Le logarithme $1\frac{1}{2}$ de Maxim Kontsevich : calculs explicites dans $\mathbb{Z}/p\mathbb{Z}$

Denise Vella-Chemla pilotant claude, juillet 2026

Je fournis à claude la traduction en français de l'article de Maxim Kontsevich ici https://denisevellachemla.eu/trad-logarithme-un-et-demi-Maxim-Kontsevich.pdf . La suite de la présente note a été produite par l'ia claude.

1. Rappel de la définition

Pour $p > 2$ premier, Maxim Kontsevich définit

$$H_p(x) \;=\; \sum_{k=1}^{p-1} \frac{x^k}{k} \pmod p, \qquad x \in \mathbb{Z}/p\mathbb{Z},$$

où $\dfrac{1}{k}$ désigne l'inverse de k dans le corps $\mathbb{Z}/p\mathbb{Z}$ (il existe car $1 \le k \le p-1$ n'est jamais multiple de p).

Cette somme ressemble à une troncature du développement en série de $\log \dfrac{1}{1-x} = \displaystyle\sum_{k\ge1} \frac{x^k}{k}$, mais ce n'en est pas un logarithme au sens usuel : il n'existe aucun homomorphisme de groupes non trivial de $(\mathbb{Z}/p\mathbb{Z})^\times \simeq \mathbb{Z}/(p-1)\mathbb{Z}$ vers $\mathbb{Z}/p\mathbb{Z}$ (les ordres des deux groupes sont premiers entre eux), donc H_p ne peut pas être "le" logarithme discret.

2. Trois équations fonctionnelles

Kontsevich montre que H_p satisfait trois identités, valables pour tout corps (et indépendantes de p) :

$$\text{(A)} \quad H(1-x) = H(x), \tag{1}$$

$$\text{(B)} \quad H(x+y) = H(y) + (1-y)\,H\!\left(\frac{x}{1-y}\right) + y\,H\!\left(-\frac{x}{y}\right) \quad (y \neq 0, 1), \tag{2}$$

$$\text{(C)} \quad x\,H\!\left(\frac{1}{x}\right) = -H(x) \quad (x \neq 0). \tag{3}$$

3. Le point remarquable : une fonction réelle bien connue

Sur $\mathbb{R}$, les équations (A), (B), (C) n'ont *qu'une seule* solution continue non nulle (à un facteur scalaire près) :

$$H_\infty(x) = -\Big(x \log|x| + (1-x)\log|1-x|\Big).$$

C'est exactement l'**entropie de Shannon** d'une variable aléatoire à deux issues, de probabilités x et $1-x$.

Kontsevich montre que H_p est, de façon analogue, l'unique solution (à un facteur scalaire près) des mêmes équations dans $\mathbb{Z}/p\mathbb{Z}$. Autrement dit : *la même identité algébrique caractérise, sur* $\mathbb{R}$,

l'entropie de Shannon, et sur $\mathbb{Z}/p\mathbb{Z}$, la fonction H_p. C'est ce qui justifie le nom que Kontsevich choisit de lui donner : un "logarithme $1\frac{1}{2}$", intermédiaire entre le logarithme ordinaire (équation à 3 termes) et le dilogarithme (équation à 5 termes) - la relation (B) comporte 4 termes.

4. Interprétation cohomologique

À H on associe la fonction homogène de degré 1

$$\varphi(x,y) = (x+y)\,H\left(\frac{x}{x+y}\right) \quad (x+y \neq 0),$$

et l'équation (B) dit exactement que φ est un 2-cocycle du groupe additif F (à coefficients triviaux). Invariant sous l'action de $F^\times$, ce cocycle définit une extension centrale du groupe affine $\mathrm{Aff}(1,F)$ - un groupe de Lie résoluble de dimension 3 lorsque $F = \mathbb{R}$.

- Sur $\mathbb{R}$: $H^2(\mathbb{R},\mathbb{R})$ (cohomologie mesurable) est non trivial ; imposer que φ soit un cobord force $\psi(x)/x = a\log|x| + b$, ce qui redonne l'entropie.
- Sur $\mathbb{Z}/p\mathbb{Z}$: la classe de φ dans $H^2(\mathbb{Z}/p\mathbb{Z},\mathbb{Z}/p\mathbb{Z}) \simeq \mathbb{Z}/p\mathbb{Z}$ est nulle (car un logarithme discret non trivial n'existe pas), ce qui donne l'unicité de H_p.

5. Lien avec l'entropie de Shannon générale

L'entropie d'une variable aléatoire à k issues se calcule par récurrence à partir du cas à deux issues via la relation de "chaînage" de l'entropie conditionnelle

$$H(\eta) = H(\xi) + H_\xi(\eta).$$

Conséquence frappante signalée par Kontsevich : si une variable aléatoire ξ ne prend que des valeurs rationnelles avec des probabilités rationnelles, on peut non seulement calculer son entropie réelle (nombre transcendant), mais aussi définir ses *réductions modulo p*, pour presque tout nombre premier p, en remplaçant H par H_p dans le même calcul récursif.

6. Pourquoi c'est intéressant

- **Un "double" algébrique d'un objet transcendant.** L'entropie de Shannon $-\sum p_i \log p_i$ est un nombre transcendant en général. Le fait qu'elle soit entièrement caractérisée par une identité purement algébrique (les équations (A), (B), (C)) permet de "réduire mod p" cette identité et d'obtenir un analogue fini, purement combinatoire, de l'entropie - sans aucune trace de transcendance. C'est le même genre de phénomène que celui qui permet de réduire mod p des périodes ou des multizêtas dans d'autres contextes arithmétiques.
- **Sa place dans la hiérarchie des polylogarithmes.** L'article auquel ce texte sert d'appendice porte sur les "poly-(ana)-logs" : des généralisations du logarithme et du dilogarithme définies par leurs équations fonctionnelles plutôt que par une série, utiles pour construire des régulateurs en K-théorie algébrique et étudier des valeurs spéciales de fonctions L. Le

logarithme usuel correspond à une relation à 3 termes et donne une classe dans H^1 ; le dilogarithme de Bloch-Wigner correspond à 5 termes et donne une classe dans $H^3(\mathrm{SL}(2,\mathbb{C}),\mathbb{R})$ (liée au volume hyperbolique). H_p, avec sa relation à 4 termes, occupe une place intermédiaire - d'où le nom "logarithme $1\frac{1}{2}$".

- **Un objet rigide.** L'unicité (à un scalaire près) dans les deux mondes, réel et fini, est ce qui rend l'analogie mathématiquement solide plutôt que purement formelle : ce n'est pas "une" fonction qui ressemble à l'entropie, c'est *la* fonction déterminée par les mêmes trois axiomes.

7. Vérification numérique

Les calculs de H_p pour les nombres premiers $p = 3, 5, 7, \ldots, 47$ (programme en annexe) confirment numériquement la propriété (A) pour chaque p, et on observe systématiquement $H_p(0) = H_p(1) = 0$: le terme $x = 0$ annule la somme, et $H_p(1) = \sum_{k=1}^{p-1} \frac{1}{k}$ est nul mod p pour $p > 2$ (fait relié à la congruence de Wolstenholme pour $p > 3$, et direct pour $p = 3$).

8. Annexe : programme pour les corps premiers jusqu'à $\mathbb{Z}/47\mathbb{Z}$

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Calcul du "logarithme 1 1/2" H_p de Maxim Kontsevich.

Reference : appendice de Kontsevich a l'article "On poly(ana)logs I"
de P. Elbaz-Vincent et H. Gangl (trad. Denise Vella-Chemla, 2025).

Pour p premier impair, on definit H_p : Z/pZ -> Z/pZ par

    H_p(x) = somme_{k=1}^{p-1}  x^k / k    (mod p)

ou "x^k / k" signifie x^k multiplie par l'inverse de k dans le corps
Z/pZ (inverse qui existe car k n'est jamais un multiple de p pour
1 <= k <= p-1).

Deux precautions de calcul :
1. L'inverse de k mod p est obtenu par le petit theoreme de Fermat :
   k^{-1} = k^{p-2} (mod p).
2. Les puissances x^k sont calculees de proche en proche
   (x^k = x^{k-1} * x, reduit mod p a chaque etape) plutot qu'en
   elevant x a la puissance k directement, afin de ne jamais
   manipuler de tres grands entiers.
"""


def Hp(p: int) -> list:
    """Retourne la liste [H_p(0), H_p(1), ..., H_p(p-1)]."""
    # Table des inverses modulaires de 1, ..., p-1
```

```
    inv = [0] * p
    for k in range(1, p):
        inv[k] = pow(k, p - 2, p)  # k^{-1} mod p (Fermat)

    resultats = []
    for x in range(p):
        s = 0
        xp = 1  # represente x^0 au depart
        for k in range(1, p):
            xp = (xp * x) % p        # x^k mod p, calcule pas a pas
            s = (s + xp * inv[k]) % p
        resultats.append(s)
    return resultats


def verifie_propriete_A(p: int, valeurs: list) -> bool:
    """Verifie H(1-x) = H(x) pour tout x dans Z/pZ."""
    return all(valeurs[x] == valeurs[(1 - x) % p] for x in range(p))


def main():
    premiers = [3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]

    with open("resultats_Hp.txt", "w", encoding="utf-8") as f:
        for p in premiers:
            valeurs = Hp(p)
            f.write(f"p = {p}\n")
            f.write("x   : " + " ".join(f"{x:2d}" for x in range(p)) + "\n")
            f.write("H(x): " + " ".join(f"{v:2d}" for v in valeurs) + "\n")
            ok = verifie_propriete_A(p, valeurs)
            f.write(f"Verification (A) H(1-x)=H(x) : {'OK' if ok else 'ECHEC'}\n")
            f.write("-" * 60 + "\n\n")

    print("Resultats ecrits dans resultats_Hp.txt")


if __name__ == "__main__":
    main()
```

Résultat d'exécution du programme

```
p = 3
x   :  0  1  2
H(x):  0  0  1
Verification (A) H(1-x)=H(x) : OK
------------------------------------------------------------

p = 5
x   :  0  1  2  3  4
H(x):  0  0  4  3  4
Verification (A) H(1-x)=H(x) : OK
------------------------------------------------------------

p = 7
```

```
x    :  0  1  2  3  4  5  6
H(x):  0  0  3  0  2  0  3
Verification (A) H(1-x)=H(x) : OK
```

```
p = 11
x    :  0  1  2  3  4  5  6  7  8  9 10
H(x):  0  0  1 10  4  5  5  5  4 10  1
Verification (A) H(1-x)=H(x) : OK
```

```
p = 13
x    :  0  1  2  3  4  5  6  7  8  9 10 11 12
H(x):  0  0  7  8  0  6  4  3  4  6  0  8  7
Verification (A) H(1-x)=H(x) : OK
```

```
p = 17
x    :  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
H(x):  0  0  8 13 11 10  7 15 15 13 15 15  7 10 11 13  8
Verification (A) H(1-x)=H(x) : OK
```

```
p = 19
x    :  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
H(x):  0  0 13  9 11  2 10 16  0 14  3 14  0 16 10  2 11  9 13
Verification (A) H(1-x)=H(x) : OK
```

```
p = 23
x    :  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
H(x):  0  0 12  6  7 20  4  5 21 10 19  3 17  3 19 10 21  5  4 20  7  6 12
Verification (A) H(1-x)=H(x) : OK
```

```
p = 29
x    :  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
       25 26 27 28
H(x):  0  0 27 15  8  6 22 16 27 15 24 22 20  8 22  1 22  8 20 22 24 15 27 16 22
       6  8 15 27
Verification (A) H(1-x)=H(x) : OK
```

```
p = 31
x    :  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
        25 26 27 28 29 30
H(x):  0  0 19 23  3  3  0 18  7 24  1 13  6 15  9 27  6 27  9 15  6 13  1 24  7
        18  0  3  3 23 19
Verification (A) H(1-x)=H(x) : OK
```

```
p = 37
x    :  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
        25 26 27 28 29 30 31 32 33 34 35 36
H(x):  0  0 35 22  9 21 21 15  1 33  7  0  3  5 23 18 19 16 11  1 11 16 19 18 23
```

```
          5  3  0  7 33  1 15 21 21  9 22 35
Verification (A) H(1-x)=H(x) : OK
-------------------------------------------------------------------------------


p = 41
x   :  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
       25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
H(x):  0  0 36 24  2 32 11  5 35 10  8 20 38 13 33 28 37 25 31 40 12 23 12 40 31
       25 37 28 33 13 38 20  8 10 35  5 11 32  2 24 36
Verification (A) H(1-x)=H(x) : OK
-------------------------------------------------------------------------------


p = 43
x   :  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
        25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
H(x):  0  0 36  1 21 33  5  0 35  5 11 27 19 29  7 14  6  1  2  6  9 22 25 22  9
        6  2  1  6 14  7 29 19 27 11  5 35  0  5 33 21  1 36
Verification (A) H(1-x)=H(x) : OK
-------------------------------------------------------------------------------


p = 47
x   :  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
        25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
H(x):  0  0  6  8 10 28 41 25  1 12  3 20 21  7  2 42 13 19 11 32 27 30 16  4 44
        4 16 30 27 32 11 19 13 42  2  7 21 20  3 12  1 25 41 28 10  8  6
Verification (A) H(1-x)=H(x) : OK
-------------------------------------------------------------------------------
```