Calcul formel

1. Premiers pas avec Sage

Le calcul formel est un domaine charnière entre mathématiques et informatique. L'objectif de ce cours est d'obtenir des algorithmes efficaces afin de manipuler des objets mathématiques abstraits tels que les fonctions, les polynômes, les matrices, etc. À notre niveau, le calcul formel ou symbolique sera synonyme de « mathématiques effectives et efficaces ».

1.1. Hello world !

Servons-nous d'abord de `Sage` comme d'une calculatrice :

> **Travaux pratiques 1.**
>
> 1. Calculer $1+2\times 3^4$.
> 2. Calculer $\frac{22}{33}$.
> 3. Calculer $\cos(\frac{\pi}{6})$.

Voilà les résultats :

> **Code 1** (*hello-world.sage*)**.**
> ```
> sage: 1+2*3^4
> 163
> sage: 22/33
> 2/3
> sage: cos(pi/6)
> 1/2*sqrt(3)
> ```

On retient que :

- `Sage` connaît les opérations classiques $+$, $-$, $\times$, $/$. La puissance a^b s'écrit `a^b` ou `a**b`.
- `Sage` fait des calculs exacts, il simplifie $\frac{22}{33}$ en $\frac{2}{3}$, contrairement à une banale calculatrice qui afficherait $0.6666\ldots$
- `Sage` manipule les fonctions et les constantes usuelles : par exemple $\cos\left(\frac{\pi}{6}\right)=\frac{\sqrt{3}}{2}$.

Dans toute la suite, on omettra l'invite de commande « `sage:` ». En fin de section vous trouverez davantage d'informations sur la mise en place de `Sage`.

1.2. Calcul formel

L'utilisation d'un système de calcul symbolique conduit le mathématicien à un type de démarche auquel il n'est traditionnellement pas habitué : l'expérimentation !

1. Je me pose des questions.
2. J'écris un algorithme pour tenter d'y répondre.
3. Je trouve une conjecture sur la base des résultats expérimentaux observés.

4. Je tente de prouver la conjecture.

Travaux pratiques 2.

1. Que vaut le nombre complexe $(1-\mathrm{i})^k$ lorsque k est un entier positif ou nul ?
2. Les nombres de la forme $F_n = 2^{(2^n)}+1$ sont-ils tous premiers ?

Nous pouvons faire des calculs avec les nombres complexes. En posant $z = 1-\mathrm{i}$, on calcule successivement la partie réelle (par la commande `real_part`), la partie imaginaire (`imag_part`), le module (`abs`) et l'argument (`arg`) de $z^0, z^1, z^2 \ldots$

Code 2 (*motiv-calcul-formel.sage (1)*).

```
z = 1-I
for k in range(10):
    print( k, z^k, real_part(z^k), imag_part(z^k), abs(z^k), arg(z^k) )
```

$$\begin{array}{llllll}
(0, & 1, & 1, & 0, & 1, & 0) \\
(1, & -i+1, & 1, & -1, & \sqrt{2}, & -\frac{\pi}{4}) \\
(2, & -2i, & 0, & -2, & 2, & -\frac{\pi}{2}) \\
(3, & -2i-2, & -2, & -2, & 2\sqrt{2}, & -\frac{3\pi}{4}) \\
(4, & -4, & -4, & 0, & 4, & \pi) \\
(5, & 4i-4, & -4, & 4, & 4\sqrt{2}, & \frac{3\pi}{4}) \\
(6, & 8i, & 0, & 8, & 8, & \frac{\pi}{2}) \\
(7, & 8i+8, & 8, & 8, & 8\sqrt{2}, & \frac{\pi}{4}) \\
(8, & 16, & 16, & 0, & 16, & 0) \\
(9, & -16i+16, & 16, & -16, & 16\sqrt{2}, & -\frac{\pi}{4})
\end{array}$$

Les expressions de z^k nous amènent à proposer les conjectures suivantes :

$$\begin{aligned}
z^{4\ell} &= (-1)^\ell 2^{2\ell} \\
z^{4\ell+1} &= (-1)^\ell 2^{2\ell}(1-\mathrm{i}) \\
z^{4\ell+2} &= (-1)^\ell 2^{2\ell+1}(-\mathrm{i}) \\
z^{4\ell+3} &= (-1)^\ell 2^{2\ell+1}(-1-\mathrm{i})
\end{aligned}$$

On remarque expérimentalement une structure assez simple. Par exemple, pour passer de $z^k = (1+\mathrm{i})^k$ à $z^{k+1} = (1+\mathrm{i})^{k+1}$, le module est multiplié par $\sqrt{2}$ alors que l'argument change de $-\frac{\pi}{4}$. On a donc une relation de récurrence $z^{k+1} = \sqrt{2}\,e^{-\frac{\mathrm{i}\pi}{4}} z^k$. Comme $z^0 = (1-\mathrm{i})^0 = 1$, on conjecture $z^k = \sqrt{2}^k e^{-\frac{k\mathrm{i}\pi}{4}}$.
Passons à la preuve : en écrivant sous la forme module-argument $z = \sqrt{2}e^{-\frac{\mathrm{i}\pi}{4}}$, on obtient bien que $z^k = \sqrt{2}^k e^{-\frac{k\mathrm{i}\pi}{4}}$. On pourrait aussi obtenir une expression encore plus simple en discutant selon les valeurs de k modulo 8.

Le calcul formel permet aussi d'éviter de passer des années à essayer de montrer un résultat qui s'avère faux au final. Pierre de Fermat pensait que tous les nombres de la forme $F_n = 2^{2^n}+1$ étaient premiers. Cent ans plus tard, Euler calcule que $F_5 = 4\,294\,967\,297 = 641 \times 6\,700\,417$ n'est pas un nombre premier.

Code 3 (*motiv-calcul-formel.sage (2)*).

```
for n in range(8):
    print(n, factor(2^(2^n)+1))
```

1.3. Calcul formel vs calcul numérique

Même si Sage est conçu pour le calcul symbolique, il sait aussi effectuer des calculs numériques. Bien que non exact, le calcul numérique possède plusieurs avantages : il est plus rapide, souvent suffisant pour les applications pratiques et peut aussi être plus lisible.

Travaux pratiques 3.
Quelle est la limite de la suite récurrente définie par :

$$u_0 = 1 \qquad u_{n+1} = \sqrt{1+u_n} \text{ pour } n \geqslant 0 \quad ?$$

Si on calcule formellement les premiers termes, on trouve :

$$u_0 = 1 \qquad u_1 = \sqrt{2} \qquad u_2 = \sqrt{1+\sqrt{2}} \qquad u_3 = \sqrt{1+\sqrt{1+\sqrt{2}}} \qquad u_4 = \sqrt{1+\sqrt{1+\sqrt{1+\sqrt{2}}}}$$

ce qui n'éclaire pas vraiment sur le comportement de la suite.

Code 4 (*calcul-numerique.sage*)**.**

```
u = 1
for i in range(10):
    u = sqrt(1 + u)
    print(u)
    print(numerical_approx(u))
```

Par contre au vu des approximations :

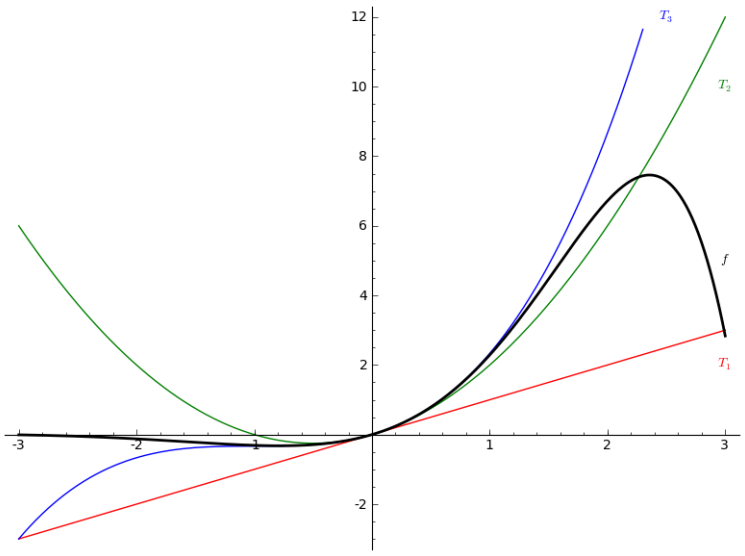
$$\begin{aligned} u_0 &= 1 \\ u_1 &= 1.4142\ldots \\ u_2 &= 1.5537\ldots \\ u_3 &= 1.5980\ldots \\ u_4 &= 1.6118\ldots \\ u_5 &= 1.6161\ldots \end{aligned}$$

on peut émettre plusieurs conjectures : la suite est croissante, elle est majorée par 2 et converge. En poussant les calculs, une approximation de la limite est $1.618033\ldots$ Les plus observateurs auront reconnu une approximation du nombre d'or $\phi = \frac{1+\sqrt{5}}{2}$, solution positive de l'équation $x^2 - x - 1$. On renvoie à un cours d'analyse pour la preuve que la suite (u_n) converge effectivement vers ϕ.

1.4. Graphiques

La production de graphiques est un formidable outil qui permet de mettre en lumière des phénomènes complexes.

Travaux pratiques 4.
Soit f la fonction définie par $f(x) = \sin(x) \cdot \exp(x)$. Calculer les premiers polynômes de Taylor associés aux développements limités de f en 0. Tracer leurs graphes. Quelles propriétés des développements limités cette illustration permet-elle de mettre en évidence ?



- Après avoir défini la fonction f par `f = sin(x)*exp(x)`, la commande `taylor(f,x,0,n)` renvoie le développement limité de f en $x = 0$ à l'ordre n, par exemple ici $f(x) = x + x^2 + \frac{1}{3}x^3 + \epsilon(x)x^3$, donc $T_1(x) = x$, $T_2(x) = x + x^2$, $T_3(x) = x + x^2 + \frac{1}{3}x^3$.
- La commande `plot(f,(a,b))` trace le graphe de f sur l'intervalle $[a, b]$.

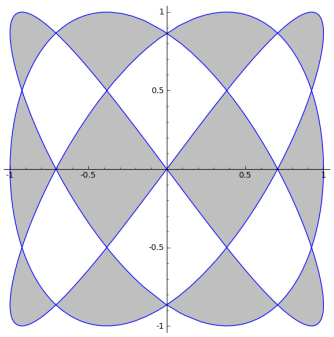
Il est perceptible que :

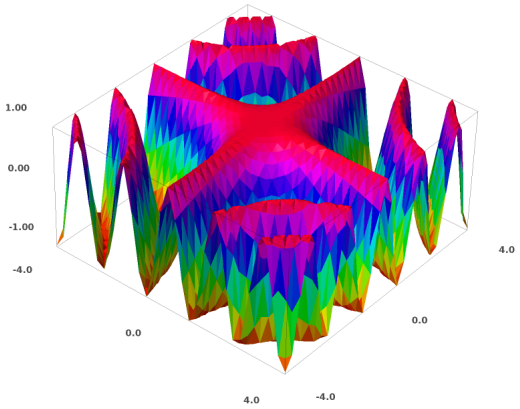
- les polynômes de Taylor sont une bonne approximation de la fonction au voisinage d'un point ;
- plus l'ordre du développement limité est élevé, meilleure est l'approximation ;
- l'approximation est seulement *locale* : loin du point considéré (ici l'origine), les polynômes de Taylor n'approchent plus du tout la fonction.

Il est possible de tracer une grande variété de graphiques. Voici par exemple la courbe de Lissajous d'équation $t \mapsto \big(\cos(3t), \sin(4t)\big)$ et le graphe de la fonction de deux variables définie par $f(x,y) = \cos(xy)$. Les commandes sont :

- `parametric_plot((cos(3*t),sin(4*t)),(t,0,2*pi))`
- `plot3d(cos(x*y),(x,-4,4),(y,-4,4))`

Attention ! Il faut avoir au préalable défini les variables utilisées : `var('t')` et `var('x,y')` (en fait, seule la variable x est définie par défaut dans Sage).





1.5. Le calcul formel peut-il tout faire ?

Le calcul formel ne résout malheureusement pas tous les problèmes de mathématique d'un coup de baguette magique !

Travaux pratiques 5.

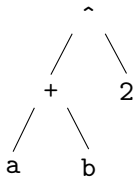
1. Pouvez-vous calculer les solutions réelles de $x^k - x - 1 = 0$ lorsque k est un entier supérieur ou égal à 2 ?
2. Est-ce que Sage sait que toutes ces expressions sont nulles ?

$$2^{10} - 1024 \qquad (x+1)^2 - x^2 - 2x - 1 \qquad \sin^2(x) + \cos^2(x) - 1$$

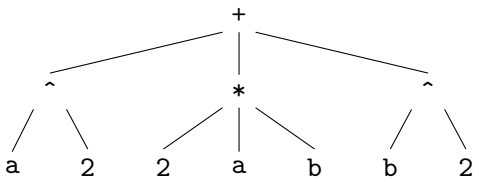
1. La première limitation est propre aux mathématiques : on ne peut pas trouver une écriture explicite des solutions de toutes les équations. Pour $x^2 - x - 1 = 0$, à l'aide de la commande `solve(x^2-x-1==0,x)`, on trouve bien les deux solutions $\frac{1+\sqrt{5}}{2}$ et $\frac{1-\sqrt{5}}{2}$. Par contre `solve(x^5-x-1==0,x)` ne renvoie pas les solutions mais l'équation qui définit les solutions (ce qui ne nous avance guère). Ce n'est pas ici un problème technique de Sage. En effet, il n'est mathématiquement pas possible d'exprimer la solution réelle de $x^5 - x - 1 = 0$ à l'aide de radicaux ($\sqrt{}$, $\sqrt{}$, $\sqrt[3]{}$, $\sqrt[4]{}$, ...), ceci est seulement possible jusqu'au degré 4. Par contre, on obtient une approximation de la solution réelle par la commande `find_root(x^5-x-1==0,-1,2)` en précisant que l'on cherche la solution dans l'intervalle $[-1, 2]$.
2. (a) Sans problème `2^10-1024` renvoie 0.
   (b) Il est nécessaire de développer l'expression pour trouver 0 : `expand((x+1)^2-x^2-2*x-1)`.
   (c) Il faut explicitement demander de simplifier l'expression trigonométrique : d'abord `f = sin(x)^2+cos(x)^2 - 1` puis on demande de simplifier `f.simplify_trig()` pour obtenir 0.

Il n'est pas du tout évident pour un ordinateur de reconnaître les identités comme $(a+b)^2 = a^2 + 2ab + b^2$ ou bien $\cos^2(x) + \sin^2(x) = 1$. Souvenez-vous d'ailleurs qu'il faut plusieurs années d'apprentissage pour les assimiler. Lorsqu'il y a plusieurs écritures d'une même expression, il n'est pas non plus évident pour l'ordinateur de savoir quelle forme est la plus adaptée pour l'utilisateur. Par exemple, selon le contexte, les trois écritures suivantes sont utiles : $(a-b)^3 = (a-b)(a^2 - 2ab + b^2) = a^3 - 3a^2b + 3ab^2 - b^3$. Il faudra donc « guider » le logiciel de calcul formel avec les fonctions `expand`, `factor`, `simplify`...

Remarque.
Pour avoir une idée de la difficulté à identifier deux expressions mathématiquement égales, voici une représentation de la façon dont sont stockées $(a+b)^2$ et $a^2+2ab+b^2$ dans le logiciel.



$(a+b)^2$



$a^2+2ab+b^2$

1.6. Un peu plus sur Sage

Ce cours n'a pas pour but d'être un manuel d'utilisation du logiciel Sage. Vous trouverez sur internet des tutoriels pour démarrer et pour un usage avancé :

Site officiel de Sage

Il existe aussi un livre gratuit :

Calcul mathématique avec Sage

Une façon simple d'obtenir de l'aide pour une commande Sage est d'utiliser le point d'interrogation : `ln?` (ou bien `help(ln)`). Vous y apprendrez que la fonction `ln` est le logarithme naturel.

Il y a deux façons d'utiliser Sage :

- *En ligne de commande* : vous obtenez une fenêtre avec l'invite `sage:` puis vous tapez vos commandes (en n'oubliant pas de faire une copie de votre travail dans un fichier texte du type `mon_programme.sage`).
- *Dans votre navigateur* : à partir de l'invite `sage:` vous tapez `notebook()` pour obtenir une interface complète et conviviale.

Voici une liste de fonctions usuelles :

`abs(x)`	$\lvert x\rvert$
`x^n` ou `x**n`	x^n
`sqrt(x)`	$\sqrt{x}$
`exp(x)`	$\exp x$
`ln(x)` ou `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
`arccos(x), arcsin(x), arctan(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)

Il existe des fonctions spécifiques qui manipulent les entiers, les vecteurs, les matrices, les polynômes, les fonctions mathématiques... Nous les découvrions au fil des chapitres.

La syntaxe de Sage est celle du langage Python. Il n'est pas nécessaire de connaître ce langage, la syntaxe sera introduite au fur et à mesure. Cependant vous pourrez étudier avec profit le chapitre « Algorithmes et mathématiques » du site Exo7.
Pour l'instant voici ce que l'on retient de l'exemple donné plus haut :

Code 5 (*motiv-calcul-formel.sage (2)*).
```
for n in range(8):
    print(n, factor(2^(2^n)+1))
```

- Une boucle `for n in range(N)` : l'indice n parcourt les entiers de 0 à $N-1$.
- Le bloc d'instructions suivant `print(n, factor(2^(2^n)+1))` est donc exécuté successivement pour $n=0$, $n=1$, $\ldots n=N-1$.
- Les espaces en début de ligne (*l'indentation*) sont essentielles car elles délimitent le début et la fin d'un bloc d'instructions.