|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| q **ĐẠI HỌC BÁCH KHOA HÀ NỘI**  **TRƯỜNG ĐIỆN – ĐIỆN TỬ**  logo_128  BÁO CÁO ĐỒ ÁN TỐT NGHIỆP CỬ NHÂN  **Tên đề tài:**  **..........**  Giảng viên hướng dẫn: PGS. TS. NGUYỄN ĐỨC MINH  TS. HOÀNG PHƯƠNG CHI  Nhóm sinh viên thực hiện:   |  |  |  |  | | --- | --- | --- | --- | | STT | Họ và tên sinh viên | MSSV | Lớp | | 1 | Lê Thành Luân | 20203494 | ET1 08 K65 | | 2 | Hà Văn Đức | 20203677 | ET1 08 K65 | | 3 | Trần Minh Đức | 20203370 | ET1 06 K65 | | 4 | Phạm Thị Xuân Mai | 20203734 | ET1 06 K65 | | 5 | Phạm Hồng Thi | 20203767 | ET1 02 K65 |   Trường/ Khoa/ Viện: Trường Điện - Điện tử  Hà Nội, 5-2024 |

**MỤC LỤC**

[DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT i](#_Toc168509343)

[DANH MỤC HÌNH VẼ ii](#_Toc168509344)

[DANH MỤC BẢNG BIỂU iii](#_Toc168509345)

[TÓM TẮT iv](#_Toc168509346)

[CHƯƠNG 1. ĐẶT VẤN ĐỀ 5](#_Toc168509347)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 6](#_Toc168509348)

[2.1 Kalman Filter 6](#_Toc168509349)

[CHƯƠNG 3. KẾT QUẢ 13](#_Toc168509350)

[CHƯƠNG 4. KẾT LUẬN 13](#_Toc168509351)

[TÀI LIỆU THAM KHẢO 13](#_Toc168509352)

# DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| Ký hiệu | Nội dung | Ý nghĩa |
| GPS | Global Positioning System | Hệ thống định vị toàn cầu |
| GNSS | Global Navigation Satellite System | Hệ thống vệ tinh định vị toàn cầu |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# DANH MỤC HÌNH VẼ

# DANH MỤC BẢNG BIỂU

# TÓM TẮT

# ĐẶT VẤN ĐỀ

# CƠ SỞ LÝ THUYẾT

## Kalman Filter

A diagram of a cycle

Description automatically generated

Tổng quan hoạt động của Kalman Filter

A diagram of a mathematical equation

Description automatically generated

Các công thức với từng giai đoạn của Kalman Filter

💬 Kí hiệu được sử dụng chung:

Chữ thường, in đậm đề cập tới vector:

Chữ in hoa, in đậm đề cập tới ma trận:

Chữ thường, không in đậm đề cập tới vô hướng hoặc những phần tử trong vector:

Chữ in hoa, không in đậm đề cập tới những phần tử trong ma trận:

💬 Kí hiệu được sử dụng chung:

F: ma trận chuyển trạng thái: mô tả cách thay đổi trạng thái của hệ thống

P: ma trận hiệp phương sai: mô tả sự k chắc chắn (uncertainty) về trạng thái ước lượng ban đầu. ( giá trị khởi tạo này phải có nhiều sự nghiên cứu mới lựa chọn được).

Đối với giá trị nhỏ:

* Mức độ tin tưởng cao vào trạng thái ban đầu: Bạn đang nói rằng bạn khá chắc chắn về trạng thái ban đầu.
* Phản ứng chậm với đo lường mới: Kalman Filter sẽ ít tin tưởng vào các đo lường mới hơn và thay vào đó dựa nhiều hơn vào ước lượng trạng thái ban đầu. Nếu ước lượng ban đầu không chính xác, quá trình lọc sẽ phản ứng chậm với các thay đổi trong đo lường.

Đối với giá trị lớn:

* Mức độ tin tưởng thấp vào trạng thái ban đầu: Bạn đang nói rằng bạn không chắc chắn lắm về trạng thái ban đầu.
* Phản ứng nhanh với đo lường mới: Kalman Filter sẽ tin tưởng vào các đo lường mới nhiều hơn và điều chỉnh trạng thái ước lượng nhanh hơn dựa trên các đo lường này. Điều này hữu ích nếu ước lượng ban đầu không chính xác hoặc nếu hệ thống có nhiều nhiễu.

**Bước 1: Khởi tạo**

Ta lấy giá trị đầu tiên và lưu vào

**Bước 2: Dự đoán**

Ở bước này ta sẽ làm việc với hai công thức sau:

|  |  |
| --- | --- |
| Tên công thức | Công thức |
| Ngoại suy trạng thái (State Extrapolation) |  |
| Ngoại suy hiệp phương sai (Covariance Extrapolation) |  |

***Công thức ngoại suy trạng thái:***

Theo phương trình chuyển động cơ học ta có tương đương với hệ phương trình sau:

Vậy ta gán cho các kí hiệu sau:

; ;

**;**

Vậy ta có được phương trình sau

|  |  |  |
| --- | --- | --- |
| Kí hiệu | Tên | Kích thước(hàng x cột) |
|  | Vecto trạng thái dự đoán |  |
|  | Vecto trạng thái hiện tại |  |
|  | Vecto đầu vào điều khiển |  |
|  | Ma trận chuyển trạng thái |  |
|  | Ma trận điều khiển |  |

Phương trình trên thể hiện vecto trạng thái tiếp theo phụ thuộc vào các với ma trận đầu vào hiện tại và ma trận điều khiển hiện tại. Đối với bài toán đã đặt ra ta có các ma trận sau:

; ;

**;**

Áp dụng ta có:

Có thể thấy đây là hệ phương trình liên hệ giữa các đại lượng vận tốc, thời gian và độ dài giữa các điểm.

***Công thức ngoại suy hiệp phương sai:***

|  |  |  |
| --- | --- | --- |
| Kí hiệu | Tên | Kích thước(hàng x cột) |
|  | Ma trận hiệp phương sai ước tính cho trạng thái tiếp theo |  |
|  | Ma trận hiệp phương sai ước tính cho trạng thái hiện tại |  |
|  | Ma trận chuyển trạng thái |  |
|  | Ma trận hiệp phương sai nhiễu quá trình ngoại suy ra trạng thái tiếp theo |  |

**;**

=

Đường chéo chính của Q là ma trận hiệp phương sai

**Bước 2: Ước tính**

Ở bước này ta sẽ làm việc với bốn công thức sau:

|  |  |
| --- | --- |
| Tên công thức | Công thức |
| Phần dư đo lường |  |
| Tính toán hệ số Kalman – trọng số (Kalman Gain) |  |
| Cập nhật trạng thái (State Update) |  |
| Ma trận cập nhật hiệp phương sai (Covariance Update) |  |

***Công thức phần dư đo lường:***

|  |  |  |
| --- | --- | --- |
| Kí hiệu | Tên | Kích thước(hàng x cột) |
|  | Vecto trạng thái dự đoán ở phần dự đoán |  |
|  | Vecto đo lường |  |
|  | Veco nhiễu ngẫu nhiên(phần dư đo lường) |  |
|  | Ma trận đo lường |  |

Ta cónên có phương trình rút gọn:

**;**

Ma trận đo lường H ở đây có thể dùng để làm tỉ lệ (scaling), lựa chọn trạng thái (state selection) và kết hợp các trạng thái (combination of states). Ở đây vì sử dụng để quan sát các giá trị x,y,z nên ma trận H như ma trận đơn vị.

Suy ra:

***Công thức độ lợi Kalman:***

|  |  |  |
| --- | --- | --- |
| Kí hiệu | Tên | Kích thước(hàng x cột) |
|  | Độ lợi Kalman (Kalman Gain) |  |
|  | Ma trận hiệp phương sai ở phần dự đoán |  |
|  | Ma trận đo lường |  |
|  | Ma trận hiệp phương sai nhiễu đo lường |  |

Ta có:nên:

Ta có phần dự đoán được gán cho :

=

***Công thức cập nhật trạng thái:***

|  |  |  |
| --- | --- | --- |
| Kí hiệu | Tên | Kích thước(hàng x cột) |
|  | Vecto trạng thái ước tính(đầu ra hệ thống) |  |
|  | Vecto trạng thái dự đoán ở phần dự đoán |  |
|  | Độ lợi Kalman (Kalman Gain) |  |
|  | Veco nhiễu ngẫu nhiên(phần dư đo lường) |  |

***Công thức ma trận cập nhật hiệp phương sai (Covariance Update)***

|  |  |  |
| --- | --- | --- |
| Kí hiệu | Tên | Kích thước(hàng x cột) |
|  | Ma trận hiệp phương sai của trạng thái hiện tại ước tính. |  |
|  | Ma trận hiệp phương sai ở phần dự đoán |  |
|  | Ma trận đo lường |  |
|  | Ma trận đơn vị |  |

Ta có nên có phương trình rút gọn:

Ta có phần dự đoán được gán cho :

=

# MÔ TẢ HỆ THỐNG

Sơ đồ thuật toán

A diagram of a process

Description automatically generated

Dòng code sau:

```python

d = distance.mahalanobis(z, H @ x, np.linalg.inv(P))

```

### Giải thích chi tiết

1. \*\*`distance.mahalanobis(z, H @ x, np.linalg.inv(P))`\*\*:

- `distance.mahalanobis` là một hàm từ thư viện `scipy.spatial.distance` dùng để tính khoảng cách Mahalanobis giữa hai vector.

- Khoảng cách Mahalanobis là một thước đo khoảng cách giữa hai điểm trong không gian đa chiều, có tính đến sự tương quan giữa các biến.

2. \*\*Các tham số của hàm `distance.mahalanobis`\*\*:

- `z`: Vector đo lường.

- `H @ x`: Vector trạng thái dự đoán.

- `H` là ma trận đo lường, được sử dụng để chuyển đổi trạng thái hệ thống từ không gian trạng thái sang không gian đo lường.

- `x` là vector trạng thái dự đoán.

- `np.linalg.inv(P)`: Ma trận hiệp phương sai nghịch đảo.

3. \*\*Ý nghĩa của khoảng cách Mahalanobis\*\*:

- Khoảng cách Mahalanobis giữa z và H @ x được tính bằng công thức:

- Khoảng cách này đo lường mức độ khác biệt giữa đo lường thực tế z và dự đoán H @ x, có tính đến hiệp phương sai của hệ thống.

### Ví dụ minh họa

Giả sử chúng ta có các giá trị sau:

```python

import numpy as np

from scipy.spatial import distance

# Vector đo lường z

z = np.array([1.0, 2.0, 3.0])

# Vector trạng thái dự đoán x

x = np.array([0.5, 1.5, 2.5])

# Ma trận đo lường H

H = np.eye(3)

# Ma trận hiệp phương sai P

P = np.array([

[4.0, 0.1, 0.2],

[0.1, 3.0, 0.1],

[0.2, 0.1, 2.5]

])

# Tính toán khoảng cách Mahalanobis

d = distance.mahalanobis(z, H @ x, np.linalg.inv(P))

print(d)

```

Kết quả sẽ là giá trị của khoảng cách Mahalanobis giữa `z` và `H @ x` dựa trên ma trận hiệp phương sai `P`.

### Ý nghĩa trong ngữ cảnh Kalman Filter

Trong ngữ cảnh của Kalman Filter, tính toán khoảng cách Mahalanobis có thể được sử dụng để:

1. \*\*Phát hiện ngoại lai\*\*: So sánh khoảng cách Mahalanobis với một ngưỡng để xác định xem đo lường hiện tại có phải là ngoại lai hay không. Nếu khoảng cách lớn hơn ngưỡng, đo lường có thể bị loại bỏ để tránh ảnh hưởng xấu đến ước lượng trạng thái.

2. \*\*Cập nhật trạng thái\*\*: Nếu đo lường không phải là ngoại lai, nó sẽ được sử dụng để cập nhật trạng thái dự đoán và ma trận hiệp phương sai.

### Tóm tắt

Dòng code `d = distance.mahalanobis(z, H @ x, np.linalg.inv(P))` tính toán khoảng cách Mahalanobis giữa vector đo lường `z` và trạng thái dự đoán `H @ x` sử dụng ma trận hiệp phương sai nghịch đảo của trạng thái `P`. Trong ngữ cảnh Kalman Filter, khoảng cách Mahalanobis được sử dụng để phát hiện ngoại lai và quyết định xem đo lường hiện tại có nên được sử dụng để cập nhật trạng thái hay không.

# KẾT QUẢ

# KẾT LUẬN

# PHỤ LỤC

**Giải thích code**

****

* **zs.shape** là một đối tượng zs truy cập vào thuộc tính shape là một mảng 2 chiều.
* **n** là số lượng vecto.
* **dim\_x** là số chiều của mỗi vecto x.

**Giải thích code**

A computer code with text

Description automatically generated with medium confidence

**v\_wls[:-1]:**

* v\_wls là một mảng NumPy.
* v\_wls[:-1] lấy tất cả các phần tử v\_wlstrừ phần tử cuối cùng.

**v\_wls[1:]:**

* v\_wls[1:] lấy tất cả các phần tử v\_wls trừ phần tử đầu tiên.

**(v\_wls[:-1] + v\_wls[1:])/2:**

* Biểu thức này tính trung bình của từng cặp tử tiếp giáp trong v\_wls.
* Kết quả sẽ là một mảng mới mà mỗi phần tử là trung bình của hai phần tử liền kề từ v\_wls.

**np.zeros([1, 3]):**

* Tạo một mảng NumPy với kích thước (1, 3) và tất cả các phần tử bằng 0.

**np.vstack([...]):**

* np.vstack()là một hàm trong NumPy để xếp chồng các mảng theo chiều dọc (tức là thêm các hàng mới).
* Trong trường hợp này, nó sẽ chồng mảng [[0, 0, 0]] lên trên kết quả mảng của (v\_wls[:-1] + v\_wls[1:])/2.

**Ví dụ minh họa:**

Giả sử v\_wls là một mảng 2 chiều:

A screenshot of a computer code

Description automatically generated

Khi áp dụng đoạn code sau:



Kết quả sẽ là:

* v\_wls[:-1] có giá trị là:

A number with different numbers

Description automatically generated with medium confidence

* v\_wls[1:] có giá trị là:

A number with blue and black text

Description automatically generated with medium confidence

* (v\_wls[:-1] + v\_wls[1:])/2 có giá trị là:

A number of numbers on a white background

Description automatically generated

* np.zeros([1, 3]) có giá trị là:



* np.vstack([np.zeros([1, 3]), (v\_wls[:-1] + v\_wls[1:])/2]) có giá trị là:

A number with numbers in the middle

Description automatically generated with medium confidence

Như vậy, mảng **v** sẽ bao gồm một hàng số không ở đầu, tiếp theo là trung bình của các cặp tử liền kề từ **v\_wls**.

# TÀI LIỆU THAM KHẢO