Bruit de quantification

Préparation à l'Agrégation de Physique

Introduction et objectifs

Numériser un signal, c'est le remplacer par une suite de nombres entiers : la tension est *arrondie* sur un nombre fini 2^n de niveaux. Cette opération ajoute au signal une erreur, le **bruit de quantification**, qui fixe la dynamique maximale d'une chaîne d'acquisition. Le montage consiste à numériser un même signal sinusoïdal avec un convertisseur analogique-numérique (CAN) dont on fait varier le nombre de bits n, puis à mesurer l'évolution du bruit avec n et à la comparer au modèle théorique.

Objectifs expérimentaux :

- visualiser l'effet du nombre de bits sur un signal sinusoïdal ;
- mesurer la puissance du bruit de quantification en fonction de n et la comparer à $q^2/12$;
- vérifier la décroissance de 6 dB par bit et en déduire le rapport signal sur bruit et le nombre effectif de bits ;
- identifier la limite de la mesure (plancher de bruit instrumental).

I. Rappels théoriques

Quantum et erreur de quantification

Un CAN à n bits de pleine échelle $U_{\max}$ découpe la plage $[0, U_{\max}]$ en $2^n - 1$ intervalles de largeur

$$\boxed{q = \frac{U_{\max}}{2^n - 1}}.$$

La tension u est remplacée par un multiple entier de q. L'erreur $\varepsilon = u_{\text{num}} - u$ est bornée par q en valeur absolue. Lorsque le signal balaie plusieurs quanta de façon « aléatoire » (signal riche, non corrélé au pas), on modélise ε par une variable **uniforme** de largeur q, indépendante du signal. Sa variance est alors

$$\sigma_q^2 = \int_{-q/2}^{q/2} \frac{\varepsilon^2}{q}\,\mathrm{d}\varepsilon = \frac{q^2}{12}.$$

Rapport signal sur bruit

Pour une sinusoïde d'amplitude A, la puissance du signal est $A^2/2$ et

$$\mathrm{SNR} = 10\log_{10}\frac{A^2/2}{q^2/12}.$$

Pour une sinusoïde d'amplitude pleine échelle ($A = 2^n q/2$) on obtient la formule classique

$$\mathrm{SNR}_{\mathrm{dB}} \simeq 6{,}02\,n + 1{,}76.$$

La puissance de bruit $q^2/12 \propto 2^{-2n}$ diminue donc de $10\log_{10} 4 \simeq 6{,}02$ dB à chaque bit supplémentaire : c'est la loi que l'on cherche à vérifier. En inversant la formule pour un SNR mesuré, on définit le **nombre effectif de bits** $\mathrm{ENOB} = (\mathrm{SNR} - 1{,}76)/6{,}02$.

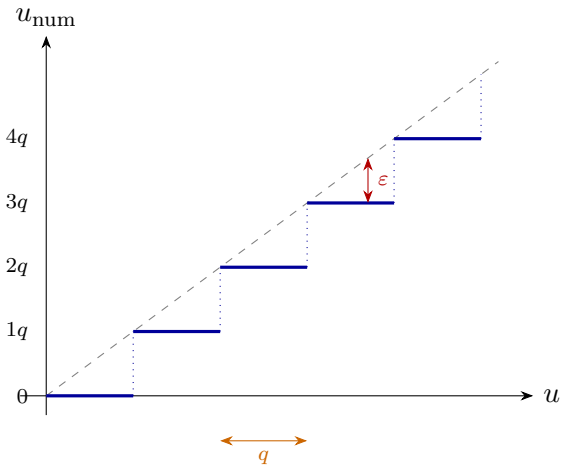

FIGURE 1 – Caractéristique d'un CAN idéal à troncature (seules cinq marches sont dessinées) et erreur de quantification associée. La droite en pointillés est la caractéristique idéale.

II. Dispositif expérimental et protocole

Principe du montage

Un générateur basse fréquence (GBF) délivre une tension sinusoïdale de fréquence 1 Hz superposée à un décalage continu (offset) afin de rester dans la plage 0–5 V de l'entrée de la carte Arduino. Le programme de la carte lit la tension sur son CAN de 10 bits et *masque* les bits de poids faible pour simuler un convertisseur à n bits ($n = 2$ à 10). Les échantillons sont envoyés à l'ordinateur et traités en Python.

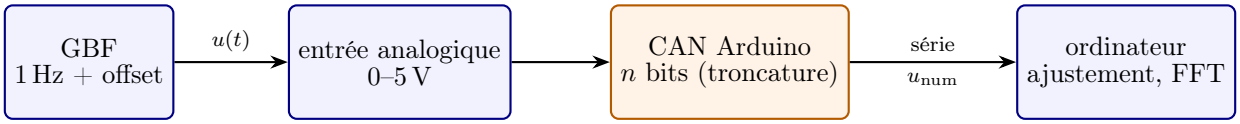

FIGURE 2 – Chaîne d'acquisition : le CAN de la carte est utilisé à n bits en ignorant les bits de poids faible.

Protocole

1. Régler le GBF sur une sinusoïde de fréquence 1 Hz, d'amplitude voisine de 2,4 V, avec un offset d'environ 2,6 V (tension toujours comprise entre 0 et 5 V).
2. Pour chaque $n = 2, \ldots, 10$, acquérir $N = 100$ points avec une période d'échantillonnage $T_e = 20$ ms, soit une durée de 2 s (environ deux périodes).
3. Enregistrer les neuf fichiers `n2.csv` à `n10.csv` (colonnes : temps en ms, tension en V).

III. Acquisition : approche qualitative

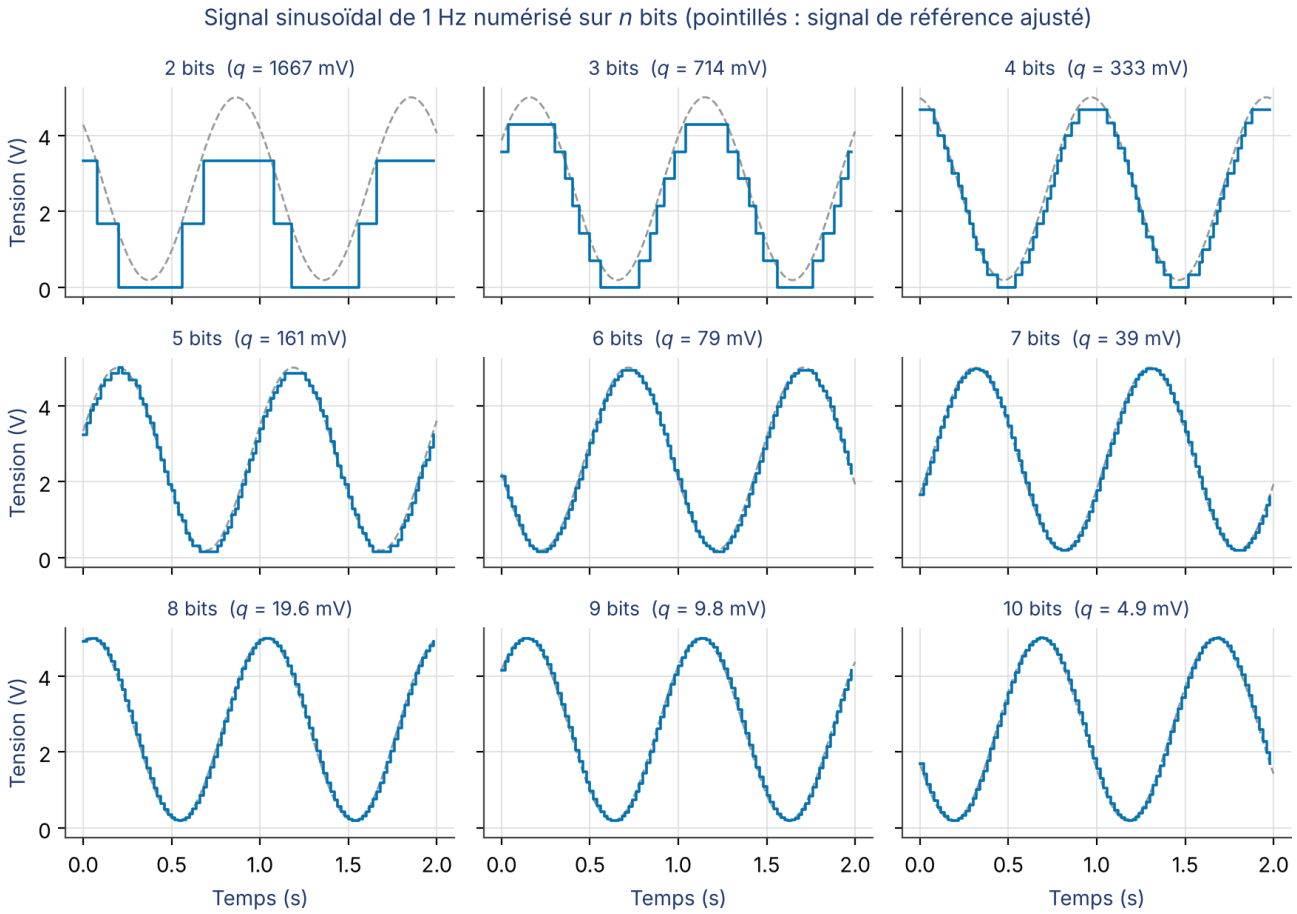

FIGURE 3 – Signal numérisé pour $n = 2$ à 10 bits (traits pleins en escalier) et sinusoïde de référence ajustée (pointillés). En dessous de 6 bits la marche d'escalier est nettement visible ; au-delà elle devient imperceptible à l'échelle de la figure.

On retrouve bien le nombre de niveaux attendu : le fichier à 2 bits ne contient que 3 valeurs (0, 1,67 et 3,33 V), celui à 3 bits en contient 7 avec un pas de 0,714 V, etc. Le niveau maximal n'atteint pas 5 V pour les faibles n (par exemple 4,67 V à 4 bits) car le signal ne dépasse pas 5 V et que la troncature ramène la valeur vers le bas.

IV. Traitement des données

Méthode retenue : résidu d'un ajustement sinusoïdal

Le bruit de quantification est ce qui reste quand on retire le signal utile. On ajuste donc sur chaque acquisition une sinusoïde $u(t) = u_0 + A\sin(2\pi f_0 t + \varphi)$:

- l'amplitude et la fréquence sont déterminées *une fois pour toutes* sur les trois acquisitions les plus fines (8, 9 et 10 bits) : $A = 2{,}399\,\text{V}$ et $f_0 = 1{,}0097\,\text{Hz}$;
- pour chaque n, seuls la phase et l'offset sont ajustés ;
- la puissance de bruit est la variance du résidu, $\sigma^2 = \langle (u_{\text{num}} - u_{\text{fit}})^2 \rangle$.

On s'affranchit ainsi de la fuite spectrale (la fenêtre contient 2,02 périodes, pas un nombre entier) qui contamine une estimation du bruit par une FFT à fenêtre rectangulaire.

Listing 1 – Traitement des données

```
import numpy as np
```

```
import pandas as pd
from scipy.optimize import curve_fit

UMAX, FE, N = 5.0, 50.0, 100       # pleine echelle (V), echantillonnage (Hz), points

def sinus(t, A, f, ph, off):
    return off + A * np.sin(2 * np.pi * f * t + ph)

# --- Lecture des 9 acquisitions (n = 2 a 10 bits) --------------------
data = {}
for n in range(2, 11):
    d = pd.read_csv(f"n{n}.csv", delimiter=";")
    data[n] = (d.iloc[:, 0].values / 1000.0, d.iloc[:, 1].values)   # s, V

# --- Signal de reference : ajustement libre sur les 3 meilleures -----
def ajuste(t, u, modele, p0s, bornes):
    best = None
    for p0 in p0s:
        try:
            p, _ = curve_fit(modele, t, u, p0=p0, bounds=bornes, maxfev=20000)
        except RuntimeError:
            continue
        r = u - modele(t, *p)
        if best is None or r.std() < best[1].std():
            best = (p, r)
    return best

phases = np.linspace(0, 2*np.pi, 12, endpoint=False)
b_libre = ([1.0, 0.9, -10.0, 1.0], [4.0, 1.1, 20.0, 4.0])      # A > 0 : evite le signe parasite
libre = [ajuste(*data[n], sinus, [[2.4, 1.01, ph, 2.6] for ph in phases], b_libre)[0]
         for n in (8, 9, 10)]
A0, f0 = np.mean([p[0] for p in libre]), np.mean([p[1] for p in libre])

# --- Bruit = residu apres ajustement (phase et offset libres) --------
def modele_fixe(t, ph, off):
    return sinus(t, A0, f0, ph, off)

res = {}
for n, (t, u) in data.items():
    p, r = ajuste(t, u, modele_fixe, [[ph, 2.6] for ph in phases], ([-10.0, 0.0], [20.0, 5.0]))
    q = UMAX / (2**n - 1)
    s2 = np.var(r)                                          # puissance du bruit (V^2)
    res[n] = dict(q=q, s2=s2, s2_th=q**2 / 12, phase=p[0], offset=p[1],
                  snr=10*np.log10(0.5*A0**2 / s2),
                  snr_th=10*np.log10(0.5*A0**2 / (q**2 / 12)))
    res[n]["enob"] = (res[n]["snr"] - 1.76) / 6.02

# --- Regression  P_dB = a n + b  avec Monte-Carlo (n = 2 a 8) --------
ns = np.arange(2, 9)
P_dB = np.array([10*np.log10(res[n]["s2"]) for n in ns])
u_dB = 0.6          # incertitude sur P_dB : sqrt(2/(N-1)) = 14 % sur une variance
rng = np.random.default_rng(0)
pentes = [np.polyfit(ns, rng.normal(P_dB, u_dB), 1)[0] for _ in range(10000)]
a_moy, a_std = np.mean(pentes), np.std(pentes)

# --- Modele  sigma^2 = q^2/12 + sigma0^2  (plancher instrumental) ----
from scipy.optimize import minimize_scalar
n_fit = np.arange(3, 11)
mesure_dB = np.array([10*np.log10(res[n]["s2"]) for n in n_fit])
q2_12 = np.array([res[n]["s2_th"] for n in n_fit])
ecart = lambda s0: np.sum((mesure_dB - 10*np.log10(q2_12 + s0))**2)
sigma0 = minimize_scalar(ecart, bounds=(0, 1e-4), method="bounded").x ** 0.5

if __name__ == "__main__":
    print(f"A = {A0:.3f} V, f = {f0:.4f} Hz")
    for n in range(2, 11):
```

```
        r = res[n]
        print(f"n={n:2d} q={r['q']*1e3:7.1f} mV  sigma={r['s2']**.5*1e3:6.1f} mV"
              f" (th {r['s2_th']**.5*1e3:6.1f})  SNR={r['snr']:5.1f} dB"
              f" (th {r['snr_th']:5.1f})  ENOB={r['enob']:.2f}")
    print(f"plancher : sigma0 = {sigma0*1e3:.1f} mV")
    print(f"pente = {a_moy:.2f} +/- {a_std:.2f} dB/bit (theorie : -6.02)")
```

Résultats

n	q (mV)	σ (mV)	$q/\sqrt{12}$ (mV)	$10\log\sigma^2$ (dB)	SNR (dB)	SNR th. (dB)	ENOB
2	1666,7	504,4	481,1	−5,9	10,5	10,9	1,46
3	714,3	204,7	206,2	−13,8	18,4	18,3	2,76
4	333,3	98,3	96,2	−20,2	24,7	24,9	3,82
5	161,3	47,2	46,6	−26,5	31,1	31,2	4,88
6	79,4	23,7	22,9	−32,5	37,1	37,4	5,87
7	39,4	13,1	11,4	−37,6	42,2	43,5	6,72
8	19,6	6,5	5,7	−43,7	48,3	49,5	7,73
9	9,8	5,0	2,8	−46,0	50,6	55,6	8,12
10	4,9	3,8	1,4	−48,4	53,0	61,6	8,51

La puissance de bruit est exprimée en dB par rapport à 1 V^2. Le SNR théorique est calculé avec l'amplitude mesurée $A = 2{,}399\,\mathrm{V}$ et $q^2/12$.

Évolution du bruit avec n

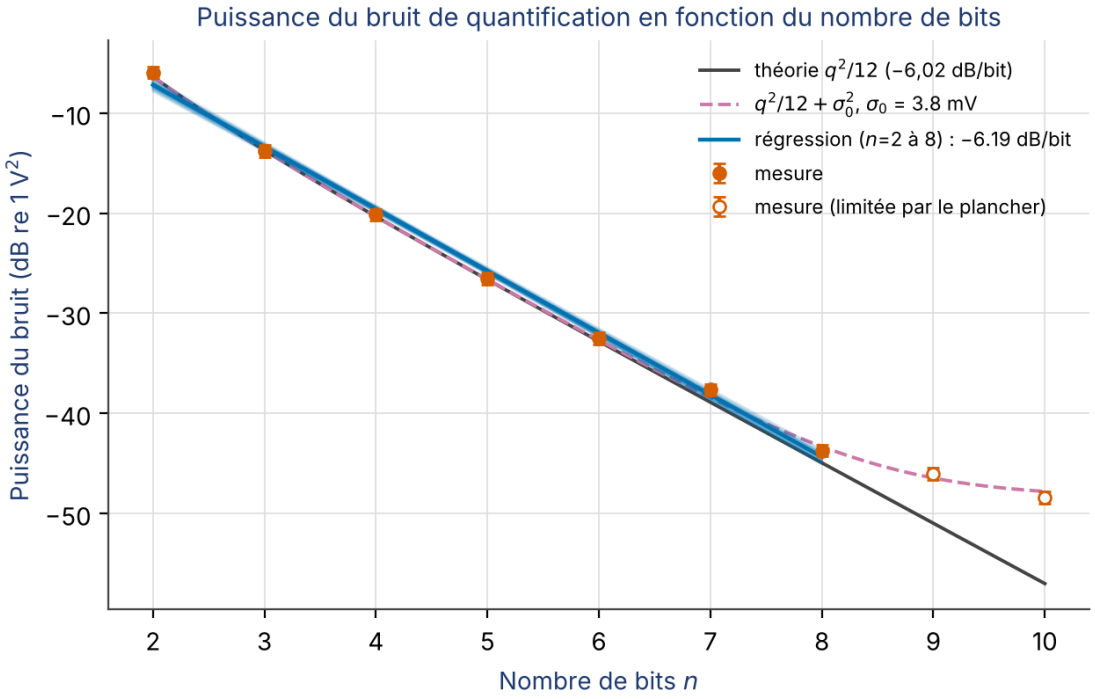

FIGURE 4 – Puissance du bruit en fonction de n. Points : mesures (barres : 0,6 dB) ; trait gris : modèle $q^2/12$; bleu : régression sur $n = 2$–8 avec 10^4 tirages Monte-Carlo ; pointillés rose : modèle avec plancher $q^2/12 + \sigma_0^2$. Les points creux ($n = 9, 10$) sont limités par le plancher.

Une régression affine $P_{\mathrm{dB}} = a\,n + b$ sur $n = 2$ à 8 donne, par Monte-Carlo (incertitude de 0,6 dB par point, voir ci-dessous),

$$\boxed{a = -6{,}19 \pm 0{,}11\ \mathrm{dB/bit}}$$

en accord à moins de deux écarts-types avec la valeur théorique $-6{,}02\,\mathrm{dB/bit}$ ($|-6{,}19+6{,}02|/0{,}11 \simeq 1{,}6$). À partir de 9 bits, la mesure s'écarte de la droite : si l'on inclut $n = 9$ et 10, la pente tombe à $-5{,}4\,\mathrm{dB/bit}$.

Analyse spectrale

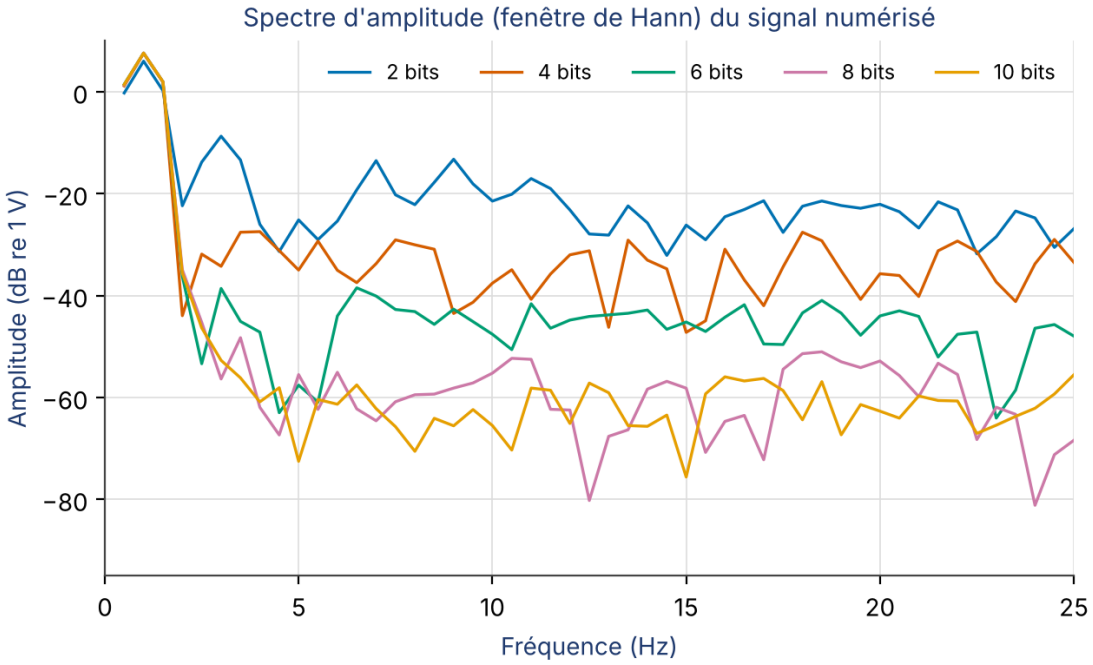

FIGURE 5 – Spectre d'amplitude (fenêtre de Hann) pour $n = 2, 4, 6, 8, 10$. Le pic à 1 Hz est commun ; le fond, dû au bruit de quantification, descend avec n.

Le spectre montre le pic du signal à 1 Hz et un fond large bande. Avec une fenêtre de Hann, la puissance moyenne du fond au-delà de 3 Hz vaut $-19{,}6$, $-32{,}6$, $-44{,}4$ et $-56{,}6\,\mathrm{dB}$ pour $n = 2, 4, 6, 8$, soit environ $-6\,\mathrm{dB}$ par bit, puis $-56{,}9$ et $-59{,}9\,\mathrm{dB}$ pour $n = 9$ et 10 : on retrouve le plancher. Ces niveaux ne se comparent pas directement aux variances du tableau (les échelles de normalisation diffèrent), seule leur évolution avec n est pertinente.

V. Analyse des incertitudes et discussion

Plancher de bruit

Au-delà de 8 bits, le bruit mesuré (5,0 puis 3,8 mV) est supérieur à $q/\sqrt{12}$ (2,8 puis 1,4 mV). On modélise un bruit indépendant de n : $\sigma^2 = q^2/12 + \sigma_0^2$. L'ajustement de σ_0 sur $n = 3$ à 10 donne $\sigma_0 \simeq 3{,}8\,\mathrm{mV}$. Plusieurs causes s'additionnent :

- **résolution des fichiers** : les tensions sont enregistrées avec deux décimales (pas de 10 mV), ce qui ajoute à lui seul un bruit de $10/\sqrt{12} \simeq 2{,}9\,\mathrm{mV}$;
- le reste, de l'ordre de $\sqrt{3{,}8^2 - 2{,}9^2} \simeq 2{,}5\,\mathrm{mV}$, vient du bruit analogique (GBF, câblage, référence de la carte) et de l'imperfection du modèle sinusoïdal ajusté (dérive de fréquence, distorsion) ;
- la formule du SNR ne peut alors plus être testée au-delà de 8 bits : le nombre effectif de bits sature vers 8,5 bits pour une carte de 10 bits.

Incertitudes

Avec $N = 100$ points, la variance du résidu est connue à $\sqrt{2/(N-1)} \simeq 14\,\%$, soit $10\log_{10}(1 \pm 0{,}14) \simeq \pm 0{,}6\,\mathrm{dB}$: c'est l'incertitude retenue sur chaque point de la figure précédente, et l'origine de l'incertitude

de 0,11 dB/bit sur la pente. L'estimation suppose des résidus indépendants ; à faible nombre de bits l'erreur est corrélée au signal (marches d'escalier), ce qui explique l'écart de 5 % entre σ et $q/\sqrt{12}$ à 2 bits (504 contre 481 mV) : le modèle du bruit blanc uniforme n'est qu'approché quand q est comparable à l'amplitude.

Validité du modèle

- Le modèle $q^2/12$ suppose une erreur uniforme et décorrélée du signal : bien vérifié pour $n \gtrsim 4$, moins pour $n = 2, 3$.
- L'acquisition ne dure que 2 s avec 100 points : l'estimation de variance est limitée et la fenêtre contient un nombre non entier de périodes.
- La formule $6{,}02n + 1{,}76$ suppose une amplitude pleine échelle ; ici $A = 2{,}4$ V pour une demi-échelle de 2,5 V, d'où un SNR inférieur d'environ 0,4 dB.

Pistes d'amélioration

- Enregistrer les tensions avec davantage de chiffres (ou directement les codes entiers du CAN) pour repousser le plancher.
- Allonger l'acquisition et la synchroniser sur un nombre entier de périodes pour utiliser la FFT sans fenêtre.

Conclusion

La numérisation d'une sinusoïde de 1 Hz sur $n = 2$ à 10 bits met en évidence le bruit de quantification. Sa puissance décroît de $-6{,}19 \pm 0{,}11$ dB par bit entre 2 et 8 bits, en accord avec la valeur théorique $-6{,}02$ dB issue de $\sigma^2 = q^2/12$, et l'erreur est compatible avec une loi uniforme, de moyenne $-q/2$ car le convertisseur tronque. Au-delà de 8 bits la mesure est limitée par un plancher de bruit instrumental de l'ordre de 3,8 mV (dont 2,9 mV dus à la résolution des fichiers), qui plafonne le nombre effectif de bits à environ 8,5.

Je n'espère pas, mais des coquilles peuvent se glisser dans ce document...
