Можно ли считать фрактальную размерность Хигучи по ЕКГ разной длительности для определения биологического возраста или нужно привести все к одной длине?

**Програма ChatGPT сказала:**

Отличный, очень правильный вопрос 👍

Когда речь идёт о вычислении **фрактальной размерности Хигучи (HFD)** по сигналам **разной длительности**, есть важные моменты:

**🔹 1. Зависимость HFD от длины сигнала**

* В идеале **фрактальная размерность — масштабно-инвариантная** величина (она не должна зависеть от длины ряда).
* Но на практике (особенно на конечных биосигналах, с шумами, трендами и нестационарностью) значение HFD **может смещаться** при слишком коротких или слишком длинных временных рядах.
* Если сигналы сильно различаются по длительности (например, 10 секунд и 10 минут), то значения HFD могут стать **несравнимыми**.

**🔹 2. Что делают в исследованиях**

* Обычно **нормализуют длительность**: берут одинаковые отрезки (например, 5–10 секунд или 1000 отсчётов ЭКГ).
* Либо применяют **скользящее окно** и усредняют значения HFD по всем окнам одинакового размера.
* Так устраняется эффект разной длины сигнала, и показатели становятся сопоставимыми.

**🔹 3. Для задач типа «биологический возраст»**

* Если цель — находить **биомаркер** (например, возраст или состояние), то **обязательно нужно привести записи к одной длине** или хотя бы к одинаковому количеству сердечных циклов.
* Иначе получится, что часть различий в HFD будет объясняться не возрастом, а просто разницей в длительности записи.

✅ **Вывод**:

* Формально считать HFD можно и на сигналах разной длины.
* Но **для сравнения между людьми (например, для оценки биологического возраста)** сигналы надо привести к одной длительности (или одинаковому числу сердечных циклов).

Как посчитать фрактальную размерность Хигучи с помощью окна на сигналах ЭКГ разной длительности?

**Програма ChatGPT сказала:**

Обробка тривала 13s

Да—это делают «скользящим окном», чтобы сравнивать записи разной длительности корректно. Ниже — короткий, практичный рецепт + готовый код.

**Что делать в общих чертах**

1. Подготовка сигнала: band-pass (напр. 0.5–40 Гц), удаление дрейфа (high-pass 0.5 Гц или полином), нормализация амплитуды (z-score) — чтобы HFD не «ловил» шум/дрейф.
2. Выбор окна и шага:
   * **По времени**: одинаковая длительность окна (напр. 5 с) и шаг (напр. 2.5 с).
   * **По сердечным циклам**: одинаковое число циклов (напр. 100 комплексов) и шаг (напр. 20 циклов). Это лучше для биосравнений, если есть R-пики.
3. Выбор параметров Хигучи:
   * kmax обычно 6–10 для «коротких» окон, 10–20 для «длинных». Чем больше окно, тем смелее kmax.
   * Минимум точек в окне: ориентируйтесь, чтобы N было как минимум в 10–20 раз больше kmax.
4. Счёт HFD в каждом окне → получается временной ряд HFD. Для сравнения между людьми берите **одно и то же окно/число циклов и шаг**, дальше усредняйте (mean/median) или анализируйте распределение.

**Готовый код (Python / NumPy)**

Код даёт:

* higuchi\_fd(x, kmax) — размерность Хигучи.
* Оконный подсчёт **по времени**: windowed\_hfd\_time(...).
* Оконный подсчёт **по циклам**: windowed\_hfd\_cycles(...) (если есть индексы R-пиков).

import numpy as np

def higuchi\_fd(x: np.ndarray, kmax: int = 10) -> float:

"""

Higuchi's Fractal Dimension.

x: 1D сигнал

kmax: максимальный масштаб (обычно 6–20)

"""

x = np.asarray(x, dtype=float)

N = x.size

if N < 2:

return np.nan

kmax = min(kmax, max(2, N // 4)) # простая защита от слишком большого kmax

Lk = []

ks = np.arange(1, kmax + 1)

for k in ks:

Lm\_vals = []

for m in range(k):

idx = np.arange(m, N, k)

if idx.size < 2:

continue

diffs = np.abs(np.diff(x[idx]))

nseg = idx.size - 1

# классическая нормировка Хигучи:

Lm = (diffs.sum() \* (N - 1)) / (nseg \* k)

Lm\_vals.append(Lm)

if len(Lm\_vals) == 0:

Lk.append(np.nan)

else:

Lk.append(np.mean(Lm\_vals))

Lk = np.array(Lk, dtype=float)

# выбросы/NaN из-за коротких окон — убираем:

mask = np.isfinite(Lk) & (Lk > 0)

ks\_eff = ks[mask]

Lk\_eff = Lk[mask]

if ks\_eff.size < 2:

return np.nan

# Регистрируем log(L(k)) на log(1/k); наклон = D

y = np.log(Lk\_eff)

xlog = -np.log(ks\_eff) # log(1/k)

slope, intercept = np.polyfit(xlog, y, 1)

D = slope

return float(D)

def windowed\_hfd\_time(x: np.ndarray, fs: float, win\_sec: float = 5.0, step\_sec: float = 2.5, kmax: int = 10):

"""

Оконный HFD по фиксированному времени.

Возвращает: t\_centers (в секундах), hfd\_values (по окнам)

"""

N = len(x)

win = int(round(win\_sec \* fs))

step = int(round(step\_sec \* fs))

if win < 2 or step < 1:

return np.array([]), np.array([])

starts = np.arange(0, N - win + 1, step)

hfd\_vals = []

t\_centers = []

for s in starts:

seg = x[s:s+win]

hfd\_vals.append(higuchi\_fd(seg, kmax=kmax))

center = (s + win/2) / fs

t\_centers.append(center)

return np.array(t\_centers), np.array(hfd\_vals)

def windowed\_hfd\_cycles(x: np.ndarray, rpeaks\_idx: np.ndarray, n\_cycles: int = 100, step\_cycles: int = 20, kmax: int = 10):

"""

Оконный HFD по фиксированному числу сердечных циклов.

rpeaks\_idx: индексы R-пиков в отсчетах (возрастающий массив)

n\_cycles: сколько последовательных комплексов берём в окно

step\_cycles: шаг по циклам

Возвращает: centers\_idx (индексы центра окна, можно перевести в секунды), hfd\_values

"""

rpeaks\_idx = np.asarray(rpeaks\_idx, dtype=int)

if rpeaks\_idx.size < n\_cycles + 1:

return np.array([]), np.array([])

# формируем пары границ по R-пикам: [R\_i, R\_{i+n\_cycles}]

starts = np.arange(0, rpeaks\_idx.size - n\_cycles - 0, step\_cycles)

hfd\_vals = []

centers = []

for i in starts:

a = rpeaks\_idx[i]

b = rpeaks\_idx[i + n\_cycles]

if b - a < 2:

hfd\_vals.append(np.nan)

centers.append((a + b) // 2)

continue

seg = x[a:b]

hfd\_vals.append(higuchi\_fd(seg, kmax=kmax))

centers.append((a + b) // 2)

return np.array(centers), np.array(hfd\_vals)

**Как применить на практике**

**Вариант А: одинаковое время**

# x — ваш канал ЭКГ, fs — частота дискретизации, Гц

t\_centers, hfd = windowed\_hfd\_time(x, fs, win\_sec=5.0, step\_sec=2.5, kmax=10)

# Сравниваем между людьми среднее или медиану hfd: np.nanmean(hfd)

**Вариант Б: одинаковое число циклов** (если есть R-пики):

centers\_idx, hfd = windowed\_hfd\_cycles(x, rpeaks\_idx, n\_cycles=100, step\_cycles=20, kmax=10)

# Для времени: t\_centers = centers\_idx / fs

**Практические советы**

* Для межчеловеческих сравнений (в т.ч. «биологический возраст»):
  + Фиксируйте **одинаковый протокол**: те же фильтры, те же окна/число циклов, одинаковый kmax.
  + Избавляйтесь от артефактов: окна с выбросами (амплитуда/EMG/движение) лучше помечать и исключать.
  + Итоговый признак делайте агрегированным: median(HFD по окнам), плюс вариативность (IQR/SD).
* Если сигнал короткий, уменьшайте kmax (например, до 6–8) — иначе регрессия по log-масштабам будет нестабильной.
* Для сопоставимости разных людей предпочтительнее окна **по числу циклов**; если остаётесь во времени, держите схожую ЧСС (например, в покое) или контролируйте её как ковариату в модели.

Если хочешь, могу адаптировать код под твой стек (MNE/NeuroKit/Matlab) или добавить автоматическую фильтрацию и детект R-пиков.