TP 5 - Calcul formel

En Python, grâce au module `sympy`, on peut réaliser des calculs formels comme on les fait sur papier en Analyse ou en Algèbre. Dans ce but, `sympy` introduit une notion de la variable *formelle* (ou *symbolique*) et **redéfinit** à sa manière les fonctions usuelles.

Ainsi, par exemple, la fonction `sin` provenant de `sympy` se comporte différemment de la fonction `sin` fournie par `numpy`. Plus généralement, les fonctions `sympy` sont **incompatibles** avec les tableaux `numpy` et les graphiques de `matplotlib`. L'importation totale (`from sympy import *`) est à éviter, surtout si vous voulez utiliser au même temps `sympy` et `numpy`.

On pourra charger le module de manière usuelle : `import sympy as sm`. Mais pour alléger l'écriture de formules mathématiques, il est préférable d'utiliser, additionnellement, l'importation *spécifique*, par exemple : `from sympy import var, sin, cos, pi`.

Exercice 1. *Manipulation des expressions*

Pour effectuer un calcul formel, il faut d'abord créer des variables symboliques. On peut le faire à l'aide de la commande `var`, par exemple : `var('x y z')`. Puis, on peut construire des expressions algébriques ou des fonctions. Tapez, par exemple : `print(x + y + 2*x + z + 1)`.

1. A l'aide de la commande `expand`, donner une expression développée de la fonction polynômiale suivante : $(x-1)(x-2)(x+3)$.

2. La commande `factor(expr)` factorise l'expression `expr` en utilisant des coefficients rationnelles (si la forme factorisée a des coefficients dans $\mathbb{R} \setminus \mathbb{Q}$ ou dans $\mathbb{C}$, `factor` ne les trouvera pas).

   Factoriser $x^3 - 39x - 70$.

3. Réduire au même dénominateur $\dfrac{1}{x+1} + \dfrac{1}{y} + \dfrac{1}{z}$ (cf. `together` ou `ratsimp`)

   Pour améliorer l'affichage du résultat, on pourra utiliser la fonction `pprint` ("pretty print").

4. Décomposer en fractions partielles l'expression suivante (cf. `apart`) :

$$\frac{2x^2 + 4x + 3}{x^3 - 39x - 70}$$

5. Simplifier (cf. `simplify`) l'expression $\cos(x+y) + \cos(x-y) + \sin(x+y) + \sin(x-y)$.

   Remarque : La commande `simplify` applique certaines règles de calcul pour obtenir une expression plus simple. Mais "plus simple" n'est pas forcément bien défini en toute généralité. Par exemple, `simplify(x**2 + 2*x + 1)` renvoie `x**2 + 2*x + 1` (essayez). Pour obtenir une forme factorisé, il faut donc utiliser une commande plus spécifique comme `factor`.

6. Pour substituer une variable dans une expression, on utilise `subs`. Par exemple :

```
p = sin(x) + cos(x)
print(p.subs(x, pi/4))
print(p)
```

   Soit $P(x) = 2x^2 - 3x + 1$. A l'aide de `subs`, afficher $P(5)$, puis une forme simple de $P(y+1) - P(y-1)$ (où y est une autre variable formelle).

7. Si `expr` est une expression `sympy` qui possède une valeur numérique, alors `expr.evalf()` calcule cette valeur (approchée) sous forme décimale. Calculer :

```
print(sqrt(10).evalf())
x = pi
print((x**2 + x + 1).evalf())
```

On peut demander une précision arbitraire : `pi.evalf(500)`.

Exercice 2. *Calcul numérique en précision arbitraire*

Refaire l'exercice 1 de la feuille TP 2 avec une précision de calcul de 500 chiffres en s'inspirant du code suivant :

```
x = sm.Float(0.23)          # construction d'un nombre à précision arbitraire (au début du calcul)
x = (4 * x * (1 - x)).evalf(500)       # (possiblement dans une boucle)
```

Afficher 20 chiffres de x_{10}, x_{20}, x_{30}, x_{40}, x_{50} et x_{60}. Comparez avec le calcul utilisant la formule développée $y_{n+1} = 4y_n - 4y_n^2$ (en grande précision) ainsi qu'avec les résultats obtenus en TP 2. Que peut-on constater ?

Exercice 3. *Equations algébriques*

Soit `expr` une expression de variable `x`. Alors la commande `solve(expr, x)` résout l'équation $expr = 0$ (par rapport à l'inconnue `x`).

On peut aussi résoudre un système d'équations (linéaires) en fournissant une liste d'expressions et une liste de variables, par exemple `s = solve([expr1, expr2, expr3], [x,y,z])`. Dans ce cas, la variable `s` est un *dictionnaire* : `s[x]` renvoie la solution pour `x`, `s[y]` celle pour `y` etc.

1. Après l'exercice précédent, `x` et `y` ne sont plus des variables symboliques (mais des valeurs de type `float` de grande précision). Avant de procéder, utiliser la commande `var` pour que les variables nécessaire redeviennent symboliques.

   Trouver les zéros de la fonction $P(x) = 3x^3 - \frac{7x^2}{2} - \frac{x}{2} + 1$.

   Vérifier à l'aide de `subs`. *Indication* : Pour calculer avec des valeurs rationnelles *exactes*, on pourra utiliser `Rational`. Par exemple, `Rational(2, 3)` signifie $\frac{2}{3}$ (or 2/3 donne toujours une valeur approchée 0.6666666666667).

2. Résoudre le système et vérifier la solution :

$$\begin{pmatrix} 0 & 2 & 5 \\ -3 & 2 & -1 \\ -1 & 2 & 2 \end{pmatrix} X = \begin{pmatrix} 3 \\ -1 \\ 5 \end{pmatrix}$$

Exercice 4. *Polynôme passant par trois points*

Soit $h > 0$ et soit $P(x) = ax^2 + bx + c$ avec $a, b, c \in \mathbb{R}$ une fonction polynomiale du seconde degré, $P : [0, h] \to \mathbb{R}$.

1. Soit u, v, w trois constantes réels. Trouver a, b, c tel que $P(0) = u$, $P(h/2) = v$ et $P(h) = w$.

2. Substituer dans P (`P = P.subs(...)`) les solutions `a, b, c` trouvées dans la question précédente. Afficher la nouvelle l'expression de P.

3. Substituer maintenant $u = 0$, $v = 2$, $w = 1$ et $h = 2$ et visualiser P à l'aide de la commande `sm.plot(P, (x, 0 ,2))`.

   Remarque: Il s'agit ici de la commande `plot` de `sympy` (et non celle de `matplotlib`).

Exercice 5. *Limite, dérivée, primitive, intégrale (TCM)*

1. En `sympy`, l'infini est note `oo` (deux petits "o"). Par exemple, `limit(x / (x**2 + 1), x, oo)` renvoie 0. Calculer les limites suivantes :

$$\lim_{x\to\infty} \sqrt{x^2+x} - \sqrt{x^2+1}, \qquad \lim_{n\to\infty} \left(1+\frac{a}{3n}\right)^{2n} \quad (a \in \mathbb{R}).$$

2. Soit `expr` une expression de variable `x`. Alors la commande `diff(expr, x)` calcule sa dérivée. La commande `integrate(expr, x)` renvoie une de ses primitives et `integrate(expr, (x, a, b))`, où `a` et `b` sont deux réels, calcule son intégrale.

   (a) Calculer une primitive des fonctions suivantes. A chaque fois vérifier le résultat en calculant la dérivée.
   $$f(x) = \frac{x}{1+\sqrt{x}}, \qquad g(x) = 3x\sqrt{1+x^2}.$$

   (b) Calculer l'intégrale $\displaystyle\int_1^2 \frac{\sqrt{\log x}}{x} dx$.

Exercice 6. *Intégration par parties.*

Dans cette question, on se propose de calculer l'intégrale

$$\int_0^1 \frac{x\log(\sqrt{x^2+3})}{\sqrt{x^2+3}}$$

en suivant la procédure d'intégration par parties. Votre script déterminera les éléments nécessaires et affichera toutes les étapes :

- les fonctions u, u', v et v';
- les éléments de la formule d'intégration par parties : $[u \cdot v]_0^1$ et $\int_0^1 u'v$;
- le résultat final.

Exercice 7. *Calcul numérique vs calcul symbolique*

On se propose de revenir, avec des nouveaux outils, sur l'exercice 2 de la feuille de TP 2.

Soit $v_n = \displaystyle\int_0^1 \frac{x^{n-1}}{a+x} dx$.

1. Calculer v_n symboliquement en toute généralité (c'est-à-dire avec variables formelles a et n). Le résultat, est-il intelligible ?
2. Refaire le calcul en substituant 40 pour la valeur de n.
3. Afficher la valeur exacte de v_{40} en substituant $a = 3$ dans l'expression obtenue dans la question précédente.
4. Afficher un développement décimal de v_{40}. Comparer avec votre résultat de TP 2.

Exercice 8. *Graphiques avec `sympy`*

Pour visualiser ses fonctions, le module `sympy` fournit sa propre commande graphique `plot`, bien différente de `plot` donnée par `matplotlib`. Plus concrètement, si `e1 e2` sont des expressions de variable `x`, alors `plot(e1, e2, (x, a, b))` trace la courbe représentative de ces expressions sur l'intervalle [`a`, `b`].

Visualiser, sur un intervalle qu'on choisira, la courbe représentative de la fonction $f(x) = \frac{1}{2}x^2 + x - 1$ et sa tangente en point d'abscisses $x = 2$.