n$$

où $\gamma = \frac{15}{2}$. Une méthode de résolution est de poser $\alpha_\ell = \frac{C(2^\ell)}{3^\ell}$ qui vérifie $\alpha_\ell = \alpha_{\ell-1} + \gamma\left(\frac{2}{3}\right)^\ell$.

D'où on tire, puisque $\alpha_0 = C(1) = 1$,

$$\alpha_\ell = \gamma \sum_{k=1}^{\ell} \left(\frac{2}{3}\right)^k + \alpha_0 = 3\gamma\left(1 - \left(\frac{2}{3}\right)^{\ell+1}\right) + 1 - \gamma$$

puis pour $n = 2^\ell$:

$$C(n) = C(2^\ell) = 3^\ell \alpha_\ell = \gamma(3^{\ell+1} - 2^{\ell+1}) + (1-\gamma)3^\ell = O(3^\ell) = O(2^{\ell \frac{\ln 3}{\ln 2}}) = O(n^{\frac{\ln 3}{\ln 2}})$$

La complexité de la multiplication de Karatsuba est donc $O(n^{\frac{\ln 3}{\ln 2}}) \simeq O(n^{1.585})$.

6.4. Optimiser ses algorithmes

Voici quelques petites astuces pour accélérer l'écriture ou la vitesse des algorithmes :

- `k ** 3` au lieu de `k * k * k` (cela économise de la mémoire, une seule variable au lieu de 3) ;
- `k ** 2 <= n` au lieu de `k <= sqrt(n)` (les calculs avec les entiers sont beaucoup plus rapides qu'avec les réels) ;
- `x += 1` au lieu de `x = x +1` (gain de mémoire) ;
- `a,b = a+b, a-b` au lieu de `newa = a+b ; newb = a-b ; a = newa ; b = newb` (gain de mémoire, code plus court).

Cependant il ne faut pas que cela nuise à la lisibilité du code : il est important que quelqu'un puisse relire et modifier votre code. Le plus souvent c'est vous même qui modifierez les algorithmes qui vous avez écrits et vous serez ravi d'y trouver des commentaires clairs et précis !

Mini-exercices

1. Faire une fonction qui renvoie le pgcd de deux polynômes.
2. Comparer les complexités des deux méthodes suivantes pour évaluer un polynôme P en une valeur $x_0 \in \mathbb{R}$: $P(x_0) = a_0 + a_1 x_0 + \cdots + a_{n-1} x_0^{n-1} + a_n x_0^n$ et $P(x_0) = a_0 + x_0\Big(a_1 + x_0\big(a_2 + \cdots + x_0(a_{n-1} + a_n x_0)\big)\Big)$ (méthode de Horner).
3. Comment trouver le maximum d'une liste ? Montrer que votre méthode est de complexité minimale (en terme du nombre de comparaisons).
4. Soit $f : [a,b] \to \mathbb{R}$ une fonction continue vérifiant $f(a) \cdot f(b) \leqslant 0$. Combien d'itérations de la méthode de dichotomie sont nécessaires pour obtenir une racine de $f(x) = 0$ avec une précision inférieure à ε ?
5. Programmer plusieurs façons de calculer les coefficients du binôme de Newton $\binom{n}{k}$ et les comparer.
6. Trouver une méthode de calcul de 2^n qui utilise peu de multiplications. On commencera par écrire n en base 2.

Auteurs

Rédaction : Arnaud Bodin
Relecture : Jean-François Barraud
Remerciements à Lionel Rieg pour son tp sur l'algorithme de Karatsuba

3 Cryptographie

Vidéo ■ partie 1. Le chiffrement de César
Vidéo ■ partie 2. Le chiffrement de Vigenère
Vidéo ■ partie 3. La machine Enigma et les clés secrètes
Vidéo ■ partie 4. La cryptographie à clé publique
Vidéo ■ partie 5. L'arithmétique pour RSA
Vidéo ■ partie 6. Le chiffrement RSA

1. Le chiffrement de César

1.1. César a dit...

Jules César a-t-il vraiment prononcé la célèbre phrase :

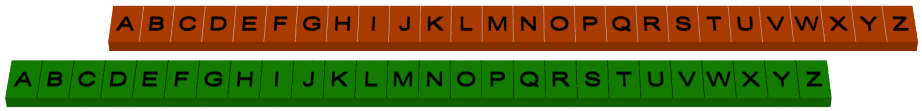
DOHD MDFWD HVW

ou bien comme le disent deux célèbres Gaulois : « Ils sont fous ces romains ! ».

En fait César, pour ses communications importantes à son armée, cryptait ses messages. Ce que l'on appelle le chiffrement de César est un décalage des lettres : pour crypter un message, A devient D, B devient E, C devient F,...

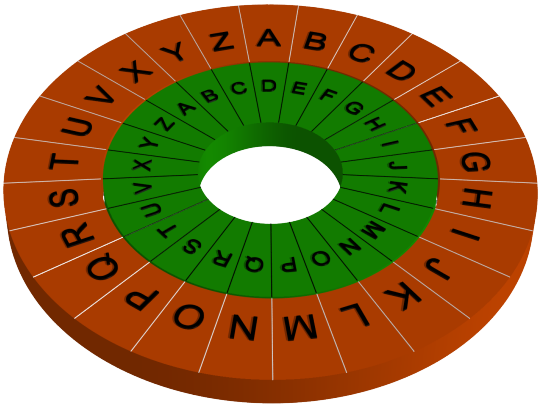
$$A \longmapsto D \quad B \longmapsto E \quad C \longmapsto F \quad \dots \quad W \longmapsto Z \quad X \longmapsto A \quad Y \longmapsto B \quad Z \longmapsto C$$

Voici une figure avec l'alphabet d'origine en haut et en **rouge**, en correspondance avec l'alphabet pour le chiffrement en-dessous et en **vert**.



Nous adopterons la convention suivante, en **vert** c'est la partie du message à laquelle tout le monde a accès (ou qui pourrait être intercepté), c'est donc le message crypté. Alors qu'en **rouge** c'est la partie du message confidentiel, c'est le message en clair.

Pour prendre en compte aussi les dernières lettres de l'alphabet, il est plus judicieux de représenté l'alphabet sur un anneau. Ce décalage est un ***décalage circulaire*** sur les lettres de l'alphabet.



Pour déchiffrer le message de César, il suffit de décaler les lettres dans l'autre sens, D se déchiffre en A, E en B,...
Et la célèbre phrase de César est :

ALEA JACTA EST

qui traduite du latin donne « Les dés sont jetés ».

1.2. Des chiffres et des lettres

Il est plus facile de manipuler des nombres que des lettres, aussi nous passons à une formulation mathématique. Nous associons à chacune des 26 lettres de A à Z un nombre de 0 à 25. En termes mathématiques, nous définissons une bijection :

$$f : \{A, B, C, \ldots, Z\} \longrightarrow \{0, 1, 2, \ldots, 25\}$$

par

$$A \longmapsto 0 \quad B \longmapsto 1 \quad C \longmapsto 2 \quad \ldots \quad Z \longmapsto 25$$

Ainsi "**A L E A**" devient "**0 11 4 0**".
Le chiffrement de César est un cas particulier de ***chiffrement mono-alphabétique***, c'est-à-dire un chiffrement lettre à lettre.
Quel est l'intérêt ? Nous allons voir que le chiffrement de César correspond à une opération mathématique très simple. Pour cela, rappelons la notion de congruence et l'ensemble $\mathbb{Z}/26\mathbb{Z}$.

1.3. Modulo

Soit $n \geqslant 2$ un entier fixé.

Définition 1

On dit que a ***est congru à*** b ***modulo*** n, si n divise $b - a$. On note alors

$$a \equiv b \pmod{n}.$$

Pour nous $n = 26$. Ce qui fait que $28 \equiv 2 \pmod{26}$, car $28 - 2$ est bien divisible par 26. De même $85 = 3 \times 26 + 7$ donc $85 \equiv 7 \pmod{26}$.
On note $\mathbb{Z}/26\mathbb{Z}$ l'ensemble de tous les éléments de $\mathbb{Z}$ modulo 26. Cet ensemble peut par exemple être représenté par les 26 éléments $\{0, 1, 2, \ldots, 25\}$. En effet, puisqu'on compte modulo 26 :

$$0,\ 1,\ 2,\ \ldots,\ 25, \quad \text{puis} \quad 26 \equiv 0,\ 27 \equiv 1,\ 28 \equiv 2, \ldots, \quad 52 \equiv 0,\ 53 \equiv 1, \ldots$$

et de même $-1 \equiv 25$, $-2 \equiv 24$,...

Plus généralement $\mathbb{Z}/n\mathbb{Z}$ contient n éléments. Pour un entier $a \in \mathbb{Z}$ quelconque, son ***représentant*** dans $\{0,1,2,\ldots,n-1\}$ s'obtient comme le reste k de la division euclidienne de a par n : $a = bn + k$. De sorte que $a \equiv k \pmod{n}$ et $0 \leqslant k < n$.

De façon naturelle l'addition et la multiplication d'entiers se transposent dans $\mathbb{Z}/n\mathbb{Z}$.

Pour $a, b \in \mathbb{Z}/n\mathbb{Z}$, on associe $a + b \in \mathbb{Z}/n\mathbb{Z}$.
Par exemple dans $\mathbb{Z}/26\mathbb{Z}$, $15 + 13$ égale 2. En effet $15 + 13 = 28 \equiv 2 \pmod{26}$. Autre exemple : que vaut $133 + 64$? $133 + 64 = 197 = 7 \times 26 + 15 \equiv 15 \pmod{26}$. Mais on pourrait procéder différemment : tout d'abord $133 = 5 \times 26 + 3 \equiv 3 \pmod{26}$ et $64 = 2 \times 26 + 12 \equiv 12 \pmod{26}$. Et maintenant sans calculs : $133 + 64 \equiv 3 + 12 \equiv 15 \pmod{26}$.

On fait de même pour la multiplication : pour $a, b \in \mathbb{Z}/n\mathbb{Z}$, on associe $a \times b \in \mathbb{Z}/n\mathbb{Z}$.
Par exemple 3×12 donne 10 modulo 26, car $3 \times 12 = 36 = 1 \times 26 + 10 \equiv 10 \pmod{26}$. De même : $3 \times 27 = 81 = 3 \times 26 + 3 \equiv 3 \pmod{26}$. Une autre façon de voir la même opération est d'écrire d'abord $27 = 1 \pmod{26}$ puis $3 \times 27 \equiv 3 \times 1 \equiv 3 \pmod{26}$.

1.4. Chiffrer et déchiffrer

Le chiffrement de César est simplement une addition dans $\mathbb{Z}/26\mathbb{Z}$! Fixons un entier k qui est le décalage (par exemple $k = 3$ dans l'exemple de César ci-dessus) et définissons la ***fonction de chiffrement de César de décalage*** k qui va de l'ensemble $\mathbb{Z}/26\mathbb{Z}$ dans lui-même :

$$C_k : \left\{ \begin{array}{ccc} \mathbb{Z}/26\mathbb{Z} & \longrightarrow & \mathbb{Z}/26\mathbb{Z} \\ x & \longmapsto & x+k \end{array} \right.$$

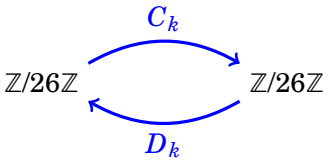
Par exemple, pour $k = 3$: $C_3(0) = 3$, $C_3(1) = 4\ldots$
Pour déchiffrer, rien de plus simple ! Il suffit d'aller dans l'autre sens, c'est-à-dire ici de soustraire. La ***fonction de déchiffrement de César de décalage*** k est

$$D_k : \left\{ \begin{array}{ccc} \mathbb{Z}/26\mathbb{Z} & \longrightarrow & \mathbb{Z}/26\mathbb{Z} \\ x & \longmapsto & x-k \end{array} \right.$$

En effet, si 1 a été chiffré en 4, par la fonction C_3 alors $D_3(4) = 4 - 3 = 1$. On retrouve le nombre original. Mathématiquement, D_k est la bijection réciproque de C_k, ce qui implique que pour tout $x \in \mathbb{Z}/26\mathbb{Z}$:

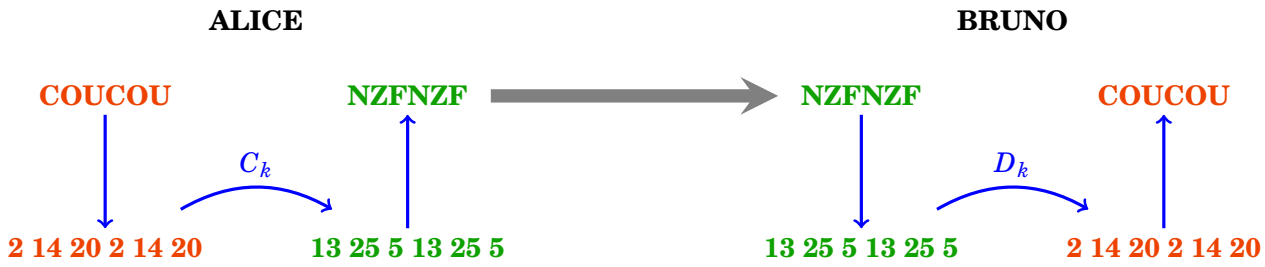
$$D_k\big(C_k(x)\big) = x$$

En d'autres termes, si x est un nombre, on applique la fonction de chiffrement pour obtenir le nombre crypté $y = C_k(x)$; ensuite la fonction de déchiffrement fait bien ce que l'on attend d'elle $D_k(y) = x$, on retrouve le nombre original x.



Une autre façon de voir la fonction de déchiffrement est de remarquer que $D_k(x) = C_{-k}(x)$. Par exemple $C_{-3}(x) = x + (-3) \equiv x + 23 \pmod{26}$.

Voici le principe du chiffrement : Alice veut envoyer des messages secrets à Bruno. Ils se sont d'abord mis d'accord sur une clé secrète k, par exemple $k = 11$. Alice veut envoyer le message "**COUCOU**" à Bruno. Elle transforme "**COUCOU**" en "**2 14 20 2 14 20**". Elle applique la fonction de chiffrement $C_{11}(x) = x + 11$ à chacun des nombres : "**13 25 5 13 25 5**" ce qui correspond au mot crypté "**NZFNZF**". Elle transmet le mot crypté à Bruno, qui selon le même principe applique la fonction de déchiffrement $D_{11}(x) = x - 11$.



Exemple 3

Un exemple classique est le "rot13" (pour rotation par un décalage de 13) :

$$C_{13}(x) = x + 13$$

et comme $-13 \equiv 13 \pmod{26}$ alors $D_{13}(x) = x + 13$. La fonction de déchiffrement est la même que la fonction de chiffrement !
Exemple : déchiffrez le mot "**PRFNE**".

Notons ici deux points importants pour la suite : tout d'abord nous avons naturellement considéré un mot comme une succession de lettres, et chaque opération de chiffrement et déchiffrement s'effectue sur un bloc d'une seule lettre. Ensuite nous avons vu que chiffrer un message est une opération mathématique (certes sur un ensemble un peu spécial).

1.5. Espace des clés et attaque

Combien existe-t-il de possibilités de chiffrement par la méthode de César ? Il y a 26 fonctions C_k différentes, $k = 0, 1, \ldots, 25$. Encore une fois, k appartient à $\mathbb{Z}/26\mathbb{Z}$, car par exemple les fonctions C_{29} et C_3 sont identiques. Le décalage k s'appelle la ***clé de chiffrement***, c'est l'information nécessaire pour crypter le message. Il y a donc 26 clés différentes et l'***espace des clés*** est $\mathbb{Z}/26\mathbb{Z}$.

Il est clair que ce chiffrement de César est d'une sécurité très faible. Si Alice envoie un message secret à Bruno et que Chloé intercepte ce message, il sera facile pour Chloé de le décrypter même si elle ne connaît pas la clé secrète k. L'attaque la plus simple pour Chloé est de tester ce que donne chacune des 26 combinaisons possibles et de reconnaître parmi ces combinaisons laquelle donne un message compréhensible.

1.6. Algorithmes

Les ordinateurs ont révolutionné la cryptographie et surtout le décryptage d'un message intercepté. Nous montrons ici, à l'aide du langage `Python` comment programmer et attaquer le chiffrement de César. Tout d'abord la fonction de chiffrement se programme en une seule ligne :

Algorithme . cesar.py (1)

```
def cesar_chiffre_nb(x,k):
    return (x+k)%26
```

Ici `x` est un nombre de $\{0,1,\ldots,25\}$ et `k` est le décalage. `(x+k)%26` renvoie le reste modulo 26 de la somme `(x+k)`. Pour le décryptage, c'est aussi simple :

Algorithme . cesar.py (2)

```
def cesar_dechiffre_nb(x,k):
    return (x-k)%26
```

Pour chiffrer un mot ou un phrase, il n'y a pas de problèmes théoriques, mais seulement des difficultés techniques :

- Un mot ou une phrase est une chaîne de caractères, qui en fait se comporte comme une liste. Si `mot` est une chaîne alors `mot[0]` est la première lettre, `mot[1]` la deuxième lettre... et la boucle `for lettre in mot:` permet de parcourir chacune des lettres.
- Pour transformer une lettre en un nombre, on utilise le code Ascii qui à chaque caractère associe un nombre, `ord(A)` vaut 65, `ord(B)` vaut 66... Ainsi `(ord(lettre) - 65)` renvoie le rang de la lettre entre 0 et 25 comme nous l'avons fixé dès le départ.
- La transformation inverse se fait par la fonction `char` : `char(65)` renvoie le caractère `A`, `char(66)` renvoie `B`...
- Pour ajouter une lettre à une liste, faites `maliste.append(lettre)`. Enfin pour transformer une liste de caractères en une chaîne, faites `"".join(maliste)`.

Ce qui donne :

Algorithme . cesar.py (3)

```
def cesar_chiffre_mot(mot,k):
    message_code = []                       # Liste vide
    for lettre in mot:                      # Pour chaque lettre
        nb = ord(lettre)-65                 # Lettre devient nb de 0 à 25
        nb_crypte = cesar_chiffre(nb,k)     # Chiffrement de César
        lettre_crypte = chr(nb_crypte+65)   # Retour aux lettres
        message_code.append(lettre_crypte) # Ajoute lettre au message
    message_code = "".join(message_code)    # Revient à chaine caractères
    return(message_code)
```

Pour l'attaque on parcourt l'intégralité de l'espace des clés : k varie de 0 à 25. Noter que pour décrypter les messages on utilise ici simplement la fonction de César avec la clé $-k$.

Algorithme . cesar.py (4)

```
def cesar_attaque(mot):
    for k in range(26):
        print(cesar_chiffre_mot(mot,-k))
    return None
```

2. Le chiffrement de Vigenère

2.1. Chiffrement mono-alphabétique

Principe

Nous avons vu que le chiffrement de César présente une sécurité très faible, la principale raison est que l'espace des clés est trop petit : il y a seulement 26 clés possibles, et on peut attaquer un message chiffré en testant toutes les clés à la main.
Au lieu de faire correspondre circulairement les lettres, on associe maintenant à chaque lettre une autre lettre (sans ordre fixe ou règle générale).
Par exemple :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
F	Q	B	M	X	I	T	E	P	A	L	W	H	S	D	O	Z	K	V	G	R	C	N	Y	J	U

Pour crypter le message

ETRE OU NE PAS ETRE TELLE EST LA QUESTION

on regarde la correspondance et on remplace la lettre **E** par la lettre **X**, puis la lettre **T** par la lettre **G**, puis la lettre **R** par la lettre **K**...
Le message crypté est alors :

XGKX DR SX OFV XGKX GXWWX XVG WF ZRXVGPDS

Pour le décrypter, en connaissant les substitutions, on fait l'opération inverse.

Avantage : nous allons voir que l'espace des clés est gigantesque et qu'il n'est plus question d'énumérer toutes les possibilités.
Inconvénients : la clé à retenir est beaucoup plus longue, puisqu'il faut partager la clé constituée des 26 lettres "FQBMX...". Mais surtout, nous allons voir que finalement ce protocole de chiffrement est assez simple à « craquer ».

Espace des clés

Mathématiquement, le choix d'une clé revient au choix d'une bijection de l'ensemble $\{A,B,\dots,Z\}$ vers le même ensemble $\{A,B,\dots,Z\}$. Il y a 26! choix possibles. En effet pour la lettre A de l'ensemble de départ, il y a 26 choix possibles (nous avions choisi F), pour B il reste 25 choix possibles (tout sauf F qui est déjà choisi), pour C il reste 24 choix... enfin pour Z il ne reste qu'une seule possibilité, la seule lettre non encore choisie. Au final il y a : $26\times25\times24\times\cdots\times2\times1$ soit 26! choix de clés. Ce qui fait environ 4×10^{26} clés. Il y a plus de clés différentes que de grains de sable sur Terre ! Si un ordinateur pouvait tester 1 000 000 de clés par seconde, il lui faudrait alors 12 millions d'années pour tout énumérer.

Attaque statistique

La principale faiblesse du chiffrement mono-alphabétique est qu'une même lettre est toujours chiffrée de la même façon. Par exemple, ici **E** devient **X**. Dans les textes longs, les lettres n'apparaissent pas avec la même fréquence. Ces fréquences varient suivant la langue utilisée. En français, les lettres les plus rencontrées sont dans l'ordre :

E S A I N T R U L O D C P M V Q G F H B X J Y Z K W

avec les fréquences (souvent proches et dépendant de l'échantillon utilisé) :

E	S	A	I	N	T	R	U	L	O	D
14.69%	8.01%	7.54%	7.18%	6.89%	6.88%	6.49%	6.12%	5.63%	5.29%	3.66%

Voici la méthode d'attaque : dans le texte crypté, on cherche la lettre qui apparaît le plus, et si le texte est assez long cela devrait être le chiffrement du **E**, la lettre qui apparaît ensuite dans l'étude des fréquences devrait être le chiffrement du **S**, puis le chiffrement du **A**... On obtient des morceaux de texte clair sous la forme d'une texte à trous et il faut ensuite deviner les lettres manquantes.

Par exemple, déchiffrons la phrase :

LHLZ HFQ BC HFFPZ WH YOUPFH MUPZH

On compte les apparitions des lettres :

H : 6 **F** : 4 **P** : 3 **Z** : 3

On suppose donc que le **H** crypte la lettre **E**, le **F** la lettre **S**, ce qui donne

***E** **ES* ** **ESS** ***E** ****SE** *****E**

D'après les statistiques **P** et **Z** devraient se décrypter en **A** et **I** (ou **I** et **A**). Le quatrième mot "**HFFPZ**", pour l'instant décrypté en "**ESS****", se complète donc en "**ESSAI**" ou "**ESSIA**". La première solution semble correcte ! Ainsi **P** crypte **A**, et **Z** crypte **I**. La phrase est maintenant :

***E*I ES* ** **ESSAI** ***E** ****ASE** ***AIE**

En réfléchissant un petit peu, on décrypte le message :

CECI EST UN ESSAI DE PHRASE VRAIE

2.2. Le chiffrement de Vigenère

Blocs

L'espace des clés du chiffrement mono-alphabétique est immense, mais le fait qu'une lettre soit toujours cryptée de la même façon représente une trop grande faiblesse. Le chiffrement de Vigenère remédie à ce problème. On regroupe les lettres de notre texte par blocs, par exemple ici par blocs de longueur 4 :

CETTE PHRASE NE VEUT RIEN DIRE

devient

CETT EPHR ASEN EVEU TRIE NDIR E

(les espaces sont purement indicatifs, dans la première phrase ils séparent les mots, dans la seconde ils séparent les blocs).

Si k est la longueur d'un bloc, alors on choisit une clé constituée de k nombres de 0 à 25 : $(n_1, n_2, \ldots, n_k)$. Le chiffrement consiste à effectuer un chiffrement de César, dont le décalage dépend du rang de la lettre dans le bloc :

- un décalage de n_1 pour la première lettre de chaque bloc,
- un décalage de n_2 pour la deuxième lettre de chaque bloc,
- ...
- un décalage de n_k pour la k-ème et dernière lettre de chaque bloc.

Pour notre exemple, si on choisit comme clé $(3,1,5,2)$ alors pour le premier bloc "**CETT**" :

- un décalage de 3 pour **C** donne **F**,
- un décalage de 1 pour **E** donne **F**,
- un décalage de 5 pour le premier **T** donne **Y**,
- un décalage de 2 pour le deuxième **T** donne **V**.

Ainsi "**CETT**" de vient "**FFYV**". Vous remarquez que les deux lettres **T** ne sont pas cryptées par la même lettre et que les deux **F** ne cryptent pas la même lettre. On continue ensuite avec le deuxième bloc...

Mathématiques

L'élément de base n'est plus une lettre mais un ***bloc***, c'est-à-dire un regroupement de lettres. La fonction de chiffrement associe à un bloc de longueur k, un autre bloc de longueur k, ce qui donne en mathématisant les choses :

$$C_{n_1,n_2,\dots,n_k} : \begin{cases} \mathbb{Z}/26\mathbb{Z} \times \mathbb{Z}/26\mathbb{Z} \times \cdots \times \mathbb{Z}/26\mathbb{Z} & \longrightarrow & \mathbb{Z}/26\mathbb{Z} \times \mathbb{Z}/26\mathbb{Z} \times \cdots \times \mathbb{Z}/26\mathbb{Z} \\ (x_1, x_2, \dots, x_k) & \longmapsto & (x_1+n_1, x_2+n_2, \dots, x_k+n_k) \end{cases}$$

Chacune des composantes de cette fonction est un chiffrement de César. La fonction de déchiffrement est juste $C_{-n_1,-n_2,\dots,-n_k}$.

Espace des clés et attaque

Il y a 26^k choix possibles de clés, lorsque les blocs sont de longueur k. Pour des blocs de longueur $k = 4$ cela en donne déjà 456 976, et même si un ordinateur teste toutes les combinaisons possibles sans problème, il n'est pas question de parcourir cette liste pour trouver le message en clair, c'est-à-dire celui qui est compréhensible !

Il persiste tout de même une faiblesse du même ordre que celle rencontrée dans le chiffrement mono-alphabétique : la lettre **A** n'est pas toujours cryptée par la même lettre, mais si deux lettres **A** sont situées à la même position dans deux blocs différents (comme par exemple "**ALPH ABET**") alors elles seront cryptées par la même lettre.

Une attaque possible est donc la suivante : on découpe notre message en plusieurs listes, les premières lettres de chaque bloc, les deuxièmes lettres de chaque bloc... et on fait une attaque statistique sur chacun de ces regroupements. Ce type d'attaque n'est possible que si la taille des blocs est petite devant la longueur du texte.

2.3. Algorithmes

Voici un petit algorithme qui calcule la fréquence de chaque lettre d'une phrase.

Algorithme . statistiques.py

```
def statistiques(phrase):
    liste_stat = [0 for x in range(26)]      # Une liste avec des 0
    for lettre in phrase:                    # On parcourt la phrase
        i = ord(lettre)-65
        if 0 <= i < 26:                      # Si c'est une vraie lettre
          liste_stat[i] = liste_stat[i] + 1
    return(liste_stat)
```

Et voici le chiffrement de Vigenère.

Algorithme . vigenere.py

```
def vigenere(mot,cle):                    # Clé est du type [n_1,...,n_k]
    message_code = []
    k = len(cle)                          # Longueur de la clé
    i = 0                                 # Rang dans le bloc
    for lettre in mot:                    # Pour chaque lettre
        nomb = ord(lettre)-65             # Lettre devient nb de 0 à 25
        nomb_code = (nomb+cle[i]) % 26    # Vigenère : on ajoute n_i
        lettre_code = chr(nomb_code+65)   # On repasse aux lettres
        i=(i+1) % k                       # On passe au rang suivant
        message_code.append(lettre_code) # Ajoute lettre au message
    message_code = "".join(message_code) # Revient à chaine caractères
    return(message_code)
```

3. La machine Enigma et les clés secrètes

3.1. Un secret parfait

L'inconvénient des chiffrements précédents est qu'une même lettre est régulièrement chiffrée de la même façon, car la correspondance d'un alphabet à un ou plusieurs autres est fixée une fois pour toutes, ce qui fait qu'une attaque statistique est toujours possible. Nous allons voir qu'en changeant la correspondance à chaque lettre, il est possible de créer un chiffrement parfait !
Expliquons d'abord le principe à l'aide d'une analogie : j'ai choisi deux entiers m et c tels que $m+c=100$. Que vaut m ? C'est bien sûr impossible de répondre car il y a plusieurs possibilités : $0+100$, $1+99$, $2+98$,... Par contre, si je vous donne aussi c alors vous trouvez m immédiatement $m=100-c$.

Voici le principe du chiffrement : Alice veut envoyer à Bruno le message secret M suivant :

ATTAQUE LE CHATEAU

Alice a d'abord choisi une clé secrète C qu'elle a transmise à Bruno. Cette clé secrète est de la même longueur que le message (les espaces ne comptent pas) et composée d'entiers de 0 à 25, tirés au hasard. Par exemple C :

[4, 18, 2, 0, 21, 12, 18, 13, 7, 11, 23, 22, 19, 2, 16, 9]

Elle crypte la première lettre par un décalage de César donné par le premier entier : **A** est décalé de 4 lettres et devient donc **E**. La seconde lettre est décalée du second entier : le premier **T** devient **L**. Le second **T** est lui décalé de 2 lettres, il devient **V**. Le **A** suivant est décalé de 0 lettre, il reste **A**... Alice obtient un message chiffré X qu'elle transmet à Bruno :

ELVALGW YL NEWMGQD

Pour le décrypter, Bruno, qui connaît la clé, n'a qu'à faire le décalage dans l'autre sens.
Notez que deux lettres identiques (par exemples les **T**) n'ont aucune raison d'être cryptées de la même façon. Par exemple, les **T** du message initial sont cryptés dans l'ordre par un **L**, un **V** et un **M**.

Formalisons un peu cette opération. On identifie A avec 0, B avec 1, ..., Z avec 25. Alors le message crypté X est juste la "somme" du message M avec la clé secrète C, la somme s'effectuant lettre à

lettre, terme à terme, modulo 26.
Notons cette opération $M \oplus C = X$.

	A	**T**	**T**	**A**	**Q**	**U**	**E**		**L**	**E**		**C**	**H**	**A**	**T**	**E**	**A**	**U**
	0	**19**	**19**	**0**	**16**	**20**	**4**		**11**	**4**		**2**	**7**	**0**	**19**	**4**	**0**	**20**
$\oplus$																		
	4	**18**	**2**	**0**	**21**	**12**	**18**		**13**	**7**		**11**	**23**	**22**	**19**	**2**	**16**	**9**
=	**4**	**11**	**21**	**0**	**11**	**6**	**22**		**24**	**11**		**13**	**4**	**22**	**12**	**6**	**16**	**3**
	E	**L**	**V**	**A**	**L**	**G**	**W**		**Y**	**L**		**N**	**E**	**W**	**M**	**G**	**Q**	**D**

Bruno reçoit X et connaît C, il effectue donc $X \ominus C = M$.
Pourquoi ce système est-il inviolable ? Pour chacune des lettres, c'est exactement le même problème que trouver m, sachant que $m + c = x$ (où $x = 100$), mais sans connaître c. Toutes les possibilités pour m pourraient être juste. Et bien sûr, dès que l'on connaît c, la solution est triviale : $m = x - c$.
Il y a trois principes à respecter pour que ce système reste inviolable :

1. La longueur de la clé est égale à la longueur du message.
2. La clé est choisie au hasard.
3. La clé ne sert qu'une seule fois.

Ce système appelé "masque jetable" ou chiffrement de Vernam est parfait en théorie, mais sa mise en œuvre n'est pas pratique du tout ! Tout d'abord il faut que la clé soit aussi longue que le message. Pour un message court cela ne pose pas de problème, mais pour envoyer une image par exemple cela devient très lourd. Ensuite, il faut trouver un moyen sûr d'envoyer la clé secrète à son interlocuteur avant de lui faire parvenir le message. Et il faut recommencer cette opération à chaque message, ou bien se mettre d'accord dès le départ sur un ***carnet de clés*** : une longue liste de clés secrètes.
Pour justifier que ce système est vraiment inviolable voici une expérience amusante : Alice veut envoyer le message M ="**ATTAQUE LE CHATEAU**" à Bruno, elle choisit la clé secrète C=[4, 18, 2, 0,...] comme ci-dessus et obtient le message chiffré X="**ELVA**..." qu'elle transmet à Bruno.
Alice se fait kidnapper par Chloé, qui veut l'obliger à déchiffrer son message. Heureusement, Alice a anticipé les soucis : elle a détruit le message M, la clé secrète C et a créé un faux message M' et une fausse clé secrète C'. Alice fournit cette fausse clé secrète C' à Chloé, qui déchiffre le message par l'opération $X \ominus C'$ et elle trouve le message bien inoffensif M' :

RECETTE DE CUISINE

Alice est innocentée !
Comment est-ce possible ? Alice avait au préalable préparé un message neutre M' de même longueur que M et calculé la fausse clé secrète $C' = X \ominus M'$. Chloé a obtenu (par la contrainte) X et C', elle déchiffre le message ainsi

$$X \ominus C' = X \ominus (X \ominus M') = (X \ominus X) \oplus M' = M'$$

Chloé trouve donc le faux message.
Ici la fausse clé C' est :

[13, 7, 19, 22, 18, 13, 18, 21, 7, 11, 10, 14, 20, 24, 3, 25]

La première lettre du message chiffré est un **E**, en reculant de 13 lettres dans l'alphabet, elle se déchiffre en **R**...

3.2. La machine Enigma

Afin de s'approcher de ce protocole de chiffrement parfait, il faut trouver un moyen de générer facilement de longues clés, comme si elles avaient été générées au hasard. Nous allons étudier deux exemples utilisés en pratique à la fin du siècle dernier, une méthode électro-mécanique : la machine Enigma et une méthode numérique : le DES.

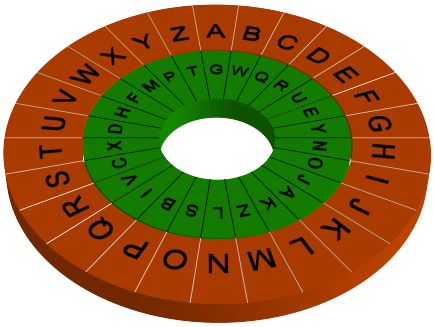
La machine Enigma est une machine électro-mécanique qui ressemble à une machine à écrire. Lorsque qu'une touche est enfoncée, des disques internes sont actionnés et le caractère crypté s'allume. Cette machine, qui sert aussi au déchiffrement, était utilisée pour les communications de l'armée allemande durant la seconde guerre mondiale. Ce que les Allemands ne savaient pas, c'est que les services secrets polonais et britanniques avaient réussi à percer les secrets de cette machine et étaient capables de déchiffrer les messages transmis par les allemands. Ce long travail d'études et de recherches a nécessité tout le génie d'Alan Turing et l'invention de l'ancêtre de l'ordinateur.

Nous symbolisons l'élément de base de la machine Enigma par deux anneaux :

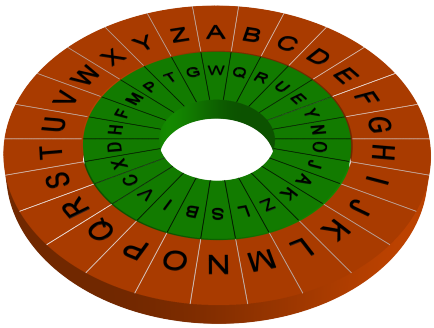
- Un anneau extérieur contenant l'alphabet "**ABCDE**..." symbolisant le clavier de saisie des messages. Cet anneau est fixe.
- Un anneau intérieur contenant un alphabet dans le désordre (sur la figure "**GWQRU**..."). Cet anneau est mobile et effectue une rotation à chaque touche tapée au clavier. Il représente la clé secrète.

Voici, dans ce cas, le processus de chiffrement du mot "**BAC**", avec la clé de chiffrement "**G**" :

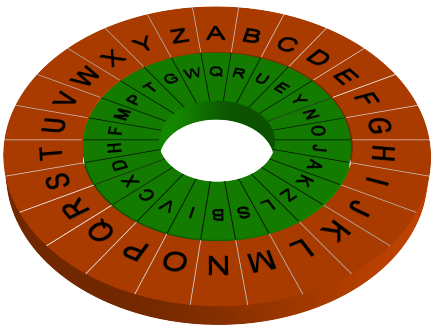
1. **Position initiale.** L'opérateur tourne l'anneau intérieur de sorte que le **A** extérieur et fixe soit en face du **G** intérieur (et donc **B** en face de **W**).



2. **Première lettre.** L'opérateur tape la première lettre du message : **B**, la machine affiche la correspondance **W**.
3. **Rotation.** L'anneau intérieur tourne de 1/26ème de tour, maintenant le **A** extérieur et fixe est en face du **W**, le **B** en face du **Q**,...



4. **Deuxième lettre.** L'opérateur tape la deuxième lettre du message **A**, la machine affiche la correspondance, c'est de nouveau **W**.
5. **Rotation.** L'anneau intérieur tourne de 1/26ème de tour, maintenant le **A** extérieur et fixe est en face du **Q**, le **B** en face du **R**, le **C** en face du **U**,...

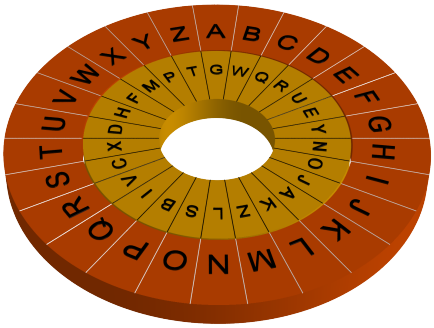


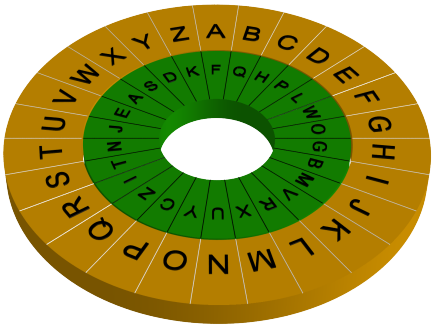
6. **Troisième lettre.** L'opérateur tape la troisième lettre du message **C**, la machine affiche la correspondance **U**.
7. **Rotation.** L'anneau intérieur effectue sa rotation.
8. **Message chiffré.** Le message crypté est donc "**WWU**"

Cette méthode de chiffrement est identique à un chiffrement de type Vigenère pour une clé de longueur 26. Il y a 26 clés différents à disposition avec un seul anneau intérieur et identifiées par lettre de la position initiale : **G**, **W**, **Q**... **T** correspondant aux alphabets : "**GWQ...PT**", "**WQR...TG**", "**QRU...GW**"...

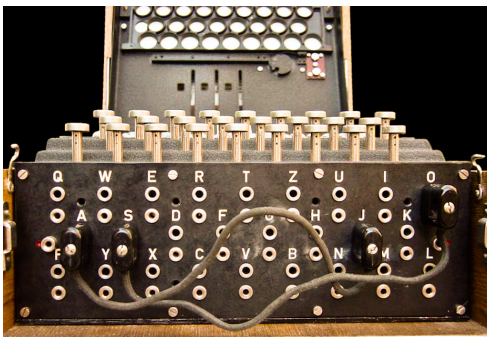
En fait, la machine Enigma était beaucoup plus sophistiquée, il n'y avait pas un mais plusieurs anneaux intérieurs. Par exemple pour deux anneaux intérieurs comme sur la figure : **B** s'envoie sur **W**, qui s'envoie sur **A** ; la lettre **B** est cryptée en **A**. Ensuite l'anneau intérieur numéro 1 effectue 1/26ème de tour. La lettre **A** s'envoie sur **W**, qui s'envoie sur **A** ; la lettre **A** est cryptée en **A**. Lorsque

l'anneau intérieur numéro 1 a fait une rotation complète (26 lettres ont été tapées) alors l'anneau intérieur numéro 2 effectue 1/26ème de tour. C'est comme sur un compteur kilométrique, lorsque le chiffre des kilomètres parcourt $0, 1, 2, 3, \ldots, 9$, alors au kilomètre suivant, le chiffre des kilomètres est 0 et celui des dizaines de kilomètres est augmenté d'une unité.





S'il y a trois anneaux, lorsque l'anneau intérieur 2 a fait une rotation complète, l'anneau intérieur 3 tourne de 1/26ème de tour. Il y a alors 26^3 clés différentes facilement identifiables par les trois lettres des positions initiales des anneaux.
Il fallait donc pour utiliser cette machine, d'abord choisir les disques (nos anneaux intérieurs) les placer dans un certain ordre, fixer la position initiale de chaque disque. Ce système était rendu largement plus complexe avec l'ajout de correspondances par fichage entre les lettres du clavier (voir photo). Le nombre de clés possibles dépassait plusieurs milliards de milliards !



3.3. La ronde des chiffres : DES

La machine Enigma génère mécaniquement un alphabet différent à chaque caractère crypté, tentant de se rapprocher d'un chiffrement parfait. Nous allons voir une autre méthode, cette fois numérique : le DES. Le DES (*Data Encryption Standard*) est un protocole de chiffrement par blocs. Il a été, entre 1977 et 2001, le standard de chiffrement pour les organisations du gouvernement des États-Unis et par extension pour un grand nombre de pays dans le monde.

Commençons par rappeler que l'objectif est de générer une clé aléatoire de grande longueur. Pour ne pas avoir à retenir l'intégralité de cette longue clé, on va la générer de façon pseudo-aléatoire à partir d'une petite clé.
Voyons un exemple élémentaire de suite pseudo-aléatoire.
Soit (u_n) la suite définie par la donnée de (a, b) et de u_0 et la relation de récurrence

$$u_{n+1} \equiv a \times u_n + b \pmod{26}.$$

Par exemple pour $a = 2$, $b = 5$ et $u_0 = 6$, alors les premiers termes de la suites sont :

$$6 \quad 17 \quad 13 \quad 5 \quad 15 \quad 9 \quad 23 \quad 25 \quad 3 \quad 11 \quad 1 \quad 7 \quad 19 \quad 17 \quad 13 \quad 5$$

Les trois nombres (a, b, u_0) représentent la clé principale et la suite des $(u_n)_{n\in\mathbb{N}}$ les clés secondaires. Avantages : à partir d'une clé principale on a généré une longue liste de clés secondaires. Inconvénients : la liste n'est pas si aléatoire que cela, elle se répète ici avec une période de longueur 12 : $17, 13, 5, ..., 17, 13, 5, ...$

Le système DES est une version sophistiquée de ce processus : à partir d'une clé courte et d'opérations élémentaires on crypte un message. Comme lors de l'étude de la machine Enigma, nous allons présenter une version très simplifiée de ce protocole afin d'en expliquer les étapes élémentaires.
Pour changer, nous allons travailler modulo 10. Lorsque l'on travaille par blocs, les additions se font *bit* par *bit*. Par exemple : [1 2 3 4] $\oplus$ [7 8 9 0] = [8 0 2 4] car ($1+7 \equiv 8 \pmod{10}$, $2+8 \equiv 0 \pmod{10}$,...)
Notre message est coupé en blocs, pour nos explications ce seront des blocs de longueur 8. La clé est de longueur 4.
Voici le message (un seul bloc) : M = [1 2 3 4 5 6 7 8] et voici la clé : C = [3 1 3 2].

Étape 0. Initialisation. On note $M_0 = M$ et on découpe M en une partie gauche et une partie droite

$$M_0 = [G_0 \parallel D_0] = [1\ 2\ 3\ 4 \parallel 5\ 6\ 7\ 8]$$

Étape 1. Premier tour. On pose

$$M_1 = [D_0 \parallel C \oplus \sigma(G_0)]$$

où σ est une permutation circulaire.
On effectue donc trois opérations pour passer de M_0 à M_1 :

1. On échange la partie droite et la partie gauche de M_0 :

$$M_0 \longmapsto [5\ 6\ 7\ 8 \parallel 1\ 2\ 3\ 4]$$

2. Sur la nouvelle partie droite, on permute circulairement les nombres :

$$\longmapsto [5\ 6\ 7\ 8 \parallel 2\ 3\ 4\ 1]$$

3. Puis on ajoute la clé secrète C à droite (ici C = [3 1 3 2]) :

$$\longmapsto [5\ 6\ 7\ 8 \parallel 5\ 4\ 7\ 3] = M_1$$

On va recommencer le même processus. Cela revient à appliquer la formule de récurrence, qui partant de $M_i = [G_i \parallel D_i]$, définit

$$M_{i+1} = [D_i \parallel C \oplus \sigma(G_i)]$$

Étape 2. Deuxième tour. On part de M_1 = [5 6 7 8 ‖ 5 4 7 3].

1. On échange la partie droite et la partie gauche de M_0 :

$$M_0 \longmapsto [5\ 4\ 7\ 3 \parallel 5\ 6\ 7\ 8]$$

2. Sur la nouvelle partie droite, on permute circulairement les nombres.

$$\longmapsto [5\ 4\ 7\ 3 \parallel 6\ 7\ 8\ 5]$$

3. Puis on ajoute la clé secrète C à droite.

$$\longmapsto [5\ 4\ 7\ 3 \parallel 9\ 8\ 1\ 7] = M_2$$

On peut décider de s'arrêter après ce tour et renvoyer le message crypté $X = M_2 = [5\ 4\ 7\ 3\ 9\ 8\ 1\ 7]$. Comme chaque opération élémentaire est inversible, on applique un protocole inverse pour déchiffrer.
Dans le vrai protocole du DES, la clé principale est de taille 64 *bits*, il y a plus de manipulations sur le message et les étapes mentionnées ci-dessus sont effectuées 16 fois (on parle de tours). À chaque tour, une clé différente est utilisée. Il existe donc un préambule à ce protocole : générer 16 clés secondaires (de longueur 48 *bits*) à partir de la clé principale, ce qui se fait selon le principe de la suite pseudo-aléatoire (u_n) expliquée plus haut.

4. La cryptographie à clé publique

Les Grecs pour envoyer des messages secrets rasaient la tête du messager, tatouaient le message sur son crâne et attendaient que les cheveux repoussent avant d'envoyer le messager effectuer sa mission !
Il est clair que ce principe repose uniquement sur le secret de la méthode.

4.1. Le principe de Kerckhoffs

Cette méthode rudimentaire va à l'encontre du principe de Kerckhoffs. Le principe de Kerckhoffs s'énonce ainsi :

«La sécurité d'un système de chiffrement ne doit reposer que sur la clé.»

Cela se résume aussi par :

«L'ennemi peut avoir connaissance du système de chiffrement.»

Voici le texte original d'Auguste Kerckhoffs de 1883 «La cryptographie militaire» paru dans le *Journal des sciences militaires*.
Il traite notamment des enjeux de sécurité lors des correspondances :

> *«Il faut distinguer entre un système d'écriture chiffré, imaginé pour un échange momentané de lettres entre quelques personnes isolées, et une méthode de cryptographie destinée à régler pour un temps illimité la correspondance des différents chefs d'armée entre eux.»*

Le principe fondamental est le suivant :

> *«Dans le second cas, [...] il faut que* **le système n'exige pas le secret**, *et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi.»*

Ce principe est novateur dans la mesure où intuitivement il semble opportun de dissimuler le maximum de choses possibles : clé et système de chiffrement utilisés. Mais l'objectif visé par Kerckhoffs est plus académique, il pense qu'un système dépendant d'un secret mais dont le mécanisme est connu de tous sera testé, attaqué, étudié, et finalement utilisé s'il s'avère intéressant et robuste.

4.2. Factorisations des entiers

Quels outils mathématiques répondent au principe de Kerckoffs ?
Un premier exemple est la toute simple multiplication ! En effet si je vous demande combien font 5×7, vous répondez 35. Si je vous demande de factoriser 35 vous répondez 5×7. Cependant ces

deux questions ne sont pas du même ordre de difficulté. Si je vous demande de factoriser 1591, vous aller devoir faire plusieurs tentatives, alors que si je vous avais directement demandé de calculer 37×43 cela ne pose pas de problème.
Pour des entiers de plusieurs centaines de chiffres le problème de factorisation ne peut être résolu en un temps raisonnable, même pour un ordinateur. C'est ce problème asymétrique qui est à la base de la cryptographie RSA (que nous détaillerons plus tard) : connaître p et q apporte plus d'information utilisable que $p \times q$. Même si en théorie à partir de $p \times q$ on peut retrouver p et q, en pratique ce sera impossible.

Formalisons ceci avec la notion de complexité. La ***complexité*** est le temps de calculs (ou le nombre d'opérations élémentaires) nécessaire pour effectuer une opération.
Commençons par la complexité de l'addition : disons que calculer la somme de deux chiffres (par exemple $6+8$) soit de complexité 1 (par exemple 1 seconde pour un humain, 1 milliseconde pour un ordinateur). Pour calculer la somme de deux entiers à n chiffres, la complexité est d'ordre n (exemple : $1234+2323$, il faut faire 4 additions de chiffres, donc environ 4 secondes pour un humain).
La multiplication de deux entiers à n chiffres est de complexité d'ordre n^2. Par exemple pour multiplier 1234 par 2323 il faut faire 16 multiplications de chiffres (chaque chiffre de 1234 est à multiplier par chaque chiffre de 2323).
Par contre la meilleure méthode de factorisation connue est de complexité d'ordre $\exp(4n^{\frac{1}{3}})$ (c'est moins que $\exp(n)$, mais plus que n^d pour tout d, lorsque n tend vers $+\infty$).
Voici un tableau pour avoir une idée de la difficulté croissante pour multiplier et factoriser des nombres à n chiffres :

n	multiplication	factorisation
3	9	320
4	16	572
5	25	934
10	100	5 528
50	2 500	2 510 835
100	10 000	115 681 968
200	40 000	14 423 748 780

4.3. Fonctions à sens unique

Il existe bien d'autres situations mathématiques asymétriques : les ***fonctions à sens unique***. En d'autres termes, étant donnée une fonction f, il est possible connaissant x de calculer «facilement» $f(x)$; mais connaissant un élément de l'ensemble image de f, il est «difficile» ou impossible de trouver son antécédent.
Dans le cadre de la cryptographie, posséder une fonction à sens unique qui joue le rôle de chiffrement n'a que peu de sens. En effet, il est indispensable de trouver un moyen efficace afin de pouvoir déchiffrer les messages chiffrés. On parle alors de ***fonction à sens unique avec trappe secrète***.

Prenons par exemple le cas de la fonction f suivante :

$$f : x \longmapsto x^3 \pmod{100}.$$

- Connaissant x, trouver $y = f(x)$ est facile, cela nécessite deux multiplications et deux divisions.

– Connaissant y image par f d'un élément x ($y = f(x)$), retrouver x est difficile.

Tentons de résoudre le problème suivant : trouver x tel que $x^3 \equiv 11 \pmod{100}$.
On peut pour cela :

– soit faire une recherche exhaustive, c'est-à-dire essayer successivement 1, 2, 3, ..., 99, on trouve alors :

$$71^3 = 357\,911 \equiv 11 \pmod{100},$$

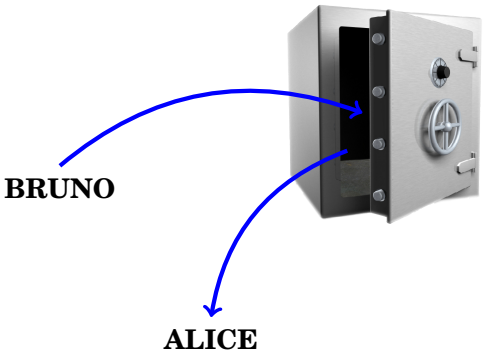
– soit utiliser la trappe secrète : $y \longmapsto y^7 \pmod{100}$ qui fournit directement le résultat !

$$11^7 = 19\,487\,171 \equiv 71 \pmod{100}.$$

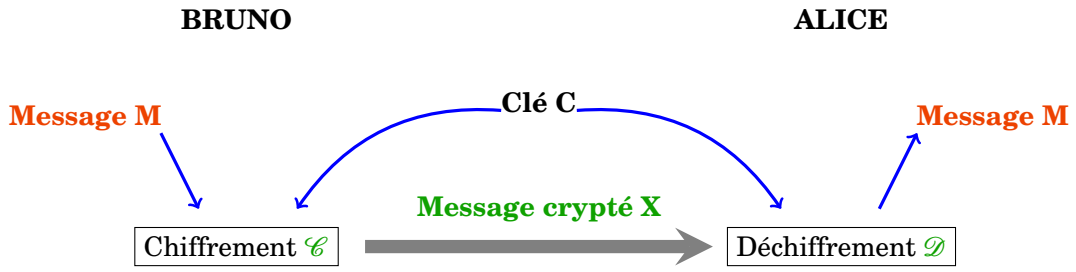
La morale est la suivante : le problème est dur à résoudre, sauf pour ceux qui connaissent la trappe secrète. (Attention, dans le cas de cet exemple, la fonction f n'est pas bijective.)

4.4. Chiffrement à clé privée

Petit retour en arrière. Les protocoles étudiés dans les chapitres précédents étaient des ***chiffrements à clé privée***. De façon imagée, tout se passe comme si Bruno pouvaient déposer son message dans un coffre fort pour Alice, Alice et Bruno étant les seuls à posséder la clé du coffre.

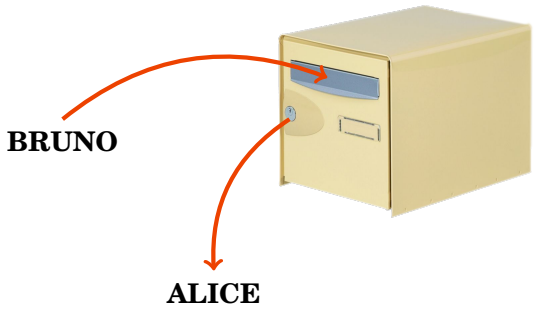


En effet, jusqu'ici, les deux interlocuteurs se partageaient une même clé qui servait à chiffrer (et déchiffrer) les messages. Cela pose bien sûr un problème majeur : Alice et Bruno doivent d'abord se communiquer la clé.

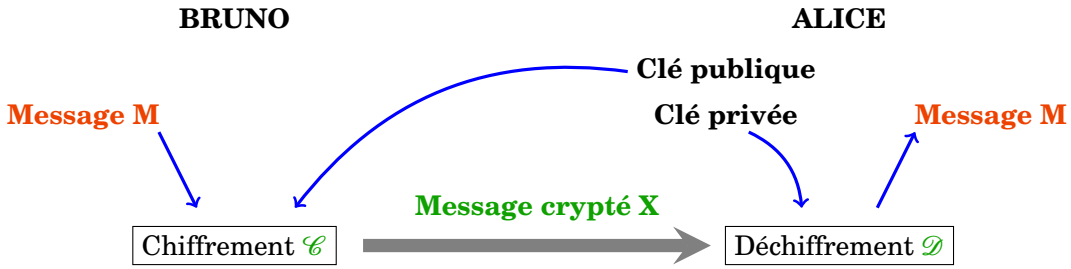


4.5. Chiffrement à clé publique

Les fonctions à sens unique à trappe donnent naissance à des protocoles de chiffrement à clé publique. L'association «clé» et «publique» peut paraître incongrue, mais il signifie que le principe de chiffrement est accessible à tous mais que le déchiffrement nécessite une clé qu'il faut bien sûr garder secrète.



De façon imagée, si Bruno veut envoyer un message à Alice, il dépose son message dans la boîte aux lettres d'Alice, seule Alice pourra ouvrir sa boîte et consulter le message. Ici la clé publique est symbolisée par la boîte aux lettre, tout le monde peut y déposer un message, la clé qui ouvre la boîte aux lettres est la clé privée d'Alice, que Alice doit conserver à l'abri.



En prenant appui sur l'exemple précédent, si le message initial est 71 et que la fonction f de chiffrement est connue de tous, le message transmis est 11 et le déchiffrement sera rapide si la trappe secrète 7 est connue du destinataire.
Les paramètres d'un protocole de ***chiffrement à clé publique*** sont donc :

- les fonctions de chiffrement et de déchiffrement : $\mathscr{C}$ et $\mathscr{D}$,
- la clé publique du destinataire qui va permettre de paramétrer la fonction $\mathscr{C}$,
- la clé privée du destinataire qui va permettre de paramétrer la fonction $\mathscr{D}$.

Dans le cadre de notre exemple Bruno souhaite envoyer un message à Alice, ces éléments sont :

- $\mathscr{C} : x \longmapsto x^? \pmod{100}$ et $\mathscr{D} : x \longmapsto x^? \pmod{100}$,
- **3** : la clé publique d'Alice qui permet de définir complètement la fonction de chiffrement :

$$\mathscr{C} : x \longmapsto x^3 \pmod{100},$$

- **7** : la clé privée d'Alice qui permet de définir complètement la fonction de déchiffrement :

$$\mathscr{D} : x \longmapsto x^7 \pmod{100}.$$

Dans la pratique, un chiffrement à clé publique nécessite plus de calculs et est donc assez lent, plus lent qu'un chiffrement à clé privée. Afin de gagner en rapidité, un protocole hybride peut être mis en place de la façon suivante :

- à l'aide d'un protocole de chiffrement à clé publique, Alice et Bruno échangent une clé,
- Alice et Bruno utilise cette clé dans un protocole de chiffrement à clé privée.

5. L'arithmétique pour RSA

Pour un entier n, sachant qu'il est le produit de deux nombres premiers, il est difficile de retrouver les facteurs p et q tels que $n = pq$. Le principe du chiffrement RSA, chiffrement à clé publique, repose sur cette difficulté.
Dans cette partie nous mettons en place les outils mathématiques nécessaires pour le calcul des clés publique et privée ainsi que les procédés de chiffrement et déchiffrement RSA.

5.1. Le petit théorème de Fermat amélioré

Nous connaissons le petit théorème de Fermat

Théorème 2. Petit théorème de Fermat

Si p est un nombre premier et $a \in \mathbb{Z}$ alors

$$a^p \equiv a \pmod{p}$$

et sa variante :

Corollaire 1

Si p ne divise pas a alors

$$a^{p-1} \equiv 1 \pmod{p}$$

Nous allons voir une version améliorée de ce théorème dans le cas qui nous intéresse :

Théorème 3. Petit théorème de Fermat amélioré

Soient p et q deux nombres premiers distincts et soit $n = pq$. Pour tout $a \in \mathbb{Z}$ tel que $\text{pgcd}(a,n) = 1$ alors :

$$a^{(p-1)(q-1)} \equiv 1 \pmod{n}$$

On note $\varphi(n) = (p-1)(q-1)$, la ***fonction d'Euler***. L'hypothèse $\text{pgcd}(a,n) = 1$ équivaut ici à ce que a ne soit divisible ni par p, ni par q. Par exemple pour $p = 5$, $q = 7$, $n = 35$ et $\varphi(n) = 4 \cdot 6 = 24$. Alors pour $a = 1,2,3,4,6,8,9,11,12,13,16,17,18,\ldots$ on a bien $a^{24} \equiv 1 \pmod{35}$.

Démonstration

Notons $c = a^{(p-1)(q-1)}$. Calculons c modulo p :

$$c \equiv a^{(p-1)(q-1)} \equiv (a^{(p-1)})^{q-1} \equiv 1^{q-1} \equiv 1 \pmod{p}$$

où l'on appliquer le petit théorème de Fermat : $a^{p-1} \equiv 1 \pmod{p}$, car p ne divise pas a.
Calculons ce même c mais cette fois modulo q :

$$c \equiv a^{(p-1)(q-1)} \equiv (a^{(q-1)})^{p-1} \equiv 1^{p-1} \equiv 1 \pmod{q}$$

où l'on appliquer le petit théorème de Fermat : $a^{q-1} \equiv 1 \pmod{q}$, car q ne divise pas a.
Conclusion partielle : $c \equiv 1 \pmod{p}$ et $c \equiv 1 \pmod{q}$.

Nous allons en déduire que $c \equiv 1 \pmod{pq}$.
Comme $c \equiv 1 \pmod{p}$ alors il existe $\alpha \in \mathbb{Z}$ tel que $c = 1 + \alpha p$; comme $c \equiv 1 \pmod{q}$ alors il existe $\beta \in \mathbb{Z}$ tel que $c = 1 + \beta q$. Donc $c - 1 = \alpha p = \beta q$. De l'égalité $\alpha p = \beta q$, on tire que $p | \beta q$.
Comme p et q sont premiers entre eux (car ce sont des nombres premiers distincts) alors par le lemme de Gauss on en déduit que $p | \beta$. Il existe donc $\beta' \in \mathbb{Z}$ tel que $\beta = \beta' p$.
Ainsi $c = 1 + \beta q = 1 + \beta' pq$. Ce qui fait que $c \equiv 1 \pmod{pq}$, c'est exactement dire $a^{(p-1)(q-1)} \equiv 1 \pmod{n}$.

5.2. L'algorithme d'Euclide étendu

Nous avons déjà étudié l'algorithme d'Euclide qui repose sur le principe que $\text{pgcd}(a,b) = \text{pgcd}(b, a \pmod b)$.
Voici sa mise en œuvre informatique.

Algorithme . euclide.py (1)

```
def euclide(a,b):
    while b !=0 :
        a , b = b , a % b
    return a
```

On profite que `Python` assure les affectations simultanées, ce qui pour nous correspond aux suites

$$\begin{cases} a_{i+1} & = & b_i \\ b_{i+1} & \equiv & a_i \pmod{b_i} \end{cases}$$

initialisée par $a_0 = a$, $b_0 = b$.
Nous avons vu aussi comment « remonter » l'algorithme d'Euclide à la main pour obtenir les coefficients de Bézout u, v tels que $au + bv = \text{pgcd}(a,b)$. Cependant il nous faut une méthode plus automatique pour obtenir ces coefficients, c'est l'***algorithme d'Euclide étendu***.
On définit deux suites (x_i), (y_i) qui vont aboutir aux coefficients de Bézout.
L'initialisation est :

$$x_0 = 1 \qquad x_1 = 0 \qquad y_0 = 0 \qquad y_1 = 1$$

et la formule de récurrence pour $i \geqslant 1$:

$$x_{i+1} = x_{i-1} - q_i x_i \qquad y_{i+1} = y_{i-1} - q_i y_i$$

où q_i est le quotient de la division euclidienne de a_i par b_i.

Algorithme . euclide.py (2)

```
def euclide_etendu(a,b):
    x = 1 ; xx = 0
    y = 0 ; yy = 1
    while b != 0 :
        q = a // b
        a , b = b , a % b
        xx , x = x - q*xx , xx
        yy , y = y - q*yy , yy
    return (a,x,y)
```

Cet algorithme renvoie d'abord le pgcd, puis les coefficients u, v tels que $au + bv = \text{pgcd}(a,b)$.

5.3. Inverse modulo n

Soit $a \in \mathbb{Z}$, on dit que $x \in \mathbb{Z}$ est un ***inverse de a modulo n*** si $\boxed{ax \equiv 1 \pmod n.}$
Trouver un inverse de a modulo n est donc un cas particulier de l'équation $ax \equiv b \pmod n$.

Proposition 5

- a admet un inverse modulo n si et seulement si a et n sont premiers entre eux.
- Si $au + nv = 1$ alors u est un inverse de a modulo n.

En d'autres termes, trouver un inverse de a modulo n revient à calculer les coefficients de Bézout associés à la paire (a,n).

Démonstration

La preuve est essentiellement une reformulation du théorème de Bézout :

$$\begin{aligned} \text{pgcd}(a,n) = 1 &\iff \exists u,v \in \mathbb{Z} \quad au + nv = 1 \\ &\iff \exists u \in \mathbb{Z} \quad au \equiv 1 \pmod{n} \end{aligned}$$

Voici le code :

Algorithme . euclide.py (3)

```
def inverse(a,n):
    c,u,v = euclide_etendu(a,n)    # pgcd et coeff. de Bézout
    if c != 1 :             # Si pgcd différent de 1 renvoie 0
        return 0
    else :
        return u % n                        # Renvoie l'inverse
```

5.4. L'exponentiation rapide

Nous aurons besoin de calculer rapidement des puissances modulo n. Pour cela il existe une méthode beaucoup plus efficace que de calculer d'abord a^k puis de le réduire modulo n. Il faut garder à l'esprit que les entiers que l'on va manipuler ont des dizaines voir des centaines de chiffres.
Voyons la technique sur l'exemple de $5^{11} \pmod{14}$. L'idée est de seulement calculer $5, 5^2, 5^4, 5^8 \ldots$ et de réduire modulo n à chaque fois. Pour cela on remarque que $11 = 8 + 2 + 1$ donc

$$5^{11} = 5^8 \times 5^2 \times 5^1.$$

Calculons donc les $5^{2^i} \pmod{14}$:

$$\begin{aligned} 5 &\equiv 5 \pmod{14} \\ 5^2 &\equiv 25 \equiv 11 \pmod{14} \\ 5^4 &\equiv 5^2 \times 5^2 \equiv 11 \times 11 \equiv 121 \equiv 9 \pmod{14} \\ 5^8 &\equiv 5^4 \times 5^4 \equiv 9 \times 9 \equiv 81 \equiv 11 \pmod{14} \end{aligned}$$

à chaque étape est effectuée une multiplication modulaire. Conséquence :

$$5^{11} \equiv 5^8 \times 5^2 \times 5^1 \equiv 11 \times 11 \times 5 \equiv 11 \times 55 \equiv 11 \times 13 \equiv 143 \equiv 3 \pmod{14}.$$

Nous obtenons donc un calcul de $5^{11} \pmod{14}$ en 5 opérations au lieu de 10 si on avait fait $5 \times 5 \times 5 \cdots$.

Voici une formulation générale de la méthode. On écrit le développement de l'exposant k en base 2 : $(k_\ell,\dots,k_2,k_1,k_0)$ avec $k_i \in \{0,1\}$ de sorte que

$$k = \sum_{i=0}^{\ell} k_i 2^i.$$

On obtient alors

$$x^k = x^{\sum_{i=0}^{\ell} k_i 2^i} = \prod_{i=0}^{\ell} (x^{2^i})^{k_i}.$$

Par exemple 11 en base 2 s'écrit (1,0,1,1), donc, comme on l'a vu :

$$5^{11} = (5^{2^3})^1 \times (5^{2^2})^0 \times (5^{2^1})^1 \times (5^{2^0})^1.$$

Voici un autre exemple : calculons $17^{154} \pmod{100}$. Tout d'abord on décompose l'exposant $k = 154$ en base 2 : $154 = 128+16+8+2 = 2^7+2^4+2^3+2^1$, il s'écrit donc en base 2 : (1,0,0,1,1,0,1,0). Ensuite on calcule $17, 17^2, 17^4, 17^8, \dots, 17^{128}$ modulo 100.

$$\begin{aligned}
&17 \equiv 17 \pmod{100}\\
&17^2 \equiv 17\times 17 \equiv 289 \equiv 89 \pmod{100}\\
&17^4 \equiv 17^2\times 17^2 \equiv 89\times 89 \equiv 7921 \equiv 21 \pmod{100}\\
&17^8 \equiv 17^4\times 17^4 \equiv 21\times 21 \equiv 441 \equiv 41 \pmod{100}\\
&17^{16} \equiv 17^8\times 17^8 \equiv 41\times 41 \equiv 1681 \equiv 81 \pmod{100}\\
&17^{32} \equiv 17^{16}\times 17^{16} \equiv 81\times 81 \equiv 6561 \equiv 61 \pmod{100}\\
&17^{64} \equiv 17^{32}\times 17^{32} \equiv 61\times 61 \equiv 3721 \equiv 21 \pmod{100}\\
&17^{128} \equiv 17^{64}\times 17^{64} \equiv 21\times 21 \equiv 441 \equiv 41 \pmod{100}
\end{aligned}$$

Il ne reste qu'à rassembler :

$$17^{154} \equiv 17^{128}\times 17^{16}\times 17^8\times 17^2 \equiv 41\times 81\times 41\times 89 \equiv 3321\times 3649 \equiv 21\times 49 \equiv 1029 \equiv 29 \pmod{100}$$

On en déduit un algorithme pour le calcul rapide des puissances.

Algorithme . puissance.py

```
def puissance(x,k,n):
    puiss = 1                       # Résultat
    while (k>0):
        if k % 2 != 0 :             # Si k est impair (i.e. k_i=1)
            puiss = (puiss*x) % n
        x = x*x % n                 # Vaut x, x^2, x^4,...
        k = k // 2
    return(puiss)
```

En fait Python sait faire l'exponentiation rapide : `pow(x,k,n)` pour le calcul de a^k modulo n, il faut donc éviter `(x ** k) % n` qui n'est pas adapté.

6. Le chiffrement RSA

Voici le but ultime de ce cours : la chiffrement RSA. Il est temps de relire l'introduction du chapitre « Arithmétique » pour s'apercevoir que nous sommes prêts !

Pour crypter un message on commence par le transformer en un –ou plusieurs– nombres. Les processus de chiffrement et déchiffrement font appel à plusieurs notions :

- *On choisit deux* **nombres premiers** *p et q que l'on garde secrets et on pose $n = p \times q$. Le principe étant que même connaissant n il est très difficile de retrouver p et q (qui sont des nombres ayant des centaines de chiffres).*
- *La clé secrète et la clé publique se calculent à l'aide de l'***algorithme d'Euclide** *et des* **coefficients de Bézout**.
- *Les calculs de cryptage se feront* **modulo** *n.*
- *Le déchiffrement fonctionne grâce à une variante du* **petit théorème de Fermat**.

Dans cette section, c'est Bruno qui veut envoyer un message secret à Alice. La processus se décompose ainsi :

1. Alice prépare une clé publique et une clé privée,
2. Bruno utilise la clé publique d'Alice pour crypter son message,
3. Alice reçoit le message crypté et le déchiffre grâce à sa clé privée.

6.1. Calcul de la clé publique et de la clé privée

Choix de deux nombres premiers

Alice effectue, une fois pour toute, les opérations suivantes (en secret) :

- elle choisit deux nombres premiers distincts p et q (dans la pratique ce sont de très grand nombres, jusqu'à des centaines de chiffres),
- Elle calcule $n = p \times q$,
- Elle calcule $\varphi(n) = (p-1) \times (q-1)$.

Exemple 1.

- $p = 5$ et $q = 17$
- $n = p \times q = 85$
- $\varphi(n) = (p-1) \times (q-1) = 64$

Vous noterez que le calcul de $\varphi(n)$ n'est possible que si la décomposition de n sous la forme $p \times q$ est connue. D'où le caractère secret de $\varphi(n)$ même si n est connu de tous.

Exemple 2.

- $p = 101$ et $q = 103$
- $n = p \times q = 10\,403$
- $\varphi(n) = (p-1) \times (q-1) = 10\,200$

Choix d'un exposant et calcul de son inverse

Alice continue :

- elle choisit un exposant e tel que $\text{pgcd}(e, \varphi(n)) = 1$,
- elle calcule l'inverse d de e module $\varphi(n)$: $d \times e \equiv 1 \pmod{\varphi(n)}$. Ce calcul se fait par l'algorithme d'Euclide étendu.

Exemple 1.

- Alice choisit par exemple $e = 5$ et on a bien $\text{pgcd}(e, \varphi(n)) = \text{pgcd}(5, 64) = 1$,

– Alice applique l'algorithme d'Euclide étendu pour calculer les coefficients de Bézout correspondant à $\text{pgcd}(e, \varphi(n)) = 1$. Elle trouve $5 \times 13 + 64 \times (-1) = 1$. Donc $5 \times 13 \equiv 1 \pmod{64}$ et l'inverse de e modulo $\varphi(n)$ est $d = 13$.

Exemple 2.

– Alice choisit par exemple $e = 7$ et on a bien $\text{pgcd}(e, \varphi(n)) = \text{pgcd}(7, 10\,200) = 1$,
– L'algorithme d'Euclide étendu pour $\text{pgcd}(e, \varphi(n)) = 1$ donne $7 \times (-1457) + 10\,200 \times 1 = 1$. Mais $-1457 \equiv 8743 \pmod{\varphi(n)}$, donc pour $d = 8743$ on a $d \times e \equiv 1 \pmod{\varphi(n)}$.

Clé publique

La ***clé publique*** d'Alice est constituée des deux nombres :

$$\boxed{n \text{ et } e}$$

Et comme son nom l'indique Alice communique sa clé publique au monde entier.

Exemple 1. $n = 85$ et $e = 5$

Exemple 2. $n = 10\,403$ et $e = 7$

Clé privée

Alice garde pour elle sa ***clé privée*** :

$$\boxed{d}$$

Alice détruit en secret p, q et $\varphi(n)$ qui ne sont plus utiles. Elle conserve secrètement sa clé privée.

Exemple 1. $d = 13$

Exemple 2. $d = 8743$

6.2. Chiffrement du message

Bruno veut envoyer un message secret à Alice. Il se débrouille pour que son message soit un entier (quitte à découper son texte en bloc et à transformer chaque bloc en un entier).

Message

Le message est un entier m, tel que $0 \leqslant m < n$.

Exemple 1. Bruno veut envoyer le message $m = 10$.

Exemple 2. Bruno veut envoyer le message $m = 1234$.

Message chiffré

Bruno récupère la clé publique d'Alice : n et e avec laquelle il calcule, à l'aide de l'algorithme d'exponentiation rapide, le message chiffré :

$$x \equiv m^e \pmod{n}$$

Il transmet ce message x à Alice

Exemple 1. $m = 10$, $n = 85$ et $e = 5$ donc

$$x \equiv m^e \pmod{n} \equiv 10^5 \pmod{85}$$

On peut ici faire les calculs à la main :

$$\begin{aligned} 10^2 &\equiv 100 \equiv 15 \pmod{85} \\ 10^4 &\equiv (10^2)^2 \equiv 15^2 \equiv 225 \equiv 55 \pmod{85} \\ x &\equiv 10^5 \equiv 10^4 \times 10 \equiv 55 \times 10 \equiv 550 \equiv 40 \pmod{85} \end{aligned}$$

Le message chiffré est donc $x = 40$.

Exemple 2. $m = 1234$, $n = 10\,403$ et $e = 7$ donc

$$x \equiv m^e \pmod{n} \equiv 1234^7 \pmod{10\,403}$$

On utilise l'ordinateur pour obtenir que $x = 10\,378$.

6.3. Déchiffrement du message

Alice reçoit le message x chiffré par Bruno, elle le décrypte à l'aide de sa clé privée d, par l'opération :

$$m \equiv x^d \pmod{n}$$

qui utilise également l'algorithme d'exponentiation rapide.
Nous allons prouver dans le lemme 1, que par cette opération Alice retrouve bien le message original m de Bruno.

Exemple 1. $c = 40$, $d = 13$, $n = 85$ donc

$$x^d \equiv (40)^{13} \pmod{85}.$$

Calculons à la main $40^{13} \equiv \pmod{85}$ on note que $13 = 8 + 4 + 1$, donc $40^{13} = 40^8 \times 40^4 \times 40$.

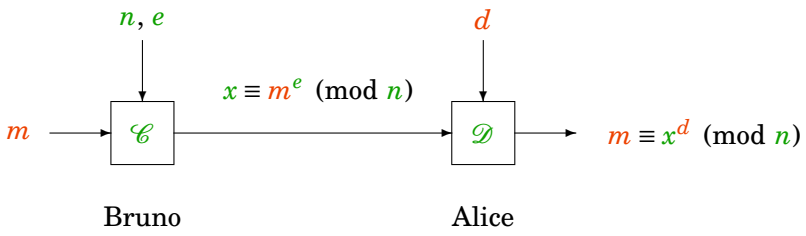
$$\begin{aligned} 40^2 &\equiv 1600 \equiv 70 \pmod{85} \\ 40^4 &\equiv (40^2)^2 \equiv 70^2 \equiv 4900 \equiv 55 \pmod{85} \\ 40^8 &\equiv (40^4)^2 \equiv 55^2 \equiv 3025 \equiv 50 \pmod{85} \end{aligned}$$

Donc

$$x^d \equiv 40^{13} \equiv 40^8 \times 40^4 \times 40 \equiv 50 \times 55 \times 40 \equiv 10 \pmod{85}$$

qui est bien le message m de Bruno.

Exemple 2. $c = 10\,378$, $d = 8743$, $n = 10\,403$. On calcule par ordinateur $x^d \equiv (10\,378)^{8743} \pmod{10\,403}$ qui vaut exactement le message original de Bruno $m = 1234$.



6.4. Schéma

Clés d'Alice :

- publique : n, e
- privée : d

6.5. Lemme de déchiffrement

Le principe de déchiffrement repose sur le petit théorème de Fermat amélioré.

Lemme 1

Soit d l'inverse de e modulo $\varphi(n)$.

$$\text{Si } x \equiv m^e \pmod{n} \text{ alors } m \equiv x^d \pmod{n}.$$

Ce lemme prouve bien que le message original m de Bruno, chiffré par clé publique d'Alice (e,n) en le message x, peut-être retrouvé par Alice à l'aide de sa clé secrète d.

Démonstration

- Que d soit l'inverse de e modulo $\varphi(n)$ signifie $d \cdot e \equiv 1 \pmod{\varphi(n)}$. Autrement dit, il existe $k \in \mathbb{Z}$ tel que $d \cdot e = 1 + k \cdot \varphi(n)$.
- On rappelle que par le petit théorème de Fermat généralisé : lorsque m et n sont premiers entre eux
$$m^{\varphi(n)} \equiv m^{(p-1)(q-1)} \equiv 1 \pmod{n}$$
- **Premier cas** $\text{pgcd}(m,n) = 1$.
Notons $c \equiv m^e \pmod{n}$ et calculons x^d :
$$x^d \equiv (m^e)^d \equiv m^{e \cdot d} \equiv m^{1+k\cdot\varphi(n)} \equiv m \cdot m^{k\cdot\varphi(n)} \equiv m \cdot (m^{\varphi(n)})^k \equiv m \cdot (1)^k \equiv m \pmod{n}$$
- **Deuxième cas** $\text{pgcd}(m,n) \neq 1$.
Comme n est le produit des deux nombres premiers p et q et que m est strictement plus petit que n alors si m et n ne sont pas premiers entre eux cela implique que p divise m ou bien q divise m (mais pas les deux en même temps). Faisons l'hypothèse $\text{pgcd}(m,n) = p$ et $\text{pgcd}(m,q) = 1$, le cas $\text{pgcd}(m,n) = q$ et $\text{pgcd}(m,p) = 1$ se traiterait de la même manière.
Étudions $(m^e)^d$ à la fois modulo p et modulo q à l'image de ce que nous avons fait dans la preuve du théorème de Fermat amélioré.
  - modulo p : $m \equiv 0 \pmod{p}$ et $(m^e)^d \equiv 0 \pmod{p}$ donc $(m^e)^d \equiv m \pmod{p}$,
  - modulo q : $(m^e)^d \equiv m \times (m^{\varphi(n)})^k \equiv m \times (m^{q-1})^{(p-1)k} \equiv m \pmod{q}$.

  Comme p et q sont deux nombres premiers distincts, ils sont premiers entre eux et on peut écrire comme dans la preuve du petit théorème de Fermat amélioré que
$$(m^e)^d \equiv m \pmod{n}$$

6.6. Algorithmes

La mise en œuvre est maintenant très simple. Alice choisit deux nombres premiers p et q et un exposant e.
Voici le calcul de la clé secrète :

Algorithme . rsa.py (1)

```
def cle_privee(p,q,e) :
    n = p * q
    phi = (p-1)*(q-1)
    c,d,dd = euclide_etendu(e,phi)          # Pgcd et coeff de Bézout
    return(d % phi)                         # Bon représentant
```

Le chiffrement d'un message m est possible par tout le monde, connaissant la clé publique (n,e).

Algorithme . rsa.py (2)

```
def codage_rsa(m,n,e):
    return pow(m,e,n)
```

Seule Alice peut déchiffrer le message crypté x, à l'aide de sa clé privée d.

Algorithme . rsa.py (3)

```
def decodage_rsa(x,n,d):
    return pow(x,d,n)
```

Pour continuer...

Bibliographie commentée :

1. **Histoire des codes secrets** de Simon Singh, Le livre de Poche.
   Les codes secrets racontés comme un roman policier. Passionnant. Idéal pour les plus littéraires.
2. **Comprendre les codes secrets** de Pierre Vigoureux, édition Ellipses.
   Un petit livre très clair et très bien écrit, qui présente un panorama complet de la cryptographie sans rentrer dans les détails mathématiques. Idéal pour les esprits logiques.
3. **Codage et cryptographie** de Joan Gómez, édition Le Monde – Images des mathématiques.
   Un autre petit livre très clair et très bien, un peu de maths, des explications claires et des encarts historiques intéressants.
4. **Introduction à la cryptographie** de Johannes Buchmann, édition Dunod.
   Un livre d'un niveau avancé (troisième année de licence) pour comprendre les méthodes mathématiques de la cryptographie moderne. Idéal pour unifier les points de vue des mathématiques avec l'informatique.
5. **Algèbre - Première année** de Liret et Martinais, édition Dunod.
   Livre qui recouvre tout le programme d'algèbre de la première année, très bien adapté aux étudiants des l'université. Pas de cryptographie.

Auteurs

Arnaud Bodin
François Recher

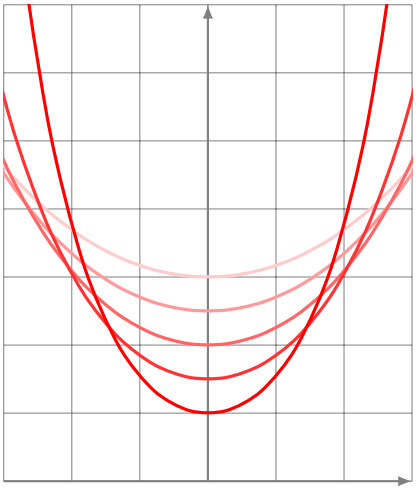
4 La chaînette

Vidéo ■ partie 1. Le cosinus hyperbolique
Vidéo ■ partie 2. Équation de la chaînette
Vidéo ■ partie 3. Longueur d'une chaînette

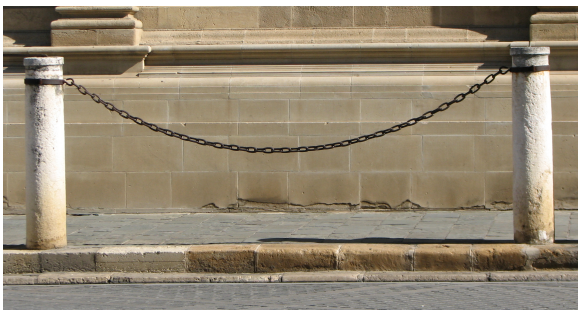
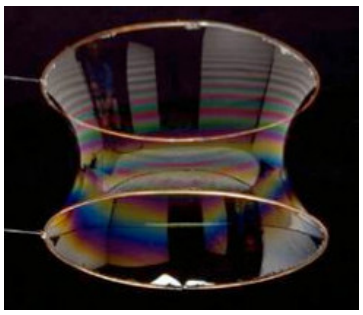
Introduction

La ***chaînette*** est le nom que porte la courbe obtenue en tenant une corde (ou un collier, un fil,...) par deux extrémités. Sans plus tarder voici l'équation d'une chaînette :

$$y = a\,\mathrm{ch}\left(\frac{x}{a}\right)$$



Ici «ch» désigne le cosinus hyperbolique, défini à partir de la fonction exponentielle : $y(x) = \frac{a}{2}\left(e^{\frac{x}{a}} + e^{-\frac{x}{a}}\right)$, nous y reviendrons.
Le paramètre a dépend de la chaînette : on peut écarter plus ou moins les mains. Ou, ce qui revient au même, si l'on garde les mains fixes, on peut prendre des cordes de différentes longueurs.
C'est donc une courbe que vous voyez tous les jours : la chaîne qui pend à votre cou ou le fil électrique entre deux pylônes. Mais on le retrouve dans des endroits plus surprenants : vous pouvez voir des chaînettes avec des films de savon. Trempez deux cercles métalliques parallèles dans de l'eau savonneuse. Il en sort une surface de révolution dont le profil est une chaînette. Enfin, si vous souhaitez faire une arche qui s'appuie sur deux piles alors la forme la plus stable est une chaînette renversée. Gaudi a beaucoup utilisé cette forme dans les bâtiments qu'il a construits.

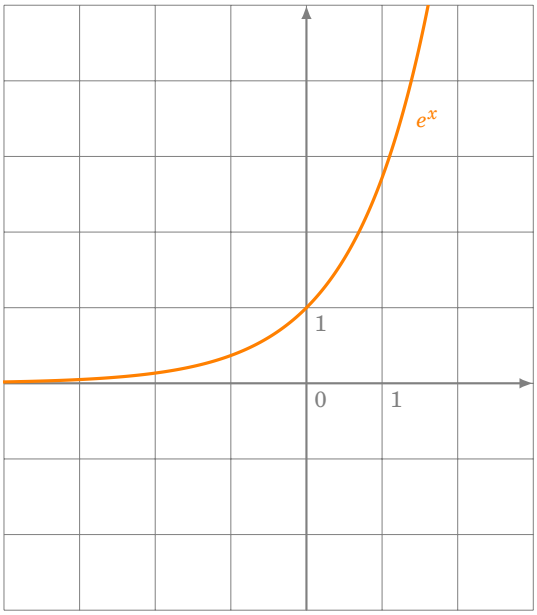

Pour finir sur un bateau, si une voile rectangulaire est maintenue par deux mats horizontaux et que le vent souffle perpendiculairement alors le profil de la voile est une chaînette.
Stop ! Place aux maths : nous allons expliquer comment calculer l'équation d'une chaînette.

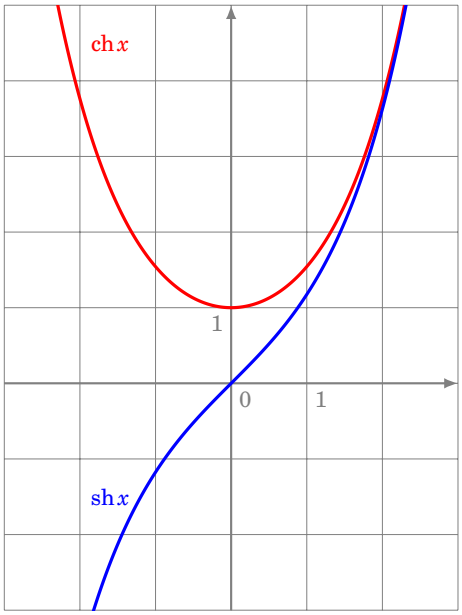
1. Le cosinus hyperbolique

1.1. Définition

Le ***cosinus hyperbolique*** et le ***sinus hyperbolique*** sont la partie paire et impaire de l'exponentielle :

$$\operatorname{ch} x = \frac{e^x + e^{-x}}{2}, \qquad \operatorname{sh} x = \frac{e^x - e^{-x}}{2}.$$





Voici quelques propriétés dont nous aurons besoin :

Proposition 6

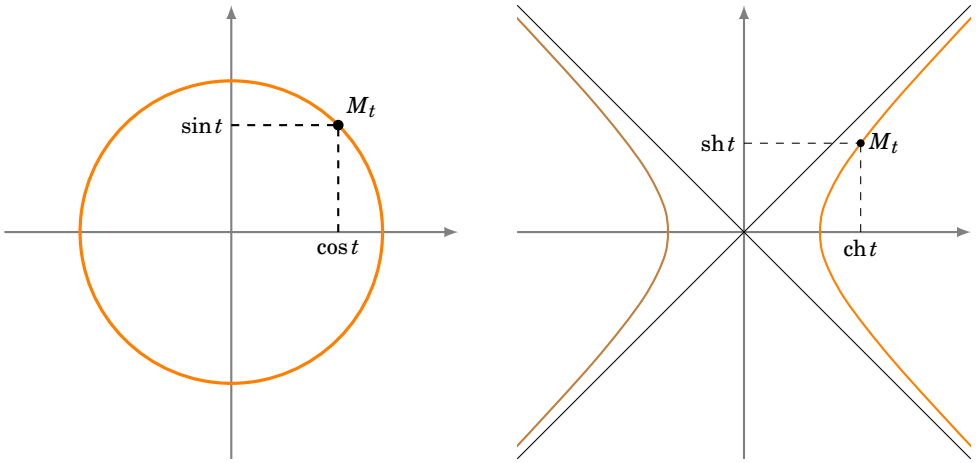
1. $\operatorname{ch}^2 x - \operatorname{sh}^2 x = 1$, pour tout $x \in \mathbb{R}$.
2. $\operatorname{ch}' x = \operatorname{sh} x$ et $\operatorname{sh}' x = \operatorname{ch} x$.

Remarque 1

Le nom cosinus hyperbolique et sinus hyperbolique ne sont pas un hasard : souvenez-vous des formules d'Euler pour le cosinus et sinus classiques (dits aussi «circulaires») :

$$\cos x = \frac{e^{\mathrm{i}x} + e^{-\mathrm{i}x}}{2}, \qquad \sin x = \frac{e^{\mathrm{i}x} - e^{-\mathrm{i}x}}{2\mathrm{i}}.$$

L'analogie avec la définition de $\operatorname{ch} x$ et $\operatorname{sh} x$ justifie les termes «cosinus» et «sinus». Reste à justifier le terme «hyperbolique».



Si nous dessinons une courbe paramétrée par $(x(t) = \cos t, y(t) = \sin t)$ alors $x(t)^2 + y(t)^2 = \cos^2 t + \sin^2 t = 1$. Donc nous avons affaire à un cercle (d'où le terme «circulaire»). Par contre si on dessine une courbe paramétrée par $(x(t) = \mathrm{ch}\, t, y(t) = \mathrm{sh}\, t)$. Alors $x(t)^2 - y(t)^2 = \mathrm{ch}^2 t - \mathrm{sh}^2 t = 1$. C'est l'équation d'une branche d'hyperbole !

1.2. Fonctions réciproques

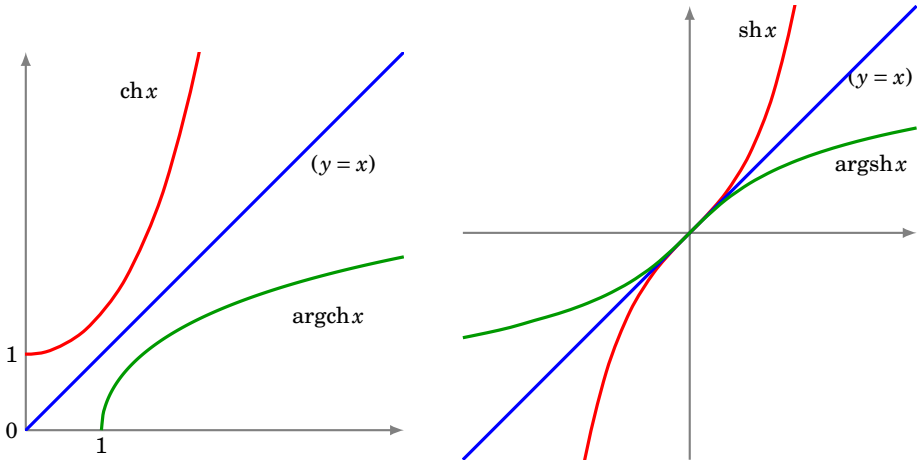
Proposition 7

– La fonction $x \mapsto \mathrm{ch}\, x$ est une bijection de $[0, +\infty[$ dans $[1, +\infty[$. Sa bijection réciproque est notée $\mathrm{argch}\, x$. Donc :

$$\mathrm{ch}\big(\mathrm{argch}(x)\big) = x \quad \forall x \in [1, +\infty[\qquad \mathrm{argch}\big(\mathrm{ch}(x)\big) = x \quad \forall x \in [0, +\infty[$$

– La fonction $x \mapsto \mathrm{sh}\, x$ est une bijection de $\mathbb{R}$ dans $\mathbb{R}$. Sa bijection réciproque est notée $\mathrm{argsh}\, x$. Donc :

$$\mathrm{sh}\big(\mathrm{argsh}(x)\big) = x \qquad \mathrm{argsh}\big(\mathrm{sh}(x)\big) = x \quad \forall x \in \mathbb{R}$$



Pour résoudre une équation différentielle nous aurons besoin de la dérivée de $\mathrm{argsh}\, x$.

Proposition 8

Les fonctions $x \mapsto \operatorname{argch} x$ et $x \mapsto \operatorname{argsh} x$ sont dérivables et

$$\operatorname{argch}' x = \frac{1}{\sqrt{x^2-1}} \qquad \operatorname{argsh}' x = \frac{1}{\sqrt{x^2+1}}.$$

1.3. Expression logarithmique

En fait, les fonctions hyperboliques inverses peuvent s'exprimer à l'aide des fonctions usuelles :

Proposition 9

$$\operatorname{argch} x = \ln\left(x + \sqrt{x^2-1}\right), \quad \text{pour } x \geqslant 1.$$
$$\operatorname{argsh} x = \ln\left(x + \sqrt{x^2+1}\right), \quad \text{pour } x \in \mathbb{R}.$$

1.4. Les démonstrations

Nous donnons les preuves des propositions précédentes pour la fonction cosinus hyperbolique. Les formules pour le sinus hyperbolique s'obtiennent de façon similaire.

Démonstration Preuve de la proposition 6

1. $\operatorname{ch}^2 x - \operatorname{sh}^2 x = \frac{1}{4}\left[(e^x+e^{-x})^2 - (e^x-e^{-x})^2\right] = \frac{1}{4}\left[(e^{2x}+2+e^{-2x}) - (e^{2x}-2+e^{-2x})\right] = 1.$
2. $\frac{d}{dx}(\operatorname{ch} x) = \frac{d}{dx}\frac{e^x+e^{-x}}{2} = \frac{e^x-e^{-x}}{2} = \operatorname{sh} x.$

Démonstration Preuve de la proposition 7

Étudions la restriction de la fonction ch : $[0,+\infty[\to [1,+\infty[$.

- Comme $\operatorname{ch}' x = \operatorname{sh} x \geqslant 0$, pour $x \geqslant 0$, alors la restriction de la fonction ch est croissante. Elle est même strictement croissante (la dérivée ne s'annule qu'en 0).
- Comme $\operatorname{ch} 0 = 1$, que $\operatorname{ch} x \to +\infty$ lorsque $x \to +\infty$, alors par continuité et la stricte croissance, la restriction ch : $[0,+\infty[\to [1,+\infty[$ est une bijection.

Par définition, la bijection réciproque de cette restriction est $\operatorname{argch} x : [1,+\infty[\to [0,+\infty[$ et vérifie :

$$\operatorname{argch}\left(\operatorname{ch} x\right) = x \qquad \text{pour tout } x \in [0,+\infty[$$
$$\operatorname{ch}\left(\operatorname{argch} x\right) = x \qquad \text{pour tout } x \in [1,+\infty[.$$

Démonstration Preuve de la proposition 8

Comme la fonction $x \mapsto \operatorname{ch}' x$ ne s'annule pas sur $]0,+\infty[$ alors la fonction argch est dérivable sur $]1,+\infty[$. On calcule la dérivée par dérivation de l'égalité $\operatorname{ch}(\operatorname{argch} x) = x$:

$$\operatorname{argch}' x \cdot \operatorname{sh}(\operatorname{argch} x) = 1$$

puis on utilise l'identité $\operatorname{ch}^2 u - \operatorname{sh}^2 u = 1$ avec $u = \operatorname{argch} x$:

$$\operatorname{argch}' x = \frac{1}{\operatorname{sh}(\operatorname{argch} x)} = \frac{1}{\sqrt{\operatorname{ch}^2(\operatorname{argch} x) - 1}} = \frac{1}{\sqrt{x^2-1}}.$$

Démonstration Preuve de la proposition 9

Notons $f(x)=\ln\left(x+\sqrt{x^2+1}\right)$.

$$f'(x)=\frac{1+\frac{x}{\sqrt{x^2+1}}}{x+\sqrt{x^2+1}}=\frac{1}{\sqrt{x^2+1}}=\operatorname{argsh}' x.$$

Comme de plus $f(0)=\ln(1)=0$ et $\operatorname{argsh}0=0$ (car $\operatorname{sh}0=0$), on en déduit que pour tout $x\in\mathbb{R}$, $f(x)=\operatorname{argsh}x$.

1.5. Dérivée des physiciens, dérivée des mathématiciens

Deux notations pour la dérivée s'affrontent : celle du mathématicien $f'(x)$ et celle du physicien $\frac{df}{dx}$. Comparons-les. La dérivée de f en x est par définition la limite (si elle existe) du taux d'accroissement :

$$\frac{f(x+h)-f(x)}{x+h-x},$$

lorsque h tend vers 0. Notons $h=dx$ et $df=f(x+h)-f(x)=f(x+dx)-f(x)$ alors le taux d'accroissement vaut $\frac{df}{dx}$ et comme dx est un nombre aussi petit que l'on veut (il est *infinitésimal*), on identifie ce quotient $\frac{df}{dx}$ avec la limite lorsque $dx\to 0$.

L'avantage de la notation des physiciens est que cela peut correspondre à un raisonnement physique. On peut raisonner sur des petits morceaux (de longueur dx petite mais pas nulle) et en déduire une relation avec des dérivées. C'est ce que nous ferons dans le paragraphe 2.3.

Autre avantage de cette notation, il est facile de retenir la formule :

$$\frac{df}{dx}=\frac{dy}{dx}\times\frac{df}{dy}.$$

Il s'agit juste de «simplifier» le numérateur avec le dénominateur.

Cette opération est justifiée, car il s'agit de la dérivée de la composée $f\circ y\,(x)=f\big(y(x)\big)$ qui est bien

$$(f\circ y)'(x)=y'(x)\times f'\big(y(x)\big).$$

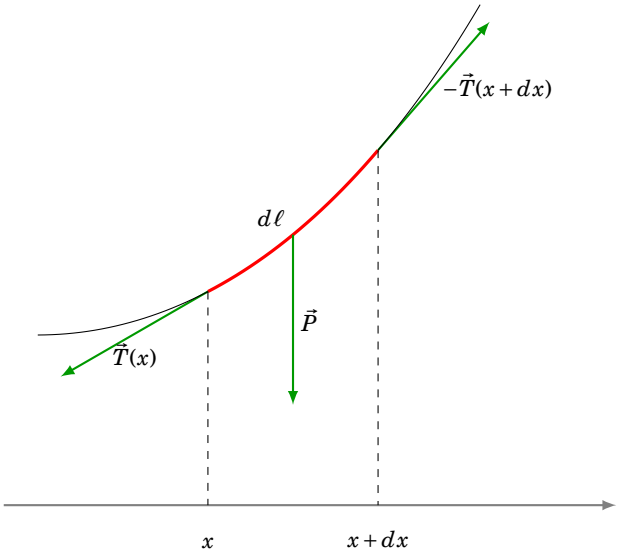
2. Équation de la chaînette

Soit $(O,\vec{i},\vec{j})$ un repère orthonormé direct, $\vec{j}$ est un vecteur vertical dirigé vers le haut (c'est-à-dire opposé au champ de pesanteur).

2.1. Découpage infinitésimal de la chaînette

Nous découpons la chaînette en petits morceaux, chaque morceau étant compris entre les abscisses x et $x+dx$. Ici dx désigne donc un réel aussi petit que l'on veut. Nous noterons $d\ell$ la longueur de ce petit morceau de chaînette.

Trois forces s'appliquent à notre mini-bout de chaînette :



- ***Le poids $\vec{P}$.*** C'est une force verticale, proportionnelle à la masse du morceau. Si μ est la masse linéique (c'est-à-dire la masse que ferait un mètre de chaîne, exprimée en kg/m), la masse de notre petit bout est $\mu \cdot d\ell$. Si g dénote la constante de gravitation (avec $g \approx 9{,}81\, m/s^2$) alors le poids est $\vec{P} = -P\vec{j} = -\mu \cdot d\ell \cdot g \cdot \vec{j}$.
- ***La tension à gauche $\vec{T}(x)$.*** La tension à gauche, s'applique au point dont l'abscisse est x. Par un principe physique, les forces de tension de notre morceau à l'équilibre sont des forces tangentes à la chaînette.
- ***La tension à droite $-\vec{T}(x+dx)$.*** La tension à droite s'applique au point d'abscisse $x+dx$. Comme notre morceau est en équilibre elle s'oppose à la tension à gauche du morceau suivant compris entre $x+dx$ et $x+2dx$. La tension à droite de notre morceau est donc l'opposée de la tension à gauche du morceau suivant, cette force est donc $-\vec{T}(x+dx)$.

Une remarque : pour cette modélisation nous supposons que dx est le même pour tous les morceaux de chaîne. Par contre x varie, et aussi la longueur du morceau de chaîne entre les abscisses x et $x+dx$, qui devrait donc plutôt être notée $d\ell(x)$ au lieu de $d\ell$. Le poids d'un morceau de chaîne dépend lui aussi de x et devrait plutôt être noté $P(x)$.

2.2. Principe fondamental de la mécanique

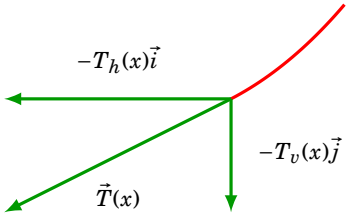
Le principe fondamental de la mécanique nous dit que, à l'équilibre, la somme des forces est nulle, donc :

$$\vec{P} + \vec{T}(x) - \vec{T}(x+dx) = \vec{0}. \tag{4.1}$$

Décomposons chaque force de tension, en une tension horizontale et une tension verticale :

$$\vec{T}(x) = -T_h(x)\vec{i} - T_v(x)\vec{j}.$$

La convention pour le choix des signes permet d'avoir des valeurs $T_h(x)$ et $T_v(x)$ positives.



Alors le principe fondamental de la mécanique devient :

$$-P\vec{j} - T_h(x)\vec{i} - T_v(x)\vec{j} - \left(-T_h(x+dx)\vec{i} - T_v(x+dx)\vec{j}\right) = \vec{0}.$$

Comme $(\vec{i},\vec{j})$ est une base, nous reformulons le principe fondamental de la mécanique en deux équations, correspondant aux forces horizontales et aux forces verticales :

$$\left\{\begin{array}{rcl} T_h(x+dx) - T_h(x) & = & 0 \\ T_v(x+dx) - T_v(x) - P & = & 0 \end{array}\right. \tag{4.2}$$

2.3. Tension horizontale

La première équation du système (4.2) nous permet de montrer que la tension horizontale est constante.

Lemme 2

La tension horizontale est indépendante de x :

$$T_h(x) = T_h.$$

Démonstration

En effet, fixons x, nous savons $T_h(x+dx) - T_h(x) = 0$, donc le rapport

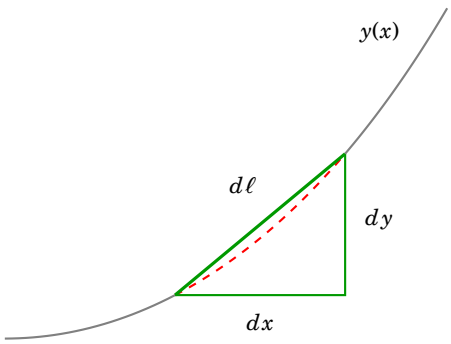
$$\frac{T_h(x+dx) - T_h(x)}{x+dx-x} = 0.$$

Ceci est vrai quelque soit l'élément infinitésimal dx. Ce taux d'accroissement étant toujours nul, la limite lorsque dx tend vers 0 est nulle. Mais la limite est –par définition– la dérivée $T_h'(x)$. Bilan : $T_h'(x) = 0$. La fonction $T_h(x)$ est donc une fonction constante comme nous l'avions annoncé.

2.4. Tension verticale et poids

Nous noterons $y(x)$ l'équation de la chaînette. Nous considérons que chaque morceau infinitésimal de la chaîne est rectiligne, nous pouvons alors appliquer le théorème de Pythagore, dans un petit triangle rectangle dont l'hypoténuse est $d\ell$:

$$d\ell^2 = dx^2 + dy^2.$$



Cela conduit à :

$$\left(\frac{d\ell}{dx}\right)^2 = 1 + \left(\frac{dy}{dx}\right)^2.$$

D'où

$$\frac{d\ell}{dx} = \sqrt{1+\left(\frac{dy}{dx}\right)^2}.$$

Nous allons maintenant nous concentrer sur la deuxième équation du principe fondamental (4.2), le poids étant $P = \mu g d\ell$:

$$T_v(x+dx) - T_v(x) = \mu g d\ell.$$

Cela donne en divisant par dx :

$$\frac{T_v(x+dx)-T_v(x)}{dx} = \mu g \frac{d\ell}{dx} = \mu g \sqrt{1+\left(\frac{dy}{dx}\right)^2}.$$

En terme de dérivée $\frac{dy}{dx}$ vaut à la limite $y'(x)$ et $\frac{T_v(x+dx)-T_v(x)}{dx}$ vaut à la limite $T_v'(x)$.
Nous avons donc montré :

$$T_v'(x) = \mu g \sqrt{1+y'(x)^2}. \tag{4.3}$$

2.5. Calcul de l'équation

Théorème 4

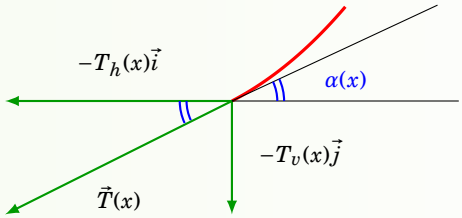
Une équation de la chaînette est

$$y(x) = a \operatorname{ch}\left(\frac{x}{a}\right)$$

où a est une constante qui vaut $a = \frac{T_h}{\mu g}$.

Démonstration

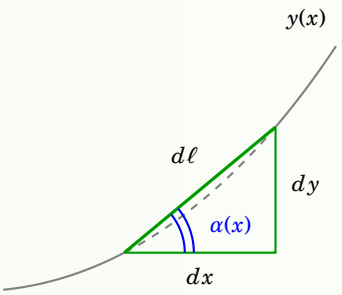
1. Lien tension verticale/tension horizontale.
Tout d'abord nous lions la tension horizontale T_h et la tension verticale T_v en fonction de l'angle que forme la chaînette avec l'horizontale. T dénote la norme de $\vec{T}$.



On obtient :

$$T_h(x) = T(x)\cos\alpha(x) \quad \text{et} \quad T_v(x) = T(x)\sin\alpha(x).$$

Ce qui conduit à $T_v(x) = T_h(x)\tan\alpha(x)$.
Maintenant, dans le triangle infinitésimal, nous avons aussi que $\tan\alpha(x) = \frac{dy}{dx} = y'(x)$.



Ce qui nous mène à la relation :

$$T_v(x) = T_h(x) \cdot y'(x).$$

2. Équations différentielles.
Nous savons que la tension horizontale est constante (lemme 2), donc en dérivant l'égalité précédente, nous avons

$$T_v'(x) = T_h \cdot y''(x).$$

Avec l'équation (4.3) nous écrivons

$$\mu g \sqrt{1 + y'(x)^2} = T_h \cdot y''(x).$$

C'est une équation différentielle du second d'ordre :

$$y''(x) = \frac{\mu g}{T_h} \sqrt{1 + y'(x)^2}. \tag{4.4}$$

Soit a la constante $a = \frac{T_h}{\mu g}$. Posons $z(x) = y'(x)$. Cela nous conduit à une équation différentielle du premier ordre $z'(x) = \frac{1}{a}\sqrt{1 + z(x)^2}$ ou encore :

$$\frac{z'(x)}{\sqrt{1 + z(x)^2}} = \frac{1}{a}.$$

3. Solutions de l'équation différentielle.
Une primitive de $\frac{z'(x)}{\sqrt{1+z(x)^2}}$ est $\operatorname{argsh} z(x)$, donc

$$\operatorname{argsh} z(x) = \frac{x}{a} + \alpha$$

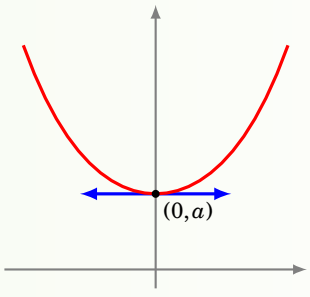
où α est une constante. En composant des deux côtés par le sinus hyperbolique :

$$y'(x) = z(x) = \operatorname{sh}\left(\frac{x}{a} + \alpha\right).$$

Une primitive de $\operatorname{sh} x$ étant $\operatorname{ch} x$, il ne reste plus qu'à intégrer :

$$y(x) = a \operatorname{ch}\left(\frac{x}{a} + \alpha\right) + \beta.$$

4. Choix des constantes.
Si l'on suppose que le point le plus bas de la chaînette a pour coordonnées $(0, a)$ alors $y(0) = a$ et $y'(0) = 0$. On a choisit $\alpha = 0$ et $\beta = 0$ pour les deux constantes.



L'équation est alors $y(x) = a \operatorname{ch}\left(\frac{x}{a}\right)$.

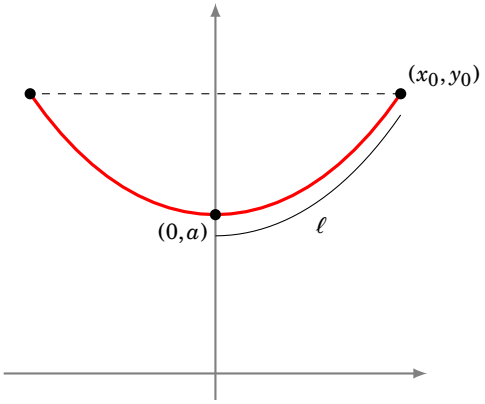
3. Longueur d'une chaînette

3.1. Longueur d'une chaînette

Proposition 10

La longueur de la portion de la chaînette de paramètre a entre le point le plus bas $(0,a)$ et le point d'abscisse x_0 est :

$$\ell = a \operatorname{sh} \frac{x_0}{a}.$$



Démonstration

On rappelle l'équation de la chaînette : $y(x) = a \operatorname{ch} \frac{x}{a}$. Par définition la longueur vaut

$$\ell = \int_0^{x_0} \sqrt{1 + y'(x)^2}\, dx.$$

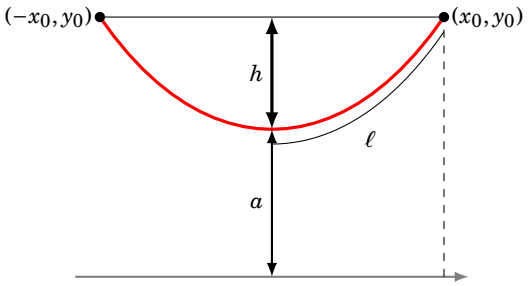
Ainsi :

$$\begin{aligned} \ell &= \int_0^{x_0} \sqrt{1 + \operatorname{sh}^2 \tfrac{x}{a}}\, dx \quad \text{car } \operatorname{ch}' \tfrac{x}{a} = \tfrac{1}{a} \operatorname{sh} \tfrac{x}{a} \\ &= \int_0^{x_0} \sqrt{\operatorname{ch}^2 \tfrac{x}{a}}\, dx \quad \text{car } 1 + \operatorname{sh}^2 u = \operatorname{ch}^2 u \\ &= \int_0^{x_0} \operatorname{ch} \tfrac{x}{a}\, dx = \left[a \operatorname{sh} \tfrac{x}{a} \right]_0^{x_0} \\ &= a \operatorname{sh} \tfrac{x_0}{a}. \end{aligned}$$

3.2. Calcul du paramètre

La chaînette ne dépend que du seul paramètre a. Ce paramètre a vaut $a = \frac{T_h}{\mu g}$ et est fonction de la masse μ du fil par unité de longueur, de la constante de gravitation g et de la tension horizontale T_h, qui elle dépend de l'écartement de deux points par lesquels passe la chaînette. Ce qui fait qu'il n'est pas facile de calculer a ainsi.

Fixons deux points, pour simplifier nous supposerons qu'ils sont à la même hauteur (même ordonnée). Prenons une chaînette de longueur 2ℓ fixée (et connue !). Nous allons calculer le paramètre a en fonction de la longueur 2ℓ et de la flèche h. La ***flèche*** est la hauteur h entre les deux points d'accroche et le point le plus bas de la chaînette.



Proposition 11

Pour une chaînette de longueur 2ℓ et de flèche h alors

$$a = \frac{\ell^2 - h^2}{2h}.$$

Démonstration

Soient $(\pm x_0, y_0)$ les coordonnées des points d'accroche. L'équation de la chaînette étant $y = a \operatorname{ch} \frac{x}{a}$, alors $y_0 = a \operatorname{ch} \frac{x_0}{a}$ qui vaut aussi $y_0 = a + h$.
Quant à la longueur elle vaut $2\ell = 2a \operatorname{sh}\left(\frac{x_0}{a}\right)$. Nous avons donc les équations :

$$\begin{cases} \ell &= a \operatorname{sh} \frac{x_0}{a} \\ h &= a \operatorname{ch} \frac{x_0}{a} - a \end{cases}$$

Nous obtenons donc :

$$\begin{aligned}\ell^2-h^2 &= a^2\operatorname{sh}^2\tfrac{x_0}{a}-\left(a\operatorname{ch}\tfrac{x_0}{a}-a\right)^2\\ &= a^2\operatorname{sh}^2\tfrac{x_0}{a}-a^2\operatorname{ch}^2\tfrac{x_0}{a}-a^2+2a^2\operatorname{ch}\tfrac{x_0}{a}\\ &= 2a\left(-a+a\operatorname{ch}\tfrac{x_0}{a}\right) \qquad \text{car } \operatorname{ch}^2u-\operatorname{sh}^2u=1\\ &= 2ah.\end{aligned}$$

Ainsi $a=\dfrac{\ell^2-h^2}{2h}$.

3.3. Équation paramétrique

Proposition 12

Une équation paramétrique de la chaînette est :

$$\begin{cases} x(t) &= a\ln t\\ y(t) &= \frac{a}{2}\left(t+\frac{1}{t}\right)\end{cases}$$

pour $t>0$.

Démonstration

Nous connaissons l'équation cartésienne $y=a\operatorname{ch}\left(\frac{x}{a}\right)$, qui est équivalente à $\operatorname{argch}\left(\frac{y}{a}\right)=\frac{x}{a}$. Utilisons la forme logarithmique de la fonction argch : $\operatorname{argch}u=\ln\left(u+\sqrt{u^2-1}\right)$ (pour $u\geqslant 1$).
Nous obtenons :

$$\ln\left(\frac{y}{a}+\sqrt{\left(\frac{y}{a}\right)^2-1}\right)=\frac{x}{a}.$$

Nous cherchons maintenant une paramétrisation $(x(t),y(t))$ de la chaînette, posons $x(t)=a\ln(t)$ (ce qui est toujours possible car ln est une bijection de $]0,+\infty[$ dans $\mathbb{R}$). Alors l'équation précédente conduit (après simplification des ln) à :

$$\frac{y(t)}{a}+\sqrt{\left(\frac{y(t)}{a}\right)^2-1}=t,$$

ou encore

$$\sqrt{\left(\frac{y(t)}{a}\right)^2-1}=t-\frac{y(t)}{a}$$

ce qui implique en élevant au carré :

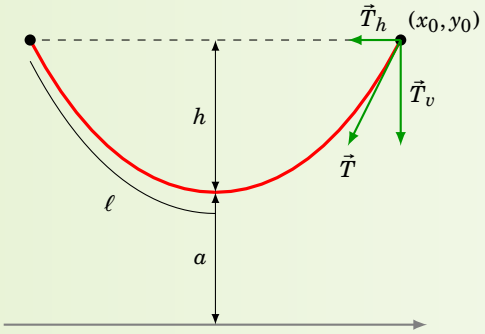
$$\left(\frac{y(t)}{a}\right)^2-1=t^2+\left(\frac{y(t)}{a}\right)^2-2t\frac{y(t)}{a}$$

d'où $\frac{y(t)}{a}=\frac{t^2+1}{2t}$, et donc $y(t)=\frac{a}{2}\left(t+\frac{1}{t}\right)$.

3.4. Calcul de la tension

Proposition 13

Nous pouvons calculer la tension en un point (x_0, y_0) de la chaînette. On note h la flèche correspondante et ℓ la longueur entre le point le plus bas et (x_0, y_0).



- La *tension horizontale* T_h est constante et vaut :

$$T_h = a\mu g = \frac{\ell^2 - h^2}{2h}\mu g.$$

- Le *tension verticale* au point (x_0, y_0) est :

$$T_v = T_h \cdot \operatorname{sh}\frac{x_0}{a} = T_h \cdot \frac{\ell}{a}.$$

- La *tension totale* au point (x_0, y_0) est :

$$T = \sqrt{T_h^2 + T_v^2} = T_h \cdot \operatorname{ch}\frac{x_0}{a} = T_h \cdot \frac{a+h}{a}.$$

La tension croît donc avec la hauteur du point.

Démonstration

- On a vu dans le lemme 2 que la tension horizontale est constante. La formule $T_h = a\mu g$ provient de la définition même de la constante a (voir le théorème 4). Enfin, la dernière égalité est donnée par la proposition 10.
- Par la preuve du théorème 4 : $T_v(x_0) = T_h \cdot y'(x_0) = T_h \cdot \operatorname{sh} \frac{x_0}{a} = T_h \cdot \frac{\ell}{a}$.
- Le vecteur tension est $\vec{T}(x) = -T_h(x)\vec{i} - T_v(x)\vec{j}$, donc la norme au point d'abscisse x_0 est $T(x_0) = \|\vec{T}(x_0)\| = \sqrt{T_h^2 + T_v^2} = T_h\sqrt{1+\operatorname{sh}^2 \frac{x_0}{a}} = T_h \cdot \operatorname{ch} \frac{x_0}{a} = T_h \cdot \frac{a+h}{a}$. La dernière égalité est juste le fait que $y_0 = a + h = a \operatorname{ch} \frac{x_0}{a}$.

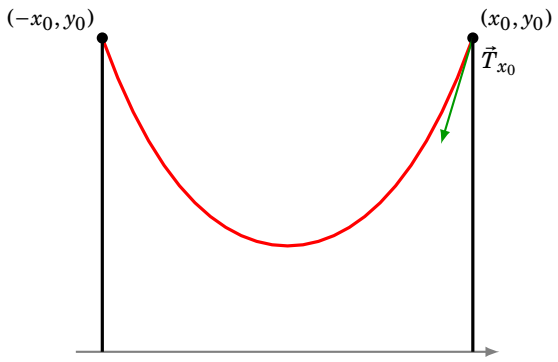
3.5. Exercices

Exercice 1. Tension minimale

On se donne deux poteaux distants d'une longueur $2x_0$ fixée et d'une hauteur suffisante. Parmi toutes les chaînettes passant par les sommets de ces poteaux, on cherche celle qui a les forces de tensions minimales.
Nous savons que la tension totale (voir la proposition 13) vaut

$$T_x(a) = a\mu g \cdot \operatorname{ch} \frac{x}{a}.$$

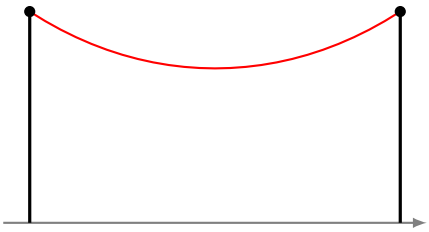
Pour une chaînette donnée, la tension est donc maximale au point d'accroche (en $x = x_0$) car le cosinus hyperbolique est une fonction croissante sur $[0, +\infty[$. Pour un a fixé, la tension maximale est donc $T_{x_0}(a)$. Notre problème, x_0 étant fixé, est de trouver le a qui minimise $T_{x_0}(a)$.



1. Considérations physiques : Que vaut la tension si la chaînette est rectiligne (la longueur

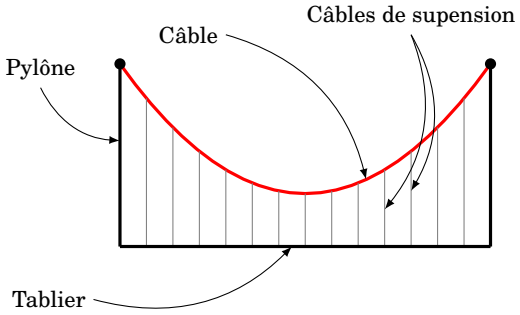
de la chaînette est celle de l'écartement) ? Que vaut la tension si la longueur de la chaînette est infinie ?

2. Montrer que l'équation $\operatorname{ch} t = t \operatorname{sh} t$ est équivalente à l'équation $(t-1)e^{2t} = t+1$. Montrer que, sur $[0, +\infty[$, cette équation a une unique solution τ. Une valeur approchée de τ est $\tau = 1,19968\ldots$
3. Montrer que la tension $T_{x_0}(a)$ est minimale en $a = \frac{x_0}{\tau}$.
4. Calculer la longueur correspondante, ainsi que la flèche.



Exercice 2. Pont suspendu

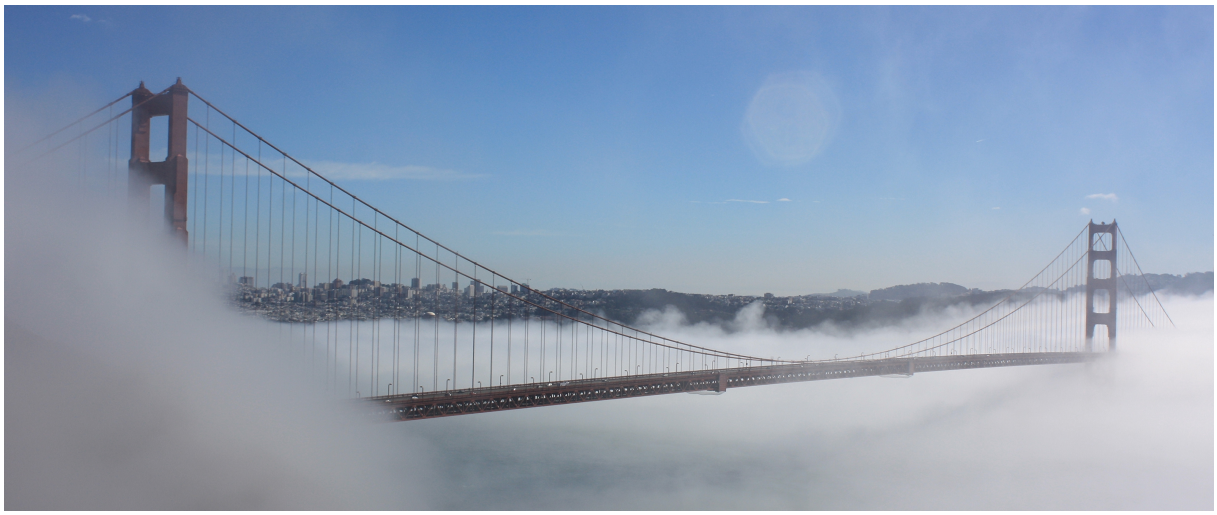
Nous allons calculer que la courbe du câble d'un pont suspendu est une parabole.
Soit le tablier d'un pont de longueur L et de masse totale M. Un gros câble est accroché entre deux pylônes. À ce câble sont accrochés un grand nombre de petits câbles de suspension verticaux reliant le gros câble au tablier.



Nous allons calculer l'équation $y(x)$ du câble. On s'inspirera pour les premières questions des calculs sur la chaînette.

1. Quelles sont les forces qui s'appliquent à une portion de câble dont l'abscisse est entre x et $x + dx$?
2. Écrire l'équation du principe fondamental de la mécanique, appliqué à cette portion.
3. Montrer que la tension horizontale est indépendante de x.
4. Montrer que la tension verticale vérifie l'équation différentielle : $T_v'(x) = -T_h \cdot y''(x)$.
5. Dans toute la suite nous supposerons que la masse du câble est négligeable devant celle du tablier. Cela revient à supposer que le poids $P(x)$ du câble est négligeable devant la charge $C(x)$ du tablier. Nous posons donc $P(x) = 0$. Montrer que le principe fondamental de la mécanique s'écrit alors :
$$T_h \cdot y''(x) = \frac{M}{L} g.$$
6. Quelle est l'équation $y(x)$ du câble ?

7. Calculer une équation du câble du *Golden Bridge* (San Francisco). Le tablier mesure 1280 mètres de long, les pylônes ont une hauteur de 160 mètres (au-dessus du tablier) et le câble descend jusqu'au tablier (au milieu du pont).



Auteurs

Arnaud Bodin
Relu par Laura Desideri.
Photos : fdecomite, soapbubble.dk, N. Jamal, G. Sivills, M. Gunn.

5 La règle et le compas

Vidéo ■ partie 1. Constructions
Vidéo ■ partie 2. Nombres constructibles
Vidéo ■ partie 3. Éléments de théorie des corps
Vidéo ■ partie 4. Corps et nombres constructibles
Vidéo ■ partie 5. Applications aux problèmes grecs

Vous avez à votre disposition une règle et un compas et bien sûr du papier et un crayon ! Avec si peu de matériel s'ouvre à vous un monde merveilleux rempli de géométrie et d'algèbre.

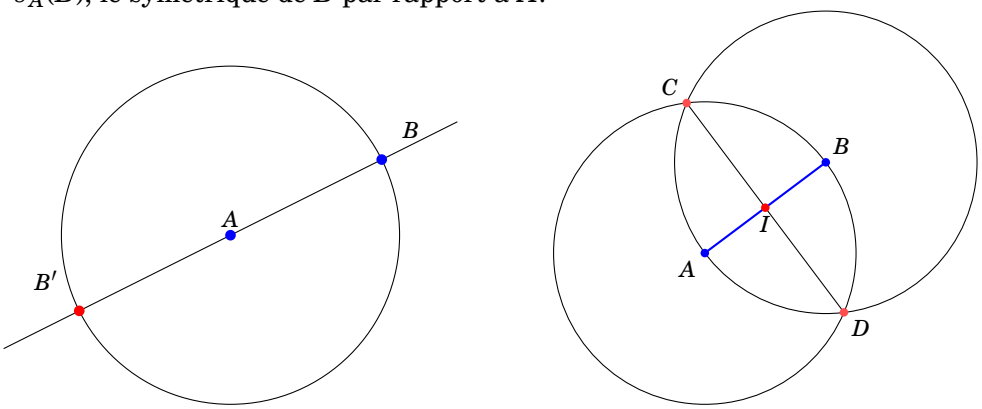
1. Constructions et les trois problèmes grecs

Nous allons voir dans cette première partie que tout un tas de constructions sont possibles. Mais le but de ce cours est de répondre à trois problèmes qui datent des mathématiciens grecs : la trisection des angles, la duplication du cube ainsi que le célèbre problème de la quadrature du cercle.

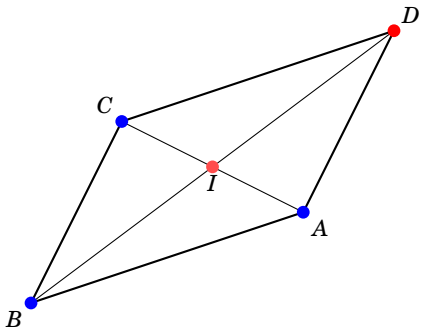
1.1. Premières constructions géométriques

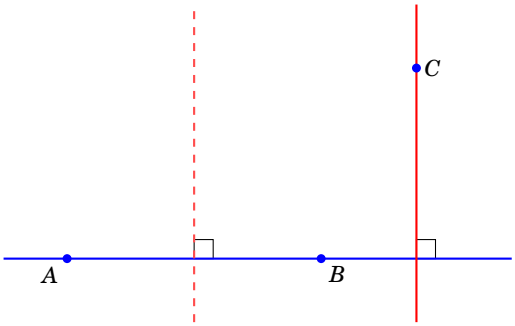
Nous avons à notre disposition un compas et une règle (non graduée). On démarre par des constructions élémentaires.

– Si A, B sont deux points donnés du plan, alors on peut construire, à la règle et au compas, le ***symétrique*** de B par rapport à A. Pour cela, il suffit juste de tracer la droite (AB) et le cercle de centre A passant par B. Cette droite et ce cercle se coupent en B bien sûr et aussi en $B' = s_A(B)$, le symétrique de B par rapport à A.



- Si A,B sont deux points donnés du plan, alors on peut construire la ***médiatrice*** de $[AB]$. Pour cela, tracer le cercle centré en A passant par B et aussi le cercle centré en B passant par A. Ces deux cercles s'intersectent en deux points C, D. Les points C, D appartiennent à la médiatrice de $[AB]$. Avec la règle on trace la droite (CD) qui est la médiatrice de $[AB]$.
- En particulier cela permet de construire le ***milieu*** I du segment $[AB]$. En effet, c'est l'intersection de la droite (AB) et de la médiatrice (CD) que l'on vient de construire.
- Si A,B,C sont trois points donnés alors on peut construire la ***parallèle*** à la droite (AB) passant par C. Tout d'abord construire le milieu I de $[AC]$. Puis construire D le symétrique de B par rapport à I. La figure $ABCD$ est un ***parallélogramme***, donc la droite (CD) est bien la parallèle à la droite (AB) passant par C.



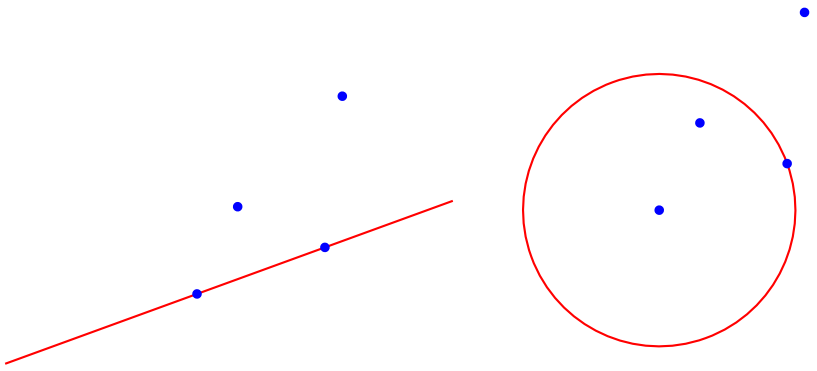


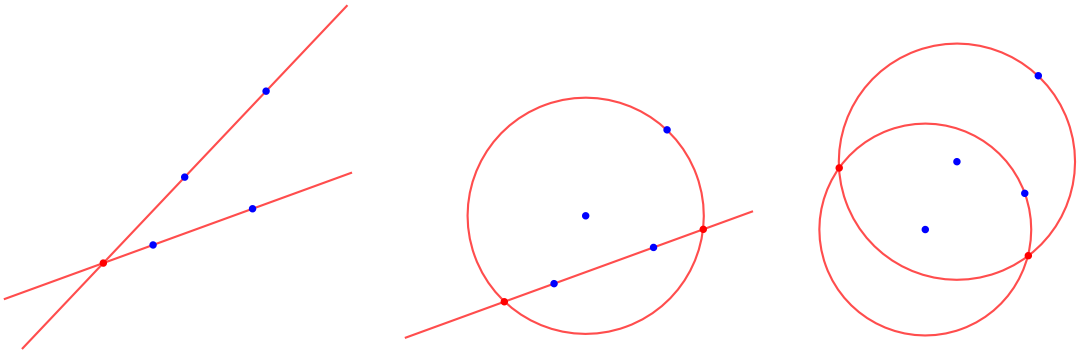
- Pour construire la ***perpendiculaire*** à (AB) passant par un point C, on construit d'abord deux points de la médiatrice de $[AB]$, puis la parallèle à cette médiatrice passant par C.

1.2. Règles du jeu

Il est peut-être temps d'expliquer ce que l'on est autorisé à faire. Voici les règles du jeu : partez de points sur une feuille. Vous pouvez maintenant tracer d'autres points, à partir de cercles et de droites en respectant les conditions suivantes :

- vous pouvez tracer une droite entre deux points déjà construits,
- vous pouvez tracer un cercle dont le centre est un point construit et qui passe par un autre point construit,
- vous pouvez utiliser les points obtenus comme intersections de deux droites tracées, ou bien intersections d'une droite et d'un cercle tracé, ou bien intersections de deux cercles tracés.

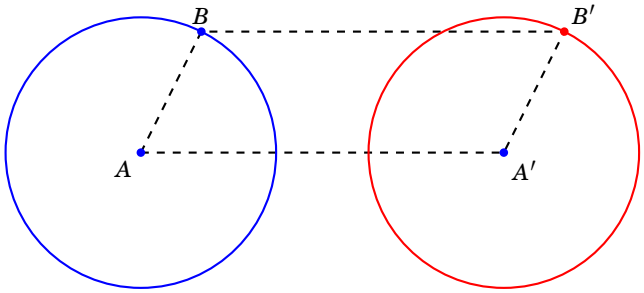




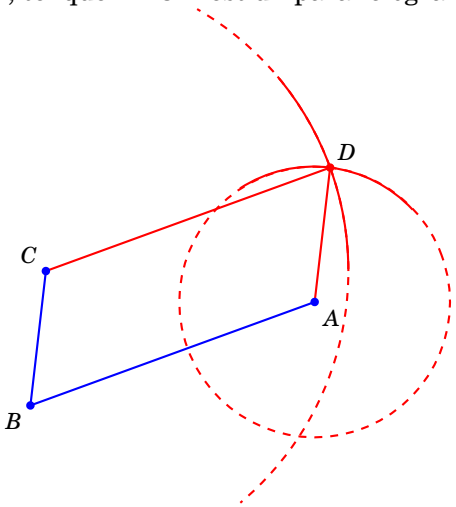
– Une remarque importante : la règle est une règle simple, qui n'est pas graduée.
– Convention pour les couleurs : les points donnés à l'avance sont les points bleus. Les constructions se font en rouge (rouge pâle pour les constructions qui viennent en premier, rouge vif pour les constructions qui viennent en dernier).

1.3. Conserver l'écartement du compas

– On peut ***conserver l'écartement du compas***. C'est une propriété importante qui simplifie les constructions.
Si l'on a placé des points A, B, A' alors on peut placer la pointe en A avec un écartement de longueur AB. C'est-à-dire que l'on peut mesurer le segment $[AB]$, puis soulever le compas en gardant l'écartement pour tracer le cercle centré en A' et d'écartement AB.
Cette opération se justifie de la façon suivante : on pourrait construire le point B' tel que $A'ABB'$ soit un parallélogramme et ensuite tracer le cercle centré en A' passant par B'.



– En conservant l'écartement du compas, nous pouvons plus facilement construire les parallélogrammes, avec seulement deux traits de compas. Donnons-nous trois points A, B, C. On mesure l'écartement $[AB]$, on trace le cercle centré en C de rayon AB. Puis on mesure l'écartement $[BC]$ et on trace le cercle centré en A de rayon BC. Ces deux cercles se recoupent en deux points, dont l'un est D, tel que $ABCD$ est un parallélogramme.

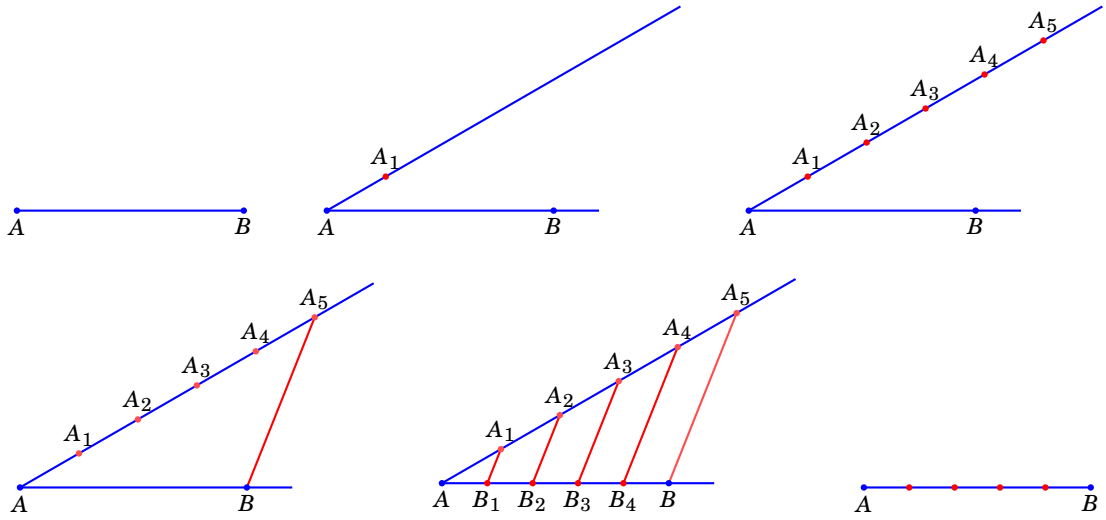


1.4. Thalès et Pythagore

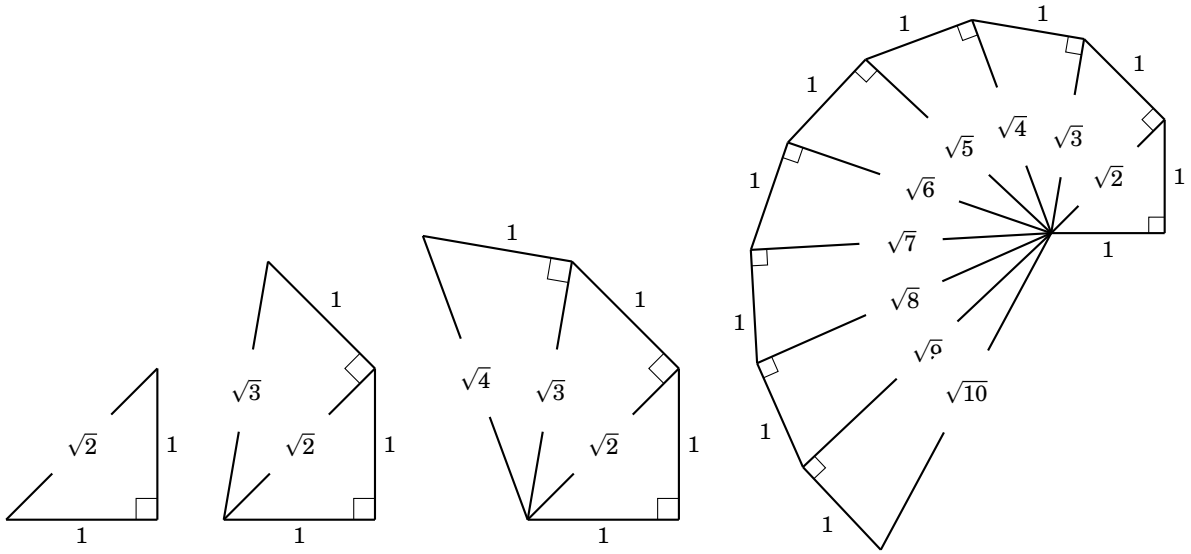
Voyons comment le théorème de Thalès nous permet de diviser un segment en n morceaux. Fixons n un entier. Voici les étapes pour diviser un segment $[AB]$ en n parts égales.

1. Tracer une droite $\mathscr{D}$ quelconque, passant par A, autre que la droite (AB).
2. Prendre un écartement quelconque du compas. Sur la droite $\mathscr{D}$ et en partant de A, tracer n segments de même longueur. On obtient des points $A_1, A_2, \ldots, A_n$.
3. Tracer la droite (A_nB). Tracer les parallèles à cette droite passant par A_i. Ces droites recoupent le segment $[AB]$ en des points $B_1, B_2, \ldots, B_{n-1}$ qui découpent l'intervalle $[AB]$ en n segments égaux.

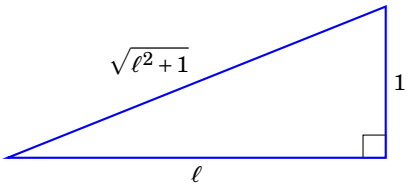
Cette construction fonctionne grâce au théorème de Thalès.



Voyons maintenant comment le théorème de Pythagore va nous permettre de faire apparaître des racines carrées. Supposons que l'on parte d'un segment de longueur 1. Il est facile de construire un segment de longueur $\sqrt{2}$: c'est la longueur de la diagonale du carré de côté 1. Repartons du segment diagonal de longueur $\sqrt{2}$: on construit un triangle rectangle avec un côté de longueur 1, et l'hypoténuse a alors pour longueur $\sqrt{3}$ (voir le calcul plus bas). Repartant de ce segment, on construit un « escargot » avec des segments de longueurs $\sqrt{4}$, $\sqrt{5}$...



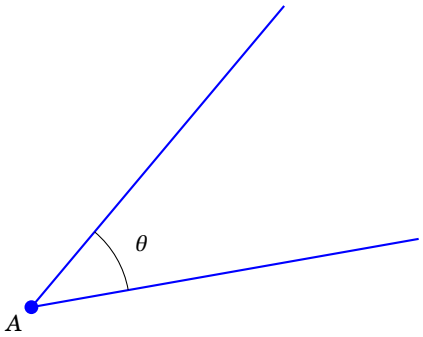
Tout ceci se justifie par le théorème de Pythagore : dans un triangle rectangle ayant un côté de longueur ℓ et un autre de longueur 1, l'hypoténuse est de longueur $\sqrt{\ell^2+1}$. En partant de $\ell_1 = 1$, on trouve $\ell_2 = \sqrt{\ell_1^2+1} = \sqrt{2}$, puis $\ell_3 = \sqrt{\ell_2^2+1} = \sqrt{3}$, $\ell_4 = \sqrt{4} = 2$, et plus généralement $\ell_n = \sqrt{n}$.

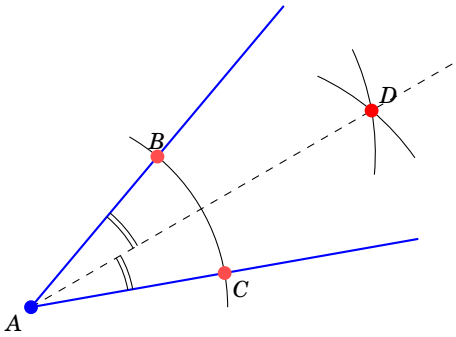


Voici maintenant trois questions qui datent de la Grèce antique et qui vont nous occuper le reste du chapitre.

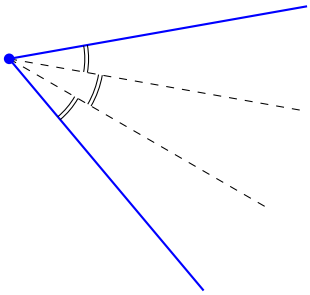
1.5. La trisection des angles

Considérons un angle θ, c'est-à-dire la donnée d'un point A et de deux demi-droites issues de ce point. Nous savons diviser cet angle en deux à l'aide d'une règle et d'un compas : il suffit de tracer la bissectrice. Pour cela on fixe un écartement de compas et on trace un cercle centré en A : il recoupe les demi-droites en des points B et C. On trace maintenant deux cercles centrés en B puis C (avec le même rayon pour les deux cercles). Si D est un point de l'intersection de ces deux cercles alors la droite (AD) est la bissectrice de l'angle.



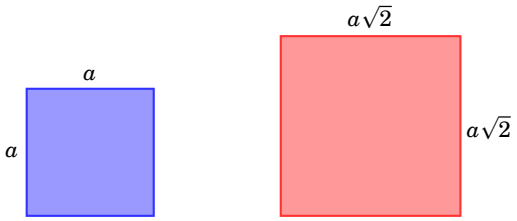


Problème de la trisection. Peut-on diviser un angle donné en trois angles égaux à l'aide de la règle et du compas ?



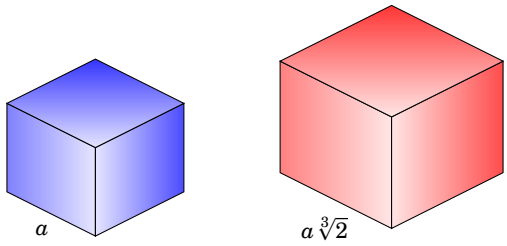
1.6. La duplication du cube

Commençons par un problème assez simple : étant donné un carré, construire (à la règle et au compas) un carré dont l'aire est le double. C'est facile, car cela revient à savoir tracer un côté de longueur $a\sqrt{2}$ à partir d'un côté de longueur a. En fait la diagonale de notre carré original a la longueur voulue $a\sqrt{2}$. Partant de cette longueur, on construit un carré dont l'aire est $(a\sqrt{2})^2 = 2a^2$: son aire est bien le double de celle du carré de départ.



$S = a^2$ $S = 2a^2$

Posons nous la question dans l'espace : étant donné un cube, peut-on construire un second cube dont le volume est le double de celui du premier ? Si le premier cube a ses côtés de longueur a, alors le second doit avoir ses côtés de longueur $a\sqrt[3]{2}$. La question se formule alors de la manière suivante :

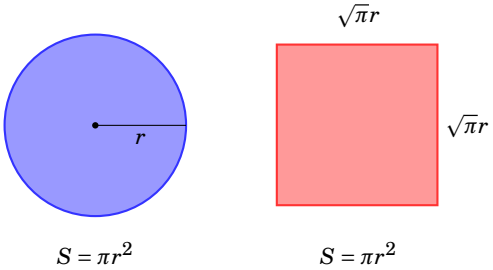


$V = a^3$ $V = 2a^3$

Problème de la duplication du cube. Étant donné un segment de longueur 1, peut-on construire à la règle et au compas un segment de longueur $\sqrt[3]{2}$?

1.7. La quadrature du cercle

Problème de la quadrature du cercle. Étant donné un cercle, peut-on construire à la règle et au compas un carré de même aire ?



$S = \pi r^2$ $S = \pi r^2$

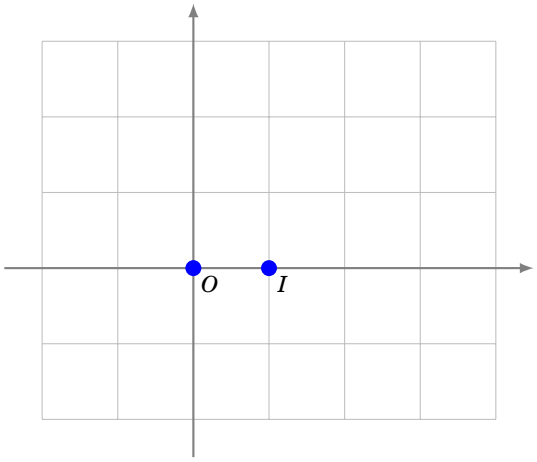
Cela revient à construire un segment de longueur $\sqrt{\pi}$ à la règle et au compas, à partir d'un segment de longueur 1.

2. Les nombres constructibles à la règle et au compas

Pour résoudre les trois problèmes grecs, il va falloir les transformer complètement. D'une question géométrique nous allons passer à une question algébrique. Dans cette partie on ramène le problème de la construction de points dans le plan à la construction de points sur la droite numérique réelle.

2.1. Nombre constructible

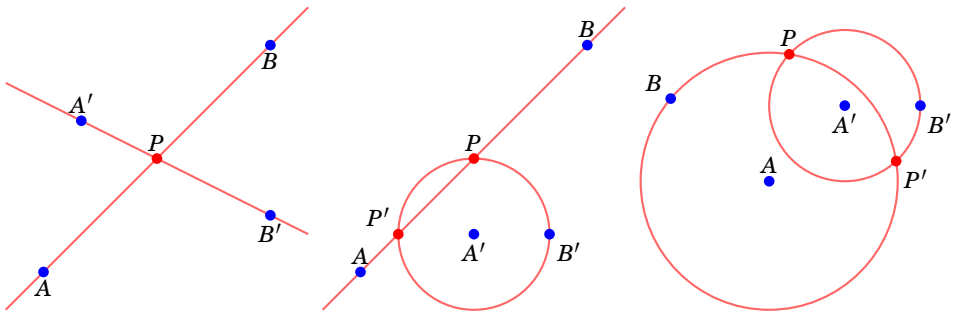
On considère le plan euclidien $\mathscr{P}$ muni d'un repère orthonormé, que l'on identifiera à $\mathbb{R}^2$ (ou $\mathbb{C}$). On définit des ensembles de points $\mathscr{C}_i \subset \mathscr{P}$ par récurrence.



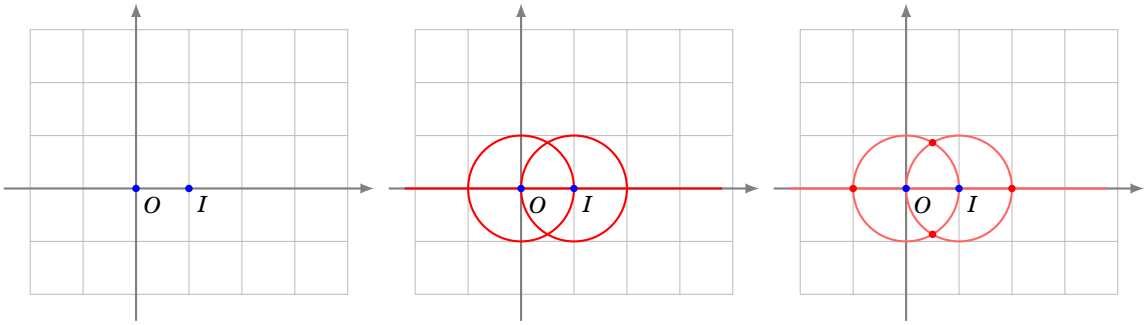
- On se donne au départ seulement deux points : $\mathscr{C}_0 = \{O, I\}$ où $O = (0,0)$ et $I = (1,0)$.
- Fixons $i \geqslant 0$, et supposons qu'un certain ensemble de points $\mathscr{C}_i$ soit déjà construit. Alors on définit $\mathscr{C}_{i+1}$ par récurrence, comme l'ensemble des ***points élémentairement constructibles*** à partir de $\mathscr{C}_i$. C'est-à-dire : $P \in \mathscr{C}_{i+1}$ si et seulement si

0. $P \in \mathscr{C}_i$
1. ou $P \in (AB) \cap (A'B')$ avec $A, B, A', B' \in \mathscr{C}_i$,
2. ou $P \in (AB) \cap \mathscr{C}(A', A'B')$ avec $A, B, A', B' \in \mathscr{C}_i$,
3. ou $P \in \mathscr{C}(A, AB) \cap \mathscr{C}(A', A'B')$ avec $A, B, A', B' \in \mathscr{C}_i$.

On a noté $\mathscr{C}(A, r)$ le cercle de centre A et de rayon r.
Il faut comprendre cette construction ainsi : si A, B, A', B' ont été construits et sont dans $\mathscr{C}_i$ alors, à partir de ces points, on peut tracer plusieurs objets à la règle et au compas : par exemple la droite (AB) – à l'aide de la règle – ou le cercle de centre A' et de rayon de longueur $A'B'$ en plaçant la pointe du compas en A' avec un écartement faisant passer le cercle par B'. Si cette droite (AB) et ce cercle $\mathscr{C}(A', A'B')$ s'intersectent alors les points d'intersection sont par définition dans $\mathscr{C}_{i+1}$. Voici les trois situations possibles. Les points A, B, A', B' en bleu sont dans $\mathscr{C}_i$, et les points P en rouge sont dans $\mathscr{C}_{i+1}$.



Voici la première étape. Partant de $\mathscr{C}_0$ (en bleu à gauche), on peut tracer une droite et deux cercles (au milieu), ce qui donne pour $\mathscr{C}_1$ quatre points supplémentaires (en rouge à droite).

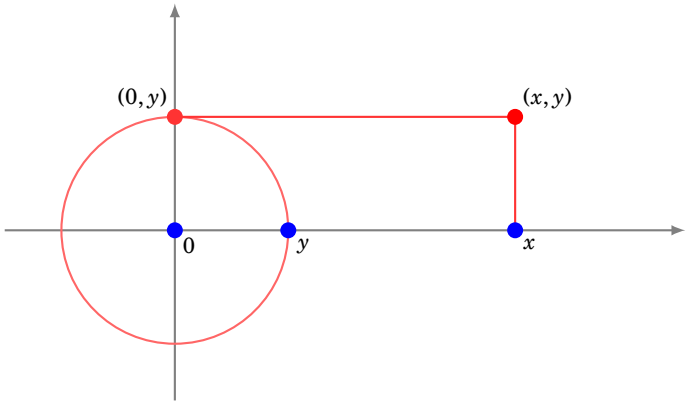


Pour $\mathscr{C}_2$ on repartirait de tous les points (rouges ou bleus) de $\mathscr{C}_1$, et on tracerait tous les cercles ou droites possibles (il y en a beaucoup !), et les points d'intersection formeraient l'ensemble $\mathscr{C}_2$.

Définition 2

- $\mathscr{C} = \bigcup_{i \geqslant 0} \mathscr{C}_i$ est l'ensemble des ***points constructibles***. Autrement dit $\mathscr{C} = \mathscr{C}_0 \cup \mathscr{C}_1 \cup \mathscr{C}_2 \cup \cdots$ De plus $P \in \mathscr{C}$ si et seulement s'il existe $i \geqslant 0$ tel que $P \in \mathscr{C}_i$.
- $\mathscr{C}_\mathbb{R} \subset \mathbb{R}$ est l'ensemble des abscisses des points constructibles : ce sont les ***nombres (réels) constructibles***.
- $\mathscr{C}_\mathbb{C} \subset \mathbb{C}$ est l'ensemble des affixes des points constructibles : ce sont les ***nombres complexes constructibles***.

Attention ! Même si deux points A, B sont constructibles et que l'on peut tracer la droite (AB), pour autant les points de (AB) ne sont pas tous constructibles. Seuls les points d'intersection de (AB) avec d'autres objets construits sont constructibles.
Déterminer les points constructibles $\mathscr{C}$ ou déterminer les nombres constructibles $\mathscr{C}_\mathbb{R}$ sont deux problèmes équivalents. En effet, si (x, y) est un point constructible alors par projection sur l'axe des abscisses nous obtenons le réel constructible x, et de même pour y projection sur l'axe des ordonnées, puis report sur l'axe des abscisses. Réciproquement on peut passer de deux nombres constructibles $x, y \in \mathbb{R}$ à un point constructible (x, y) dans le plan. Voici comment : partant du point $(y, 0)$ on construit $(0, y)$ sur l'axe des ordonnées par un coup de compas en reportant y. Une fois que $(x, 0)$ et $(0, y)$ sont construits, il est facile de construire (x, y).



2.2. Premières constructions algébriques

Proposition 14

Si x, x' sont des réels constructibles alors :

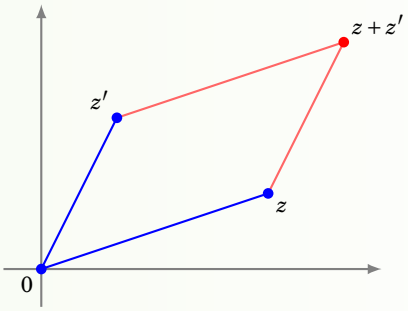
1. $x + x'$ est constructible,
2. $-x$ est constructible,
3. $x \cdot x'$ est constructible.
4. Si $x' \neq 0$, alors x/x' est constructible.

Tous ces résultats sont valables si l'on remplace x, x' par des nombres complexes z, z'.

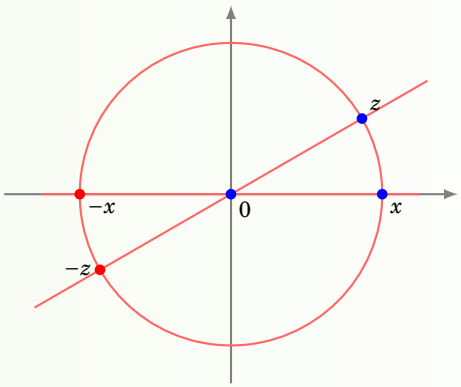
Démonstration

1. La construction pour le réel $x + x'$ est facile en utilisant le report du compas (on reporte la longueur x' à partir de x). Une autre méthode est de construire d'abord le milieu $\frac{x+x'}{2}$ puis le symétrique de 0 par rapport à ce milieu : c'est $x + x'$.

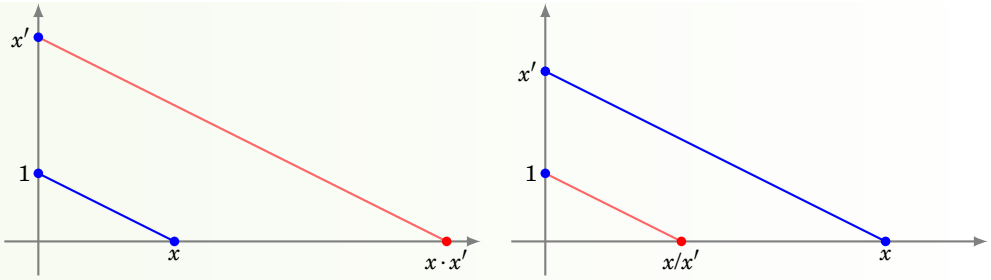
   La somme de deux nombres complexes $z + z'$ correspond à la construction d'un parallélogramme de sommets $0, z, z', z + z'$: les points d'affixes 0, z, z' étant supposés constructibles, on construit un parallélogramme de sorte que $z + z'$ soit le quatrième sommet.



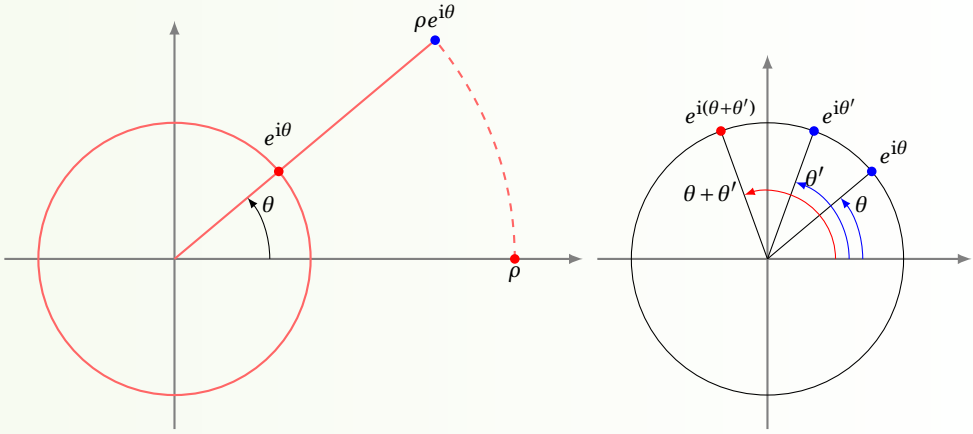
2. L'opposé du réel x (resp. du complexe z) s'obtient comme symétrique par rapport à l'origine : tracez la droite passant par 0 et x (resp. z) ; tracez le cercle de centre 0 passant par x (resp. z) ; ce cercle recoupe la droite en $-x$ (resp. $-z$).



3. Commençons par le produit de deux nombres réels $x \cdot x'$. On suppose construits les points $(x, 0)$ et $(0, x')$ (dessin de gauche). On trace la droite $\mathscr{D}$ passant par $(x, 0)$ et $(0, 1)$. On construit ensuite – à la règle et au compas – la droite $\mathscr{D}'$ parallèle à $\mathscr{D}$ et passant par $(0, x')$. Le théorème de Thalès prouve que $\mathscr{D}'$ recoupe l'axe des abscisses en $(x \cdot x', 0)$.



4. Pour le quotient la méthode est similaire (dessin de droite).
5. Il reste à s'occuper du produit et du quotient de deux nombres complexes. Tout d'abord, si $z = \rho e^{i\theta}$ est un nombre complexe constructible, alors ρ est constructible (considérer le cercle centré à l'origine qui passe par z ; il recoupe l'axe des abscisses en $(\rho, 0)$). Le nombre $e^{i\theta}$ est aussi constructible : c'est l'intersection de la droite passant par l'origine et z avec le cercle unité. Réciproquement avec ρ et $e^{i\theta}$ on construit facilement $z = \rho e^{i\theta}$.



Maintenant si $z = \rho e^{i\theta}$ et $z' = \rho' e^{i\theta'}$ alors $z \cdot z' = (\rho \cdot \rho') e^{i(\theta+\theta')}$. Le réel $\rho \cdot \rho'$ est constructible comme nous l'avons vu au-dessus. Il reste à construire le nombre complexe $e^{i(\theta+\theta')}$, qui correspond à la somme de deux angles θ et θ'. Cela se fait simplement, à partir du cercle unité, en reportant au compas la mesure d'un angle à partir de l'extrémité de l'autre.

Pour le quotient la méthode est similaire.

Corollaire 2

$$\mathbb{N} \subset \mathscr{C}_{\mathbb{R}} \qquad \mathbb{Z} \subset \mathscr{C}_{\mathbb{R}} \qquad \mathbb{Q} \subset \mathscr{C}_{\mathbb{R}}$$

Autrement dit, tous les rationnels (et en particulier tous les entiers) sont des nombres réels constructibles.

La preuve découle facilement de la proposition :

Démonstration

- Puisque 1 est un nombre constructible alors $2 = 1 + 1$ est constructible, mais alors $3 = 2 + 1$ est constructible et par récurrence tout entier $n \geqslant 0$ est un élément de $\mathscr{C}_{\mathbb{R}}$.
- Comme tout entier $n \geqslant 0$ est constructible alors $-n$ l'est aussi ; donc tous les entiers $n \in \mathbb{Z}$ sont constructibles.
- Enfin pour $\frac{p}{q} \in \mathbb{Q}$, comme les entiers p, q sont constructibles, alors le quotient $\frac{p}{q}$ est constructible et ainsi $\mathbb{Q} \subset \mathscr{C}_{\mathbb{R}}$.

Nous allons voir que $\mathscr{C}_{\mathbb{R}}$ contient davantage de nombres que les rationnels.

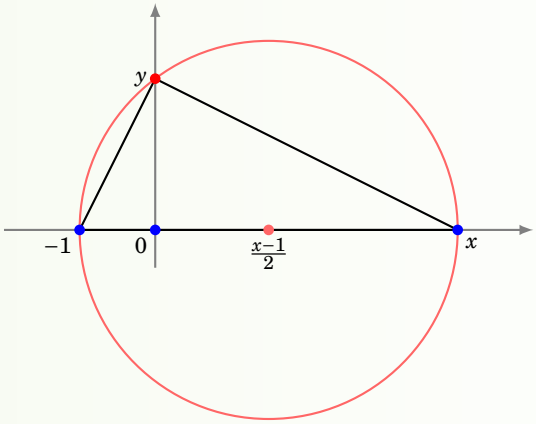
Proposition 15

Si $x \geqslant 0$ est un nombre constructible, alors $\sqrt{x}$ est constructible.

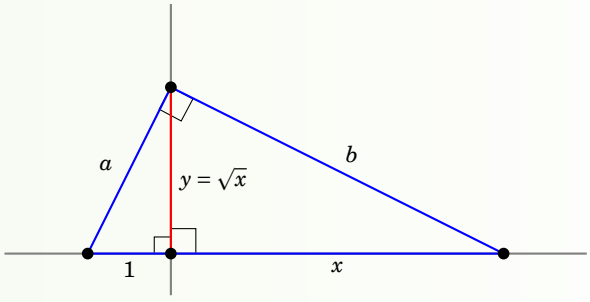
Remarques :

1. La réciproque est vraie. En effet si $x' = \sqrt{x}$ est un nombre constructible, alors par la proposition 14 : $x' \cdot x'$ est constructible. Or $x' \cdot x' = \sqrt{x} \cdot \sqrt{x} = x$, donc x est constructible.
2. On peut en déduire aussi que si $z \in \mathbb{C}$ est constructible alors les racines carrées (complexes) de z sont constructibles. On utilise pour cela la racine carrée du module et la bissection de l'argument comme on l'a vue au paragraphe 1.5.
3. En particulier $\sqrt{2}$, $\sqrt{3}$, ... sont des nombres constructibles (comme on l'avait vu en première partie).

Démonstration



Soient les nombres constructibles $0, -1, x$ placés sur l'axe des abscisses. Traçons le cercle dont le diamètre est $[-1, x]$ (cela revient à construire le centre du cercle $\frac{x-1}{2}$; voir la proposition 14). Ce cercle recoupe l'axe des ordonnées en $y \geqslant 0$.



On applique le théorème de Pythagore dans trois triangles rectangles, pour obtenir :

$$\begin{cases} a^2 + b^2 &= (1+x)^2 \\ 1 + y^2 &= a^2 \\ x^2 + y^2 &= b^2. \end{cases}$$

On en déduit $a^2 + b^2 = (1+x)^2 = 1 + x^2 + 2x$ d'une part et $a^2 + b^2 = 1 + x^2 + 2y^2$ d'autre part. Ainsi $1 + x^2 + 2x = 1 + x^2 + 2y^2$ d'où $y^2 = x$. Comme $y \geqslant 0$ alors $y = \sqrt{x}$.
Une autre méthode consiste à remarquer que le triangle de sommets $(0,0), (-1,0), (0,y)$ et le triangle de sommets $(0,0), (x,0), (0,y)$ sont semblables donc $\frac{x}{y} = \frac{y}{1}$, d'où $x = y^2$, donc $y = \sqrt{x}$.

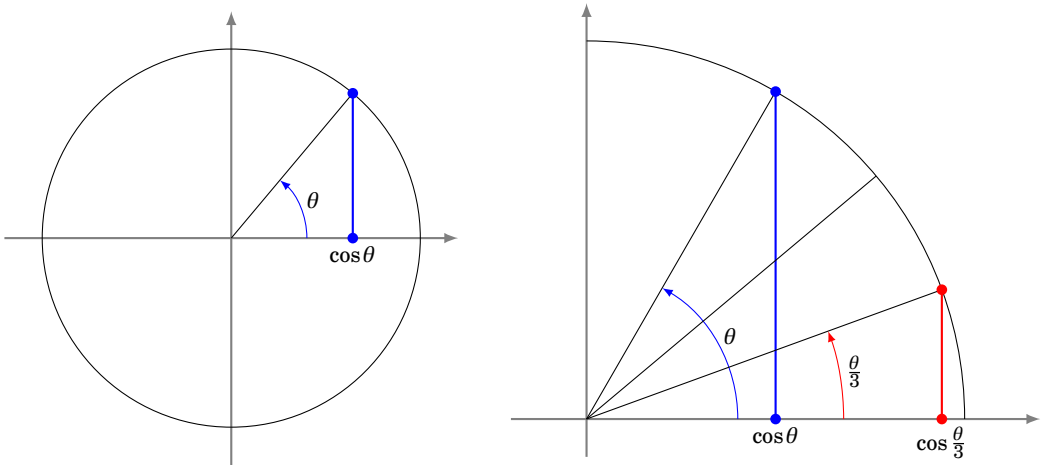
2.3. Retour sur les trois problèmes grecs

Avec le langage des nombres constructibles les problèmes historiques s'énoncent ainsi :

- ***La duplication du cube.*** Est-ce que $\sqrt[3]{2}$ est un nombre constructible ?
- ***La quadrature du cercle.*** Est-ce que π est un nombre constructible ?
- ***La trisection des angles.*** Étant donné un réel constructible $\cos\theta$, est-ce que $\cos\frac{\theta}{3}$ est aussi constructible ?

Si vous n'êtes pas convaincu voici les preuves :

- Si on a un cube de volume a^3, alors il faut construire un cube de volume $2a^3$. Fixons a un réel constructible. Que $\sqrt[3]{2}$ soit aussi constructible équivaut à $a\sqrt[3]{2}$ constructible. Un segment de longueur $a\sqrt[3]{2}$ définit bien un cube de volume $2a^3$. On aurait résolu la duplication du cube.
- Soit construit un cercle de rayon r, donc d'aire πr^2. Que π soit constructible équivaut à $\sqrt{\pi}$ constructible. Construire un segment de longueur $\sqrt{\pi}r$, correspond à un carré d'aire πr^2, donc de même aire que le cercle initial. Nous aurions construit un carré de même aire que le cercle ! On aurait résolu la quadrature du cercle.
- Remarquons que construire un angle géométrique de mesure θ est équivalent à construire le nombre réel $\cos\theta$ (voir la figure de gauche). Partons d'un angle géométrique θ, c'est-à-dire partons d'un réel $\cos\theta$ constructible. Construire $\cos\frac{\theta}{3}$ est équivalent à construire un angle géométrique de mesure $\frac{\theta}{3}$. On aurait résolu la trisection des angles.



2.4. Les ensembles

Une dernière motivation à propos des nombres constructibles concerne les ensembles. Nous avons les inclusions d'ensembles :

$$\mathbb{N}\subset\mathbb{Z}\subset\mathbb{Q}\subset\mathbb{R}\subset\mathbb{C}.$$

Le passage d'un ensemble à un ensemble plus grand se justifie par la volonté de résoudre davantage d'équations :

- passage de $\mathbb{N}$ à $\mathbb{Z}$, pour résoudre des équations du type $x+7=0$,
- passage de $\mathbb{Z}$ à $\mathbb{Q}$, pour résoudre des équations du type $5x=4$,
- passage de $\mathbb{Q}$ à $\mathbb{R}$, pour résoudre des équations du type $x^2=2$,
- passage de $\mathbb{R}$ à $\mathbb{C}$, pour résoudre des équations du type $x^2=-1$.

Mais en fait le passage de $\mathbb{Q}$ à $\mathbb{R}$ est un saut beaucoup plus « grand » que les autres : $\mathbb{Q}$ est un ensemble dénombrable (il existe une bijection entre $\mathbb{Z}$ et $\mathbb{Q}$) alors que $\mathbb{R}$ ne l'est pas.
Nous allons définir et étudier deux ensembles intermédiaires :

$$\mathbb{Q}\subset\mathscr{C}_{\mathbb{R}}\subset\overline{\mathbb{Q}}\subset\mathbb{R}$$

où

- $\mathscr{C}_{\mathbb{R}}$ est l'ensemble des nombres réels constructibles à la règle et au compas,
- $\overline{\mathbb{Q}}$ est l'ensemble des nombres algébriques : ce sont les réels x qui sont solutions d'une équation $P(x)=0$, pour un polynôme P à coefficients dans $\mathbb{Q}$.

3. Éléments de théorie des corps

La théorie des corps n'est pas évidente et mériterait un chapitre entier. Nous résumons ici les grandes lignes utiles à nos fins. Il est important de bien comprendre le paragraphe suivant ; les autres paragraphes peuvent être sautés lors de la première lecture.

3.1. Les exemples à comprendre

Premier exemple. Soit l'ensemble

$$\mathbb{Q}(\sqrt{2}) = \left\{a+b\sqrt{2} \mid a,b \in \mathbb{Q}\right\}.$$

C'est un sous-ensemble de $\mathbb{R}$, qui contient par exemple 0, 1, $\frac{1}{3}$ et tous les éléments de $\mathbb{Q}$, mais aussi $\sqrt{2}$ (qui n'est pas rationnel !), $\frac{1}{2}-\frac{2}{3}\sqrt{2}$.
Voici quelques propriétés :

- Soient $a+b\sqrt{2}$ et $a'+b'\sqrt{2}$ deux éléments de $\mathbb{Q}(\sqrt{2})$. Alors leur somme $(a+b\sqrt{2})+(a'+b'\sqrt{2})$ est encore un élément de $\mathbb{Q}(\sqrt{2})$. De même $-(a+b\sqrt{2}) \in \mathbb{Q}(\sqrt{2})$.
- Plus surprenant, si $a+b\sqrt{2}, a'+b'\sqrt{2} \in \mathbb{Q}(\sqrt{2})$ alors $(a+b\sqrt{2})\times(a'+b'\sqrt{2}) = aa'+2bb'+(ab'+a'b)\sqrt{2}$ est aussi un élément de $\mathbb{Q}(\sqrt{2})$. Enfin l'inverse d'un élément non nul $a+b\sqrt{2}$ est $\frac{1}{a+b\sqrt{2}} = \frac{1}{a^2-2b^2}(a-b\sqrt{2})$: c'est encore un élément de $\mathbb{Q}(\sqrt{2})$.

Ces propriétés font de $\mathbb{Q}(\sqrt{2})$ un ***corps***. Comme ce corps contient $\mathbb{Q}$ on parle d'une ***extension*** de $\mathbb{Q}$. De plus, il est étendu avec un élément du type $\sqrt{\delta}$: on parle alors d'une ***extension quadratique***. Notez que, même si $\delta \in \mathbb{Q}$, $\sqrt{\delta}$ n'est généralement pas un élément de $\mathbb{Q}$.

Deuxième série d'exemples. On peut généraliser l'exemple précédent : si K est lui-même un corps et δ est un élément de K alors

$$K(\sqrt{\delta}) = \left\{a+b\sqrt{\delta} \mid a,b \in K\right\}$$

est un corps. On vérifie comme ci-dessus que la somme et le produit de deux éléments restent dans $K(\sqrt{\delta})$, ainsi que l'opposé et l'inverse.
Cela permet de construire de nouveaux corps : partant de $K_0=\mathbb{Q}$, on choisit un élément, disons $\delta_0=2$ et on obtient le corps plus gros $K_1=\mathbb{Q}(\sqrt{2})$. Si on prend $\delta_1=3$ alors $\sqrt{3} \notin \mathbb{Q}(\sqrt{2})$ et donc $K_2=K_1(\sqrt{3})$ est un nouveau corps (qui contient K_1). Le corps K_2 est :

$$K_2 = K_1(\sqrt{3}) = \mathbb{Q}(\sqrt{2})(\sqrt{3}) = \left\{a+b\sqrt{2}+c\sqrt{3}+d\sqrt{2}\sqrt{3} \mid a,b,c,d \in \mathbb{Q}\right\}.$$

On pourrait continuer avec $\delta_2=11$ et exprimer chaque élément de $\mathbb{Q}(\sqrt{2})(\sqrt{3})(\sqrt{11})$ comme une somme de 8 éléments $a_1+a_2\sqrt{2}+a_3\sqrt{3}+a_4\sqrt{11}+a_5\sqrt{2}\sqrt{3}+a_6\sqrt{2}\sqrt{11}+a_7\sqrt{3}\sqrt{11}+a_8\sqrt{2}\sqrt{3}\sqrt{11}$ avec les $a_i \in \mathbb{Q}$.
En partant de $K_1=\mathbb{Q}(\sqrt{2})$, on aurait pu considérer $\delta_1=1+\sqrt{2}$ et $K_2=K_1(\sqrt{1+\sqrt{2}})=\mathbb{Q}(\sqrt{2})(\sqrt{1+\sqrt{2}})$. Chaque élément de K_2 peut s'écrire comme une somme de 4 éléments $a+b\sqrt{2}+c\sqrt{1+\sqrt{2}}+d\sqrt{2}\sqrt{1+\sqrt{2}}$.

Une propriété. Il faut noter que chaque élément de $\mathbb{Q}(\sqrt{2})$ est racine d'un polynôme de degré au plus 2 à coefficients dans $\mathbb{Q}$. Par exemple $3+\sqrt{2}$ est annulé par $P(X)=(X-3)^2-2=X^2-6X+7$. Les

nombres qui sont annulés par un polynôme à coefficients rationnels sont les ***nombres algébriques***. Plus généralement, si K est un corps et $\delta \in K$, alors tout élément de $K(\sqrt{\delta})$ est annulé par un polynôme de degré 1 ou 2 à coefficients dans K. On en déduit que chaque élément de $\mathbb{Q}(\sqrt{2})(\sqrt{3})$ (ou de $\mathbb{Q}(\sqrt{2})(\sqrt{1+\sqrt{2}})$) est racine d'un polynôme de $\mathbb{Q}[X]$ de degré 1, 2 ou 4. Et chaque élément de $\mathbb{Q}(\sqrt{2})(\sqrt{3})(\sqrt{11})$ est racine d'un polynôme de $\mathbb{Q}[X]$ de degré 1, 2, 4 ou 8, etc.

Nous allons maintenant reprendre ces exemples d'une manière plus théorique.

3.2. Corps

Un corps est un ensemble sur lequel sont définies deux opérations : une addition et une multiplication.

Définition 3

Un ***corps*** $(K,+,\times)$ est un ensemble K muni des deux opérations $+$ et $\times$, qui vérifient :

0. $+$ et $\times$ sont des lois de composition interne, c'est à dire $x+y \in K$ et $x \times y \in K$ (pour tout $x,y \in K$).
1. $(K,+)$ est un groupe commutatif, c'est-à-dire :
   - Il existe $0 \in K$ tel que $0+x=x$ (pour tout $x \in K$).
   - Pour tout $x \in K$ il existe $-x$ tel que $x+(-x)=0$.
   - $+$ est associative : $(x+y)+z=x+(y+z)$ (pour tout $x,y,z \in K$).
   - $x+y=y+x$ (pour tout $x,y \in K$).
2. $(K \setminus \{0\},\times)$ est un groupe commutatif, c'est-à-dire :
   - Il existe $1 \in K \setminus \{0\}$ tel que $1 \times x = x$ (pour tout $x \in K$).
   - Pour tout $x \in K \setminus \{0\}$, il existe x^{-1} tel que $x \times x^{-1} = 1$.
   - $\times$ est associative : $(x \times y) \times z = x \times (y \times z)$ (pour tout $x,y,z \in K \setminus \{0\}$).
   - $x \times y = y \times x$ (pour tout $x,y \in K \setminus \{0\}$).
3. $\times$ est distributive par rapport à $+$: $(x+y) \times z = x \times z + y \times z$ (pour tout $x,y,z \in K$).

Voici des exemples classiques :

- $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ sont des corps. L'addition et la multiplication sont les opérations usuelles.
- Par contre $(\mathbb{Z},+,\times)$ n'est pas un corps. (Pourquoi ?)

Voici des exemples qui vont être importants pour la suite :

- $\mathbb{Q}(\sqrt{2}) = \{a+b\sqrt{2} \mid a,b \in \mathbb{Q}\}$ est un corps (avec l'addition et la multiplication habituelles des nombres réels). Voir les exemples introductifs.
- $\mathbb{Q}(\mathrm{i}) = \{a+\mathrm{i}b \mid a,b \in \mathbb{Q}\}$ est un corps (avec l'addition et la multiplication habituelles des nombres complexes).
- Par contre $\{a+b\pi \mid a,b \in \mathbb{Q}\}$ n'est pas un corps (où $\pi = 3,14\ldots$). (C'est une conséquence du fait que π n'est pas un nombre algébrique comme on le verra plus loin.)

La proposition 14 de la première partie se reformule avec la notion de corps en :

Proposition 16

L'ensemble des nombre réels constructibles $(\mathscr{C}_{\mathbb{R}}, +, \times)$ est un corps inclus dans $\mathbb{R}$.

On a aussi que $(\mathscr{C}_{\mathbb{C}}, +, \times)$ est un corps inclus dans $\mathbb{C}$.

3.3. Extension de corps

Nous cherchons des propositions qui lient deux corps, lorsque l'un est inclus dans l'autre. Les résultats de ce paragraphe seront admis.

Proposition 17

Soient K, L deux corps avec $K \subset L$. Alors L est un espace vectoriel sur K.

Définition 4

L est appelé une ***extension*** de K. Si la dimension de cet espace vectoriel est finie, alors on l'appelle le ***degré*** de l'extension, et on notera :

$$[L : K] = \dim_K L.$$

Si ce degré vaut 2, nous parlerons d'une ***extension quadratique***.

Proposition 18

Si K, L, M sont trois corps avec $K \subset L \subset M$ et si les extensions ont un degré fini alors :

$$[M : K] = [M : L] \times [L : K].$$

Exemple 4

- $\mathbb{Q}(\sqrt{2})$ est une extension de $\mathbb{Q}$. De plus, comme $\mathbb{Q}(\sqrt{2}) = \{a + b\sqrt{2} \mid a, b \in \mathbb{Q}\}$, alors $\mathbb{Q}(\sqrt{2})$ est un espace vectoriel (sur $\mathbb{Q}$) de dimension 2 : en effet $(1, \sqrt{2})$ en est une base. Attention : ici 1 est un vecteur et $\sqrt{2}$ est un autre vecteur. Le fait que $\sqrt{2} \notin \mathbb{Q}$ se traduit en : ces deux vecteurs sont linéairement indépendants sur $\mathbb{Q}$. C'est un peu déroutant au début !
- $\mathbb{C}$ est une extension de degré 2 de $\mathbb{R}$ car tout élément de $\mathbb{C}$ s'écrit $a + ib$. Donc les vecteurs 1 et i forment une base de $\mathbb{C}$, vu comme un espace vectoriel sur $\mathbb{R}$.
- Notons $\mathbb{Q}(\sqrt{2}, \sqrt{3}) = \mathbb{Q}(\sqrt{2})(\sqrt{3}) = \{a + b\sqrt{3} \mid a, b \in \mathbb{Q}(\sqrt{2})\}$. Alors $\mathbb{Q} \subset \mathbb{Q}(\sqrt{2}) \subset \mathbb{Q}(\sqrt{2}, \sqrt{3})$. Calculer le degré des extensions. Expliciter une base sur $\mathbb{Q}$ de $\mathbb{Q}(\sqrt{2}, \sqrt{3})$.

Pour $x \in \mathbb{R}$, on note $\mathbb{Q}(x)$ le plus petit corps contenant $\mathbb{Q}$ et x : c'est le ***corps engendré*** par x. C'est cohérent avec la notation pour les extensions quadratiques $\mathbb{Q}(\sqrt{\delta})$, qui est bien le plus petit corps contenant $\sqrt{\delta}$.
Par exemple, si $x = \sqrt[3]{2} = 2^{\frac{1}{3}}$, alors il n'est pas dur de calculer que

$$\mathbb{Q}(\sqrt[3]{2}) = \left\{a + b\sqrt[3]{2} + c\sqrt[3]{2}^2 \mid a, b, c \in \mathbb{Q}\right\}.$$

En effet $\mathbb{Q}(\sqrt[3]{2})$ contient $x, x^2, x^3, \ldots$ mais aussi $\frac{1}{x}, \frac{1}{x^2}, \ldots$ Mais comme $x^3 = 2 \in \mathbb{Q}$ et $\frac{1}{x} = \frac{x^2}{2}$, alors $a + bx + cx^2$, avec $a, b, c \in \mathbb{Q}$, engendrent tous les éléments de $\mathbb{Q}(x)$. Conclusion : $[\mathbb{Q}(\sqrt[3]{2}) : \mathbb{Q}] = 3$.

3.4. Nombre algébrique

L'ensemble des ***nombres algébriques*** est

$$\overline{\mathbb{Q}} = \{x \in \mathbb{R} \mid \text{ il existe } P \in \mathbb{Q}[X] \text{ non nul tel que } P(x) = 0\}.$$

Proposition 19

$\overline{\mathbb{Q}}$ est un corps.

Démonstration

L'addition et la multiplication définies sur $\overline{\mathbb{Q}}$ sont celles du corps $(\mathbb{R}, +, \times)$. Ainsi beaucoup de propriétés découlent du fait que l'ensemble des réels est un corps (on parle de sous-corps).
La première chose que l'on doit démontrer, c'est que $+$ et $\times$ sont des lois de composition interne, c'est-à-dire que si x et y sont des nombres réels algébriques alors $x+y$ et $x \times y$ le sont aussi. Ce sera prouvé dans le corollaire 3.

1. $(\overline{\mathbb{Q}}, +)$ est un groupe commutatif, car :
   - $0 \in \overline{\mathbb{Q}}$ (prendre $P(X) = X$) et $0 + x = x$ (pour tout $x \in \overline{\mathbb{Q}}$).
   - Si $x \in \overline{\mathbb{Q}}$ alors $-x \in \overline{\mathbb{Q}}$ (si $P(X)$ est un polynôme qui annule x alors $P(-X)$ annule $-x$).
   - $+$ est associative : cela découle de l'associativité sur $\mathbb{R}$.
   - $x + y = y + x$: idem.
2. $(\overline{\mathbb{Q}} \setminus \{0\}, \times)$ est un groupe commutatif, car :
   - $1 \in \overline{\mathbb{Q}} \setminus \{0\}$ et $1 \times x = x$ (pour tout $x \in \overline{\mathbb{Q}} \setminus \{0\}$).
   - Si $x \in \overline{\mathbb{Q}} \setminus \{0\}$ alors $x^{-1} \in \overline{\mathbb{Q}} \setminus \{0\}$: en effet, si $P(X)$ est un polynôme de degré n annulant x, alors $X^n P(\frac{1}{X})$ est un polynôme annulant $\frac{1}{x}$.
   - $\times$ est associative : cela découle de l'associativité sur $\mathbb{R} \setminus \{0\}$.
   - $x \times y = y \times x$: idem.
3. $\times$ est distributive par rapport à $+$: cela découle de la distributivité sur $\mathbb{R}$.

Si $x \in \overline{\mathbb{Q}}$ est un nombre algébrique, alors le plus petit degré, parmi tous les degrés des polynômes $P \in \mathbb{Q}[X]$ tels que $P(x) = 0$, est le ***degré algébrique*** de x. Par exemple, calculons le degré algébrique de $\sqrt{2}$: un polynôme annulant ce nombre est $P(X) = X^2 - 2$ et il n'est pas possible d'en trouver de degré 1, donc le degré algébrique de $\sqrt{2}$ vaut 2. Plus généralement $\sqrt{\delta}$ avec $\delta \in \mathbb{Q}$ est de degré algébrique égal à 1 ou 2 (de degré algébrique 1 si $\sqrt{\delta} \in \mathbb{Q}$, de degré 2 sinon). Par contre $\sqrt[3]{2}$ est de degré 3, car il est annulé par $P(X) = X^3 - 2$ mais pas par des polynômes de degré plus petit.

Proposition 20

1. Soit L une extension finie du corps $\mathbb{Q}$. Si $x \in L$, alors x est un nombre algébrique.
2. Si x un nombre algébrique alors $\mathbb{Q}(x)$ est une extension finie de $\mathbb{Q}$.
3. Si x est un nombre algébrique alors le degré de l'extension $[\mathbb{Q}(x) : \mathbb{Q}]$ et le degré algébrique de x coïncident.

Démonstration

1. Soit L une extension finie de $\mathbb{Q}$, et soit $n = [L : \mathbb{Q}]$. Fixons $x \in L$. Les $n+1$ éléments $(1, x, x^2, \ldots, x^n)$ forment une famille de $n+1$ vecteurs dans un espace vectoriel de dimension n. Donc cette famille est liée. Il existe donc une combinaison linéaire nulle non triviale, c'est-à-dire il existe $a_i \in \mathbb{Q}$ non tous nuls tels que $\sum_{i=0}^{n} a_i x^i = 0$. Si l'on définit $P(X) = \sum_{i=0}^{n} a_i X^i$, alors $P(X) \in \mathbb{Q}[X]$, $P(X)$ n'est pas le polynôme nul et $P(x) = 0$. C'est exactement dire que x est un nombre algébrique.

2. Soit $P(X)=\sum_{i=0}^{n} a_i X^i$ non nul qui vérifie $P(x)=0$. En écartant le le cas trivial $x=0$, on peut donc supposer que $a_0 \neq 0$ et $a_n \neq 0$. Alors $x^n = -\frac{1}{a_n}\sum_{i=0}^{n-1} a_i x^i$ et $\frac{1}{x} = \frac{1}{a_0}\sum_{i=1}^{n} a_i x^{i-1}$. Ce qui prouve que $x^n \in \text{Vect}(1,x,\dots,x^{n-1})$ et $\frac{1}{x} \in \text{Vect}(1,x,\dots,x^{n-1})$. De même pour tout $k \in \mathbb{Z}$, $x^k \in \text{Vect}(1,x,\dots,x^{n-1})$, donc $\mathbb{Q}(x) \subset \text{Vect}(1,x,\dots,x^{n-1})$. Ce qui prouve que $\mathbb{Q}(x)$ est un espace vectoriel de dimension finie sur $\mathbb{Q}$.
3. Ce sont à peu près les mêmes arguments. Si $m=[\mathbb{Q}(x):\mathbb{Q}]$ alors il existe $a_i \in \mathbb{Q}$ non tous nuls tels que $\sum_{i=0}^{m} a_i x^i = 0$. Donc il existe un polynôme non nul de degré m annulant x. Donc le degré algébrique de x est inférieur ou égal à m.
   Mais s'il existait un polynôme $P(X)=\sum_{i=0}^{m-1} b_i X^i$ non nul de degré strictement inférieur à m qui annulait x, alors nous aurions une combinaison linéaire nulle non triviale $\sum_{i=0}^{m-1} b_i x^i = 0$. Cela impliquerait que $x^{m-1} \in \text{Vect}(1,x,\dots,x^{m-2})$ et plus généralement que $\mathbb{Q}(x) \subset \text{Vect}(1,x,\dots,x^{m-2})$, ce qui contredirait le fait que $\mathbb{Q}(x)$ soit un espace vectoriel de dimension m sur $\mathbb{Q}$.
   Bilan : le degré algébrique de x est exactement $[\mathbb{Q}(x):\mathbb{Q}]$.

Corollaire 3

Si x et y sont des nombres réels algébriques alors $x+y$ et $x\times y$ aussi.

Démonstration

Comme x est un nombre algébrique alors $L=\mathbb{Q}(x)$ est une extension finie de $K=\mathbb{Q}$. Posons $M=\mathbb{Q}(x,y)=\big(\mathbb{Q}(x)\big)(y)$. Comme y est un nombre algébrique alors M est une extension finie de $\mathbb{Q}(x)$. Par la proposition 18 $M=\mathbb{Q}(x,y)$ est une extension finie de $K=\mathbb{Q}$.
Comme $x+y \in \mathbb{Q}(x+y) \subset \mathbb{Q}(x,y)$ et que $\mathbb{Q}(x,y)$ est une extension finie de $\mathbb{Q}$ alors par la proposition 20, $x+y$ est un nombre algébrique.
C'est la même preuve pour $x\times y \in \mathbb{Q}(x\times y) \subset \mathbb{Q}(x,y)$.

4. Corps et nombres constructibles

Cette partie est la charnière de ce chapitre. Nous expliquons à quoi correspondent algébriquement les opérations géométriques effectuées à la règle et au compas.

4.1. Nombre constructible et extensions quadratiques

Voici le résultat théorique le plus important de ce chapitre. C'est Pierre-Laurent Wantzel qui a démontré ce théorème en 1837, à l'âge de 23 ans.

Théorème 5. Théorème de Wantzel

Un nombre réel x est constructible si et seulement s'il existe des extensions quadratiques

$$\mathbb{Q}=K_0 \subset K_1 \subset \cdots \subset K_r$$

telles que $x \in K_r$.

Chacune des extensions est quadratique, c'est-à-dire $[K_{i+1}:K_i]=2$. Autrement dit, chaque extension est une extension quadratique de la précédente : $K_{i+1}=K_i(\sqrt{\delta_i})$ pour un certain $\delta_i \in K_i$. Donc en partant de $K_0=\mathbb{Q}$, les extensions sont :

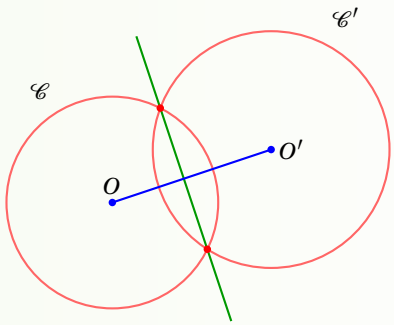
$$\mathbb{Q} \subset \mathbb{Q}(\sqrt{\delta_0}) \subset \mathbb{Q}(\sqrt{\delta_0})(\sqrt{\delta_1}) \subset \cdots$$

Démonstration

Il y a un sens facile : comme on sait construire les racines carrées des nombres constructibles (voir la proposition 15) alors on sait construire tout élément d'une extension quadratique $K_1 = \mathbb{Q}(\sqrt{\delta_0})$, puis par récurrence tout élément de $K_2, K_3, \ldots$

Passons au sens difficile. Rappelons-nous que les points constructibles sont construits par étapes $\mathscr{C}_0$, $\mathscr{C}_1$, $\mathscr{C}_2, \ldots$
L'ensemble $\mathscr{C}_{j+1}$ s'obtient à partir de $\mathscr{C}_j$ en ajoutant les intersections des droites et des cercles que l'on peut tracer à partir de $\mathscr{C}_j$. Nous allons voir que ce passage correspond à une extension quadratique.
Soit donc K le plus petit corps contenant les coordonnées des points de $\mathscr{C}_j$. Nous considérons P un point de $\mathscr{C}_{j+1}$. Ce point P est l'intersection de deux objets (deux droites ; une droite et un cercle ; deux cercles). Distinguons les cas :

- *P est l'intersection de deux droites.* Ces droites passent par des points de $\mathscr{C}_j$ donc elles ont pour équations $ax+by=c$ et $a'x+b'y=c'$ et il est important de noter que l'on peut prendre a,b,c,a',b',c' comme étant des éléments de K. Par exemple une équation de la droite passant par $A(x_A,y_A)$ et $B(x_B,y_B)$ (avec $x_A,y_A,x_B,y_B \in K$) est $y = \frac{y_B-y_A}{x_B-x_A}(x-x_A)+y_A$, ce qui donne bien une équation à coefficients dans K. Les coordonnées de P sont donc
$$\left(\frac{cb'-c'b}{ab'-a'b}, \frac{ac'-a'c}{ab'-a'b}\right).$$
Comme K est un corps alors l'abscisse et l'ordonnée de ce P sont encore dans K. Dans ce cas il n'y a pas besoin d'extension : le plus petit corps contenant les coordonnées des points de $\mathscr{C}_j$ et de P est K.
- *P appartient à l'intersection d'une droite et d'un cercle.* Notons l'équation de la droite $ax+by=c$ avec $a,b,c \in K$ et $(x-x_0)^2+(y-y_0)^2=r^2$ l'équation du cercle. On note que x_0, y_0, r^2 (mais pas nécessairement r) sont des éléments de K car les coordonnées du centre et d'un point du cercle sont dans K. Il reste à calculer les intersections de la droite et du cercle : en posant
$$\delta = -2x_0a^3by_0 + 2y_0a^2cb - b^2{y_0}^2a^2 + b^2r^2a^2 + 2a^3x_0c - a^4{x_0}^2 - a^2c^2 + a^4r^2 \in K,$$
on trouve deux points (x,y), (x',y') avec
$$x = -\frac{b}{a}\frac{1}{a^2+b^2}\left(-x_0ab + y_0a^2 + cb - \frac{c}{b}(a^2+b^2) + \sqrt{\delta}\right) \quad \text{et} \quad y = \frac{c-ax}{b},$$
$$x' = -\frac{b}{a}\frac{1}{a^2+b^2}\left(-x_0ab + y_0a^2 + cb - \frac{c}{b}(a^2+b^2) - \sqrt{\delta}\right) \quad \text{et} \quad y' = \frac{c-ax'}{b}.$$
Les coordonnées sont bien de la forme $\alpha+\beta\sqrt{\delta}$ avec $\alpha,\beta \in K$ et c'est le même $\delta \in K$ pour x,y,x',y'. Donc les coordonnées de P sont bien dans l'extension quadratique $K(\sqrt{\delta})$.
- *P appartient à l'intersection de deux cercles.* On trouve aussi deux points (x,y), (x',y') et x,y,x',y' sont aussi de la forme $\alpha+\beta\sqrt{\delta}$ pour un certain $\delta \in K$ fixé et $\alpha,\beta \in K$. Les formules sont plus longues à écrire et on se contentera ici de faire un exemple (voir juste après).
On pourrait donner un autre argument : l'intersection du cercle $\mathscr{C}$ centré en O et du cercle $\mathscr{C}'$ centré en O' est aussi l'intersection du cercle $\mathscr{C}$ avec la médiatrice de $[OO']$. (Exercice : justifier que cette médiatrice est constructible sans étendre le corps.) On se ramène donc au cas de l'intersection d'un cercle et d'une droite.

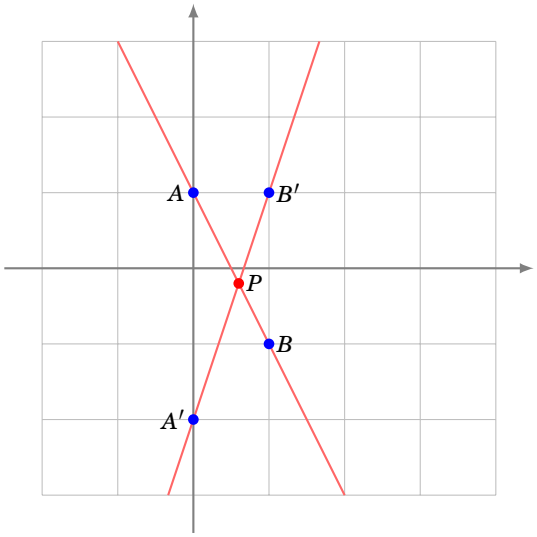


En résumé, dans tous les cas, les coordonnées de P sont dans une extension quadratique du corps K, qui contient les coefficients qui servent à construire P. Par récurrence, cela donne le résultat souhaité.

Exemple 5

Donnons les extensions nécessaires dans chacun des trois cas de la preuve sur un exemple concret.

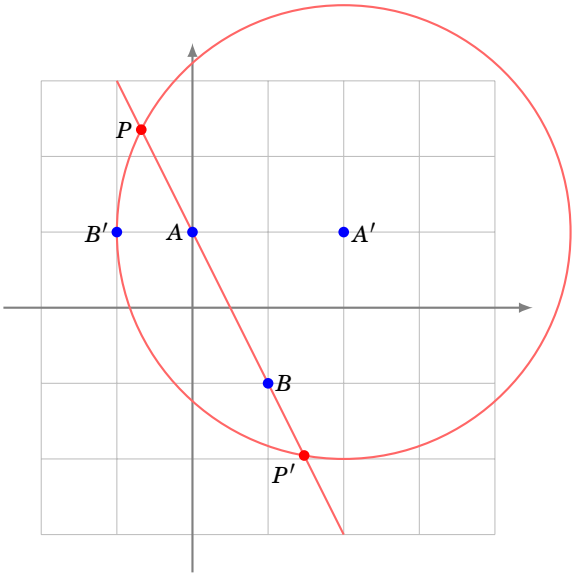
1. P est l'intersection de deux droites (AB) et $(A'B')$ avec par exemple $A(0,1)$, $B(1,-1)$, $A'(0,-2)$, $B'(1,1)$ dont les coordonnées sont dans $K=\mathbb{Q}$. Les équations sont $2x+y=1$ et $3x-y=2$; le point d'intersection P a pour coordonnées $(\frac{3}{5},-\frac{1}{5})$, donc l'abscisse et l'ordonnée sont dans $\mathbb{Q}$. Il n'y a pas besoin d'étendre le corps.



2. P et P' sont les intersections de la droite passant par $A(0,1)$ et $B(1,-1)$ et du cercle de centre $A'(2,1)$ passant par le point $B'(-1,1)$ (et donc de rayon 3). Les équations sont alors $2x+y=1$ et $(x-2)^2+(y-1)^2=9$. Les deux solutions sont les points :

$$\left(\frac{1}{5}\left(2-\sqrt{29}\right),\frac{1}{5}\left(1+2\sqrt{29}\right)\right),\left(\frac{1}{5}\left(2+\sqrt{29}\right),\frac{1}{5}\left(1-2\sqrt{29}\right)\right).$$

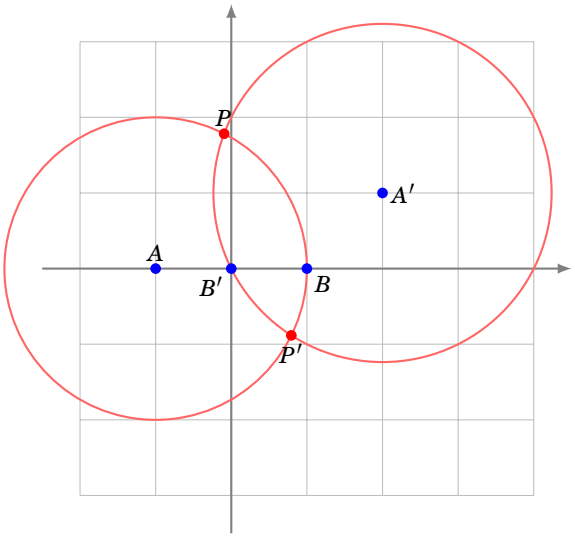
Donc si on pose $\delta=29$ (qui est bien un rationnel) alors les coordonnées des points d'intersection sont de la forme $\alpha+\beta\sqrt{\delta}$ $(\alpha,\beta\in\mathbb{Q})$, c'est-à-dire appartiennent à l'extension quadratique $\mathbb{Q}(\sqrt{29})$.



3. Soient le cercle $\mathscr{C}((-1,0),2)$ (qui a pour centre $A(-1,0)$ et passe par $B(1,0)$) et le cercle $\mathscr{C}((2,1),\sqrt{5})$ (qui a pour centre $A'(2,1)$ et passe par $B'(0,0)$). Les équations sont $(x+1)^2+y^2=4$, $(x-2)^2+(y-1)^2=5$. Les deux points d'intersection sont :

$$\left(\frac{1}{20}\left(7-\sqrt{79}\right),\frac{3}{20}\left(3+\sqrt{79}\right)\right),\left(\frac{1}{20}\left(7+\sqrt{79}\right),\frac{3}{20}\left(3-\sqrt{79}\right)\right).$$

Encore une fois, pour le rationnel $\delta=79$, les abscisses et ordonnées des points d'intersection sont de la forme $\alpha+\beta\sqrt{\delta}$ avec $\alpha,\beta\in\mathbb{Q}$; l'extension quadratique qui convient est donc $\mathbb{Q}(\sqrt{79})$.



4.2. Corollaires

La conséquence la plus importante du théorème de Wantzel est donnée par l'énoncé suivant. C'est ce résultat que l'on utilisera dans la pratique.

Corollaire 4

Tout nombre réel constructible est un nombre algébrique dont le degré algébrique est de la forme 2^n, $n \geqslant 0$.

Démonstration

Soit x un nombre constructible. Par le théorème de Wantzel, il existe des extensions quadratiques $\mathbb{Q} = K_0 \subset K_1 \subset \cdots \subset K_r$ telles que $x \in K_r$. Donc x appartient à une extension de $\mathbb{Q}$ de degré fini. Ainsi, par la proposition 20, x est un nombre algébrique.
On sait de plus que $[K_{i+1} : K_i] = 2$, donc par la proposition 18, nous avons $[K_r : \mathbb{Q}] = 2^r$. Il nous reste à en déduire le degré algébrique $[\mathbb{Q}(x) : \mathbb{Q}]$. Comme $\mathbb{Q}(x) \subset K_r$, alors nous avons toujours par la proposition 18 que : $[K_r : \mathbb{Q}(x)] \times [\mathbb{Q}(x) : \mathbb{Q}] = [K_r : \mathbb{Q}] = 2^r$. Donc $[\mathbb{Q}(x) : \mathbb{Q}]$ divise 2^r et est donc de la forme 2^n.

Voici une autre application plus théorique du théorème de Wantzel, qui caractérise les nombres constructibles.

Corollaire 5

$\mathscr{C}_\mathbb{R}$ est le plus petit sous-corps de $\mathbb{R}$ stable par racine carrée, c'est-à-dire tel que :
- $(x \in \mathscr{C}_\mathbb{R}$ et $x \geqslant 0) \Rightarrow \sqrt{x} \in \mathscr{C}_\mathbb{R}$,
- si K est un autre sous-corps de $\mathbb{R}$ stable par racine carrée alors $\mathscr{C}_\mathbb{R} \subset K$.

La preuve est à faire en exercice.

5. Applications aux problèmes grecs

Nous allons pouvoir répondre aux problèmes de la trisection des angles, de la duplication du cube et de la quadrature du cercle, tout cela en même temps ! Il aura fallu près de 2 000 ans pour répondre à ces questions. Mais pensez que, pour montrer qu'une construction est possible, il suffit de l'exhiber (même si ce n'est pas toujours évident). Par contre pour montrer qu'une construction n'est pas possible, c'est complètement différent. Ce n'est pas parce que personne n'a réussi une construction qu'elle n'est pas possible ! Ce sont les outils algébriques qui vont permettre de résoudre ces problèmes géométriques.
Rappelons le corollaire au théorème de Wantzel, qui va être la clé pour nos problèmes.

Corollaire 6

1. Si un nombre réel x est constructible alors x est un nombre algébrique. C'est-à-dire qu'il existe un polynôme $P \in \mathbb{Q}[X]$ tel que $P(x) = 0$.
2. De plus le degré algébrique de x est de la forme 2^n, $n \geqslant 0$. C'est-à-dire que le plus petit degré, parmi tous les degrés des polynômes $P \in \mathbb{Q}[X]$ vérifiant $P(x) = 0$, est une puissance de 2.

5.1. L'impossibilité de la duplication du cube

La duplication du cube ne peut pas s'effectuer à la règle et au compas.

Cela découle du fait suivant :

Théorème 6

$\sqrt[3]{2}$ n'est pas un nombre constructible.

Démonstration

$\sqrt[3]{2}$ est une racine du polynôme $P(X) = X^3 - 2$. Ce polynôme est unitaire et irréductible dans $\mathbb{Q}[X]$, donc $\sqrt[3]{2}$ est un nombre algébrique de degré 3. Ainsi son degré algébrique n'est pas de la forme 2^n. Bilan : $\sqrt[3]{2}$ n'est pas constructible.

5.2. L'impossibilité de la quadrature du cercle

La quadrature du cercle ne peut pas s'effectuer à la règle et au compas.

C'est une reformulation du théorème suivant, dû à Ferdinand von Lindemann (en 1882) :

Théorème 7

π n'est pas un nombre algébrique (donc n'est pas constructible).

Comme π n'est pas constructible, alors $\sqrt{\pi}$ n'est pas constructible non plus (c'est la contraposée de $x \in \mathscr{C}_{\mathbb{R}} \implies x^2 \in \mathscr{C}_{\mathbb{R}}$).

Nous ne ferons pas ici la démonstration que π n'est pas un nombre algébrique, mais c'est une démonstration qui n'est pas si difficile et abordable en première année.

5.3. L'impossibilité de la trisection des angles

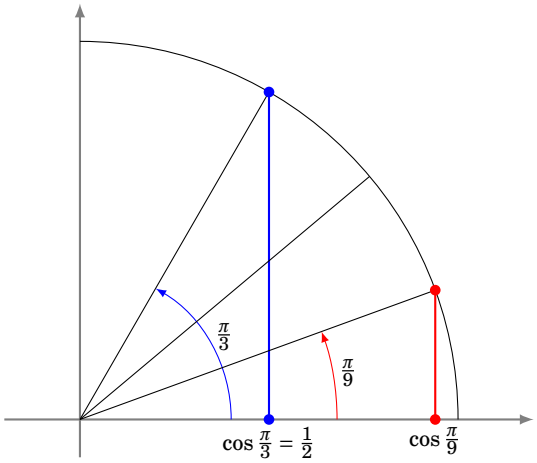
La trisection des angles ne peut pas s'effectuer à la règle et au compas.

Plus précisément nous allons exhiber un angle que l'on ne peut pas couper en trois.

Théorème 8

L'angle $\frac{\pi}{3}$ est constructible mais ne peut pas être coupé en trois car $\cos\frac{\pi}{9}$ n'est pas un nombre constructible.

Bien sûr l'angle $\frac{\pi}{3}$ est constructible car $\cos\frac{\pi}{3}=\frac{1}{2}$. La preuve de la non constructibilité de l'angle $\frac{\pi}{9}$ fait l'objet d'un exercice : $\cos\frac{\pi}{9}$ est un nombre algébrique de degré algébrique 3, donc il n'est pas constructible.



La trisection n'est donc pas possible en général, mais attention, pour certains angles particuliers c'est possible : par exemple les angles π ou $\frac{\pi}{2}$!

Pour aller plus loin voici quelques références :

- *Théorie des corps. La règle et le compas.* J.-L. Carrega, Hermann, 2001.
  Un livre complet sur le sujet !
- *Nombres constructibles.* V. Vassallo, Ph. Royer, IREM de Lille, 2002.
  Avec un point de vue pour le collège et le lycée.
- *Sur les nombres algébriques constructibles à la règle et au compas.* A. Chambert-Loir, Gazette des mathématiciens 118, 2008.
  Vous trouverez dans cet article une réciproque du corollaire au théorème de Wantzel, prouvée de façon « élémentaire », c'est-à-dire sans faire usage de la théorie de Galois.

Auteurs

Arnaud Bodin.
Relu par Vianney Combet.

6 Leçons de choses

Vidéo ■ partie 2. L'alphabet grec
Vidéo ■ partie 3. LaTeX en cinq minutes
Vidéo ■ partie 4. Formules de trigonométrie : sinus, cosinus, tangente
Vidéo ■ partie 5. Formulaire: trigonométrie circulaire et hyperbolique
Vidéo ■ partie 6. Développements limités
Vidéo ■ partie 7. Primitives

1. Travailler avec les vidéos

Les vidéos ne remplacent pas les vrais cours. Cependant elle peuvent vous aider pour préparer, approfondir ou réviser vos connaissances. Nous vous offrons deux outils supplémentaires pour travailler : les polycopié de cours et les vidéos. Voici quelques conseils pour optimiser le visionnage.

1.1. Les vidéos

- Les deux outils de bases : ***papier & crayon***. Notez les points qui vous échappent pour pouvoir y revenir plus tard, faites des petits croquis, résolvez les mini-exercices,... Soyez actifs devant votre écran !
- Limitez-vous : ***une ou deux vidéos*** d'affilée c'est déjà beaucoup de travail. Il vaut mieux privilégier la régularité (par exemple une vidéo de cours par jour et deux vidéos d'exercices). Si vous enchaînez les vidéos comme une séance de cinéma, vous oublierez tout au bout de trois jours.
- Profitez des fonctions ***pause & retour en arrière*** pour prendre le temps de bien comprendre les notions, quitte à repassez la séquence trois fois. Les vidéos vont quatre à cinq fois plus vite que la « vraie vie » : *une vidéo de* 15 *minutes correspond à un cours d'une heure, un exercice corrigé en* $5-6$ *minutes en vidéo serait corrigé en une demi-heure en TD*.
- Il faut du ***temps*** et du ***travail***. Les mathématiques exigent pas mal d'efforts, mais cela vaut vraiment le coup. Tout le monde peut réussir, il n'y a pas besoin d'un don spécial ni d'être un génie des maths. Cependant ne vous leurrez pas, il y a des notions difficiles : bien sûr les profs et les vidéos sont là pour vous aider à les surmonter, mais l'apprentissage repose avant tout sur la qualité et la quantité de votre travail personnel.
- À titre d'exemple le chapitre *Nombres complexes* c'est 1h15 de cours en vidéos et aussi 1h15 d'exercices en vidéos. Cela correspond à 6 heures de cours dans la réalité et 12 heures de séances d'exercices (sur 2 à 3 semaines). Pensez aussi que les étudiants, en plus d'assister

aux cours et aux td, doivent fournir un travail personnel conséquent : à une heure de cours correspond une heure de travail personnel en plus ! Ainsi le chapitre *Nombres complexes* c'est plus de 30 heures de travail en tout et pas seulement 3 heures de visionnage.
Retenez donc le facteur 10 : **Une vidéo de** 12 **minutes c'est** 120 **minutes de travail.**

1.2. Pour les cours

Il faut :

- ***Recopier le cours*** au fur et à mesure du visionnage : écrire permet de mémoriser et d'adopter un rythme plus lent que celui de la vidéo.
- Travailler avec ***le poly*** qui contient plus de détails.
- ***Comprendre*** le cours.
- ***Apprendre*** le cours. Les définitions, les théorèmes et les propositions doivent être appris par cœur. Bien sûr une notion bien comprise est beaucoup plus facile à apprendre !
- Faire les ***mini-exercices***.
- Faire les fiches d'***exercices***.

1.3. Pour les exercices

- ***Chercher*** d'abord à résoudre l'exercice tout seul, sans regarder la correction (ni écrite, ni vidéo). Chercher demande du temps et de la persévérance. Cela permet de vérifier si l'on connaît bien son cours. Les exercices ne sont pas une suite d'astuces à retenir, mais un moyen de travailler par vous-même.
- Le ***lendemain*** seulement, vous pouvez regarder la correction.
- La vidéo de correction et la correction écrite sont complémentaires. Étudiez les deux.

1.4. Note aux collègues enseignants

Si vous êtes enseignants ces vidéos peuvent être utiles de plusieurs façons :

- Vous pouvez proposer les vidéos en compléments ou en révision de vos cours.
- Vous pouvez les proposer comme compléments ou comme sujet d'exposés à faire par les étudiants.
- Vous pouvez passez une vidéos dans vos cours : le support audiovisuel est mieux mémorisé qu'un cour classique, cela permet en plus de regagner l'attention des étudiants en diversifiant les types d'activités.
- Vous pouvez donner à vos étudiants à étudier seul un chapitre à l'avance, sur lequel vous revenez dans votre cours.

Vous trouverez des conseils efficaces dans le livre *Enseigner à l'université* de Markus Brauer. Si vous utilisez ces vidéos d'une façon ou d'une autre nous serions ravis d'avoir un retour de votre expérience !

1.5. D'autres sources pour travailler

Rien ne remplace un vrai prof dans une vraie salle de cours !

Voici deux livres papiers : *Algèbre* et *Analyse* de François Liret, Dominique Martinais aux éditions Dunod.

- Deux livres qui recouvrent le programme de première année.
- Adaptés aux étudiants de l'université.
- Un peu cher !

Voici un cours de première année accessible en ligne : *Cours concis de mathématiques – Première année* de Pierre Guillot.

- Cours concis et complet (370 pages).
- Adapté aux étudiants de l'université.
- Gratuit !

Et un livre accessible gratuitement en ligne *Cours de mathématiques – Math Sup* (attention gros fichier : 11 Mo) d'Alain Soyeur, François Capaces, Emmanuel Vieillard-Baron.

- Cours très complet (1200 pages !).
- Adapté aux élèves des classes prépas.
- Gratuit !

2. Alphabet grec

α		alpha
β		beta
γ	Γ	gamma
δ	Δ	delta
ε		epsilon
ζ		zeta
η		eta
θ	Θ	theta
ι		iota
κ		kappa
λ	Λ	lambda
μ		mu

ν		nu
ξ		xi
o		omicron
π	Π	pi
ρ, ϱ		rho
σ	Σ	sigma
τ		tau
υ		upsilon
ϕ, φ	Φ	phi
χ		chi
ψ	Ψ	psi
ω	Ω	omega

On rencontre aussi "nabla" ∇, l'opérateur de dérivée partielle ∂ (dites "d rond"), et aussi la première lettre de l'alphabet hébreu "aleph" $\aleph$.

3. Écrire des mathématiques : LaTeX en cinq minutes

3.1. Les bases

Pour écrire des mathématiques, il existe un langage pratique et universel, le langage LaTeX (prononcé [latek]). Il est utile pour rédiger des textes contenant des formules, mais aussi accepté sur certains blogs et vous permet d'écrire des maths dans un courriel ou un texto.

Une formule s'écrit entre deux dollars `$\pi^2$` qui donne π^2 ou entre double dollars si l'on veut la centrer sur une nouvelle ligne ; `$$\lim u_n = +\infty$$` affichera :

$$\lim u_n = +\infty$$

Dans la suite on omettra les balises dollars.

3.2. Premières commandes

Les exposants s'obtiennent avec la commande `^` et les indices avec `_` : a^2 s'écrit `a^2` ; u_n s'écrit `u_n` ; a_i^2 s'écrit `\alpha_i^2`. Les accolades `{ }` permettent de grouper du texte : `2^{10}` pour 2^{10} ; `a_{i,j}` pour $a_{i,j}$.
Il y a ensuite toute une liste de commandes (qui commencent par \) dont voici les plus utiles :

`\sqrt`	racine	$\sqrt{a}$	`\sqrt{a}`
		$\sqrt{1+\sqrt{2}}$	`\sqrt{1+\sqrt{2}}`
		$\sqrt[3]{x}$	`\sqrt[3]{x}`
`\frac`	fraction	$\frac{a}{b}$	`\frac{a}{b}`
		$\frac{\pi^3}{12}$	`\frac{\pi^3}{12}`
		$\frac{1}{2+\frac{3}{4}}$	`\frac{1}{2 + \frac{3}{4}}`
		$\gamma^{\frac{1}{n}}$	`\gamma^{\frac{1}{n}}`
`\lim`	limite	$\lim_{n\to+\infty} u_n = 0$	`\lim_{n \to +\infty} u_n = 0`
		$\lim_{x\to 0^+} f(x) < \varepsilon$	`\lim_{x \to 0^+} f(x) < \epsilon`
`\sum`	somme	$\sum_{i=1}^{n} \frac{1}{i}$	`\sum_{i=1}^n \frac{1}{i}`
		$\sum_{i\geqslant 0} a_i$	`\sum_{i \ge 0} a_i`
`\int`	intégrale	$\int_a^b \phi(t)dt$	`\int_a^b \phi(t) dt`

3.3. D'autres commandes

Voici d'autres commandes, assez naturelles pour les anglophones.

$f : E \to F$	`f : E \to F`
$+\infty$	`+\infty`
$a \leqslant 0$	`a \le 0`
$a > 0$	`a > 0`
$a \geqslant 1$	`a \ge 1`
δ	`\delta`
Δ	`\Delta`

$a \in E$	`a \in E`
$A \subset E$	`A \subset E`
$P \Longrightarrow Q$	`P \implies Q`
$P \Longleftrightarrow Q$	`P \iff Q`
$\forall$	`\forall`
$\exists$	`\exists`
$\cup$	`\cup`
$\cap$	`\cap`

3.4. Pour allez plus loin

Il est possible de créer ses propres commandes avec `\newcommand`. Par exemple avec l'instruction

```
\newcommand{\Rr}{\mathbb{R}}
```

vous définissez une nouvelle commande `\Rr` qui exécutera l'instruction `\mathbb{R}` et affichera donc $\mathbb{R}$.

Autre exemple, après avoir défini

```
\newcommand{\monintegrale}{\int_0^{+\infty} \frac{\sin t}{t} dt}
```

la commande `\monintegrale` affichera $\int_0^{+\infty} \frac{\sin t}{t} dt$.

Pour (beaucoup) plus de détails, consultez le manuel *Une courte (?) introduction à LaTeX*.

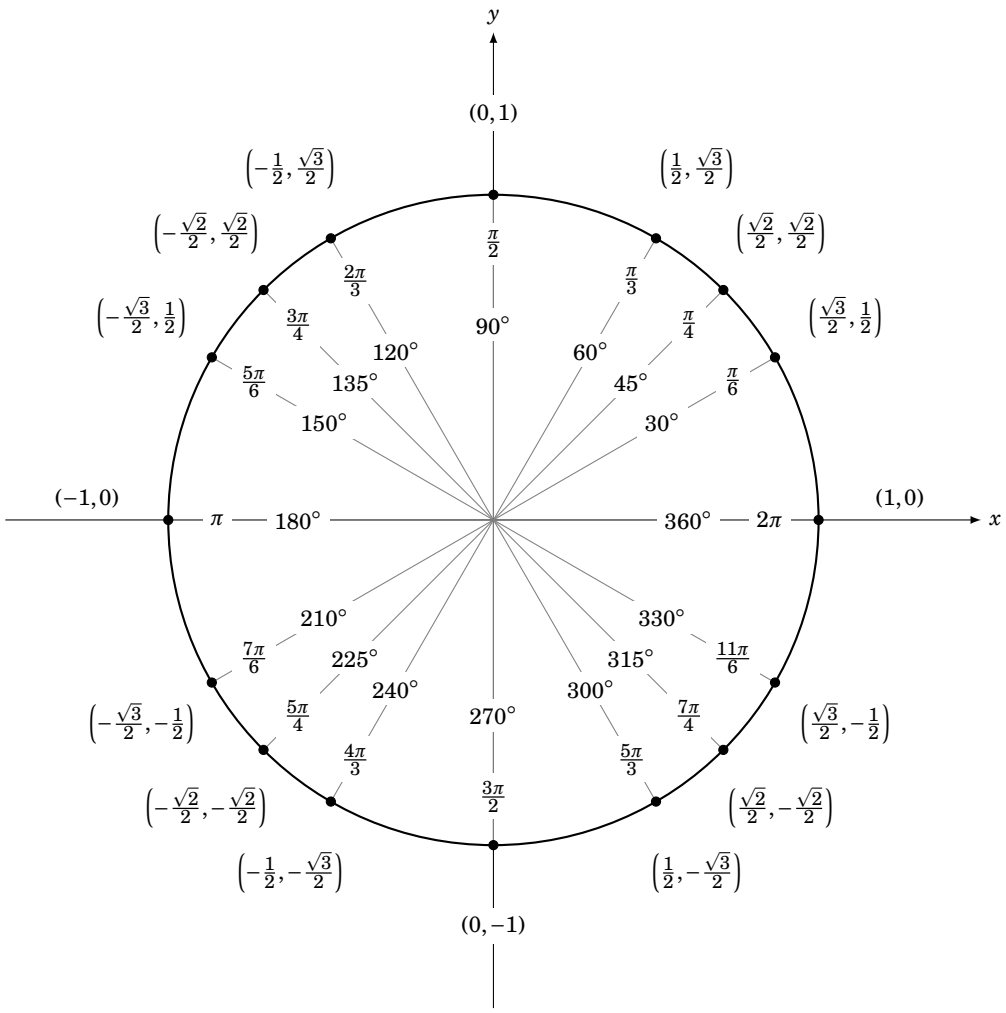
3.5. Mini-exercices

Écrire en LaTeX toutes ces formules (qui par ailleurs sont vraies !).

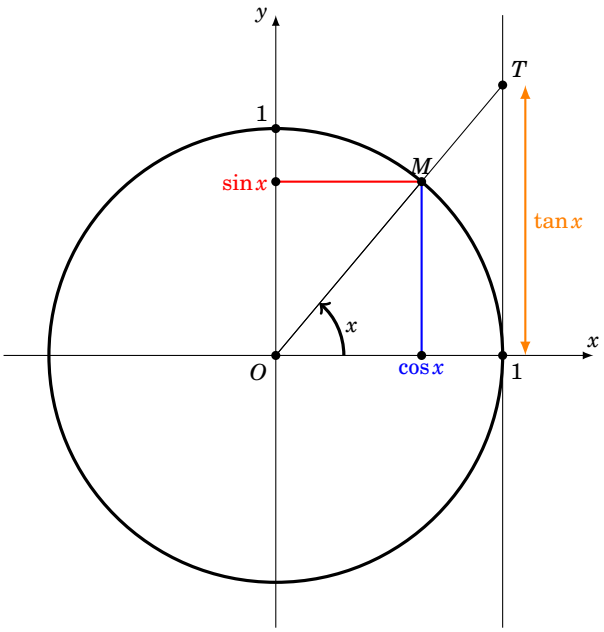
1. $\sqrt{a} - \sqrt{b} = \dfrac{a-b}{\sqrt{a}+\sqrt{b}}$
2. $\displaystyle\sum_{n=1}^{+\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$
3. $\displaystyle\lim_{R \to +\infty} \int_{-R}^{+R} e^{-t^2} dt = \sqrt{\pi}$
4. $\forall \varepsilon > 0 \quad \exists \delta \geqslant 0 \quad (|x - x_0| < \delta \implies |\ln(x) - \ln(x_0)| < \varepsilon)$
5. $\displaystyle\sum_{k=0}^{+\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right) = \pi$

4. Formules de trigonométrie : sinus, cosinus, tangente

4.1. Le cercle trigonométrique



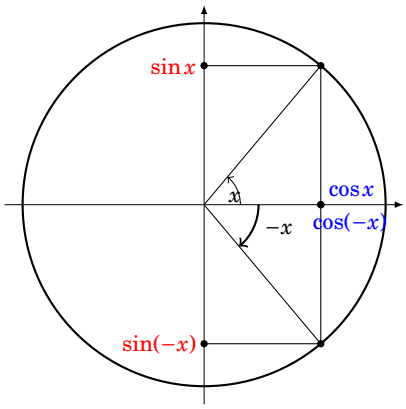
Voici le cercle trigonométrique (de rayon 1), le sens de lecture est l'inverse du sens des aiguilles d'une montre. Les angles remarquables sont marqués de 0 à 2π (en radian) et de 0° à 360°. Les coordonnées des points correspondant à ces angles sont aussi indiquées.



Le point M a pour coordonnées $(\cos x, \sin x)$. La droite (OM) coupe la droite d'équation $(x = 1)$ en T, l'ordonnée du point T est $\tan x$.

Les formules de base :

$$\cos^2 x + \sin^2 x = 1$$
$$\cos(x + 2\pi) = \cos x$$
$$\sin(x + 2\pi) = \sin x$$



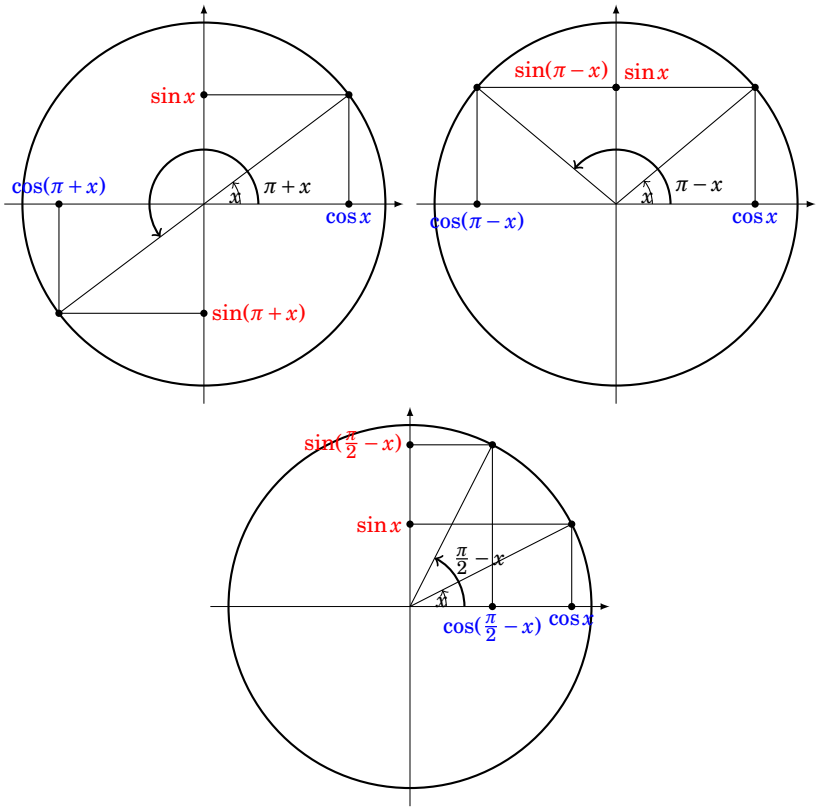
Nous avons les formules suivantes :

$$\cos(-x) = \cos x$$
$$\sin(-x) = -\sin x$$

On retrouve graphiquement ces formules à l'aide du dessin des angles x et $-x$.

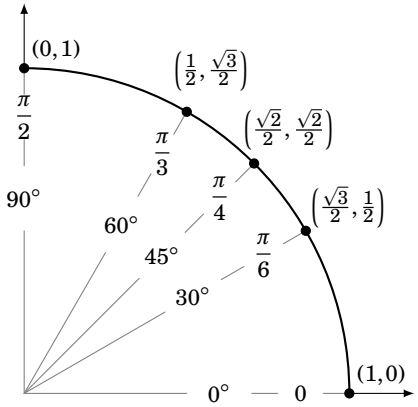
Il en est de même pour les formules suivantes :

$$\cos(\pi + x) = -\cos x \qquad \cos(\pi - x) = -\cos x \qquad \cos(\frac{\pi}{2} - x) = \sin x$$
$$\sin(\pi + x) = -\sin x \qquad \sin(\pi - x) = \sin x \qquad \sin(\frac{\pi}{2} - x) = \cos x$$



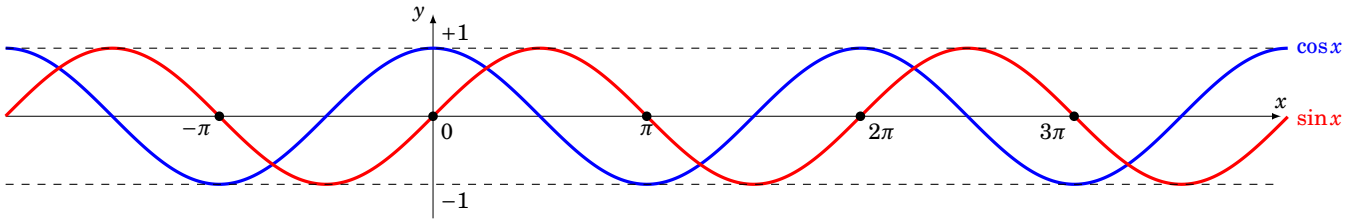
x	0	$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$
$\cos x$	1	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{1}{2}$	0
$\sin x$	0	$\frac{1}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{3}}{2}$	1
$\tan x$	0	$\frac{1}{\sqrt{3}}$	1	$\sqrt{3}$	

Valeurs que l'on retrouve bien sur le cercle trigonométrique.

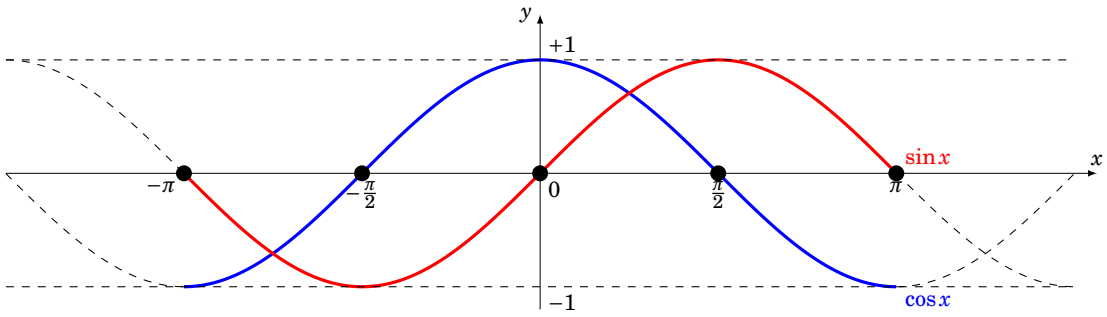


4.2. Les fonctions sinus, cosinus, tangente

La fonction cosinus est périodique de période 2π et elle paire (donc symétrique par rapport à l'axe des ordonnées). La fonction sinus est aussi périodique de période de 2π mais elle impaire (donc symétrique par rapport à l'origine).



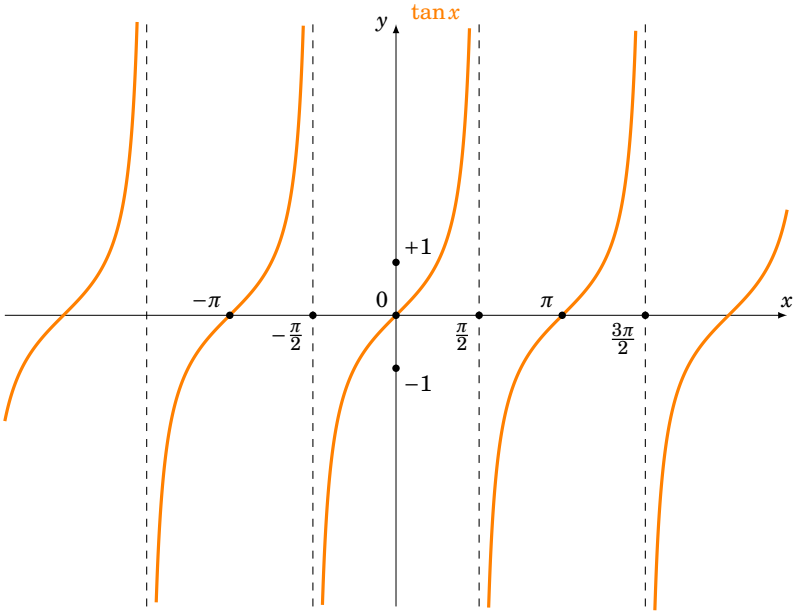
Voici un zoom sur l'intervalle $[-\pi, \pi]$.



Pour tout x n'appartenant pas à $\{\ldots, -\frac{\pi}{2}, \frac{\pi}{2}, \frac{3\pi}{2}, \frac{5\pi}{2}, \ldots\}$ la tangente est définie par

$$\tan x = \frac{\sin x}{\cos x}$$

La fonction $x \mapsto \tan x$ est périodique de période π ; c'est une fonction impaire.



Voici les dérivées :

$$\cos' x = -\sin x$$
$$\sin' x = \cos x$$
$$\tan' x = 1 + \tan^2 x = \frac{1}{\cos^2 x}$$

4.3. Les formules d'additions

$$\cos(a+b) = \cos a \cdot \cos b - \sin a \cdot \sin b$$
$$\sin(a+b) = \sin a \cdot \cos b + \sin b \cdot \cos a$$
$$\tan(a+b) = \frac{\tan a + \tan b}{1 - \tan a \cdot \tan b}$$

On en déduit immédiatement :

$$\cos(a-b) = \cos a \cdot \cos b + \sin a \cdot \sin b$$
$$\sin(a-b) = \sin a \cdot \cos b - \sin b \cdot \cos a$$
$$\tan(a-b) = \frac{\tan a - \tan b}{1 + \tan a \cdot \tan b}$$

Il est bon de connaître par cœur les formules suivantes (faire $a = b$ dans les formules d'additions) :

$$\begin{aligned}\cos 2a &= 2\cos^2 a - 1\\ &= 1 - 2\sin^2 a\\ &= \cos^2 a - \sin^2 a\end{aligned}$$
$$\sin 2a = 2\sin a \cdot \cos a$$
$$\tan 2a = \frac{2\tan a}{1 - \tan^2 a}$$

4.4. Les autres formules

Voici d'autres formules qui se déduisent des formules d'additions. Il n'est pas nécessaire de les connaître mais il faut savoir les retrouver en cas de besoin.

$$\cos a \cdot \cos b = \frac{1}{2}\big[\cos(a+b) + \cos(a-b)\big]$$
$$\sin a \cdot \sin b = \frac{1}{2}\big[\cos(a-b) - \cos(a+b)\big]$$
$$\sin a \cdot \cos b = \frac{1}{2}\big[\sin(a+b) + \sin(a-b)\big]$$

Les formules précédentes se reformulent aussi en :

$$\cos p + \cos q = 2\cos\frac{p+q}{2}\cdot\cos\frac{p-q}{2}$$
$$\cos p - \cos q = -2\sin\frac{p+q}{2}\cdot\sin\frac{p-q}{2}$$
$$\sin p + \sin q = 2\sin\frac{p+q}{2}\cdot\cos\frac{p-q}{2}$$
$$\sin p - \sin q = 2\sin\frac{p-q}{2}\cdot\cos\frac{p+q}{2}$$

Enfin les formules de la «tangente de l'arc moitié» permettent d'exprimer sinus, cosinus et tangente en fonction de $\tan\frac{x}{2}$.

$$\text{Avec} \quad t = \tan\frac{x}{2} \quad \text{on a} \quad \begin{cases} \cos x &= \frac{1-t^2}{1+t^2} \\ \sin x &= \frac{2t}{1+t^2} \\ \tan x &= \frac{2t}{1-t^2} \end{cases}$$

Ces formules sont utiles pour le calcul de certaines intégrales par changement de variable, en utilisant en plus la relation $dx = \dfrac{2dt}{1+t^2}$.

4.5. Mini-exercices

1. Montrer que $1+\tan^2 x = \frac{1}{\cos^2 x}$.
2. Montrer la formule d'addition de $\tan(a+b)$.
3. Prouver la formule pour $\cos a \cdot \cos b$.
4. Prouver la formule pour $\cos p + \cos q$.
5. Prouver la formule : $\sin x = \dfrac{2\tan\frac{x}{2}}{1+(\tan\frac{x}{2})^2}$.
6. Montrer que $\cos\frac{\pi}{8} = \frac{1}{2}\sqrt{\sqrt{2}+2}$. Calculer $\cos\frac{\pi}{16}$, $\cos\frac{\pi}{32}$,...
7. Exprimer $\cos(3x)$ en fonction $\cos x$; $\sin(3x)$ en fonction $\sin x$; $\tan(3x)$ en fonction $\tan x$.

5. Formulaire : trigonométrie circulaire et hyperbolique

Fonctions circulaires et hyperboliques

Propriétés trigonométriques *: remplacer* cos *par* ch *et* sin *par* i·sh.

$$\cos^2 x + \sin^2 x = 1$$

$$\begin{aligned}\cos(a+b) &= \cos a \cdot \cos b - \sin a \cdot \sin b\\ \sin(a+b) &= \sin a \cdot \cos b + \sin b \cdot \cos a\\ \tan(a+b) &= \frac{\tan a + \tan b}{1 - \tan a \cdot \tan b}\end{aligned}$$

$$\begin{aligned}\cos(a-b) &= \cos a \cdot \cos b + \sin a \cdot \sin b\\ \sin(a-b) &= \sin a \cdot \cos b - \sin b \cdot \cos a\\ \tan(a-b) &= \frac{\tan a - \tan b}{1 + \tan a \cdot \tan b}\end{aligned}$$

$$\begin{aligned}\cos 2a &= 2\cos^2 a - 1\\ &= 1 - 2\sin^2 a\\ &= \cos^2 a - \sin^2 a\end{aligned}$$

$$\sin 2a = 2 \sin a \cdot \cos a$$

$$\tan 2a = \frac{2\tan a}{1 - \tan^2 a}$$

$$\begin{aligned}\cos a \cdot \cos b &= \frac{1}{2}\left[\cos(a+b) + \cos(a-b)\right]\\ \sin a \cdot \sin b &= \frac{1}{2}\left[\cos(a-b) - \cos(a+b)\right]\\ \sin a \cdot \cos b &= \frac{1}{2}\left[\sin(a+b) + \sin(a-b)\right]\end{aligned}$$

$$\begin{aligned}\cos p + \cos q &= 2\cos\frac{p+q}{2} \cdot \cos\frac{p-q}{2}\\ \cos p - \cos q &= -2\sin\frac{p+q}{2} \cdot \sin\frac{p-q}{2}\\ \sin p + \sin q &= 2\sin\frac{p+q}{2} \cdot \cos\frac{p-q}{2}\\ \sin p - \sin q &= 2\sin\frac{p-q}{2} \cdot \cos\frac{p+q}{2}\end{aligned}$$

$$\mathrm{ch}^2 x - \mathrm{sh}^2 x = 1$$

$$\begin{aligned}\mathrm{ch}(a+b) &= \mathrm{ch}\,a \cdot \mathrm{ch}\,b + \mathrm{sh}\,a \cdot \mathrm{sh}\,b\\ \mathrm{sh}(a+b) &= \mathrm{sh}\,a \cdot \mathrm{ch}\,b + \mathrm{sh}\,b \cdot \mathrm{ch}\,a\\ \mathrm{th}(a+b) &= \frac{\mathrm{th}\,a + \mathrm{th}\,b}{1 + \mathrm{th}\,a \cdot \mathrm{th}\,b}\end{aligned}$$

$$\begin{aligned}\mathrm{ch}(a-b) &= \mathrm{ch}\,a \cdot \mathrm{ch}\,b - \mathrm{sh}\,a \cdot \mathrm{sh}\,b\\ \mathrm{sh}(a-b) &= \mathrm{sh}\,a \cdot \mathrm{ch}\,b - \mathrm{sh}\,b \cdot \mathrm{ch}\,a\\ \mathrm{th}(a-b) &= \frac{\mathrm{th}\,a - \mathrm{th}\,b}{1 - \mathrm{th}\,a \cdot \mathrm{th}\,b}\end{aligned}$$

$$\begin{aligned}\mathrm{ch}\,2a &= 2\,\mathrm{ch}^2 a - 1\\ &= 1 + 2\,\mathrm{sh}^2 a\\ &= \mathrm{ch}^2 a + \mathrm{sh}^2 a\end{aligned}$$

$$\mathrm{sh}\,2a = 2\,\mathrm{sh}\,a \cdot \mathrm{ch}\,a$$

$$\mathrm{th}\,2a = \frac{2\,\mathrm{th}\,a}{1 + \mathrm{th}^2 a}$$

$$\operatorname{ch} a \cdot \operatorname{ch} b = \frac{1}{2}\left[\operatorname{ch}(a+b) + \operatorname{ch}(a-b)\right]$$
$$\operatorname{sh} a \cdot \operatorname{sh} b = \frac{1}{2}\left[\operatorname{ch}(a+b) - \operatorname{ch}(a-b)\right]$$
$$\operatorname{sh} a \cdot \operatorname{ch} b = \frac{1}{2}\left[\operatorname{sh}(a+b) + \operatorname{sh}(a-b)\right]$$

$$\operatorname{ch} p + \operatorname{ch} q = 2 \operatorname{ch} \frac{p+q}{2} \cdot \operatorname{ch} \frac{p-q}{2}$$
$$\operatorname{ch} p - \operatorname{ch} q = 2 \operatorname{sh} \frac{p+q}{2} \cdot \operatorname{sh} \frac{p-q}{2}$$
$$\operatorname{sh} p + \operatorname{sh} q = 2 \operatorname{sh} \frac{p+q}{2} \cdot \operatorname{ch} \frac{p-q}{2}$$
$$\operatorname{sh} p - \operatorname{sh} q = 2 \operatorname{sh} \frac{p-q}{2} \cdot \operatorname{ch} \frac{p+q}{2}$$

avec $t = \tan\frac{x}{2}$ on a $\begin{cases} \cos x &= \frac{1-t^2}{1+t^2} \\ \sin x &= \frac{2t}{1+t^2} \\ \tan x &= \frac{2t}{1-t^2} \end{cases}$

avec $t = \mathrm{th}\frac{x}{2}$ on a $\begin{cases} \mathrm{ch}\, x &= \frac{1+t^2}{1-t^2} \\ \mathrm{sh}\, x &= \frac{2t}{1-t^2} \\ \mathrm{th}\, x &= \frac{2t}{1+t^2} \end{cases}$

Dérivées *: la multiplication par i n'est plus valable*

$$\cos' x = -\sin x$$
$$\sin' x = \cos x$$
$$\tan' x = 1 + \tan^2 x = \frac{1}{\cos^2 x}$$

$$\mathrm{ch}' x = \mathrm{sh}\, x$$
$$\mathrm{sh}' x = \mathrm{ch}\, x$$
$$\mathrm{th}' x = 1 - \mathrm{th}^2 x = \frac{1}{\mathrm{ch}^2 x}$$

$$\mathrm{Arccos}' x = \frac{-1}{\sqrt{1-x^2}} \quad (|x| < 1)$$
$$\mathrm{Arcsin}' x = \frac{1}{\sqrt{1-x^2}} \quad (|x| < 1)$$
$$\mathrm{Arctan}' x = \frac{1}{1+x^2}$$

$$\mathrm{Argch}' x = \frac{1}{\sqrt{x^2-1}} \quad (x > 1)$$
$$\mathrm{Argsh}' x = \frac{1}{\sqrt{x^2+1}}$$
$$\mathrm{Argth}' x = \frac{1}{1-x^2} \quad (|x| < 1)$$

6. Formules de développements limités

Développements limités usuels *(au voisinage de 0)*

$$e^x = 1+\frac{x}{1!}+\frac{x^2}{2!}+\cdots+\frac{x^n}{n!} + o(x^n) = \sum_{k=0}^{n}\frac{x^k}{k!} + o(x^n)$$

$$\cos x = 1-\frac{x^2}{2!}+\frac{x^4}{4!}-\cdots+(-1)^n\cdot\frac{x^{2n}}{(2n)!} + o(x^{2n+1}) = \sum_{k=0}^{n}(-1)^k\frac{x^{2k}}{(2k)!} + o(x^{2n+1})$$

$$\sin x = x-\frac{x^3}{3!}+\frac{x^5}{5!}-\cdots+(-1)^n\cdot\frac{x^{2n+1}}{(2n+1)!} + o(x^{2n+2}) = \sum_{k=0}^{n}(-1)^k\frac{x^{2k+1}}{(2k+1)!} + o(x^{2n+2})$$

$$\tan x = x+\frac{x^3}{3}+\frac{2}{15}x^5+\frac{17}{315}x^7 + o(x^8)$$

$$\operatorname{ch} x = 1+\frac{x^2}{2!}+\frac{x^4}{4!}+\cdots+\frac{x^{2n}}{(2n)!} + o(x^{2n+1}) = \sum_{k=0}^{n}\frac{x^{2k}}{(2k)!} + o(x^{2n+1})$$

$$\operatorname{sh} x = x+\frac{x^3}{3!}+\frac{x^5}{5!}+\cdots+\frac{x^{2n+1}}{(2n+1)!} + o(x^{2n+2}) = \sum_{k=0}^{n}\frac{x^{2k+1}}{(2k+1)!} + o(x^{2n+2})$$

$$\operatorname{th} x = x-\frac{x^3}{3}+\frac{2}{15}x^5-\frac{17}{315}x^7 + o(x^8)$$

$$\ln(1+x) = x-\frac{x^2}{2}+\frac{x^3}{3}-\cdots+(-1)^{n-1}\cdot\frac{x^n}{n} + o(x^n) = \sum_{k=1}^{n}(-1)^{k+1}\frac{x^k}{k} + o(x^n)$$

$$(1+x)^\alpha = 1+\alpha x+\frac{\alpha(\alpha-1)}{2!}x^2+\cdots+\frac{\alpha(\alpha-1)\cdots(\alpha-n+1)}{n!}x^n + o(x^n) = \sum_{k=0}^{n}\binom{\alpha}{k}x^k + o(x^n)$$

$$\frac{1}{1+x} = 1-x+x^2-\cdots+(-1)^n x^n + o(x^n) = \sum_{k=0}^{n}(-1)^k x^k + o(x^n)$$

$$\frac{1}{1-x} = 1+x+x^2+\cdots+x^n + o(x^n) = \sum_{k=0}^{n}x^k + o(x^n)$$

$$\sqrt{1+x} = 1+\frac{x}{2}-\frac{1}{8}x^2-\cdots+(-1)^{n-1}\cdot\frac{1\cdot1\cdot3\cdot5\cdots(2n-3)}{2^n n!}x^n + o(x^n)$$

$$\frac{1}{\sqrt{1+x}} = 1-\frac{x}{2}+\frac{3}{8}x^2-\cdots+(-1)^n\cdot\frac{1\cdot3\cdot5\cdots(2n-1)}{2^n n!}x^n + o(x^n)$$

$$\arccos x = \frac{\pi}{2}-x-\frac{1}{2}\frac{x^3}{3}-\frac{1\cdot3}{2\cdot4}\frac{x^5}{5}-\cdots-\frac{1\cdot3\cdot5\cdots(2n-1)}{2\cdot4\cdot6\cdots(2n)}\frac{x^{2n+1}}{2n+1} + o(x^{2n+2})$$

$$\arcsin x = x+\frac{1}{2}\frac{x^3}{3}+\frac{1\cdot3}{2\cdot4}\frac{x^5}{5}+\cdots+\frac{1\cdot3\cdot5\cdots(2n-1)}{2\cdot4\cdot6\cdots(2n)}\frac{x^{2n+1}}{2n+1} + o(x^{2n+2})$$

$$\arctan x = x-\frac{x^3}{3}+\frac{x^5}{5}+\cdots+(-1)^n\cdot\frac{x^{2n+1}}{2n+1} + o(x^{2n+2})$$

7. Formulaire : primitives

Primitives usuelles

C désigne une constante arbitraire. Les intervalles sont à préciser.

$$\int e^{\alpha t}\,dt = \frac{e^{\alpha t}}{\alpha} + C \quad (\alpha \in \mathbb{C}^*)$$

$$\int t^{\alpha}\,dt = \frac{t^{\alpha+1}}{\alpha+1} + C \quad (\alpha \neq -1)$$

$$\int \frac{dt}{1+t^2} = \operatorname{Arctan} t + C$$

$$\int \frac{dt}{\sqrt{1-t^2}} = \operatorname{Arcsin} t + C$$

$$\int \cos t\,dt = \sin t + C$$

$$\int \sin t\,dt = -\cos t + C$$

$$\int \frac{dt}{\cos^2 t} = \tan t + C$$

$$\int \frac{dt}{\sin^2 t} = -\operatorname{cotan} t + C$$

$$\int \frac{dt}{\cos t} = \ln\left|\tan\left(\frac{t}{2}+\frac{\pi}{4}\right)\right| + C$$

$$\int \frac{dt}{\sin t} = \ln\left|\tan\frac{t}{2}\right| + C$$

$$\int \tan t\,dt = -\ln|\cos t| + C$$

$$\int \operatorname{cotan}\, t\,dt = \ln|\sin t| + C$$

$$\int \frac{dt}{t} = \ln|t| + C$$

$$\int \frac{dt}{1-t^2} = \frac{1}{2}\ln\left|\frac{1+t}{1-t}\right| + C$$

$$\int \frac{dt}{\sqrt{t^2+\alpha}} = \ln\left|t+\sqrt{t^2+\alpha}\right| + C$$

$$\int \operatorname{ch} t\,dt = \operatorname{sh} t + C$$

$$\int \operatorname{sh} t\,dt = \operatorname{ch} t + C$$

$$\int \frac{dt}{\operatorname{ch}^2 t} = \operatorname{th} t + C$$

$$\int \frac{dt}{\operatorname{sh}^2 t} = -\coth t + C$$

$$\int \frac{dt}{\operatorname{ch} t} = 2\operatorname{Arctan} e^t + C$$

$$\int \frac{dt}{\operatorname{sh} t} = \ln\left|\operatorname{th}\frac{t}{2}\right| + C$$

$$\int \operatorname{th} t\,dt = \ln(\operatorname{ch} t) + C$$

$$\int \coth t\,dt = \ln|\operatorname{sh} t| + C$$

Les auteurs

Les auteurs des chapitres sont cités à la fin de chaque chapitre.

Les exercices en vidéos sont de Arnaud Bodin et Léa Blanc-Centi (université Lille 1).

La musique du générique est de Victor Fleurant.