**Federated Kalman Filter**

Our model is a simple constant velocity in 2D space, represented by the following:

State vector

State Transition matrix

Process noise matrix

And the G matrix:

The complete state space model:

Such that

The measurement model  
Measurement matrix

Each sensor has it's own noise characteristics

And the measurement equation for each sensor is

Estimation:

Each local Kalman filter is responsible for estimating (propagating / updating) position. Meaning

Federated implementation

The global filter state vector is comprised of N local state vectors .

Such that each local vector is comprised of the common state variables, and bias (which is for now excluded)

The global covariance matrix is comprised of local covariance matrices:

In general, the cross covariance matrices are not zero.

Using the upper limit bound discussed in Carlson's paper, we can use a larger local covariance matrix for each matrix in the diagonal, and cross diagonal matrices are zero. In this case, we can use a LS method to find the globally optimal solution for estimate of the full system state x.

* לכל קלמן פילטר מקומי יש את אותו המודל של המרחב מצב
* כל קלמן פילטר מקומי אחראי על קידום/ עדכון ווקטור המצב המשוערך בעזרת מדידה מחיישן מסוים
* כל קלמן פילטר מקומי מאותחל לאותם ערכי מטריצת קוואריאנס ושערוך התחלתי.
* לכל קלמן פילטר זמני עדכון מדידה שונים. למשל ייתכן ופילטר אחד יתעדכן כל 5 איטרציות בעוד שפילטר אחר כל 10.
* קיימת שגרה FederatedKF האחראית על תהליך הקידום, עדכון והיתוך של כל הפילטרים במערכת
* תדירות ההיתוך נקבעת מראש, למשל כל 100 איטרציות
* ברגע ההיתוך, ווקטור המצב המשותף מתעדכן להיות מעיין ממוצע משוקלל של מטריצות הקוואריאנס ווקטורי המצב המשוערכים של כל פילטר
* לאחר מכן, כל ווקטור מצב לוקלי מתעדכן להיות ווקטור המצב לאחר התכה, והאלגוריתם ממשיך

**פגישה עם אושמן**

**שאלות ליעקב**

* אני חושב שיש בעייתיות במימוש שלי, למרות שאני לא מצליח למצוא איפה הטעות אז אני מעדיף לוודא. יוצא לי כי עבור תדירות התכה גבוהה מדי, התוצאות לא נכונות. וכן עבור תדירות התכה נמוכה מדי. כלומר קיימת תדירות התכה מסוימת המניבה תוצאות אופטימליות. אפילו במקרה הקצה בו כל פילטר מתעדכן בכל איטרציה, וכן ההתכה מתבצעת באותה האיטרציה (לאחר העדכון).
* חשבתי על כיוון שאוכל לקחת אליו את הפרויקט לאחר שווידאתי שהמימוש נכון. מאחר ולכל חיישן יש תכונות שונות (יש חיישנים יותר מדויקים מהשאר), קיימת דילמה האם בהינתן והאלגוריתם הגיע לרגע התכה של מספר חיישנים, כאשר החיישנים העדכני ביותר הוא דווקא מהחיישנים המדויקים פחות, האם כדאי לחכות עם ההתכה עד לרגע בו החיישן המדויק יותר מתעדכן. או באופן כללי לחשוב על איזה אלגוריתם לניהול חכם של מחזורי ההתכה.
* האם המודל הפשוט מספיק בשביל הפרויקט או שכדאי ליישם מודל קצת יותר מורכב

**לבדוק האם היישום נכון עבור מטריצת קוואריאנס אלכסונית בלבד**

**ההגדלה נועדה לפתור בעיה של חזרה על אותו עיבוד מידע**

**להשוות את ה FKF ל CKF בתנאים אופטימליים**

**לבדוק את ההתכה המלאכותית**

**אחד היישומים המרכזיים של FKF הוא בהתמודדות עם תקלות בחיישנים. להכניס תקלה באחד החיישנים ולראות איך כל אחד מהפילטרים יתמודדו איתה. והאם אפשר להתמודד איתה בצורה טובה עם הFKF**

**Complementary filtering – יכול לעזור בזיהוי חיישן תקול**

**קודם – גילוי הבעיה, ואז זיהוי מהי הבעיה**

**תקלה Hard ו soft. למשל בsoft הטייה שגדלה עם הזמן**

**במקרה של תקלה, לבדוק מה הן הפתרונות האפשריים**

**08.09.2025 – המשך הפרויקט**

* FKF עובד כראוי (בוצעה השוואה עם CKF).
* לבדוק השפעת מדידות שגויות על ביצועי CKF ו FKF ולהסיק מסקנות .
* לבחור אחת מן הדרכים להתמודדות עם מדידות שגויות

1. **Normalized Innovation Spectral Norm Test** (Okatana, 2008)
2. **Adaptive Avg Chi-square Residual Monitoring** (Wu et al., 2023)

מצורף הכיתוב עם הצ'אט

**1. How faulty data may affect your filters**

* **Local Filters (each sensor)**:
  + A sudden bias or outlier measurement skews the local state—this error propagates forward.
* **Federated Fusion**:
  + Since fusion blends all local estimates, a faulty sensor can contaminate the global estimate.
  + The filters could become overconfident on corrupted data, especially if you treat covariances incorrectly.
* **Centralized KF**:
  + Also vulnerable—an outlier corrupts the single KF unless you detect and reject it.

**2. Paper-based fault detection methods for FKF**

I found a few relevant approaches:

**A. Normalized Innovation Spectral Norm Test (Okatana, 2008)**

* Builds a matrix of normalized innovations from each sensor at each time step.
* Calculates its **spectral norm**; if it breaches statistically-derived bounds, a fault is flagged.
* Works **without prior knowledge of fault magnitude**—just leverages innovation consistency.  
  [gavsispanel.gelisim.edu.tr](https://gavsispanel.gelisim.edu.tr/Document/aokatan/20181128163139716_52dfa991-6dc5-4cda-9ad0-9c730e0da5f0.pdf?utm_source=chatgpt.com)

**B. Dual Fault Detection (Dual FD) (Kim et al., 2020)**

* Combines measurement-based fault detection (e.g., parity checks, chi-square tests) with system-based detection via the FKF.
* Uses **weighted fusion of fault metrics**, dynamically adapting detection sensitivity.
* Designed to be both fast and accurate across various fault types.  
  [Sejong University](https://sejong.elsevierpure.com/en/publications/a-dual-fault-detection-algorithm-based-on-the-federated-kalman-fi?utm_source=chatgpt.com)

**C. Adaptive Avg Chi-square Residual Monitoring (Wu et al., 2023)**

* Uses chi-square thresholds (based on residual covariance) to detect sensor faults.
* Upon detecting a fault, replaces the faulty local estimate with the master (fused) estimate before further fusion.
* Helps maintain stability even when sub-filters fail.  
  [MDPI](https://www.mdpi.com/2076-3417/13/9/5790?utm_source=chatgpt.com)

**3. Recommendations for you**

**Next Steps**

1. **Simulate sensor faults** in your FKF demo:
   * Types: constant bias, variance increase, intermittent outliers.
   * Observe how local and fused errors degrade.
2. **Implement one fault detection method**:
   * Start simple: chi-square test on each local filter’s innovation.
   * If innovation > threshold, mark that sensor as faulty.
3. **Design a handling strategy**:
   * *Exclude* the faulty sensor from fusion for a period.
   * *Replace* its output with the global fused state (like Wu et al., 2023) when faulty.
   * Possibly schedule *re-inclusion* once the sensor returns healthy.
4. **Compare performance**:
   * Compare RMSE and consistency before and after implementing fault detection.
   * Use both FKF and centralized KF for benchmark.

**יישום האלגוריתם לזיהוי וטיפול תקלות**

**Algorithm Overview**

The fault detection system operates on each local filter independently before fusion, using residual-based statistical tests to identify measurement anomalies. The approach combines two complementary methods: instantaneous chi-square testing and sliding window analysis.

**Mathematical Foundation**

**1. Residual Calculation**

For each local filter i at time k, the measurement residual is:

r\_{i,k} = z\_{i,k} - ẑ\_{i,k|k-1}

where:

* z\_{i,k} is the actual measurement
* ẑ\_{i,k|k-1} = H\_i × x̂\_{i,k|k-1} is the predicted measurement

**2. Residual Covariance**

Under normal (fault-free) conditions, the residual follows a zero-mean Gaussian distribution with covariance:

W\_{i,k} = E[r\_{i,k} × r\_{i,k}^T] = H\_{i,k} × P\_{i,k|k-1} × H\_{i,k}^T + R\_{i,k}

This represents the theoretical residual covariance combining prediction uncertainty and measurement noise.

**3. Chi-Square Test Statistic**

The fault detection statistic is:

λ\_{i,k} = r\_{i,k}^T × W\_{i,k}^{-1} × r\_{i,k}

Under the null hypothesis (no fault), λ\_{i,k} follows a chi-square distribution with m degrees of freedom, where m is the measurement dimension.

**4. Fault Detection Threshold**

The threshold is determined by the desired confidence level α:

T\_d = χ²\_{1-α,m}

For α = 0.05 (95% confidence), this gives the critical value from the chi-square table.

**5. Basic Fault Detection Rule**

Fault detected if: λ\_{i,k} ≥ T\_d

Normal operation if: λ\_{i,k} < T\_d

**Enhanced Sliding Window Detection**

**6. Actual Residual Covariance**

Over a sliding window of N samples:

a\_r = (1/N) × Σ\_{j=k-N+1}^k [r\_{i,j} × r\_{i,j}^T]

This estimates the actual residual behavior from recent data.

**7. Deviation Ratio**

The ratio between theoretical and actual covariances:

η = tr(W\_{i,k}) / tr(a\_r)

**8. Enhanced Fault Criteria**

Fault detected if: η > 2 or η < 0.2

Normal range: 0.5 ≤ η ≤ 1.5

**Algorithm Implementation Flow**

**Step 1: Initialization**

* Calculate chi-square thresholds for each local filter based on measurement dimensions
* Initialize sliding window buffers for residual history
* Set all fault flags to normal (1)

**Step 2: Prediction Phase**

Each local filter performs time update:

x̂\_{k|k-1} = F × x̂\_{k-1|k-1}

P\_{k|k-1} = F × P\_{k-1|k-1} × F^T + Q

**Step 3: Fault Detection Phase**

For each incoming measurement:

**3a. Calculate Residual:**

r = z - H × x̂\_{k|k-1}

W = H × P\_{k|k-1} × H^T + R

**3b. Compute Test Statistic:**

λ = r^T × W^{-1} × r

**3c. Apply Detection Rules:**

if λ ≥ T\_d:

fault\_flag = 0 (fault detected)

else:

fault\_flag = 1 (normal)

**3d. Update Sliding Window:**

* Add current residual to history buffer
* Maintain window size N
* Calculate deviation ratio η if sufficient data

**Step 4: Measurement Update Phase**

if fault\_flag == 1:

Perform normal Kalman update

K = P × H^T × (H × P × H^T + R)^{-1}

x̂ = x̂ + K × r

P = (I - K × H) × P × (I - K × H)^T + K × R × K^T

else:

Skip update (maintain prediction)

**Step 5: Fusion Phase**

Only healthy local filters contribute to fusion:

* Collect estimates from non-faulty filters
* Apply information fusion algorithms
* Redistribute fused state to all local filters

**Key Algorithm Properties**

**Statistical Guarantees**

* Type I error (false alarm) rate ≤ α
* Optimal detection for Gaussian noise assumptions
* Robust to model uncertainties through sliding window

**Computational Complexity**

* O(m³) per measurement for matrix inversion
* O(N×m²) for sliding window maintenance
* Minimal overhead compared to standard Kalman filtering

**Fault Handling Strategy**

The implementation uses a conservative approach:

1. **Detection Only**: Currently focuses on fault identification rather than correction
2. **Isolation**: Faulty measurements are rejected rather than corrected
3. **System Continuity**: Fusion continues with remaining healthy filters

**Limitations and Considerations**

**Assumptions**

* Gaussian noise models
* Linear measurement relationships
* Independent measurement faults

**Tuning Parameters**

* **Confidence Level (α)**: Trade-off between false alarms and missed detections
* **Window Size (N)**: Balance between responsiveness and statistical reliability
* **Deviation Thresholds**: May need adjustment for specific applications

The algorithm provides a principled statistical framework for real-time fault detection in federated filtering systems, combining immediate anomaly detection with longer-term trend analysis for robust operation in uncertain environments.

**09.09.25 – סקירה כללית של היום**

מה נעשה?

* נוספו תקלות מדידה יזומות בהסתברות כלשהי לכל מדיד
* נבחרה האפשרות השלישית מתוך המאמרים להטמעת הפילטר המשופר (**Adaptive Avg Chi-square Residual Monitoring (Wu et al., 2023))**
* מחלקת FederatedKF כך שתכיל את שני המבחנים הסטטיסטים – מבחן חי בריבוע וחלון נע.
* נוספה מחלקה נוספת בשם FederatedKFTraditional אשר אינה מכילה אף מבחן (גרסת ברירת מחדל)
* נוספה סימולציית מונטה קרלו להשוואה של האלגוריתמים
* נוספו שקופיות למצגת

מה יש לעשות?

* לבצע מעבר על הקוד מחדש ולוודא תקינות ורענון
* לבדוק תיאוריה של מבחן חי בריבוע וחלון נע
* להמשיך את המצגת
* להתחיל לעבוד על קובץ וורד המסכם את התהליך לפי מבנה מוסכם

**11.09.25 – סקירה כללית של היום**

מה נעשה?

* נוספו שקופיות למצגת
* נוספו שתי שיטות לטיפול בשגיאות חיישן (מפורט במצגת)

מה יש לעשות?

* לבדוק תרחישים נוספים (אי סינכרון מדידים, תדירות היתוך שונה ועוד אם מוצא) ולהסיק מסקנות
* להוסיף את הגרפים + מסקנות למצגת
* להתחיל לעבוד על קובץ וורד המסכם את התהליך לפי מבנה מוסכם

**17.09.25 – סקירה כללית של היום**

מה נעשה?

* התחלתי את הקובץ המסכם
* נפתח ענף חדש בשם calib-develop ולהלן התיאור שלו

**Calib-Develop**

* על מנת לייחד את הפרויקט מהמאמר הנוסף (בנוסף שמתי לב כי לא הוספתי את השיטה במאמר לחלוקה נכונה של הקוואריאנסים בעזרת ISF) , נבחן אלגוריתם הנתון בתנאים מיוחדים.
* במצב שבו עבור משך זמן מסוים יש לנו גישה לחיישן שאנחנו יודעים שהוא אמין בצורה גבוהה מאוד, ניתן להשתמש בו כרפרנס.
* שגרת FederatedKF שונתה כך שתחולק לשני שלבים:
  1. שלב הקליברציה – בשלב זה הפילטר הראשי משתמש בפילטר הרפרנס (הפילטר אשר החיישן שלו הוא חיישן הרפרנס) על מנת לבצע חלוקה של המשקלים בין החיישנים האחרים.
  2. שלב האופרציה – בשלב זה פילטר הרפרנס כבר אינו זמין ובתקווה כיילנו את הפילטרים המשניים בצורה טובה בשלב הקליברציה כך שביצועי הפילטר יהיו טובים.
* שיטות ההתמודדות עם שגיאות מדיד עדיין קיימות, אך יש לבדוק איך הן משתלבות

**ובאופן כללי**

* שגרת demo\_fkf\_linear שונתה כך שתכיל גם פילטר FKF רגיל להשוואה וגם תקטע עבור כל הפילטרים את חיישן הרפרנס ברגע האופרציה
* שגרת FederatedKFTraditional עודכנה קלות על מנת שתוכל להתמודד עם שינוי פתאומי של פילטרים

מה יש לעשות?

* ראשית והכי חשוב, **יש להבין לעומק איך עובד האלגוריתם. להיעזר בצ'אט ולוודא שאין פערים.**
* **לוודא שהשילוב של זיהוי תקלות וטיפול בהן עדיין טוב.** לשקול הסרה של השיטות השונות ולהישאר רק עם זו שעובדת טוב
* במידה והאלגוריתם טוב, יש לבדוק מספר מקרים לפני שנעבור למונטה קרלו. להתייעץ עם הצ'אט גם כן
* אם האלגוריתם צלח במספר המקרים שבדקנו (קיבלנו תוצאות הגיוניות), לסדר את שגרת המונטה קרלו בשביל הריצות הסופיות
* אם התוצאות מספקות – **לעצור כאן בפיתוח**. לסיים קודם מצגת ואז לעבור לקובץ סיכום.