TPE : Calcul Symbolique

Groupe
Filière Informatique — Option Data Science

Université de Yaoundé I
Superviseur : Pr. Paulin MELATAGIA

2 octobre 2025

Sommaire

Présentation du calcul scientifique

Définition : branche des mathématiques appliquées utilisant des méthodes numériques et symboliques pour résoudre des problèmes (EDO, optimisation, modélisation).

Exemple : résolution exacte d'équations, calcul symbolique des dérivées et primitives.

Applications : physique, ingénierie, data science, optimisation.

Théorèmes et corollaires utiles

Convexité (1D)

Si $f : \mathbb{R} \to \mathbb{R}$ est C^2 et $f''(x) \geq 0 \ \forall x$ alors f est convexe.

Hessienne (nD)

Si $f : \mathbb{R}^n \to \mathbb{R}$ est C^2 et sa Hessienne $H_f(x)$ est semi-définie positive $\forall x$, alors f est convexe.

Corollaire

Obtenir un ∇f exact (symbolique) réduit les erreurs d'approximation lors d'une descente de gradient.

Dériver fonctions coût & appliquer la descente de gradient

- Ex. : coût MSE univarié : $J(a, b) = \frac{1}{n} \sum_i (y_i - (ax_i + b))^2$.
  Calcul symbolique : $\nabla J = \left(\frac{\partial J}{\partial a}, \frac{\partial J}{\partial b}\right)$.
- Descente : $\theta \leftarrow \theta - \eta \nabla J(\theta)$ (avec gradient obtenu symboliquement).
- Avantage : exactitude théorique des formules ; puis conversion en numérique pour exécution.

Présentation de Sympy

- Bibliothèque Python pour le calcul symbolique : dérivation, intégration, simplification, matrices, résolution d'équations.
- Installation : `pip install sympy`.
- Fonction clé : `lambdify` (convertit expressions symboliques $\rightarrow$ fonctions numériques).

Exemples (1/3) — Convexité et Hessienne

```
from sympy import symbols, diff, hessian, Matrix
x, y = symbols('x y')
f = x**2 + y**2 + x*y

# dérivées partielles
df_dx = diff(f, x)
df_dy = diff(f, y)

# Hessienne et test de convexité
H = hessian(f, (x, y))
# valeurs propres >= 0 => convexe
H_eigs = Matrix(H).eigenvals()
print(H, H_eigs)
```

Exemples (2/3) — Régression linéaire (1 var) et multivariée

```
from sympy import symbols, diff, Matrix, lambdify
# 1-variable
a, b, x, y = symbols('a b x y')
f = a*x + b
J = (y - f)**2
dJa = diff(J, a)
dJb = diff(J, b)

# multivarié (notation compacte)
# w = Matrix(symbols('w0:3'))  # exemple 2 ou 3 dim
# X, y  => J = (1/(2*m))*(X*w - y).T*(X*w - y)
# grad = diff(J, w_i)  (on peut vectoriser avec sympy Matrix)
```

Exemples (3/3) — Logistique, lambdify, descente

```
from sympy import symbols, exp, log, lambdify
w1, w2, b, x1, x2, y = symbols('w1 w2 b x1 x2 y')
z = w1*x1 + w2*x2 + b
sigma = 1/(1+exp(-z))
L = - (y*log(sigma) + (1-y)*log(1-sigma))

grad_w1 = diff(L, w1)
grad_w2 = diff(L, w2)

# conversion -> fonctions numpy
grad_w1_num = lambdify((w1,w2,b,x1,x2,y), grad_w1, 'numpy')
# boucle de descente (schéma)
# w1 -= eta * grad_w1_num(w1,w2,b, x1_i, x2_i, y_i)
```

Conclusion

- Le calcul symbolique (Sympy) relie rigueur mathématique et implémentation pratique.
- Permet : vérifier convexité (2 dérivée / Hessienne), obtenir gradients/Hessiennes exacts, convertir en fonctions numériques (lambdify) et lancer des expériences de descente.
- Perspectives : comparaison avec auto-diff (TensorFlow/PyTorch), optimisation à grande échelle.