**Kalman Filter** là một mô hình **Linear-Gaussian State Space Model** thuộc nhóm thuật toán dự đoán chuỗi thời gian. Thuật toán được lấy tên theo **Rudolf E. Kálmán**, một nhà khoa học ảnh hưởng quan trọng trong quá trình phát triển thuật toán.

A drawing of a rocket and mountains

Description automatically generated

Nếu là một kỹ sư điều khiển hệ thống, bạn hiểu rằng điều khiển hệ thống không đơn thuần nghĩa là đo rồi điều khiển những gì mình đo được. Đo lường chính xác nhường nào, hệ thống điều khiển càng hoạt động ổn định nhường nấy.

Hãy quay ngược thời gian về những năm 1960 với những khó khăn của những kỹ sư thiết kế phi thuyền Apollo trong xử lý tín hiệu. Dữ liệu thô đo được từ máy tính, được đo từ các cảm biến như con quay hồi chuyển, gia tốc kế, dữ liệu rađa vốn dĩ đầy nhiễu, đầy lỗi ngẫu nhiên và đặc biệt lộn xộn không chính xác. Khi lao về mặt trăng ở tốc độ cao, ai mà chắc chắn rằng điều gì sẽ xảy ra với những gì bạn điều khiển, bạn sẽ không muốn Apollo lao thẳng vào nhà mình đâu!

Kalman Filter là một công cụ mạnh mẽ kết hợp thông tin không chắc chắn ở thời điểm hiện tại cùng với thông tin đầy nhiễu loạn của môi trường sang một dạng thông tin mới đáng tin cậy hơn để phục vụ dự đoán tương lai. Điểm mạnh của Kalman Filter là chạy rất nhanh và tính ổn định cao.

Một trong những ứng dụng đầu tiên của Kalman Filter là được áp dụng vào phi thuyền Apollo:

*The Apollo computer used 2k of magnetic core RAM and 36k wire rope […]. Clock speed was under 100 kHz […]. The fact that the MIT engineers were able to pack such good software (one of the very first applications of the Kalman filter) into such a tiny computer is truly remarkable.  
— Phỏng vấn****Jack Crenshaw****bởi****Matthew Reed****, TRS-80.org (2009)*

**1. Ý tưởng cơ bản thuật toán**

Thuật toán có rất nhiều ứng dụng vào các bài toán khác nhau. Tuy nhiên để dể hình dung, bây giờ chúng ta hãy tưởng tượng mình là một nhà sáng chế phi thuyền Thetalo (một phiên bản lỗi từ Apollo).

Phi thuyền sẽ gửi về thông tin trạng thái sau mỗi khoảng thời gian nhất định thông qua bộ cảm biến, nhiệm vụ của chúng ta là dự đoán trạng thái trên phi thuyền để phục vụ tác vụ điều khiển hệ thống.

Gọi xt=[positionvelocity]∈R2𝑥𝑡=[𝑝𝑜𝑠𝑖𝑡𝑖𝑜𝑛𝑣𝑒𝑙𝑜𝑐𝑖𝑡𝑦]∈𝑅2 là trạng thái vị trí và vận tốc hiện tại của phi thuyền. Tại mỗi thời điểm t𝑡, bạn không hề có thông tin chính xác về xt𝑥𝑡.

Vì sao vậy? Hệ mà chúng ta điều khiển không nằm trong một môi trường lý tưởng, nhiều tác động từ môi trường xung quanh mà chúng ta không đoán trước ở sẽ ảnh hưởng những tính toán ban đầu.

Biết đâu được một cơn gió vô tình, một cơn mưa ngang qua, một trận sấm sét cuồng phong sẽ làm mọi tính toán sai lầm. Trực cảm cho ta thấy nó đang ở đâu đó trong không gian này, tại một số vùng nào đó khả năng xảy ra cao hơn, tại một số vùng nào đó khả năng xảy ra thấp hơn. Phân bố xác suất như một bản đồ về sự chắc chắn, một niềm tin về những sự kiện xảy ra.

Lựa chọn phân bố xác suất phù hợp là điều cần thiết như thể lựa chọn cho mình một niềm tin, trong mô hình Kalman Filter phân bố xác suất được chọn là phân bố chuẩn nhiều chiều. Sở dĩ chọn phân bố chuẩn nhiều chiều là vì đây là phân bố liên tục phù hợp bài toán đang xét, một vài tính chất đặc biệt của phân bố chuẩn nhiều chiều thuận lợi cho tác vụ ước lượng tham số.

Tại mỗi thời điểm t𝑡, giả định vector ngẫu nhiên xt𝑥𝑡 có phân bố chuẩn nhiều chiều hay xt∼N(^xt,Σt)𝑥𝑡∼𝑁(𝑥^𝑡,Σ𝑡) với ^xt𝑥^𝑡 là kỳ vọng của xt𝑥𝑡, ΣtΣ𝑡 là ma trận hiệp phương sai với mỗi phần tử tại hàng i𝑖 cột j𝑗 thể hiện độ biến thiên cùng nhau của hai biến ngẫu nhiên thứ i𝑖 và j𝑗. Công thức tổng quát của hàm mật độ xác suất vector ngẫu nhiên xt𝑥𝑡 với kỳ vọng ^xt𝑥^𝑡, ma trận hiệp phương sai ΣtΣ𝑡 với số chiều D𝐷 (trong ví dụ đang xét, D=2𝐷=2) là:

p(xt)=1(2π)D/2|Σt|1/2exp[−12(xt−^xt)⊺Σ−1t(xt−^xt)]𝑝(𝑥𝑡)=1(2𝜋)𝐷/2|Σ𝑡|1/2exp[−12(𝑥𝑡−𝑥^𝑡)⊺Σ𝑡−1(𝑥𝑡−𝑥^𝑡)]

A diagram of a graph

Description automatically generated

Lập luận theo cách trên lại có một ưu thế nữa, trong nhiều bài toán khoa học kỹ thuật, đôi lúc bạn cần biết xác suất xảy ra trong một vùng nào đấy, chẳng hạn:

* Đôi lúc chúng ta chỉ quan tâm đến một vùng nào đó của một đại lượng, chẳng hạn từ 9595 đến 110110 độ C thì CPU máy tính phải tắt vì nhiệt độ vượt ngưỡng an toàn, xác suất để nhiệt độ rơi vào vùng này là bao nhiêu nhỉ? Tương tự với một tập đại lượng, chúng ta quan tâm đến một vùng giá trị nào đó thật sự quan trọng trong bài toán đang xét.
* Với hàm mật độ xác suất liên tục p(xt)𝑝(𝑥𝑡) chúng ta có thể dể dàng tính xác suất xảy ra trong vùng E𝐸 bằng tích phân ∫Ep(xt)dxt∫𝐸𝑝(𝑥𝑡)𝑑𝑥𝑡, thật tiện lợi đúng không nào!

Giả định rằng tại thời điểm t𝑡 phi thuyền có:

^xt=[−0.20.1]𝑥^𝑡=[−0.20.1]

Σt=[0.40.350.350.6]Σ𝑡=[0.40.350.350.6]

A graph with a circular diagram

Description automatically generated with medium confidence

Thông tin mà chúng ta có không chỉ có những gì chúng ta tiên lượng về nó (cách chúng ta đoán về xt𝑥𝑡). Trên phi thuyền được gắn một cảm biến đặc biệt cho phép đo trạng thái hiện tại và gửi thông tin về trụ sở.

Tuy nhiên, mặc dù vậy, cảm biến này lại **không chính xác**, bộ cảm biến cho chúng ta biết trạng thái hiện tại là yt=[2−2]𝑦𝑡=[2−2]

A graph with a diagram of a galaxy

Description automatically generated with medium confidence

Trong trường hợp này, chúng ta giả định rằng:yt=Gtxt+vt𝑦𝑡=𝐺𝑡𝑥𝑡+𝑣𝑡với vt𝑣𝑡 là một vector ngẫu nhiên nhiễu có phân bố chuẩn nhiều chiều vt∼N(0,Rt)𝑣𝑡∼𝑁(0,𝑅𝑡), chúng ta gọi vt𝑣𝑡 là nhiễu mô hình quan sát (observation noise).

Giả định trên có thể hiểu yt𝑦𝑡 được xem như là “nhiễu quan sát được từ xt𝑥𝑡“, yt𝑦𝑡 là kết quả của tổ hợp tuyến tính từng thành phần xt𝑥𝑡 dựa trên ma trận Gt𝐺𝑡 và bị ảnh hưởng cộng thêm một lượng nhiễu môi trường với vector ngẫu nhiên vt𝑣𝑡. Người ta gọi công thức trên là mô hình quan sát, nó thể hiện quá trình xt𝑥𝑡 biến đổi thành yt𝑦𝑡 đồng thời bị ảnh hưởng nhiễu từ môi trường.

Giờ đây, chúng ta có niềm tin của chúng ta về trạng thái phi thuyền qua hàm mật độ xác suất p(xt)𝑝(𝑥𝑡) và một sự kiện đã xảy ra là thông tin thu được từ bộ cảm biến - trạng thái yt𝑦𝑡, liệu rằng chúng ta có thể kết hợp cả hai thông tin đang có thành một thông tin mới có ý nghĩa hơn, giúp chúng ta hiểu hơn về trạng thái hiện tại hệ thống hay không?

Biết yt𝑦𝑡 sẽ giúp chúng ta cập nhật niềm tin về xt𝑥𝑡 như thế nào? Nếu bạn đang tìm một giải pháp như trên, có lẽ định lý Bayes là một câu trả lời chúng ta đang tìm:

p(xt|yt)=p(yt|xt)p(xt)p(yt)𝑝(𝑥𝑡|𝑦𝑡)=𝑝(𝑦𝑡|𝑥𝑡)𝑝(𝑥𝑡)𝑝(𝑦𝑡)

* p(xt)𝑝(𝑥𝑡) hàm mật độ xác suất tiên nghiệm, niềm tin của chúng ta không phụ thuộc vào sự kiện yt𝑦𝑡 xảy ra.
* p(yt|xt)𝑝(𝑦𝑡|𝑥𝑡) hàm mật độ xác suất khả dĩ, mật độ xác suất có điều kiện khi biết trạng thái xt𝑥𝑡 xảy ra, vì chúng ta biết rằng yt𝑦𝑡 là một vector ngẫu nhiên sao cho yt=Gtxt+vt𝑦𝑡=𝐺𝑡𝑥𝑡+𝑣𝑡, khi biết xt𝑥𝑡 xảy ra nghĩa là Gtxt𝐺𝑡𝑥𝑡 là một vector hằng, vector ngẫu nhiên yt𝑦𝑡 được biểu diễn bằng một vector hằng cộng thêm một vector ngẫu nhiên có phân bố chuẩn nhiều chiều vt∼N(0,Rt)𝑣𝑡∼𝑁(0,𝑅𝑡), hay nói cách khác yt|xt𝑦𝑡|𝑥𝑡 hiện tại là một phân bố chuẩn nhiều chiều. Phân bố yt|xt∼N(Gtxt,Rt)𝑦𝑡|𝑥𝑡∼𝑁(𝐺𝑡𝑥𝑡,𝑅𝑡).
* p(yt)𝑝(𝑦𝑡) hàm mật độ xác suất biên không phụ thuộc vào xt𝑥𝑡 đóng vai trò như một hằng số chuẩn hóa.

Phân bố xt|yt𝑥𝑡|𝑦𝑡 được tìm như thế nào? Trên nền tảng Linear Gaussian Systems (LGS - mô hình phân bố chuẩn tuyến tính), lời giải của định lý Bayes cho bài toán này như sau:

| **Định lý Bayes cho Linear Gaussian Systems** |
| --- |
| Giả định có hai vector ngẫu nhiên m∈RDm𝑚∈𝑅𝐷𝑚 và n∈RDn𝑛∈𝑅𝐷𝑛 với m𝑚 là vectơ ẩn, n𝑛 là vector nhiễu quan sát được từ m𝑚 với:   * m∼N(μm,Σm)𝑚∼𝑁(𝜇𝑚,Σ𝑚) * n|m∼N(Cm+d,Σn)𝑛|𝑚∼𝑁(𝐶𝑚+𝑑,Σ𝑛)   Phân bố hậu nghiệm m|n𝑚|𝑛 khi biết n𝑛 là m|n∼N(μm|y,Σm|n)𝑚|𝑛∼𝑁(𝜇𝑚|𝑦,Σ𝑚|𝑛) với:   * Σm|n=(Σ−1m+C⊺Σ−1nC)−1Σ𝑚|𝑛=(Σ𝑚−1+𝐶⊺Σ𝑛−1𝐶)−1 * μm|n=Σm|n[C⊺Σ−1n(n−d)+Σ−1mμm]𝜇𝑚|𝑛=Σ𝑚|𝑛[𝐶⊺Σ𝑛−1(𝑛−𝑑)+Σ𝑚−1𝜇𝑚]   **TL;DR:** phần chứng minh định lý sẽ không được bàn ở đây, trong một dịp nào đó nếu có thể mình sẽ viết về các mô hình phân bố chuẩn nhiều chiều, bạn đọc quan tâm có thể đọc thêm tài liệu đính kèm. (**Kevin P. Murphy phần 6 tài liệu [7]**, Chris Bracegirdle tài liệu [6] phần hệ luận Corollary 5) |

Tuy nhiên nếu tính toán trực tiếp ma trận hiệp phương sai Σxt|ytΣ𝑥𝑡|𝑦𝑡 như trên thì có lẽ bạn vừa bỏ qua một vài “điều thú vị” trong đại số tuyến tính khiến giải pháp của chúng ta thú vị hơn!

| **Đồng nhất thức ma trận Woodbury (hay còn gọi matrix inversion lemma)** |
| --- |
| Cho 44 ma trận A𝐴 (n×n)(𝑛×𝑛), U𝑈 (n×k)(𝑛×𝑘), C𝐶 (k×k)(𝑘×𝑘), V𝑉 (k×n)(𝑘×𝑛) Đồng nhất thức Woodbury nói rằng:(A+UCV)−1=A−1−A−1U(C−1+VA−1U)−1VA−1(𝐴+𝑈𝐶𝑉)−1=𝐴−1−𝐴−1𝑈(𝐶−1+𝑉𝐴−1𝑈)−1𝑉𝐴−1Nghĩa là khi bạn có nhu cầu tính toán công thức bên trái, bạn có thể xử dụng công thức bên phải để thay thế! Nó có lợi điểm gì trong bài toán này nhỉ?  **TL;DR:** phần chứng minh này bạn có thể xem thêm tài liệu [8]. |

Trước khi giải thích ý nghĩa của đồng nhất thức ma trận Woodbury trong Kalman Filter, chúng ta hãy cùng nhau viết lại lời giải phân bố hậu nghiệm xt|yt𝑥𝑡|𝑦𝑡 vừa tìm được, chúng ta gọi đây là “phân bố lọc” (Filtering Distribution):

xFt∼xt|yt∼N(^xF,ΣF)𝑥𝑡𝐹∼𝑥𝑡|𝑦𝑡∼𝑁(𝑥^𝐹,Σ𝐹)

Với tham số kỳ vọng và ma trận hiệp phương sai của phân bố chuẩn nhiều chiều:

* ΣFt=(Σ−1t+G⊺tR−1tGt)−1=Σt−ΣtG⊺t(R+GtΣtG⊺t)−1GtΣtΣ𝑡𝐹=(Σ𝑡−1+𝐺𝑡⊺𝑅𝑡−1𝐺𝑡)−1=Σ𝑡−Σ𝑡𝐺𝑡⊺(𝑅+𝐺𝑡Σ𝑡𝐺𝑡⊺)−1𝐺𝑡Σ𝑡
* ^xFt=^xt+ΣtG⊺t(GtΣtG⊺t+Rt)−1(yt−Gt^xt)𝑥^𝑡𝐹=𝑥^𝑡+Σ𝑡𝐺𝑡⊺(𝐺𝑡Σ𝑡𝐺𝑡⊺+𝑅𝑡)−1(𝑦𝑡−𝐺𝑡𝑥^𝑡)

Ý nghĩa của đồng nhất thức ma trận Woodbury: Trong nhiều ứng dụng theo thời gian thực, thường thì chỉ có một số ít trạng bạn nhận được từ bộ cảm biến tại một thời điểm, hay nói cách khác số chiều của vector ngẫu nhiên yt𝑦𝑡 rất nhỏ, lúc này chi phí tìm ma trận nghịch đảo của (Rt+GtΣtG⊺t)−1(𝑅𝑡+𝐺𝑡Σ𝑡𝐺𝑡⊺)−1 tương đối nhỏ, đồng thời các ma trận khác đã có rồi, việc tính toán còn lại chỉ là nhân ma trận sẽ nhanh hơn rất nhiều so với bài toán gốc nhân rồi tính một ma trận nghịch đảo rất lớn!

Giả định tiếp tục với ví dụ trên với:Rt=[0.260.22750.22750.39]𝑅𝑡=[0.260.22750.22750.39]

Gt=[1001]𝐺𝑡=[1001]

A diagram of a galaxy

Description automatically generated

Cuối cùng chúng ta cũng kết hợp được hai thông tin thành một thông tin mới hữu ích hơn để hiểu rõ về trạng thái của phi thuyển!

Ngay bây giờ chúng ta muốn dự đoán tương lai, trạng thái của phi thuyền sẽ đi đâu về đâu?

Trạng thái mới có thể biểu diễn dưới dạng trạng thái dịch chuyển và bị thêm một ít nhiễu từ trạng thái cũ:

xt+1=Atxt+Btut+wt𝑥𝑡+1=𝐴𝑡𝑥𝑡+𝐵𝑡𝑢𝑡+𝑤𝑡

Với:

* At𝐴𝑡: ma trận mô hình chuyển đổi trạng thái.
* Bt𝐵𝑡: ma trận mô hình điều khiễn đầu vào (dùng để kết hợp với ut𝑢𝑡 từ người dùng).
* ut𝑢𝑡: vector điều khiển (người dùng nhập để kết hợp với Bt𝐵𝑡).
* wt𝑤𝑡: vector ngẫu nhiên nhiễu hệ thống (system noise), có phân bố chuẩn nhiều chiều w∼N(0,Qt)𝑤∼𝑁(0,𝑄𝑡).

Thường thì Bt𝐵𝑡 và ut𝑢𝑡 bị khuyết, Btut𝐵𝑡𝑢𝑡 được xem là sửa lỗi hệ thống được thêm bởi người dùng, việc thêm vào cũng giống như việc thêm một đại lượng không đổi vào các công thức, việc biến đổi với các tính chất kỳ vọng và hiệp phương sai không khó. Nên để gọn phần trình bày dưới đây chúng ta sẽ xem như Btut𝐵𝑡𝑢𝑡 bị khuyết.

Vì vậy chúng ta xem xt+1𝑥𝑡+1 như là:

xt+1=Atxt+wt𝑥𝑡+1=𝐴𝑡𝑥𝑡+𝑤𝑡

Tuy nhiên chúng ta lại trực nhớ ra rằng, xác suất tiên nghiệm của chúng ta đã được thay thế bằng một xác suất hậu nghiệm nhìn có vẻ hợp lí hơn, do đó chúng ta biểu diễn xt+1𝑥𝑡+1 dưới dạng:

xt+1=AtxFt+wt𝑥𝑡+1=𝐴𝑡𝑥𝑡𝐹+𝑤𝑡

Vì tổ hợp tuyến tính của các phân bố chuẩn là phân bố chuẩn, nên xt+1𝑥𝑡+1 cũng là phân bố chuẩn, mà là phân bố chuẩn nhiều chiều thì chúng ta cần tìm hai tham số, kỳ vọng và ma trận hiệp phương sai.

May mắn thay, nhờ vào tính chất kỳ vọng và ma trận hiệp phương sai với vector ngẫu nhiên, ta có:

^xt+1=E[AtxFt+wt]=AtExFt+Ewt=At^xFt𝑥^𝑡+1=𝐸[𝐴𝑡𝑥𝑡𝐹+𝑤𝑡]=𝐴𝑡𝐸𝑥𝑡𝐹+𝐸𝑤𝑡=𝐴𝑡𝑥^𝑡𝐹Σt+1=Var[AtxFt+wt]=AtVar[xFt]A⊺t+Qt=AtΣFtA⊺t+QtΣ𝑡+1=Var[𝐴𝑡𝑥𝑡𝐹+𝑤𝑡]=𝐴𝑡Var[𝑥𝑡𝐹]𝐴𝑡⊺+𝑄𝑡=𝐴𝑡Σ𝑡𝐹𝐴𝑡⊺+𝑄𝑡

Đặt Kt=ΣtG⊺t(GtΣtG⊺t+Rt)−1𝐾𝑡=Σ𝑡𝐺𝑡⊺(𝐺𝑡Σ𝑡𝐺𝑡⊺+𝑅𝑡)−1 hệ số này được gọi là Kalman Gain, chúng ta có thể viết phân bố mới ở dạng gọn hơn:xt+1∼N(^xt+1,Σt+1)𝑥𝑡+1∼𝑁(𝑥^𝑡+1,Σ𝑡+1)

Viết gọn lại:

* ^xt+1=At^xFt=At(^xt+Kt(yt−Gt^xt))𝑥^𝑡+1=𝐴𝑡𝑥^𝑡𝐹=𝐴𝑡(𝑥^𝑡+𝐾𝑡(𝑦𝑡−𝐺𝑡𝑥^𝑡))
* Σt+1=AtΣFtA⊺t+Qt=At(I−KtGt)ΣtA⊺t+QtΣ𝑡+1=𝐴𝑡Σ𝑡𝐹𝐴𝑡⊺+𝑄𝑡=𝐴𝑡(𝐼−𝐾𝑡𝐺𝑡)Σ𝑡𝐴𝑡⊺+𝑄𝑡

(với I𝐼 là ma trận đơn vị)

Giả định rằng ta có:

At=[1.2500−0.4]𝐴𝑡=[1.2500−0.4]

Qt=[0.120.1050.1050.18]𝑄𝑡=[0.120.1050.1050.18]

Chúng ta có phân bố dự đoán tương lai như sau:

A graph of a graph with circles and a arrow pointing to the center

Description automatically generated

**2. Tổng kết và thủ tục đệ quy:**

A diagram of a diagram of a number of objects

Description automatically generated with medium confidence

Kalman Filter sẽ thực hiện thủ tục đệ quy như gồm 2 thủ tục “đo lường” và “dự đoán”:

* Thủ tục cập nhật đo lường (Measurement):
  1. Tính Kalman Gain:  
     Kt=ΣtG⊺t(GtΣtG⊺t+Rt)−1𝐾𝑡=Σ𝑡𝐺𝑡⊺(𝐺𝑡Σ𝑡𝐺𝑡⊺+𝑅𝑡)−1
  2. Cập nhật trạng thái hiện tại thành phân bố lọc:  
     ^xt:=^xFt𝑥^𝑡:=𝑥^𝑡𝐹 với ^xFt=^xt+Kt(yt−Gt^xt)𝑥^𝑡𝐹=𝑥^𝑡+𝐾𝑡(𝑦𝑡−𝐺𝑡𝑥^𝑡)  
     Σt:=ΣFtΣ𝑡:=Σ𝑡𝐹 với ΣFt=(I−KtGt)ΣtΣ𝑡𝐹=(𝐼−𝐾𝑡𝐺𝑡)Σ𝑡  
     (với I𝐼 là ma trận đơn vị)
* Dự đoán tương lai (Prediction):
  1. Cập nhật trạng thái tương lai:  
     ^xt+1=At^xt+Btut𝑥^𝑡+1=𝐴𝑡𝑥^𝑡+𝐵𝑡𝑢𝑡  
     (trong các hệ điều khiển đơn giản Btut𝐵𝑡𝑢𝑡 khuyết)
  2. Cập nhập ma trận hiệp phương sai:  
     Σt+1=AtΣtA⊺t+QtΣ𝑡+1=𝐴𝑡Σ𝑡𝐴𝑡⊺+𝑄𝑡

A diagram of a computer program

Description automatically generated with medium confidence

**3. Cài đặt thuật toán và những vấn đề tính toán**

Chi phí tính toán của thuật toán chủ yếu đến từ việc nhân và lấy ma trận nghịch đảo, như đã trình bày ở trên nếu như vector ngẫu nhiên quan sát được yt𝑦𝑡 có số chiều nhỏ hơn nhiều so với xt𝑥𝑡 chúng ta có thể tính toán bằng khai triển ma trận nghịch đảo như trên để tính toán hiệu quả.

Việc thiết lập ma trận At𝐴𝑡, Qt𝑄𝑡, Gt𝐺𝑡, Rt𝑅𝑡 thường là do kỹ sư thiết lập dựa trên hiểu biết về dữ liệu của mình (các đại lượng vật lý, kinh tế,… thường có mối quan hệ mật thiết với nhau). Tuy nhiên có rất nhiều phương pháp học tham số cho mô hình Kalman Filter mà bạn đọc có thể tự tham khảo thêm (trong cộng đồng lý thuyết điều khiển thì vấn đề này được biết đến như systems identification, thường cách đơn giản nhất là bạn sẽ thu thập dữ liệu, xây dựng hàm mất mát và giải bài toán bình phương tối tiểu).

Nếu chú ý kĩ bạn sẽ thấy ma trận hiệp phương sai Σt+1Σ𝑡+1 chỉ phụ thuộc vào ΣtΣ𝑡 không phụ thuộc vào sự kiện quan sát được trả về từ bộ cảm biến yt𝑦𝑡, trong một số trường hợp đặc biệt, chúng ta có thể tính toán ma trận này trước khi yt𝑦𝑡 xuất hiện, việc này đòi hỏi bạn cần hiểu về sự hội tụ của ma trận ΣtΣ𝑡 xảy ra như thế nào (bạn đọc quan tâm tìm hiểu thêm về **Ricatti equations**).

Trong những ứng dụng thương mại lớn, Kalman Filter được thiết kế “tương đối phức tạp” vì tính ổn định tính toán số, thời gian chạy,… Nếu quan tâm đến những vấn đề này, bạn đọc có thể truy cập vào kho tài liệu đồ sộ về Kalman Filter ở đây: <http://www.cs.unc.edu/~welch/kalman/>

Hiện tại có rất nhiều thư viện cài đặt Kalman Filter sẳn như OpenCV (C/C++, Python), filterpy (Python), Matlab.

Dưới đây là một bài **mô phỏng đơn giản trên phân bố chuẩn 1 chiều** dựa trên ví dụ của Greg Welch và Gary Bishop, giả sử có một nguồn điện với hiệu điện thế không thay đổi theo thời gian (cho lý tưởng là vậy nhé, nghĩa là x=0.28𝑥=0.28 volts mãi luôn). Bạn thực sự không biết giá trị hiệu điện thế của nguồn điện này đâu, bạn quyết định đi tìm giá trị này như một thú vui lúc rảnh rỗi. May mắn thay, bạn tìm được một cái vôn kế đời cũ không chính xác. Rất may là nhà sản xuất có để lại thông tin, vôn kế này bị nhiễu theo phân bố N(0,0.01)𝑁(0,0.01). Biết rằng mỗi lần đo kế tiếp, hiệu điện thế không đổi gì nhiều (thiết lập ma trận chuyển trạng thái A𝐴), mô hình quan sát của được đo từ vôn kế bằng giá trị thực tế cộng thêm nhiễu (giả định ma trận G𝐺). Không rõ độ lệch chuẩn của nguồn điện phóng ra khi sử dụng bình thường là bao nhiêu, bạn tạm đoán là 11 volt (thiết lập ma trận ΣΣ), tại thời điểm ban đầu chưa biết bạn đoán nguồn điện 0.50.5 volt (thiết lập ^x𝑥^).