Calcul symbolique et son utilisation avec Sympy

Devoir de groupe

KUITCHE AROLLE NACHARD 22T2931
KONZOU SODEA ALAN PEREC 22T2957
KENFACK LEKANE FRANK 22T2847
VOUKENG DJIOKENG CHRISTIAN ROUSSEL 22U2053

Université de Yaoundé I
Département d'Informatique

Plan

Qu'est-ce que le calcul symbolique ?

- Manipulation d'expressions mathématiques sous forme exacte.
- Contraire du calcul numérique (approximation).
- Exemple :

Calcul Symbolique (Exact)

- Constante: $\sqrt{2}$
- Expression: $(x+1)^2$

Calcul Numérique (Approximatif)

- Constante: $\sqrt{2} \approx 1.41421356$
- Évaluation: $(1.5+1)^2 = 6.25$

Historique

- **Origines anciennes** : manipulation algébrique par Euclide, al-Khwârizmî.
- **1960s** : Macsyma (MIT), premier système de calcul formel.
- **1970s-80s** : Maple, puis Mathematica.
- **Aujourd'hui** : Sympy (2007–...), open-source, intégré à Python.

Domaines d'application

- Mathématiques pures (algèbre, équations, démonstrations).
- Physique et ingénierie (formules exactes).
- Informatique (cryptographie, Datascience).
- Outils : Maple, Mathematica, **Sympy** (Python, open-source).

Introduction à Sympy

- Librairie Python pour le calcul symbolique.
- Intégrée à l'écosystème scientifique (NumPy, SciPy, Jupyter,).
- Permet : simplifications, dérivées, intégrales, équations, matrices.

Manipulation symbolique

```
import sympy as sp
x, y = sp.symbols('x y')

expr = (x + 1)**2
sp.expand(expr)     # D veloppement
sp.factor(x**2+2*x+1)  # Factorisation
```

Calcul différentiel et intégral

```
f = sp.sin(x**2)
sp.diff(f, x)    # d riv e

sp.integrate(sp.exp(x), x)  # int grale
```

Résolution d'équations

```
solutions = sp.solve(x**2 - 5*x + 6, x)
# R sultats exacts : [2, 3]
```

Exact vs numérique

```
val_exact = sp.sqrt(2)
val_num = sp.N(val_exact, 10)  # approx   10 chiffres
```

Pourquoi l'exactitude compte ?

Le calcul symbolique permet de manipuler des constantes irrationnelles (comme π, $\sqrt{2}$) ou des fractions sans **perte de précision** avant l'évaluation finale.

Optimisation et calcul symbolique

- Calcul de dérivées exactes pour trouver des minima/maxima.
- Calcul automatique de gradients et hessien.
- Gestion des contraintes avec les multiplicateurs de Lagrange.
- Symbolique = exactitude ; Numérique = rapidité.

Le Pont Symbolique-Numérique

Les expressions exactes du gradient et du hessien calculées par SymPy peuvent être utilisées pour accélérer et rendre plus précis les algorithmes d'optimisation **numériques** (comme le Newton-Raphson).

Exemple : points critiques

```
f = x**3 - 6*x**2 + 9*x + 1
df = sp.diff(f, x)
crit_points = sp.solve(df, x)
# Points critiques exacts
```

Exemple : gradient et hessien

```
f = x**2 + y**2 + x*y
grad = [sp.diff(f, var) for var in (x, y)]
hess = sp.hessian(f, (x, y))
```

Utilité du Hessien

Le Hessien permet d'appliquer la **condition du second ordre** pour déterminer si un point critique est un minimum, un maximum ou un point de selle.

Avantages et limites

Avantages :

- Résultats exacts.
- Idéal pour l'enseignement, la recherche et la vérification.
- Gratuit et accessible.

Limites :

- Peu adapté aux problèmes massifs.
- Plus lent que le calcul numérique sur de grands systèmes.

Conclusion

- Le calcul symbolique permet de manipuler les mathématiques de manière exacte.
- Sympy rend cette pratique accessible dans Python.
- En optimisation, il aide à dériver, simplifier et analyser les conditions exactes.

Merci !