Plan Projektu: Analiza Porównawcza Algorytmów Głosujących w Systemach Odpornych na Uszkodzenia

Grupa: NiDUC-g4-Korwin-Kula-Majer

6 listopada 2025

1 Cel i Zakres Projektu (Etap 1)

Celem projektu jest implementacja i analiza porównawcza algorytmów głosujących stosowanych w systemach z redundancją (N-Modular Redundancy), ze szczególnym uwzględnieniem algorytmu **Smoothing Voter**. Projekt bazuje na metodologii i eksperymentach opisanych we wspomnianym artykule.

1.1 Model Systemu

System bazuje na architekturze TMR (Triple Modular Redundancy) oraz stanowisku testowym, składającym się z:

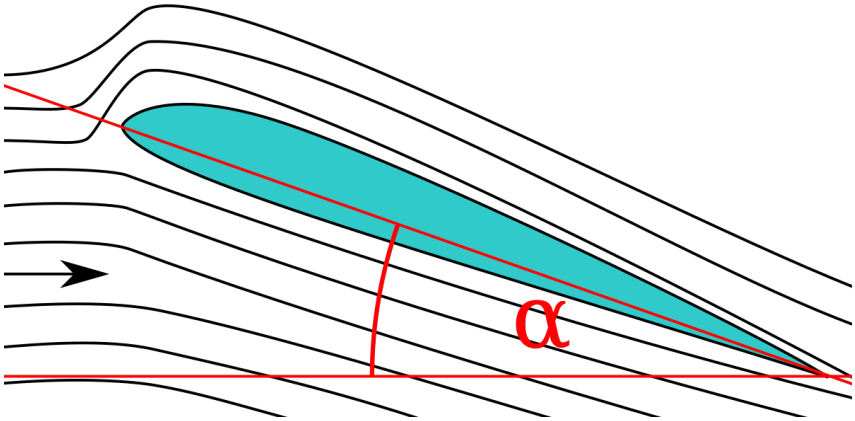
- **Generatora sygnału wzorcowego ($u(t)$):** Generuje idealny sygnał wejściowy. Zgodnie z artykułem, przyjmujemy:
$$u(t) = 100 \cdot \sin(t) + 100$$
- **Modułów zakłóceń:** Trzy moduły symulujące zakłócenia (błędy) wprowadzane do idealnego sygnału.
- **Modułu Głosującego:** Implementacja badanych algorytmów.
- **Komparatora:** Porównuje wynik głosowania z sygnałem wzorcowym $u(t)$ w celu klasyfikacji wyniku.

1.2 Studium Przypadku

Aby osadzić projekt w realnym kontekście inżynierskim, nasza symulacja będzie modelować krytyczny dla bezpieczeństwa (safety-critical) podsystem awioniki: system odczytu **kąta natarcia (Angle of Attack – AoA)** w samolocie.

W większości nowoczesnych samolotów pasażerskich i wojskowych (np. Boeing 737/787, Airbus A320/A350) system AoA nie opiera się na jednym, ale na **wielu niezależnych czujnikach** (łopatkach), zazwyczaj dwóch lub trzech, umieszczonych na różnych stronach kadłuba. Jest to podstawowa forma redundancji. Czujniki AoA są narażone na kilka typów uszkodzeń:

- **Uszkodzenia stałe (Hard Faults)**: Awaria mechaniczna łopatki, awaria elektroniki czujnika, utrata zasilania.
- **Uszkodzenia przejściowe (Transient Faults)**: Zablokowanie łopatki przez lód, zanieczyszczenia, piasek (np. podczas startu/lądowania), co powoduje zafałszowanie odczytu.
- **Błędy obliczeniowe**: Błędy w jednostce przetwarzającej dane z czujników.



Rysunek 1: Kąt między cięciwą skrzydła samolotu a kierunkiem napływającego powietrza

Błędny odczyt AoA może prowadzić do przeciągnięcia i katastrofy. Dlatego w systemie TMR (Triple Modular Redundancy) stosuje się trzy niezależne czujniki AoA. Nasz model symulacyjny jest mapowany na ten problem w następujący sposób:

- **Generator Sygnału ($u(t)$):** Reprezentuje *idealny, fizyczny* kąt natarcia samolotu w danym momencie.
- **Zakłócenia (3 moduły):** Modelują trzy *realne czujniki AoA*, które mogą ulec awarii (np. na skutek oblodzenia, uderzenia czy zakłóceń EMI).
- **Moduł Głosujący (Voter):** Odpowiada logice *Komputera Sterowania Lotem (Flight Control Computer)*, który musi wybrać jedną, zaufaną wartość AoA na podstawie trzech (potencjalnie sprzecznych) odczytów.

W tym kontekście, nasze miary wyjściowe nabierają krytycznego znaczenia:

- **Poprawne wyjście (n_c/n):** Dostępność (Availability). Autopilot działa poprawnie.
- **Niepoprawne wyjście (n_{ic}/n):** Błąd Katastroficzny (Safety Failure). Komputer podaje autopilotowi błędną wartość AoA, co grozi katastrofą. **Minimalizacja tej miary jest naszym priorytetem.**
- **Brak decyzji (n_d/n):** Błąd Łagodny (Benign Failure). Komputer (np. MAJ lub SM) wykrywa niespójność, rozłącza autopilota i alarmuje pilota. Jest to pożądane zachowanie w przypadku awarii, gdyż stawia bezpieczeństwo ponad dostępność.

1.3 Parametry Wejściowe Symulacji

- **Amplituda błędu (e_{max}):** Główny parametr badań, określający maksymalną wartość perturbacji.
- **Model błędu:** Scenariusze z 2 lub 3 uszkodzonymi modułami; błędy permanentne lub transjentowe.
- **Próg ϵ (Voter threshold):** Próg akceptacji dla algorytmów MAJ i SM (przyjęty 0.5).
- **Próg β (Smoothing threshold):** Kluczowy parametr algorytmu Smoothing Voter.

2 Implementacja Symulatora (Etap 2)

Symulator zostanie zaimplementowany w języku **Python**, z wykorzystaniem bibliotek **NumPy** (do operacji na sygnałach i zakłóceniach) oraz **Matplotlib** (do wizualizacji wyników).

Zaimplementowane zostaną następujące algorytmy głosujące:

1. **Majority Voter (MAJ):** Wymaga zgodności $\lceil (n+1)/2 \rceil$ wejść (w naszym przypadku 2-z-3) w ramach progu ϵ.
2. **Weighted Average (WA):** Oblicza średnią ważoną.
3. **Smoothing Voter (SM):** Nasz główny obiekt badań. W przypadku braku większości (wg MAJ), algorytm wybiera kandydata x_k najbliższego poprzedniemu poprawnemu wyjściu X_{prev}, pod warunkiem spełnienia kryterium wygładzania:

$$\text{IF } d(x_k, X_{prev}) \leq \beta \text{ THEN Output} = x_k \text{ ELSE Output} = \text{No Result}$$

3 Badania Symulacyjne (Etap 3)

Przeprowadzone zostaną eksperymenty symulacyjne (metodą Monte Carlo) dla $n = 10000$ cykli. Celem jest odtworzenie wykresów z artykułu poprzez badanie wpływu parametru e_{max} na miary wyjściowe dla różnych scenariuszy błędów.

4 Miary Oceny i Analiza (Etap 4)

Będziemy obserwować trzy kluczowe miary:

- **Znormalizowane poprawne wyjścia (n_c/n):** Miara *Dostępności* (Availability). Stosunek poprawnych decyzji do wszystkich cykli.
- **Znormalizowane niepoprawne wyjścia (n_{ic}/n):** Miara *Bezpieczeństwa* (Safety). Stosunek błędów katastroficznych.
- **Znormalizowane braki decyzji (n_d/n):** Miara *Detekcji* (Benign errors). Stosunek błędów łagodnych, gdzie algorytm (MAJ lub SM) zasygnalizował błąd.

Analiza polegać będzie na porównaniu tych miar dla różnych algorytmów w tych samych scenariuszach błędów.

5 Dokumentacja (Etap 5)

Finalne produkty projektu będą zgodne z wytycznymi:

1. Sprawozdanie końcowe (format: **pdf**, wygenerowane z LaTeX).
2. Archiwum ZIP (`.zip`) zawierające kompletny kod źródłowy w Pythonie (`.py`).
3. Prezentacja końcowa (format: pdf lub pptx).