Lien ; <http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/>

Je l'ai depuis longtemps été interressés dans déclarants Kalman et comment ils fonctionnent, je également utilisé un filtre de Kalman pour mon [robot de](http://blog.tkjelectronics.dk/2012/03/the-balancing-robot/)équilibrage, mais je jamais expliqué comment elle a effectivement été mis en œuvre. En fait, je ne l'avais jamais pris le temps de s'asseoir avec un stylo et un morceau de papier et d'essayer de faire le calcul par moi-même, donc je ne savent réellement pas comment il a été mis en œuvre.   
Il est avéré être une bonne chose, comme je l'ai fait découvert une erreur dans le code original, mais je vais y revenir plus tard.

En fait, je écrit sur ​​le filtre de Kalman que ma mission principale à l'école secondaire en Décembre 2011. Mais je ne utilisé le filtre de Kalman pour calculer la véritable tension d'un signal continu modulé par bruit blanc gaussien connue. Ma mission peut être trouvée dans le fichier zip suivant:http://www.tkjelectronics.dk/uploads/Kalman\_SRP.zip. Il est en danois, mais vous pouvez utiliser correctement Google traduction pour traduire certaines d'entre elles. Si vous avez des questions spécifiques concernant l'affectation, puis demander dans les commentaires ci-dessous.

D'accord, mais revenir à l'objet. Comme je l'ai triste, je ne l'avais jamais pris le temps de s'asseoir et de faire le calcul concernant le filtre de Kalman sur la base d'un accéléromètre et d'un gyroscope. Il n'a pas été aussi difficile que je m'y attendais, mais je dois avouer que je ne l'ai toujours pas étudié la théorie plus profonde derrière, pourquoi cela fonctionne réellement. Mais pour moi, et la plupart des gens là-bas, je suis plus interressés dans l'application du filtre, que dans la théorie plus profonde derrière et pourquoi les équations travaux.  
  
Avant de commencer, vous devez avoir quelques connaissances de base sur les matrices comme la multiplication de matrices et de transposition de matrices. Si non, s'il vous plaît jeter un oeil sur les sites suivants:

* <http://en.wikipedia.org/wiki/Matrix_multiplication#Matrix_product_.28two_matrices.29>
* <http://www.mathwarehouse.com/algebra/matrix/multiply-matrix.php>
* <http://en.wikipedia.org/wiki/Transpose>
* <http://en.wikipedia.org/wiki/Covariance_matrix>

Pour ceux d'entre vous qui ne savent pas ce qu'est un filtre de Kalman est, il est un algorithme qui utilise une série de mesures observées au fil du temps, dans ce contexte, un accéléromètre et un gyroscope. Ces mesures vont contenir bruit qui contribue à l'erreur de la mesure. Le filtre de Kalman va alors essayer d'estimer l'état du système, sur la base des états actuels et antérieurs, qui ont tendance à être plus précis que les mesures que les seuls.

Dans ce contexte, le problème est que l'accéléromètre est en général très bruit quand il est utilisé pour mesurer l'accélération de la pesanteur puisque le robot se déplace d'avant en arrière. Le problème avec le gyroscope est qu'il dérive au fil du temps -. Tout comme un rouet-gyro va commencer à tomber quand il est en perte de vitesse   
En bref, vous pouvez dire que vous ne pouvez faire confiance au gyroscope sur un court terme tandis que vous pouvez seulement confiance à l'accéléromètre sur un long terme.

Il est en fait un moyen très facile de régler ce problème en utilisant un filtre gratuit, qui fondamentalement juste se composent d'un filtre numérique passe-bas sur l'accéléromètre et filtre passe-haut numérique sur les lectures de gyroscope. Mais il est pas aussi précis que le filtre de Kalman, mais d'autres personnes ont avec succès construire des robots d'équilibrage en utilisant un filtre gratuit affiné.

Plus d'informations sur des gyroscopes, accéléromètres et des filtres gratuits peut être trouvé dans ce pdf. Une comparaison entre un filtre gratuit et un filtre de Kalman peut être trouvé dans la suiteblog.

Le filtre de Kalman fonctionne en produisant une estimation statistiquement optimale de l'état du système sur la base de la mesure (s). Pour ce faire, il aura besoin de connaître le bruit de l'entrée du filtre appelé le bruit de mesure, mais aussi le bruit du système lui-même appelé le bruit de processus. Pour ce faire, le bruit doit être [une distribution gaussienne](http://en.wikipedia.org/wiki/Normal_distribution) et avoir une moyenne de zéro, heureusement pour nous, le bruit plus aléatoire ont cette caractéristique.   
Pour plus d'informations à propos de la théorie derrière le filtre de jeter un oeil sur les pages suivantes:

* <http://en.wikipedia.org/wiki/Kalman_filter>
* <http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf>
* <http://academic.csuohio.edu/simond/courses/eec644/kalman.pdf>

**L'état du système\ boldsymbol {x} _k**  
La prochaine de cet article peut sembler très déroutant pour certains, mais je vous promets, si vous prenez un stylo et un morceau de papier et essayez de suivre, il est pas si difficile si vous êtes raisonnable en maths.

Si vous, comme moi, ne disposez pas d'un programme de calculatrice ou un ordinateur qui peut fonctionner avec des matrices, alors je vous recommande la calculatrice en ligne gratuit [Wolfram](http://www.wolframalpha.com/" \t "_blank)Alpha. Je l'ai utilisé pour tous les calculs de cet article.

Je vais utiliser la même notation que [l'article de](http://en.wikipedia.org/wiki/Kalman_filter)wikipedia, mais je vais faire remarquer que lorsque les matrices sont constantes et ne dépend pas de l'heure actuelle, vous ne devez pas écrire le k après eux. Ainsi, par exemple F_kpeut être simplifiée Fpour.

Aussi je voudrais écrire une petite explication sur les autres aspects des notations.   
Je vais d'abord faire une note sur ce qui est appelé *l'état*précédent:

\ boldsymbol {\ hat {x}} _ {k-1 |  k-1}

Quel est l'état estimé précédente sur la base de l'état précédent et les estimations des Etats dont il est saisi.

La prochaine est le *état ​​a*priori:

\ boldsymbol {\ hat {x}} _ {k |  k-1}

A priori donc l'estimation de la matrice de l'Etat à l'heure actuelle k basée sur l'état antérieur du système et les estimations des Etats dont il est saisi.

Le dernier est appelé *un état*​​posteriori:

\ boldsymbol {\ hat {x}} _ {k |  k}

Est l'estimation de l'état à l'instant k observations données jusqu'à et y compris à l'instant k.

Le problème est que l'état du système lui-même est caché et ne peut être observée à travers z_kl'observation. Il est également appelé un [modèle de Markov](http://en.wikipedia.org/wiki/Hidden_Markov_model)caché.

Cela signifie que l'État sera fondé sur l'état à l'instant k et tous les états précédents. Cela signifie également que vous ne pouvez pas faire confiance à l'estimation de l'état avant le filtre de Kalman a stabilisé - jetez un oeil à le graphique à la page d'accueil de [mon](http://www.tkjelectronics.dk/uploads/Kalman_SRP.zip)affectation.

Le chapeau sur les \ hat {x}moyens qui est l'estimation de l'état. Contrairement à un seul Xce qui signifie l'état réel - celui que nous essayons d'estimer.   
Donc, la notation de l'état à l'instant k est:

\ boldsymbol {x} _k 

L'état du système à l'instant k si elle est donnée par:

\ boldsymbol {x} _k = \ boldsymbol {F} {x_ k-1} + \ boldsymbol {B} u_k + w_k

Où x_kest la matrice de l'Etat qui est donné par:

\ boldsymbol {x} _k = \ begin {} bmatrix \ theta \\ \ dot {\ theta} _b \ end {} _k bmatrix

Comme vous pouvez voir la sortie du filtre sera l'angle \ thetamais aussi le parti pris \ dot {\ theta} _bsur la base des mesures de l'accéléromètre et du gyroscope. Le biais est le montant le gyroscope a dérivé. Cela signifie que l'on peut obtenir le taux réel en soustrayant le biais de la mesure du gyroscope.

La prochaine est la Fmatrice, qui est le modèle de transition d'état qui est appliqué à l'état de x_ {k-1}prevouis.

Dans ce cas Fest défini comme:

\ boldsymbol {F} = \ begin {} bmatrix 1 & - \ Delta t \\ 0 & 1 \ end {} bmatrix

Je sais que le - \ Delta tpeut sembler déroutant, mais il sera logique plus tard (jetez un oeil à moncommentaire).

Le prochain est l'entrée de Royaume-Unicommande, dans ce cas, il est la mesure du gyroscope en degrés par seconde (° / s) à l'instant k, ceci est aussi appelé le \ dot {\ theta}taux. Nous allons effectivement réécrire l'équation d'état que:

\ boldsymbol {x} _k = \ boldsymbol {F} {x_ k-1} + \ boldsymbol {B} {\ dot {\ theta}} + w_k _k

La prochaine chose est la Bmatrice. Qui est appelé le modèle de contrôle d'entrée, qui est défini comme:

\ boldsymbol {B} = \ begin {bmatrix} \ Delta t \\ 0 \ end {} bmatrix

Cela rend parfaitement logique que vous obtiendrez l'angle \ thetalorsque vous multipliez le taux \ dot {\ theta}par le temps delta \ Delta tet puisque nous ne pouvons pas calculer le biais directement basé sur le taux, nous allons régler le bas de la matrice à 0.

w_kest le bruit de processus qui est une distribution gaussienne avec une moyenne nulle et de covariance Qà l'instant k:

\ w} {boldsymbol _k \ sim N \ left (0, \ {Q} boldsymbol _k \ right)

Q_kest la matrice de covariance de bruit de processus et dans ce cas, la matrice de covariance de l'estimation de l'état de l'accéléromètre et de partialité. Dans ce cas, nous allons examiner l'estimation de la partialité et l'accéléromètre pour être indépendant, il est donc en fait juste égale à la variance de l'estimation de l'accéléromètre et de partialité.   
La matrice finale est définie comme ceci:

\ {Q} boldsymbol _k = \ begin {} bmatrix Q_ \ theta & 0 0 \\ & Q _ {\ dot {\ theta} _b} \ end {bmatrix} \ Delta t 

Comme vous pouvez voir la Q_kmatrice de covariance dépend du courant k de temps, de sorte que la variance de l'accéléromètre Q_ \ thetaet la variance de la partialité Q _ {\ dot {\ theta _b}}est multiplié par le temps \ Delta tdelta.   
Cela est logique que le bruit de processus sera plus grande que plus de temps, il est depuis la dernière mise à jour de l'État. Par exemple, le gyroscope aurait dérivé.   
Vous aurez besoin de connaître ces constantes pour le filtre de Kalman à travailler.   
Notez que si vous définissez une valeur plus grande, plus le bruit dans l'estimation de l'état. Ainsi, par exemple si l'angle estimé commence à dériver, vous devez augmenter la valeur Q _ {\ dot {\ theta _b}}de. Sinon, si l'estimation tend à être lent, vous approuvez l'estimation de l'angle trop et devriez essayer de diminuer la valeur de Q_ \ thetala rendre plus réactive.

**La mesure\ boldsymbol {z} _k**  
Maintenant, nous allons jeter un oeil à l'observation ou la mesure z_kde l'état x_kvrai. L'observation z_kest donnée par:

\ boldsymbol {z} _k = \ boldsymbol {H} x_ {k} + v_k

Comme vous pouvez voir la mesure z_kest donnée par l'état actuel x_kmultiplié par la Hmatrice plus le bruit de v_kmesure.

Hest appelé le modèle d'observation et est utilisé pour cartographier l'espace d'état vrai dans l'espace observé. Le véritable état ​​ne peut pas être observé. Etant donné que la mesure est simplement la mesure de l'accéléromètre, Hest donnée par:

\ boldsymbol {H} = \ begin {} bmatrix 1 & 0 \ end {} bmatrix

Le bruit de la mesure à être une distribution gaussienne ainsi avec une moyenne de zéro et Rque la covariance:

\ {c} boldsymbol _k \ sim N \ left (0, \ boldsymbol {R} \ right)

Mais, Rest une matrice pas le bruit de mesure est juste égale à la variance de la mesure, puisque la covariance de la même variable est égale à la variance. Voir ce [la page](http://en.wikipedia.org/wiki/Covariance_matrix#Generalization_of_the_variance) pour plus d'informations.   
Maintenant, nous pouvons définir Rcomme si:

\ boldsymbol {R} = E \ begin {} bmatrix v_k & {} ^ v_k T \ end {bmatrix} = var (v_k) 

Plus d'informations sur la covariance peut être trouvée sur [Wikipedia](http://en.wikipedia.org/wiki/Covariance_matrix" \l "Conflicting_nomenclatures_and_notations" \t "_blank) et dans [ma](http://www.tkjelectronics.dk/uploads/Kalman_SRP.zip)mission.

Nous supposons que le bruit de mesure est le même et ne dépend pas du temps k:

var (v_k) = var (v) 

Notez que si vous définissez la variance du bruit de mesure var (v)trop élevée, le filtre va répondre très lentement car il a confiance de nouvelles mesures moins, mais si elle est trop petite la valeur pourrait dépassement et être bruyant car nous faisons confiance aux mesures de l'accéléromètre trop.

Donc, pour arrondir vous devez trouver les variances de bruit de processus Q_ \ thetaet Q _ {\ dot {\ theta _b}}et la variance de mesure du bruit de var (v)mesure. Il ya plusieurs façons de les trouver, mais il est hors de l'aspect de cet article.

**Les équations du filtre de Kalman**  
Bon maintenant aux équations nous allons utiliser pour estimer le véritable état ​​du système à l'instant \ hat {x} _kk. Certains gars intelligents sont venus avec des équations trouvées ci-dessous pour estimer l'état du système.   
Les équations peuvent être écrites plus compact, mais je préfère avoir les a déployés, de sorte qu'il est plus facile à mettre en œuvre et de comprendre les différentes étapes.

**Prédire**  
Dans les deux premières équations, nous allons essayer de prédire l'état actuel et l'erreur de matrice de covariance à l'instant k. D'abord, le filtre va essayer d'estimer l'état actuel, basé sur tous les états antérieurs et la mesure du gyroscope:

\ boldsymbol {\ hat {x}} _ {k |  k-1} = \ boldsymbol {F} \ hat {x} _ {k-1 |  k-1} + \ boldsymbol {B} {\ dot {\ theta}} _k

Voilà aussi pourquoi il est appelé une entrée de commande, car nous l'utilisons comme une entrée supplémentaire pour estimer l'état à l'heure actuelle k appelé l'état a priori \ hat {x} _ {k |  k-1}comme décrit dans le début de l'article.

La prochaine chose est que nous allons essayer d'estimer a priori la matrice de covariance d'erreur P_ {k |  k-1}sur la base de la matrice de covariance d'erreur P_ {k-1 |  k-1}précédente, qui est défini comme:

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

Cette matrice est utilisée pour estimer combien nous faisons confiance aux valeurs actuelles de l'état estimé. Le petit plus nous espérons l'état actuel estimé. Le principe de l'équation ci-dessus est en fait assez facile à comprendre, car il est assez évident que la covariance d'erreur va augmenter puisque nous Dernière mise à jour de l'estimation de l'état, donc nous avons multiplié la matrice de covariance des erreurs par le modèle de transition d'état Fet de la transposition de la que F ^ Tet ajouter le bruit de processus en cours Q_kà l'instant k.

L'erreur de matrice de covariance Pdans notre cas est une matrice 2 × 2:

\ boldsymbol {P} = \ begin {} bmatrix P_ {00} {01 & P_} \\ P_ {10} & P_ {11} \ end {} bmatrix

**Mettre à jour**  
Le poing chose que nous allons faire est de calculer la différence entre la mesure z_ket l'état a x_ {k |  k-1}priori, ceci est aussi appelé l'innovation:

\ boldsymbol {\ tilde {y}} _ k = \ {z} boldsymbol _k - \ boldsymbol {H} \ hat {x} _ {k |  k-1}

Le modèle d'observation Hest utilisée pour cartographier l'état a priori x_ {k |  k-1}dans l'espace observé qui est la mesure de l'accéléromètre, donc de l'innovation est pas une matrice

\ boldsymbol {\ tilde {y}} _ k = \ begin {bmatrix} \ boldsymbol {\ tilde {y}} \ end {} _k bmatrix

La prochaine chose que nous allons faire est de calculer ce qui est appelé la covariance de l'innovation:

\ boldsymbol {S} _k = \ boldsymbol {H} \ boldsymbol {P} _ {k |  k-1} \ boldsymbol H {} ^ T + \ boldsymbol {R}

Ce qu'il fait est qu'il tente de prédire combien nous devrions faire confiance la mesure fondée sur l'a priori erreur matrice de covariance P_ {k |  k-1}et la matrice de covariance Rmesure. Le modèle d'observation Hest utilisée pour cartographier l'a priori erreur matrice de covariance P_ {k |  k-1}dans l'espace observé.

Plus la valeur du bruit de mesure plus la valeur Sde, cela signifie que nous ne faisons pas confiance la mesure entrant tant que ça.   
Dans ce cas, Sne sont pas une matrice et est simplement écrit:

\ boldsymbol {S} _k = \ begin {bmatrix} \ boldsymbol {S} \ end {} _k bmatrix

La prochaine étape consiste à calculer le gain de Kalman. Le gain de Kalman est utilisé pour indiquer combien nous avons confiance de l'innovation et se définit comme:

\ boldsymbol {K} _k = \ boldsymbol {P} _ {k |  k-1} \ boldsymbol H {} ^ T \ boldsymbol {S} _k ^ {- 1}

Vous pouvez voir que si nous ne faisons pas confiance l'innovation que beaucoup de la covariance de l'innovation Ssera élevé et si nous faisons confiance à l'estimation de l'état alors la matrice de covariance d'erreur Psera faible gain de Kalman sera donc petite et oppesite si nous faisons confiance l'innovation mais ne pas faire confiance à l'estimation de l'état actuel.

Si vous prenez un regard plus profond, vous pouvez voir que la transposition du modèle d'observation Hest utilisée pour cartographier l'état de l'erreur de matrice de covariance Pdans l'espace observé. On compare alors la matrice de covariance d'erreur en multipliant par l'inverse de la covariance de Sl'innovation.

Ce sentiment de maquillage que nous allons utiliser le modèle d'observation Hpour extraire des données à partir de l'erreur covariance de l'Etat et les comparer avec l'estimation actuelle de la covariance de l'innovation.

Notez que si vous ne connaissez pas l'état au démarrage, vous pouvez définir la matrice de covariance d'erreur comme ceci:

\ boldsymbol {P} = \ begin {} bmatrix L & 0 0 \\ & L \ end {} bmatrix

Où Lreprésentent un grand nombre.

Pour ma robots équilibrage Je sais que l'angle de départ et je trouve le biais du gyroscope au démarrage en calibrant, donc je suppose que l'Etat sera connu au démarrage, donc je initialiser la matrice de covariance d'erreur comme ceci:

\ boldsymbol {P} = \ begin {} bmatrix 0 & 0 \\ 0 & 0 \ end {} bmatrix

Jetez un oeil à [ma routine d'étalonnage](https://github.com/TKJElectronics/BalancingRobotArduino/blob/master/BalancingRobotArduino.ino#L448-L454) pour plus d'informations.

Dans ce cas, le gain de Kalman est une matrice 2 × 1:

\ boldsymbol {K} = \ begin {} bmatrix K_0 \\ K_1 \ end {} bmatrix

Maintenant, nous pouvons mettre à jour l'estimation de l'état actuel a posteriori:

\ boldsymbol {\ hat {x}} _ {k |  k} = \ boldsymbol {\ hat {x}} _ {k |  k-1} + \ boldsymbol {K} _k \;  \ boldsymbol {\ tilde {y}} _ k

Cela se fait en ajoutant l'état a priori \ hat {x} _ {k |  k-1}avec le gain de Kalman multiplié par le \ tilde {y} _kinnovation.   
Rappelez-vous que l'innovation \ tilde {y} _kest la différence entre la mesure z_ket l'état priori \ hat {x} _ {k |  k-1}estimée,de sorte que le l'innovation peut être à la fois positive et négative.

Un peu simplifié l'équation peut être comprise comme nous corrigeons simplement l'estimation de l'état a \ hat {x} _ {k |  k-1}priori, qui a été calculée en utilisant l'état précédent et la mesure du gyroscope, avec la mesure - dans ce cas, l'accéléromètre.

La dernière chose que nous allons faire est de mettre à jour la matrice de covariance d'erreur a posteriori:

\ boldsymbol {P} _ {k |  k} = (\ boldsymbol {I} - \ boldsymbol {K} _k \ boldsymbol {H}) \ boldsymbol {P} _ {k |  k-1}

Lorsque jel'on appelle la matrice identité et est défini comme:

\ boldsymbol {I} = \ begin {} bmatrix 1 & 0 \\ 0 & 1 \ end {} bmatrix

Ce que le filtre est fait est qu'il est fondamentalement auto-correction de la matrice de covariance d'erreur sur la base de combien nous avons corrigé l'estimation. Ce sentiment de faire comme nous corrigée en fonction de l'état l'a priori erreur matrice de P_ {k |  k-1}covariance, mais aussi la covariance de S_kl'innovation.

**Mise en œuvre du filtre**  
Dans cette section, je vais utiliser l'équation d'en haut pour mettre en œuvre le filtre dans un code C ++ simple qui peut être utilisé pour [l'équilibrage des](http://youtu.be/N28C_JqVhGU)robots, [quadcopters](http://blog.tkjelectronics.dk/2012/03/quadcopters-how-to-get-started/" \t "_blank) et d'autres applications où vous avez besoin pour calculer l'angle, la partialité ou le taux.

Dans le cas où vous voulez le code à côté de vous, il peut être trouvé à github:https://github.com/TKJElectronics/KalmanFilter.

Je vais simplement écrire les équations en haut de chaque étape, puis de les simplifier, après que je vais écrire comment il est peut être fait i C et finalement je vais créer un lien vers calculs effectués dans [Wolfram Alpha](http://wolframalpha.com/) dans le fond de chaque étape, comme je le faisais qu'ils fassent le calcul.

**Étape 1:**

\ boldsymbol {\ hat {x}} _ {k |  k-1} = \ boldsymbol {F} \ hat {x} _ {k-1 |  k-1} + \ boldsymbol {B} {\ dot {\ theta}} _k

 = \ Begin {} bmatrix \ theta - \ dot {\ theta} _b \ Delta t + \ dot {\ theta} \ Delta t \\ \ dot {\ theta} _b \ end {} bmatrix 

 = \ Begin {} bmatrix \ theta + \ Delta t (\ dot {\ theta} - \ dot {\ theta} _b) \\ \ dot {\ theta} _b \ end {} bmatrix 

Comme vous pouvez voir l'estimation de l'angle a priori \ hat {\ theta} {k _ |  k-1}est égale à l'estimation de l'état \ hat {\ theta} _ {k-1 |  k-1}précédent, plus les temps de taux impartiales le temps \ Delta tdelta.   
Puisque nous ne pouvons pas mesurer directement le biais de l'estimation du biais a priori est juste égal à la le précédent.

Cela peut être écrit en C comme ceci:

taux = newRate - polarisation;  
angle + = dt \* taux;

Notez que je calcule le taux impartiale, de sorte qu'il peut être utilisé par l'utilisateur ainsi.

Liens Wolfram Alpha:

* [Eq. 1.1](http://www.wolframalpha.com/input/?i=%7B%7B1%2C-t%7D%2C%7B0%2C1%7D%7D*%7B%7Btheta%7D%2C%7Bb%7D%7D%2B%7B%7Bt%7D%2C%7B0%7D%7D*r)

**Étape 2:**

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

Les équations ci-dessus peuvent être écrites en C comme ceci:

P [ 0 ] [ 0 ] += dt \* ( dt \* P [ 1 ] [ 1 ] - P [ 0 ] [ 1 ] - P [ 1 ] [ 0 ] + Q\_angle ) ;  
P [ 0 ] [ 1 ] -= dt \* P [ 1 ] [ 1 ] ;  
P [ 1 ] [ 0 ] -= dt \* P [ 1 ] [ 1 ] ;  
P [ 1 ] [ 1 ] += Q\_gyroBias \* dt ;

Notez que ceci est la partie du code qu'il y avait une erreur dans le code d'origine que je l'habitude.

Liens Wolfram Alpha:

* [Eq. 2.1](http://www.wolframalpha.com/input/?i=%7B%7B1%2C-t%7D%2C%7B0%2C1%7D%7D%2A%7B%7BP_00%2CP_01%7D%2C%7BP_10%2CP_11%7D%7D)
* [Eq. 2.2](http://www.wolframalpha.com/input/?i=%7B%7BP_00-t*P_10%2CP_01-t*P_11%7D%2C%7BP_10%2CP_11%7D%7D*%7B%7B1%2C0%7D%2C%7B-t%2C1%7D%7D)
* [Eq. 2.3](http://www.wolframalpha.com/input/?i=%7B%7B1%2C-t%7D%2C%7B0%2C1%7D%7D*%7B%7BP_00%2CP_01%7D%2C%7BP_10%2CP_11%7D%7D*%7B%7B1%2C0%7D%2C%7B-t%2C1%7D%7D%2B%7B%7BQ_a%2C0%7D%2C%7B0%2CQ_b%7D%7D*t)
* [Eq. 2.4](http://www.wolframalpha.com/input/?i=P_0-t*P_10-t*%28P_1-t*P_11%29%2Bt*Q_1)

**Etape 3:**

\ boldsymbol {\ tilde {y}} _ k = \ {z} boldsymbol _k - \ boldsymbol {H} \ hat {x} _ {k |  k-1}

= \ {Z} boldsymbol _k - \ begin {} bmatrix 1 & 0 \ end {bmatrix} \ begin {} bmatrix \ theta \\ \ dot {\ theta} _b \ end {} bmatrix _ {k |  k-1}

= \ {Z} boldsymbol _k - \ theta_ {k |  k-1}

L'innovation peut être calculé de C comme ceci:

y = newAngle - angle;

Liens Wolfram Alpha:

* [Eq. 3.1](http://www.wolframalpha.com/input/?i=%7B%7B1%2C0%7D%7D*%7B%7Ba%7D%2C%7Bb%7D%7D)

**Étape 4:**

\ boldsymbol {S} _k = \ boldsymbol {H} \ boldsymbol {P} _ {k |  k-1} \ boldsymbol H {} ^ T + \ boldsymbol {R}

= {{P_ 00}} _ {k |  k-1} + \ boldsymbol {R}

= {{P_ 00}} _ {k |  k-1} + var (v)

Encore une fois le code C est assez simple:

S = P [0] [0] + R\_measure;

Liens Wolfram Alpha:

* [Eq. 4.1](http://www.wolframalpha.com/input/?i=%7B%7B1%2C0%7D%7D*%7B%7BP_00%2CP_01%7D%2C%7BP_10%2CP_11%7D%7D*%7B%7B1%7D%2C%7B0%7D%7D)

**Etape 5:**

\ boldsymbol {K} _k = \ boldsymbol {P} _ {k |  k-1} \ boldsymbol H {} ^ T \ boldsymbol {S} _k ^ {- 1}

= \ Begin {} bmatrix P_ {00} \\ P_ {10} \ end {} bmatrix _ {k |  k-1} \ boldsymbol {S} _k ^ {- 1}



Notez que dans d'autres cas Speut être une matrice et vous ne pouvez pas tout simplement diviserpar. Au lieu de cela, vous devez calculer l'inverse de la matrice. Voir ci-dessous [la page](http://mathworld.wolfram.com/MatrixInverse.html) pour plus d'informations sur la façon de le faire.PS

La mise en œuvre de C ressemble à ceci:

K [0] = P [0] [0] / S; K [1] = P [1] [0] / S; 

Liens Wolfram Alpha:

* [Eq. 5.1](http://www.wolframalpha.com/input/?i=%7B%7BP_00%2CP_01%7D%2C%7BP_10%2CP_11%7D%7D*%7B%7B1%7D%2C%7B0%7D%7D)

**Etape 6:**

\ boldsymbol {\ hat {x}} _ {k |  k} = \ boldsymbol {\ hat {x}} _ {k |  k-1} + \ boldsymbol {K} _k \;  \ boldsymbol {\ tilde {y}} _ k

Encore une fois, l'équation finissent assez court, et peut être écrit comme si dans C:

angle + = K [0] \* y; biais + = K [1] \* y; 

**Etape 7:**

\ boldsymbol {P} _ {k |  k} = (\ boldsymbol {I} - \ boldsymbol {K} _k \ boldsymbol {H}) \ boldsymbol {P} _ {k |  k-1}

. Rappelez-vous que nous réduisons à nouveau la matrice de covariance d'erreur, puisque l'erreur de l'estimation de l'état a été diminué   
Le code C ressemble à ceci:

float P00\_temp = P [ 0 ] [ 0 ] ;   
float P01\_temp = P [ 0 ] [ 1 ] ;  
  
P [ 0 ] [ 0 ] -= K [ 0 ] \* P00\_temp ;  
P [ 0 ] [ 1 ] -= K [ 0 ] \* P01\_temp ;  
P [ 1 ] [ 0 ] -= K [ 1 ] \* P00\_temp ;  
P [ 1 ] [ 1 ] -= K [ 1 ] \* P01\_temp ;

Liens Wolfram Alpha:

* [Eq. 7.1](http://www.wolframalpha.com/input/?i=%7B%7BK_0%7D%2C%7BK_1%7D%7D*%7B%7B1%2C0%7D%7D)
* [Eq. 7.2](http://www.wolframalpha.com/input/?i=%7B%7B1%2C0%7D%2C%7B0%2C1%7D%7D*%7B%7BP_00%2CP_01%7D%2C%7BP_10%2CP_11%7D%7D)
* [Eq. 7.3](http://www.wolframalpha.com/input/?i=%7B%7BK_0%2C0%7D%2C%7BK_1%2C0%7D%7D*%7B%7BP_00%2CP_01%7D%2C%7BP_10%2CP_11%7D%7D)

Notez que je l'ai constaté que les écarts suivants fonctionne parfaitement pour la plupart des IMU:

flotter Q\_angle = 0,001;   
flotter Q\_gyroBias = 0,003;   
flotter R\_measure = 0,03;

Rappelez-vous qu'il est très important de régler l'angle de la cible au démarrage si vous avez besoin d'utiliser la sortie au démarrage. Pour plus d'informations, voir le [programme d'étalonnage pour mon robot de](https://github.com/TKJElectronics/BalancingRobotArduino/blob/master/BalancingRobotArduino.ino#L448-L454)l'équilibrage.

Dans le cas où vous l'avez manqué ici est la bibliothèque je l'ai écrit une bibliothèque qui peut être utilisé par tout microcontrôleur qui prend en charge les mathématiques flottante. Le code source peut être trouvé à github: https://github.com/TKJElectronics/KalmanFilter.

Si vous préférez une explication vidéo sur le filtre de Kalman, je recommande la série vidéo suivante: http://www.youtube.com/watch?v=FkCT\_LV9Syk.

Notez que vous ne pouvez pas utiliser la bibliothèque si vous avez besoin de représenter quelque chose dans une pleine orientations 3D, comme des angles de Euler souffrent de ce qu'on appelle[Gimbal verrouiller](http://en.wikipedia.org/wiki/Gimbal_lock) vous devrez utiliser Quaternions de le faire, mais qui est toute une histoire nother. Pour l'instant, jetez un oeil à ce qui suit [la](http://www.x-io.co.uk/node/8#open_source_imu_and_ahrs_algorithms)page.

Ceci est tout pour savoir, je l'espère que vous trouverez i serviable, si vous faites ou avez des questions ont chuté de laisser un commentaire ci-dessous - il prend en charge [la syntaxe LaTeX](http://en.wikibooks.org/wiki/LaTeX/Mathematics)ainsi, si vous avez besoin d'écrire des équations.   
Si vous repérez des erreurs s'il vous plaît laissez-moi savoir aussi.