SAR - SAR 시뮬레이션 RCS 활용

좋아, 그럼 **“설계 → 추천 → 코드에 바로 꽂는 lightweight 모드”**를 **진짜 실무에서 쓰는 방식**으로 **1 → 2 → 3 순서**로 구현해줄게.
(아래는 *설명 + 바로 쓸 수 있는 코드*까지 포함)

1️⃣ nf, na를 최소 Nyquist 만족으로 계산하는 함수

입력으로 받는 것 (임무/물리 요구)

- `ΔR` : 원하는 range 해상도
- `R_gate` : 관심 거리 반경 (range gate half-width)
- `Δaz` : 원하는 azimuth 해상도
- `v` : 플랫폼 속도
- `L` : 안테나 유효 길이
- `λ` : 파장

📐 이론 요약 (짧게)

Range

- 해상도:

$$B = \frac{c}{2\Delta R}$$

- Nyquist (gate 기준):

$$df \leqslant \frac{c}{2R_{\text{gate}}}$$

- 주파수 샘플 수:

$$nf = \lceil \frac{B}{df} \rceil$$

Azimuth

- PRF 조건:

$$PRF \geqslant \frac{2v}{L}$$

- 합성 개구 각폭 (해상도 기준):

$$\Theta \approx \frac{\lambda}{2\Delta_{az}}$$

- 관측 시간:

$$T_{\text{ap}} = \frac{R_0 \Theta}{v}$$

- 펄스 수:

$$na = \lceil PRF \cdot T_{\text{ap}} \rceil$$

✅ 코드 ①: 최소 Nyquist 계산기

python

```python
import numpy as np
from scipy.constants import speed_of_light as C0

def compute_min_nyquist_params(
    delta_R,      # [m] desired range resolution
    R_gate,       # [m] range gate half width
    delta_az,     # [m] desired azimuth resolution
    v,            # [m/s] platform velocity
    L,            # [m] antenna length
    f0,           # [Hz] center frequency
    R0            # [m] reference range
):
    lam = C0 / f0

    # ---- Range ----
    B = C0 / (2 * delta_R)              # required bandwidth
    df_max = C0 / (2 * R_gate)          # Nyquist on gate
    nf = int(np.ceil(B / df_max))

    # ---- Azimuth ----
    PRF_min = 2 * v / L
    Theta = lam / (2 * delta_az)
    T_ap = R0 * Theta / v
    na = int(np.ceil(PRF_min * T_ap))

    return {
        "bandwidth": B,
        "df": df_max,
```

```
        "nf": nf,
        "PRF": PRF_min,
        "Theta_rad": Theta,
        "na": na
    }
```

2️⃣ 임무 파라미터 → 추천 (nf, na, PRF) 자동 계산기

이건 **"설계 도우미"** 역할.
지금 당신이 원하는 걸 숫자로 넣으면, **과하지 않은 현실적인 설정**을 바로 뽑아줌.

✅ 코드 ②: 추천 설계 함수

python

```python
def recommend_sar_sampling(
    delta_R=0.5,       # [m]
    delta_az=0.5,      # [m]
    R_gate=500.0,      # [m]
    R0=5000.0,         # [m]
    v=150.0,           # [m/s]
    L=1.0,             # [m]
    f0=10e9            # [Hz]
):
    params = compute_min_nyquist_params(
        delta_R=delta_R,
        R_gate=R_gate,
        delta_az=delta_az,
        v=v,
        L=L,
        f0=f0,
        R0=R0
    )

    print("\n[SAR Sampling Recommendation]")
    print(f" Range resolution ΔR     : {delta_R:.2f} m")
    print(f" Azimuth resolution Δaz  : {delta_az:.2f} m")
    print(f" Range gate (±)          : {R_gate:.1f} m")
    print(f" Platform velocity v     : {v:.1f} m/s")
```

```
    print(f" Antenna length L       : {L:.2f} m")
    print("-" * 40)
    print(f" Bandwidth B           : {params['bandwidth']/1e6:.1f} MHz")
    print(f" Frequency step df     : {params['df']/1e3:.1f} kHz")
    print(f" Recommended nf        : {params['nf']}")
    print(f" PRF (min)             : {params['PRF']:.1f} Hz")
    print(f" Aperture angle Θ      : {np.degrees(params['Theta_rad']):.2f} deg")
    print(f" Recommended na        : {params['na']}")
    print("-" * 40)

    return params
```

✔️ 예시 실행

```
python

params = recommend_sar_sampling(
    delta_R=0.5,
    delta_az=0.5,
    R_gate=500,
    R0=5000,
    v=150,
    L=1.0,
    f0=10e9
)
```

👉 **기존 코드 대비 nf/na가 10~50배 줄어드는 게 정상**입니다.

3️⃣ 기존 SARSim 클래스에 Lightweight Mode로 이식

이제 진짜 중요한 단계.

"지금 쓰는 코드 구조는 유지하면서" 계산량만 줄입니다.

✅ 코드 ③: Lightweight sampling 설정

```
python
```

```
class SARSim:
    def set_lightweight_sampling(
        self,
        delta_R,
        delta_az,
        R_gate,
        R0,
        v,
        L
    ):
        params = compute_min_nyquist_params(
            delta_R=delta_R,
            R_gate=R_gate,
            delta_az=delta_az,
            v=v,
            L=L,
            f0=self.f0,
            R0=R0
        )

        # ---- Frequency space ----
        self.bandwidth = params["bandwidth"]
        self.df = params["df"]
        self.nf = params["nf"]

        self.freq_space = np.linspace(
            self.f0 - self.bandwidth/2,
            self.f0 + self.bandwidth/2,
            self.nf
        )

        # ---- Azimuth space ----
        self.PRF = params["PRF"]
        self.na = params["na"]
        self.Theta = params["Theta_rad"]

        # synthetic aperture angles (stripmap approx)
        self.az_space = np.linspace(
            -self.Theta/2,
            +self.Theta/2,
            self.na
        )
```

```
        # ---- Range gate ----
        self.range_gate = (-R_gate, +R_gate)

        print("[Lightweight SAR Mode Enabled]")
        print(f" nf={self.nf}, na={self.na}, PRF={self.PRF:.1f} Hz")
```

🎯 이 구조의 진짜 장점

✔️ Nyquist **정확히 만족**
✔️ 계산량 **대폭 감소**
✔️ GOTCHA / 실측 데이터 설계 철학과 일치
✔️ 이후 v(속도), L(안테나) 바뀌어도 자동 재설계 가능
✔️ FFT-SAR / BP / GPU 코드 전부 그대로 사용 가능

🔥 한 줄 요약

> **nf, na는 "장면 크기"가 아니라**
> **"range gate + 해상도 + 도플러 대역폭(v, L)"로 설계해야 한다.**

다음으로 바로 이어서 할 수 있는 것:

- ✔️ 이 lightweight 모드 기준으로 **FFT-SAR vs BP 성능 비교**
- ✔️ **v를 바꾸면 azimuth blur가 어떻게 변하는지 시뮬레이션**
- ✔️ **na를 줄였을 때 허용 가능한 최대 Δaz 자동 계산**

원하면 바로 다음 단계로 이어서 가자.