TP n°2 : Filtrage de Kalman

1st Picard Hugo
Intelligence artificcielle embarquée
ESIEE – 2025
France
hugo.picard@edu.esiee.fr

Résumé – Dans ce compte rendu nous allons vous expliquer comment nous avons implémenté le filtre de Kalman et comment nous l'avons expérimenté à travers différentes applications. Dans ce TP nous avons considéré des systèmes linéaires, pouvant être décrits par un modèle d'état comme :

$$\underline{x}(n+1)=F(n+1,n)\cdot\underline{x}(n)+\underline{u}(n) \quad \textbf{équation d'état}$$
$$\underline{y}(n)=C(n)\cdot\underline{x}(n)+\underline{w}(n) \quad \textbf{équation d'observation}$$

où :

- $\underline{x}(n)$ **est le vecteur d'état à l'instant** ***n*****, de taille** ***Mx1*****,**

- $F(n+1,n)$ **est la matrice de transition d'état de l'instant** ***n*** **à l'instant** ***n*****+1 (de taille** ***MxM*****),**

- $\underline{u}(n)$ **est le vecteur d'erreur de modélisation du processus, de taille** ***Mx1*****, supposé être un bruit blanc de matrice de covariance** $E[\underline{u}(n)\cdot\underline{u}(k)^H]=\begin{Bmatrix} R_u(n) \, si \, n=k \\ 0_{MxM} \, si \, n\neq k \end{Bmatrix}$

- **y(n) est le vecteur d'observation (mesure) à l'instant** ***n*****, de taille** ***Nx1*****,**

- **C(n) est la matrice de taille** ***NxM*** **qui modélise le processus de mesure,**

- **w(n) est le vecteur d'erreur d'observation à l'instant** ***n*****, de taille** ***Nx1*** **, supposé être un bruit blanc de matrice de covariance** $E[\underline{w}(n)\cdot\underline{w}(k)^H]=\begin{Bmatrix} R_w(n) \, si \, n=k \\ 0_{NxN} \, si \, n\neq k \end{Bmatrix}$

I – Génération de vecteurs aléatoires gaussiens

On considère un vecteur aléatoire $\underline{x}$ de moyenne $\underline{m_x}$ et de matrice de covariance R_x. On rappelle que, par définition, $\underline{m_x} = E[\underline{x}]$ *et* $R_x = E[(\underline{x} - \underline{m_x}) \cdot (\underline{x} - \underline{m_x})^H]$, et que, par conséquent, la matrice R_x est toujours hermitienne et définie positive.

Soit le vecteur aléatoire $\underline{y}$ lié au vecteur $\underline{x}$ par la relation suivante : $\underline{y} = A \cdot \underline{x} + \underline{b}$, où A et b sont déterministes.

On cherche à calculer théoriquement $\underline{m_y}$ et $\underline{y}$ et sa matrice de covariance $R_y = E[(\underline{y} - \underline{m_y})(\underline{y} - \underline{m_y})^H]$

1.1 Calcul de $\underline{m_y}$

On sait que :

$$\underline{m_y} = E[\underline{y}]$$

On sait aussi que :

$$\underline{y} = A \cdot \underline{x} + \underline{b}$$

Ainsi on a :

$$\underline{m_y} = E[\underline{y}] = E[A \cdot \underline{x} + \underline{b}] = A \cdot E[\underline{x}] + \underline{b} = A \cdot \underline{m_x} + \underline{b}$$

Car on rappelle :

$$\underline{m_x} = E[\underline{x}]$$

1.2 Calcul de R_y

On connaît pour commencer l'équation à laquelle répond la matrice R_y :

$$R_y = E[(\underline{y} - \underline{m_y})(\underline{y} - \underline{m_y})^H]$$

On connaît $\underline{y}$ et $\underline{m_y}$ donc on sait que :

$$\underline{y} - \underline{m_y} = A \cdot \underline{x} + \underline{b} - (A \cdot \underline{m_x} + \underline{b}) = A \cdot (\underline{x} + \underline{m_x})$$

On peut donc réécrire : $R_y = E[(\underline{y} - \underline{m_y})(\underline{y} - \underline{m_y})^H]$

$$R_y = E[(\underline{y} - \underline{m_y})(\underline{y} - \underline{m_y})^H] = E[(A \cdot (\underline{x} + \underline{m_x})) \cdot (A \cdot (\underline{x} + \underline{m_x}))^H]$$

On rappelle la propriétés basiques de la transposée :

$$(A \cdot B)^H = B^H \cdot A^H$$

on obtient alors :

$$R_y = E[(A \cdot (\underline{x} - \underline{m_x})) \cdot (A \cdot (\underline{x} - \underline{m_x}))^H] = E[A \cdot (\underline{x} - \underline{m_x}) \cdot (\underline{x} - \underline{m_x})^H \cdot A^H]$$

On reconnaît R_x dans l'équation :

$$R_y = E[A \cdot (\underline{x} - \underline{m_x}) \cdot (\underline{x} - \underline{m_x})^H \cdot A^H] = A \cdot E[(\underline{x} - \underline{m_x}) \cdot (\underline{x} - \underline{m_x})^H] \cdot A^H = A \cdot R_x \cdot A^H$$

Résumé :

$$R_y = A \cdot R_x \cdot A^H \text{ et } \underline{m_y} = A \cdot \underline{m_x} + \underline{b}$$

2. Moyenne nulle et covariance égale à l'identité

2.1 Explication du code

On nous demande faire réaliser un script permettant de générez un nuage de 1000 points représentant K réalisations d'un vecteur aléatoire gaussien $\underline{x}$ de R^2, supposé centré et de matrice de covariance égale à l'identité. On présentera donc le script ainsi que la fonction trace_ellipse utilisé pour tracer les ellipses de confiance de niveaux 0.9 ,0.99 , et 0.999 associées à $\underline{x}$.

```
6
7       K = 1000;
8
```

Donc pour commencer on fixe le nombre de réalisations aléatoires. Ici on souhaite générer 1000 points pour avoir un nuage suffisamment représentatif d'une loi gaussienne. On attribut ainsi 1000 à K.

```
9
10      x = randn(2, K);
11
```

Ici on génère un vecteur aléatoire gaussien bidimensionnel, de moyenne nulle, de variance unitaire et composantes indépendantes (Doc MATLAB *randn*() en annexe) Chaque colonne de x correspond à une réalisation du vecteur aléatoire. La dimension de x est (2 x K) 2 lignes et K colonnes.

```
12      figure;
13      plot(x(1,:), x(2,:),'.');
14      axis equal;
15      grid on;
16      hold on;
```

On passe au tracé, on ouvre une nouvelle fenêtre graphique pour l'affichage puis on trace le nuage de points dans le plan avec la fonction *plot()* . La première composante est placée sur l'axe horizontal et la seconde sur l'axe vertical.
On impose la même échelle sur les deux axes afin de ne pas déformer la géométrie du nuage avec *axis equal ;* et on affiche une grille pour faciliter la lecture graphique avec *grid on ;*. Et pour finir on conserve la graphique courant afin de pouvoir superposer les ellipses de confiance.

```
18      mx = [0;0];
19      Rx =[1 0;0 1];
20
21      eta_list = [0.9 0.99 0.999];
22      for eta = eta_list
23          trace_ellipse(mx, Rx, eta, 'r');
24
```

Par la suite on initialise les composantes nécessaire à la fonction *trace_ellipse()*. On commence par définir la moyenne théorique du vecteur aléatoire qui est nulle ici car supposé centré. Ensuite on définit la matrice de covariance théorique, égale à l'identité d'après l'énoncé, ce qui signifie que les deux composantes sont indépendantes et de même variance. Et pour finir on définit les

niveaux de confiance pour lesquels on souhaite tracer les ellipses et par la suite on parcourt successivement chaque niveau de confiance pour faire les ellipses.

Remarque : code est disponible intégralement en annexe.

On va maintenant passer à l'explication de la fonction *trace_ellipse* . Pour commencer cette fonction prend en entrée la moyenne du vecteur gaussien (m_x), la matrice de covoriance R_x, le niveau de confiance (*eta)* et le style graphique de l'ellipse *(style)*.

```
1   function trace_ellipse(mx,Rx,eta,style)
```

La première étape est seulement ici pour rendre la fonction plus « robuste », si on ne fournis aucun style, un style par défaut est appliqué et le tracé sera noir.

```
8   if exist('style')==0,
9       style='k';
10  end
```

La fonction commence vraiment à partir de cette étape. On définit un paramètre angulaire $s \in [0 \quad 2\cdot\pi]$. Avec une incrémentation toutes les 0.01. On avance de 0.01 radian à chaque point. S est donc un vecteur 1D contenant tous les angles nécessaires pour décrire un cercle complet.

```
2   s=[0:0.01:2*pi];
```

On est maintenant confronté à une belle équation donc on va séparer les éléments afin de l'expliquer au mieux.

Premièrement on a un vecteur qui représente les points du cercle unité. $c(s)=\begin{pmatrix}\cos(s)\\ \sin(s)\end{pmatrix}$

```
x=mx*ones(size(s))+sqrt(-2*log(1-eta))*chol(Rx,'lower')*[cos(s);sin(s)];
```

Ensuite on a une décomposition de Cholesky : $R_x = L\cdot L^T$ avec L triangulaire inférieure
Cette décomposition encode l'orientation et l'étirement de l'ellipse. Cette étape réalise en faite la transformation suivante : $C \rightarrow L\cdot C$ qui transforme le cercle unité en une ellipse de covariance R_x.

```
x=mx*ones(size(s))+sqrt(-2*log(1-eta))*chol(Rx,'lower')*[cos(s);sin(s)];
```

On a ensuite le facteur *sqrt(-2*log(1-etat))* qui intervient. Il correspond à : $\sqrt{-2\ln(1-\eta)}$
C'est directement en lien avec l'annexe de l'énoncé. Plus η est grand plus l'ellipse est large.

```
x=mx*ones(size(s))+sqrt(-2*log(1-eta))*chol(Rx,'lower')*[cos(s);sin(s)];
```

Cette expression construit donc l'ellipse centrée en 0 et correspond à cette partie de la démonstration de l'annexe.

Par définition, l'ellipse de confiance de niveau η $(0 \le \eta \le 1)$ associée à un vecteur aléatoire gaussien $\underline{x}$ de $\mathbb{R}^2$ de moyenne $\underline{m}_x$ et de matrice de covariance $\underline{\underline{R}}_x$ est la plus petite région D telle que la probabilité que $\underline{x}$ appartienne à D soit égale à η. Cette région est délimitée par une courbe de niveau de la densité de probabilité du vecteur aléatoire $\underline{x}$, c'est-à-dire par une courbe de niveau de forme quadratique :

$$\varepsilon_{\underline{x}} = \left\{ \underline{x} \in \mathbb{R}^2 \text{ tq } (\underline{x}-\underline{m}_x)^T \, \underline{\underline{R}}_x^{-1} (\underline{x}-\underline{m}_x) = a^2 \right\}.$$

La matrice $\underline{\underline{R}}_x$ étant symétrique et définie positive, elle peut être réécrite en utilisant la factorisation de Cholesky. En effet, il existe alors une matrice réelle triangulaire inférieure $\underline{\underline{L}}$ telle que : $\underline{\underline{R}}_x = \underline{\underline{L}}\,\underline{\underline{L}}^T$. On a alors :

$\underline{\underline{R}}_x^{-1} = \left(\underline{\underline{L}}^T\right)^{-1} \underline{\underline{L}}^{-1}$, et par suite :

$$(\underline{x}-\underline{m}_x)^T \, \underline{\underline{R}}_x^{-1} (\underline{x}-\underline{m}_x) = (\underline{x}-\underline{m}_x)^T \left(\underline{\underline{L}}^T\right)^{-1} \underline{\underline{L}}^{-1} (\underline{x}-\underline{m}_x) = \left(\underline{\underline{L}}^{-1}(\underline{x}-\underline{m}_x)\right)^T \underline{\underline{L}}^{-1}(\underline{x}-\underline{m}_x) = \left\| \underline{\underline{L}}^{-1}(\underline{x}-\underline{m}_x) \right\|_2^2.$$

On peut donc écrire :

$$\begin{aligned}
\varepsilon_{\underline{x}} &= \left\{ \underline{x} \in \mathbb{R}^2 \text{ tq } \left\| \underline{\underline{L}}^{-1}(\underline{x}-\underline{m}_x) \right\|_2^2 = a^2 \right\} \\
&= \left\{ \underline{x} \in \mathbb{R}^2 \text{ tq } \left\| \frac{1}{a} \underline{\underline{L}}^{-1}(\underline{x}-\underline{m}_x) \right\|_2^2 = 1 \right\} \\
&= \left\{ \underline{x} \in \mathbb{R}^2 \text{ tq } \frac{1}{a} \underline{\underline{L}}^{-1}(\underline{x}-\underline{m}_x) \in \mathcal{C} \right\} \quad \text{où } \mathcal{C} \text{ désigne le cercle unité} \\
&= \underline{m}_x + a \underline{\underline{L}} \mathcal{C}
\end{aligned}$$

avec $a > 0$.
L'ellipse $\varepsilon_{\underline{x}}$ peut donc être définie par la représentation paramétrique suivante :

$$\underline{x}(s) = \underline{m}_x + a \underline{\underline{L}} \begin{bmatrix} \cos(s) \\ \sin(s) \end{bmatrix} \text{ avec } 0 \le s \le 2\pi .$$

Pour choisir a, on remarque que :

$$\eta = \Pr(\underline{x} \in D) = \Pr\left((\underline{x}-\underline{m}_x)^T \, \underline{\underline{R}}_x^{-1} (\underline{x}-\underline{m}_x) \le a^2 \right) = \Pr\left(\left\| \underline{\underline{L}}^{-1}(\underline{x}-\underline{m}_x) \right\|_2^2 \le a^2 \right) = \Pr\left(z \le a^2 \right)$$

où $z = \left\| \underline{\underline{L}}^{-1}(\underline{x}-\underline{m}_x) \right\|_2^2$ suit la loi du χ^2 à deux degrés de liberté, dont la densité de probabilité est donnée par :

$$f_Z(z) = \begin{cases} \frac{1}{2} \exp\left(-\frac{z}{2}\right) & \text{si } z \ge 0 \\ 0 & \text{sinon.} \end{cases}$$

On a donc $\eta = \Pr(\underline{x} \in D) = \Pr\left(z \le a^2\right) = \int_0^{a^2} \frac{1}{2} \exp\left(-\frac{z}{2}\right) dz = 1 - \exp\left(-\frac{a^2}{2}\right)$, ce qui permet d'en déduire la valeur de a à choisir pour avoir une probabilité d'appartenance à l'ellipse égale à η $\quad a = \sqrt{-2\ln(1-\eta)}$.

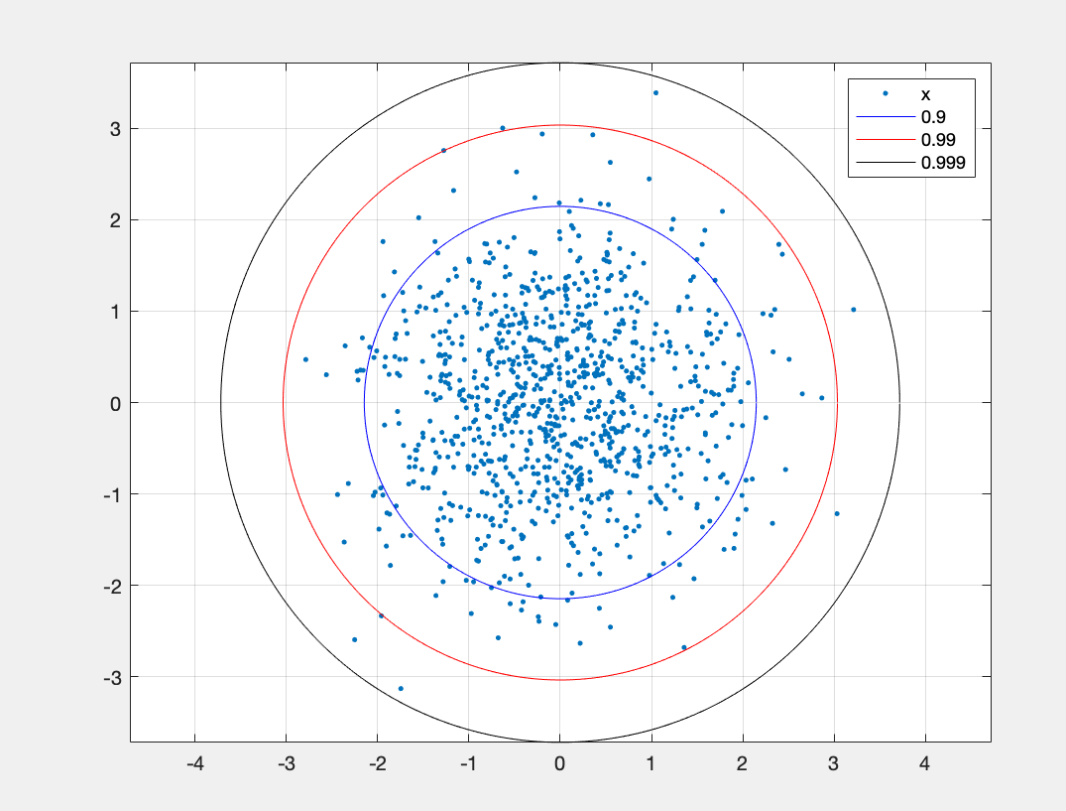
On remarque qu'il nous manque le terme m_x. Si on regarde les autres termes de l'équation on remarque *mx*ones(size(s))* on vient en faite créer un vecteur contenant la moyenne m_x répétée pour chaque point. Cet élément sert à translater l'ellipse pour la centrer sur la moyenne.

```
x=mx*ones(size(s))+sqrt(-2*log(1-eta))*chol(Rx,'lower')*[cos(s);sin(s)];
```

Avec ce dernier terme on retrouve donc exactement la formule exposé dans la démonstration de l'ellipse de confiance.

2.2 Analyse des résultats

Voici le résultat du script présenté dans la sous partie précédente :



On retrouve donc bien un nuage de 1000 points. Ce nuage représente des réalisations d'un vecteur aléatoire gaussien bidimensionnel centré, de covariance identité. La densité des points est maximale autour de l'origine et décroît progressivement lorsqu'on s'en éloigne, ce qui est caractéristiques d'une loi normale. Et l'absence de direction privilégiée confirme que les deux composantes sont indépendantes et de variance identique.

Concernant la forme et la position des ellipses. Il y a trois ellipses noire, rouge et bleu elles correspondent respectivement au niveau de probabilité 0.999, 0.99, 0.9. Elles sont toutes centrées sur l'origine ce qui est cohérent avec la moyenne nulle du vecteur aléatoire. Leur forme est

circulaire, ce qui s'explique par la matrice de covariance égale à l'identité. Les variances sont identiques sur chaque axe et il n'existe aucune corrélation entre les composantes. Les observations coïncident donc à la théorie !

Si on apporte une analyse davantage probabiliste chaque ellipse délimite une région dans laquelle se trouve une proportion donnée des réalisations. Environ 90 % des points sont contenus à l'intérieur de l'ellipse de niveau 0.9, environ 99 % dans celle à 0.99 et quasiment l'ensemble des points dans celle à 0.999 de niveau de confiance.

Puis on peut aussi faire un lien entre le rayon des cercles et la théorie derrière la fonction *trace_ellipse*. En effet cette région de confiance est délimité par l'inégalité suivante :

$$(x-m_x)^T \cdot R_x^{-1} \cdot (x-m_x) \leqslant a^2$$
$$\text{avec } a = \sqrt{(-2 \cdot \ln(1-\eta))}$$

Le facteur *$-2\ln(1-\eta)$* explique l'augmentation progressive de la taille des cercles lorsque η tend vers 1. Le fait que les cercles soient concentriques et de forme circulaire valide l'expression théorique utilisée.

On peut faire une petite remarque concernant la taille du nuage de points. Avec K=1000 on a sert déjà une bonne idée visuelle de la forme gaussienne mais ça reste un échantillon fini. La proportion de points à l'intérieur des ellipses va naturellement fluctuer autour des valeurs théoriques. Du coup selon les tirages on peut très bien en observer 0, 1, 2 ou même 3 hors du niveau de confiance à 0.999, et on peut donc avoir l'impression que ça ne colle pas avec la théorie alors que c'est simplement de la variabilité statistique.

3. Moyenne (2,3) et une matrice de covariance égale à (2 1,1 4)

3.1 Explication du script

Dans cette partie on cherche à générer un vecteur aléatoire gaussien y de moyenne m_y et de matrice de covariance R_Y, à partir d'un vecteur gaussien centré réduit x. Pour cela on a utilisé la décomposition de Cholesky de la matrice de covariance R_y, qui permet d'écrire $R_y = AA^T$, où A est une matrice triangulaire inférieure.

On a choisi de passer par une transformation linéaire $y = A \cdot x + m_y$ qui permet de construire un vecteur aléatoire dont la moyenne est m_y et la covariance est R_y. Cette méthode est bien adaptée car elle repose sur le fait qu'une transformation linéaire d'un vecteur gaussien conserve la nature gaussienne de la variable aléatoire tout en modifiant sa position selon la matrice appliquée.

On a ensuite tracé les ellipses de confiance aux niveaux 0.9-0.99-0.999 et on analysera les résultats après avoir expliquer/présenté le code.

On commence donc par initialiser la moyenne cible m_y ainsi que la covariance cible R_y. Le moyenne m_y a pour objectif de centrer le nuage autour du point (2,3) et la matrice de covariance R_y impose de la dispersion (avec la variance) et de la corrélation (les termes hors de la diagonale) entre y_1 et y_2.

```
34    my = [2 ; 3];
35    Ry = [2 1;1 4];
36
```

On a ensuite le calcul de la décomposition de Cholesky $R_y = A \cdot A^T$ A va servir à transformer un bruit blanc donc une covariance identité en un nuage ayant la covariance R_y.

```
7     A = chol(Ry,'lower');
```

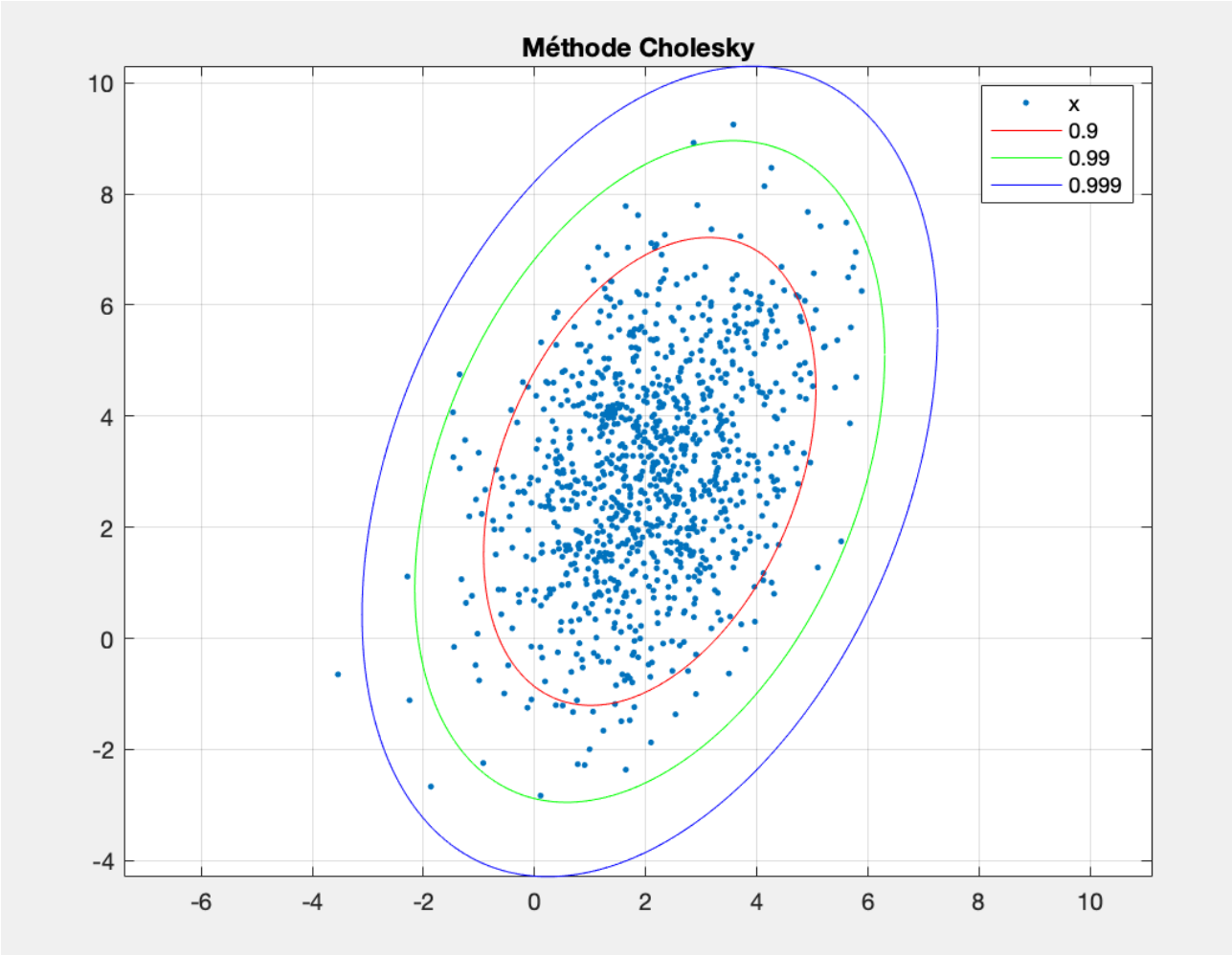
On construit le nuage de points y.

```
38    b=my;
39    y = (A * x) + b;
```

Et on affiche le nuage ainsi que les ellipses de confiance dont on a déjà expliqué le fonctionnement dans la partie précédente.

```
42    figure;
43    plot(y(1,:), y(2,:), '.');
44    axis equal; grid on; hold on;
45    title('Méthode Cholesky');
46    trace_ellipse(mean(y,2), cov(y.'), 0.9,'r');
47    trace_ellipse(mean(y,2), cov(y.'), 0.99,'g');
48    trace_ellipse(mean(y,2), cov(y.'), 0.999,'b');
49    hold off;
50    legend("x","0.9","0.99","0.999");
51
```

3.2 Explication des résultats



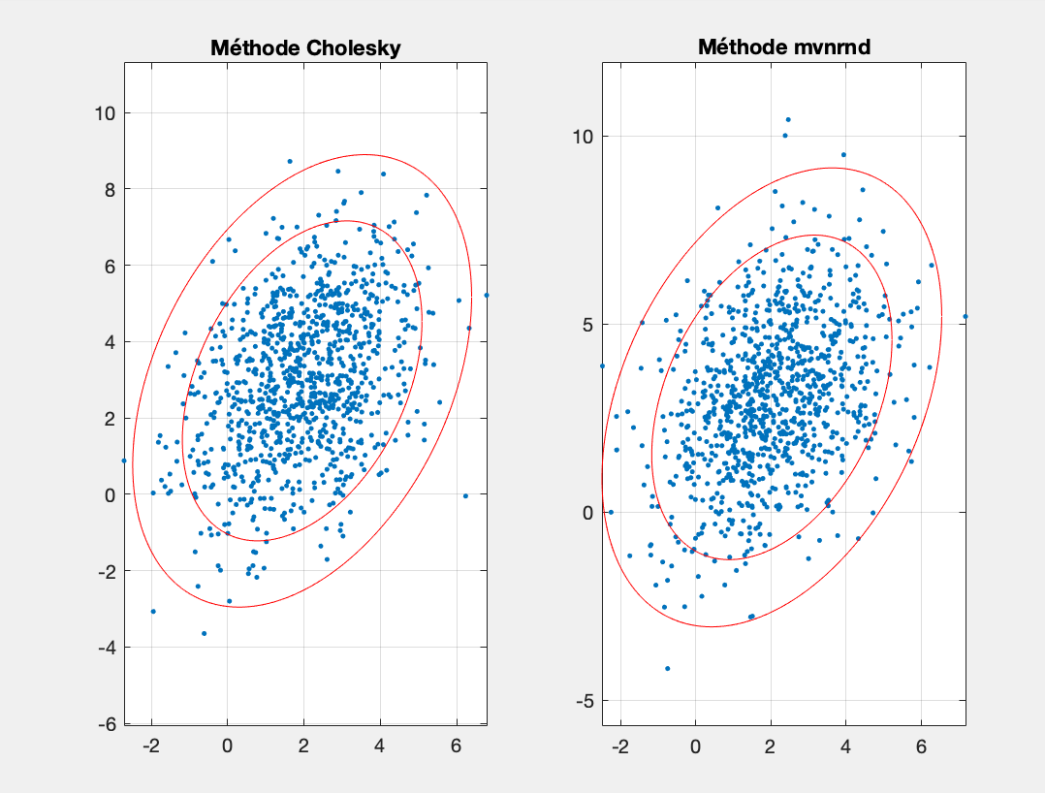
Le nuage est bien centré autour du point (2,3), ce qui confirme que la translation par $b=m_y$ est correctement appliquée. La dispersion du nuage est allongée et inclinée ce qui reflète la structure de la matrice de covariance R_y, en particulier la présence de termes hors diagonale traduisant une corrélation entre les deux composantes.

Les ellipses de confiance tracées aux niveaux 0,9, 0,99 et 0,999 épousent correctement la forme et l'orientation du nuage. Leur centre coïncide avec la moyenne empirique, et leur orientation correspond aux directions propres de la matrice de covariance. À mesure que le niveau de confiance augmente, les ellipses s'élargissent et englobent une proportion croissante des points, ce qui est conforme à la théorie des lois gaussiennes.

Cette figure nous permet donc de valider/vérifier que l'utilisation de la décomposition de Cholesky permet bien de générer un vecteur gaussien de moyenne et de covariance imposées, et que les propriétés statistiques théoriques sont retrouvées dans les résultats.

4. Comparaison avec la méthode *mvnrnd*

On va maintenant valider la méthode de Cholesky utilisée dans la sous partie en la comparant à une fonction MATLAB de référence (*mvnrnd)*, qui génère directement des réalisations d'un vecteur gaussien de moyenne et covariance imposées.

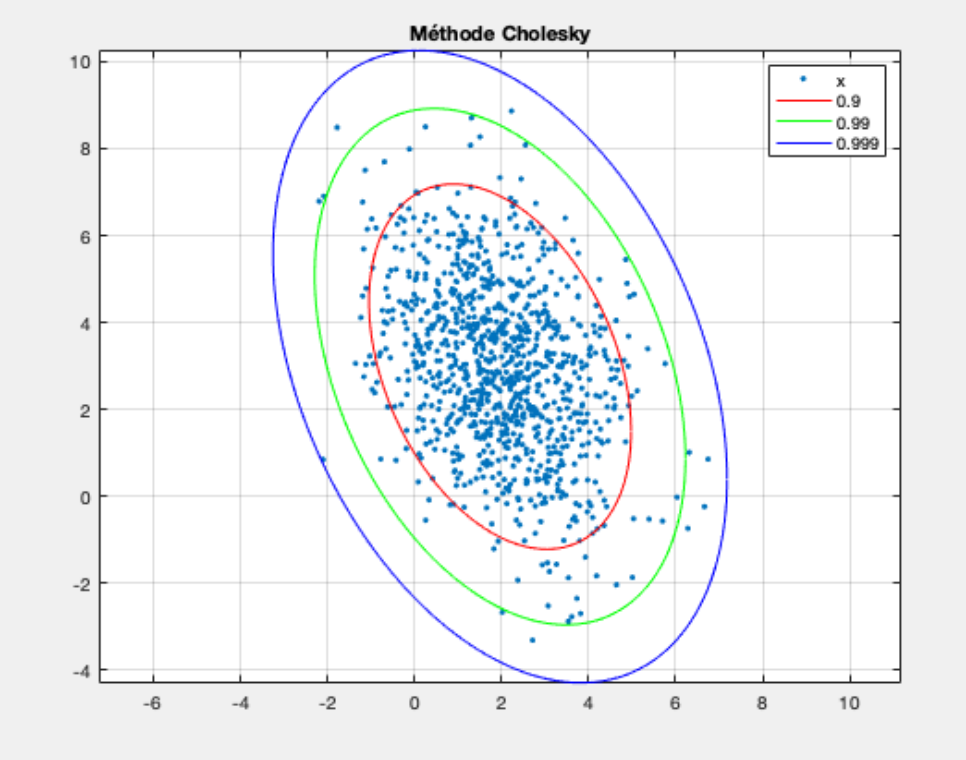


Les nuages générés par la méthode de Cholesky et par *mvnrnd* présentent la même forme, la même orientation et sont centrés autour de la même moyenne. Les ellipses de confiance associées aux deux nuages se superposent quasiment, ce qui montre que les matrices de covariance sont les mêmes. Les légères différences qu'on peut observer proviennent du caractère aléatoire de la simulation et du nombre fini d'échantillons.

Donc cette comparaison montre que la méthode avec Cholesky permet bien de générer un vecteur gaussien de moyenne et de covariance imposées.

5. Influence de la matrice de covariance

Dans cette sous partie on va expliquer l'influence de la matrice de covariance en utilisant le même script que la sous partie 3.



Pour obtenir ce résultat on a pris la moyenne $m_y = (4 \quad 2)$ et la matrice de covariance $Ry = \begin{bmatrix} 2 & -1 \\ -1 & 4 \end{bmatrix}$.

La seule différence que l'on remarque est que l'orientation du nuage de points et des ellipses de confiance est inversée. On remarque donc bien l'influence des termes hors diagonale de la covariance. En faite les valeurs hors diagonales de la matrice de covariance traduisent la corrélation entre les deux composantes du vecteur aléatoire. Leur signe a une interprétation géométrique directe sur l'orientation du nuage de points et des ellipses.

Lorsque les termes hors diagonale sont positifs, cela signifie que les deux composantes ont tendance à évoluer dans le même sens (lorsque l'une augmente l'autre aussi). Graphiquement cette corrélation positive se traduit comme on a pu le voir sur les figures précédentes par une ellipses inclinée suivant une direction de pente positive. Et on remarque l'inverse (voir ci dessus) quand les composantes hors diagonale sont négatives. Et lorsque elles sont nulles alors on se retrouve avec les mêmes matrices qu'on a observé dans la sous partie 2, donc les ellipses de

confiance sont alors alignée avec les axes du repère car les composantes du vecteur ne sont pas corrélées.

Ainsi pour conclure, le signe des termes hors diagonale de la matrice de covariance ne modifie pas la taille de l'ellipse mais uniquement son orientation, en révélant la nature de la dépendance entre les composantes du vecteur.

II – Filtrage de Kalman : étude pas à pas sur un premier exemple

Dans cette partie deux nous avons commencer par écrire une fonction qui met en œuvre une seule itération du filtre de Kalman. Elle a comme syntaxe d'appel :

[G,x_post,P_post,x_prio_suiv,P_prio_suiv]=iter_kalman(x_prio,P_prio,y,F,C,Ru,Rw)

L'objectif final de la fonction *iter_kalman* est d'estimer au mieux l'état réel d'un système à partir de mesures bruitées, tout en mesurant l'incertitude associée à cette estimation, et de préparer cette estimation pour un instant suivant.

Cette fonction que vous pouvez retrouver en annexe prends comme entrées :

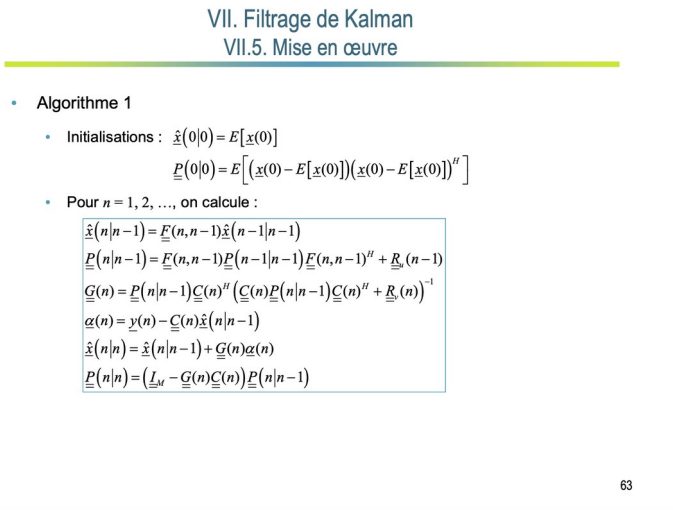
- *x_prio* qui est l'estimation à priori. C'est ce que l'on pense de l'état avant de regarder la mesure.
- *P_prio* qui est la covariance à priori. C'est à quel point on doute de *x_prio.* Elle définit la taille de l'ellipse de confiance avant la mesure.lorsqu'on a une grande valeur l'estimation est peu fiable et lorsqu'elle est petite l'estimation est assez sûre.
- *y* qui est une mesure y(n). C'est ce que le capteur observe réellement. Par exemple les données d'un capteur de vitesse ou radar.
- *F* qui est la matrice d'évolution de l'état. C'est comment l'état évolue dans le temps selon le modèle.
- *C* qui est la matrice de mesure. C'est le lien entre ce que l'on mesure et l'état. Elle indique quelles composantes de l'état sont observées.
- *Ru* qui est la covariance du bruit de modèle. Donc c'est à quel point le modèle est imparfait. Par exemple des perturbations non modélisées.
- *Rw* qui est la covariance du bruit de mesure. Ici c'est à quel point le capteur est bruité. Cela représente la qualité des mesures, plus *Rw* est grand, plus la mesure est incertaine.

Et fournie en sortie :

- *G* qui est le gain de Kalman. C'est le coefficient qui arbitre entre le modèle et la mesure. Si *G* est grand on fait davantage confiance à la mesure. À l'inverse si *G* est petit on va avoir tendance à faire davantage confiance au modèle.
- *x_post* qui est l'estimation à posteriori. C'est la meilleure estimation de l'état après avoir vu la mesure. C'est l'estimation finale à l'instant n.
- *P_post* qui est la covariance à posteriori. C'est l'incertitude restante après la correction. Plus l'elispse est petite, meilleure est la confiance.
- *x_prio_suiv* qui est l'estimation à priori à n+1. C'est la projection de l'estimation vers l'instant suivant. C'est le point de départ de la prochaine itération.
- *P_prio_suiv* qui est la covariance à priori à n+1. C'est l'incertitude mais projeté à l'instant n+1.

```
1  function [G, x_post, P_post, x_prio_suiv, P_prio_suiv] = iter_kalman(x_prio, P_prio, y, F, C, Ru, Rw)
2
```

Maintenant que l'on connaît les entrées/sorties on peut expliquer le code mis en place ligne par ligne mais surtout le sens physique derrières ces lignes. Déjà pour commencer la fonction fait **une** itération complète du filtre de Kalman à l'instant n. On commence par une correction, on combine l'estimation a priori et la mesure puis on fait la prédiction où l'on projette l'état corrigé à l'instant n+1. On a suivi l'algorithme proposé dans le cours de F.NADAL page 63 (voir ci dessous) [1]:



VII. Filtrage de Kalman
VII.5. Mise en œuvre

- Algorithme 1
  - Initialisations : $\underline{\hat{x}}(0|0) = E[\underline{x}(0)]$

    $\underline{\underline{P}}(0|0) = E\left[\left(\underline{x}(0) - E[\underline{x}(0)]\right)\left(\underline{x}(0) - E[\underline{x}(0)]\right)^H\right]$
  - Pour $n = 1, 2, \ldots$, on calcule :

$$\underline{\hat{x}}(n|n-1) = \underline{\underline{F}}(n,n-1)\underline{\hat{x}}(n-1|n-1)$$
$$\underline{\underline{P}}(n|n-1) = \underline{\underline{F}}(n,n-1)\underline{\underline{P}}(n-1|n-1)\underline{\underline{F}}(n,n-1)^H + \underline{\underline{R}}_w(n-1)$$
$$\underline{\underline{G}}(n) = \underline{\underline{P}}(n|n-1)\underline{\underline{C}}(n)^H\left(\underline{\underline{C}}(n)\underline{\underline{P}}(n|n-1)\underline{\underline{C}}(n)^H + \underline{\underline{R}}_v(n)\right)^{-1}$$
$$\underline{\alpha}(n) = \underline{y}(n) - \underline{\underline{C}}(n)\underline{\hat{x}}(n|n-1)$$
$$\underline{\hat{x}}(n|n) = \underline{\hat{x}}(n|n-1) + \underline{\underline{G}}(n)\underline{\alpha}(n)$$
$$\underline{\underline{P}}(n|n) = \left(\underline{\underline{I}}_M - \underline{\underline{G}}(n)\underline{\underline{C}}(n)\right)\underline{\underline{P}}(n|n-1)$$

63

On a commencé par calculer le gain de Kalman G :

$$Gn = P_{n|n-1} \cdot C^T \cdot \left(C \cdot P_{n|n-1} \cdot C^T + R_w\right)^{-1}$$

Dans le sens physique $C \cdot P_{n|n-1} \cdot C^T$ est l'incertitude sur la mesure due à l'incertitude de l'état. R_w est le bruit du capteur. Et l'ensemble $C \cdot P_{n|n-1} \cdot C^T + R_w$ mesure à quel point la mesure est fiable.

Si R_w est petit la mesure est fiable et du coup G est grand. Si *P_prio* est grand on doute du modèle et G est grand aussi. Et si la mesure est bruitée on a G qui diminue. Donc cela confirme ce que l'on a dit dans la présentation des paramètres, « *C'est le coefficient qui arbitre entre le modèle et la mesure. Si G est grand on fait davantage confiance à la mesure. À l'inverse si G est petit on va avoir tendance à faire davantage confiance au modèle.* »

```
3   % G = P_prio C^T (C * P_prio * C.' + Rw)^{-1}
4
5   G = (P_prio * C.') / (C * P_prio * C.' + Rw);
6
```

Après avoir calculé le gain de Kalman on a calculé l'innovation alpha qui s'exprime suivant :

$$\alpha_n = y_n - C \cdot \hat{x}_{\cdot n|n-1}$$

Au sens physique on a que $C \cdot x_{prio}$ est ce que le modèle prévoit comme mesure et y ce que le capteur a réellement mesuré. Si l'innovation est proche de zéro, la mesure ne surprend pas le modèle.

```
 7    % alpha = y - C x_prio
 8
 9    innov = y - C * x_prio;
10
```

On a ensuite calculé l'estimation a posteriori qui est donnée suivant :

$$\hat{x_{n|n}} = \hat{x_{n|n-1}} + G_n \cdot \alpha_n$$

Dans sens physique on part de l'estimation a priori et on la corrige. Une correction faible si la mesure est peu fiable ou une correction forte si la mesure est crédible.

```
11    % x_post = x_prio + G * innov
12
13    x_post = x_prio + G * innov;
14
```

On a par la suite calculé la covariance a posteriori comme dans l'algorithme proposé dans le cours :

$$P_{n|n} = (I - G_n \cdot C) P_{n|n-1}$$

Dans le sens physique après avoir utilisé la mesure, on est plus sûr de l'état donc l'incertitude diminue. Les ellipses de confiance se resserrent.

```
15    % P_post = (I - G C) P_prio page 62
16
17    I = eye(M);
18    P_post = (I - G * C) * P_prio;
19
```

On a ensuite calculé la prédiction de l'état avec :

$$\hat{x}_{n+1|n} = F \cdot \hat{x}_{n|n}$$

Dans le sens physique on fait évoluer l'état corrigé selon le modèle dynamique.

```
20    % x_prio_suiv = F x_post
21
22    x_prio_suiv = F * x_post;
23
```

Et pour finir on a calculé la prédiction de la covariance :

$$P_{n+1|n} = F \cdot P_{n|n} \cdot F^T + R_u$$

```
24    % P_prio_suiv = F P_post F^T + Ru
25
26    P_prio_suiv = F * P_post * F.' + Ru;
27
28    end
```

On va maintenant mettre en pratique l'utilisation de cette fonction sur un petit exemple. On réalisera dans un premier temps le modèle d'état théorique puis un petit script utilisant la fonction afin d'estimer deux paramètres constants x_1 et x_2.

On commence par définir l'état :

$$x(n) = \begin{pmatrix} x_1(n) \\ x_2(n) \end{pmatrix}$$

On dit que ces paramètres sont constants donc :

$$x(n+1) = x(n) + c(n)$$

On néglige c(n) car on considère qu'il est quasiment égale à zéro.

$$x(n+1) = x(n)$$

On a donc la matrice d'évolution suivante :

$$F = I_2$$

L'énoncé donne :

$$y(n) = \alpha(n) \cdot x_1 + \beta(n) \cdot x_2 + w(n)$$

Donc on a :

$$y(n) = C(n) \cdot x(n) + w(n) \text{ avec } C(n) = (\alpha(n)\ , \beta(n)\ )$$

On a alors *y(n)* scalaire et *C(n)* une matrice 1x2.
D'après l'énoncé le bruit est de variance 9 donc :

$$R_w = [9]$$

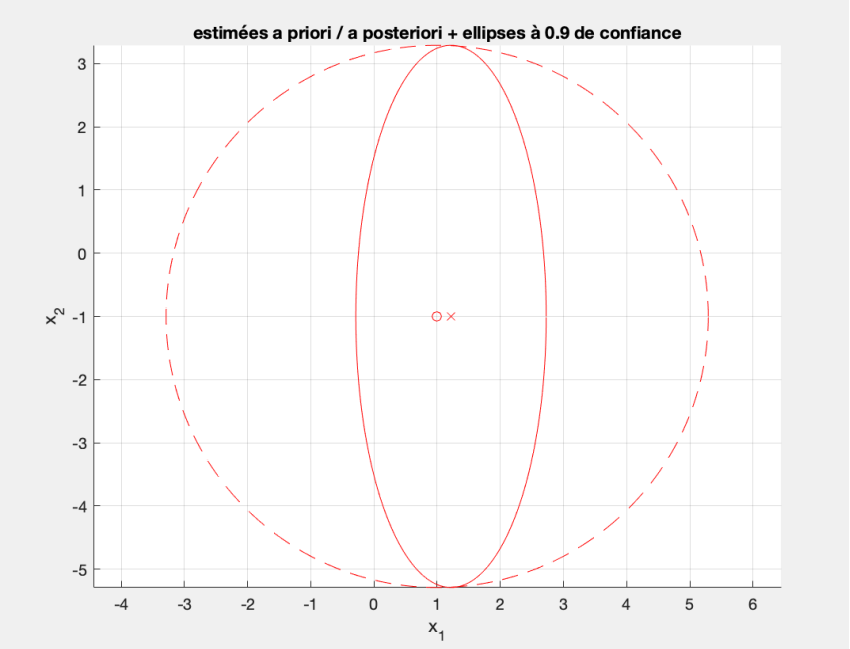
Pour le bruit de modèle on a deux solutions. Soit on considère des paramètres strictement constants (*Ru=0)* Soit une petite valeur (*Ru=facteur.*I_2*).*

L'énoncé dit : «*A priori x1≈1, x2≈−1 avec variance 4*»

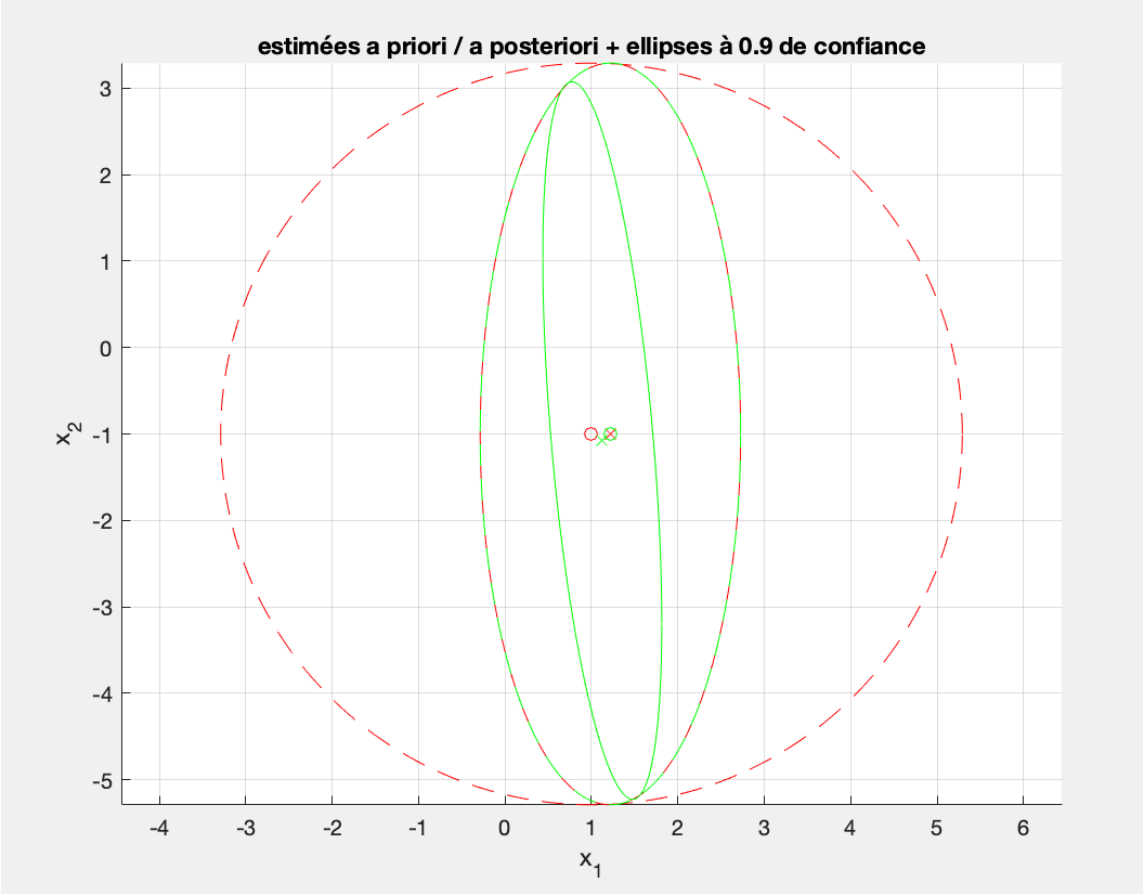
Donc pour l'initialisation on a :

$$\hat{x_{1|0}} = \begin{pmatrix} 1 \\ -1 \end{pmatrix},\ P_{1|0} = 4 \cdot I_2$$

Après avoir réalisé ce modèle d'état on a testé notre fonction *iter_kalman* sur cinq itérations !



Sur ce graphique, on observe à la fois l'estimation a priori et l'estimation a posteriori des paramètres, ainsi que leurs ellipses de confiance au niveau 0,9. Le point a posteriori est très proche du point a priori, ce qui montre que la mesure utilisée à cet instant est cohérente avec la prédiction issue du modèle et n'entraîne donc qu'une correction limitée de l'estimation. En revanche, la différence est beaucoup plus marquée au niveau des ellipses : l'ellipse a posteriori est nettement plus resserrée que l'ellipse a priori, signe que la mesure a apporté une information significative et a permis de réduire l'incertitude sur les paramètres estimés. On remarque également que l'ellipse reste allongée principalement selon l'axe x_2 ce qui indique que l'incertitude sur ce paramètre demeure plus importante que sur x_1. Cela traduit une observabilité inégale des deux paramètres pour cette mesure, liée aux coefficients du modèle d'observation.



Même raisonnement que pour la figure précédente mais ici pourquoi la position a priori est la même que la position a posteriori précédente ? En regardant les équations du filtrage de Kalman on remarque :

$$\hat{x}(n+1|n)=F(n+1,n)\cdot\hat{x}(n|n)$$

On rappelle que F est une matrice identité de taille (2x2) donc on a :

$$\hat{x}(n+1|n)=\hat{x}(n|n)$$

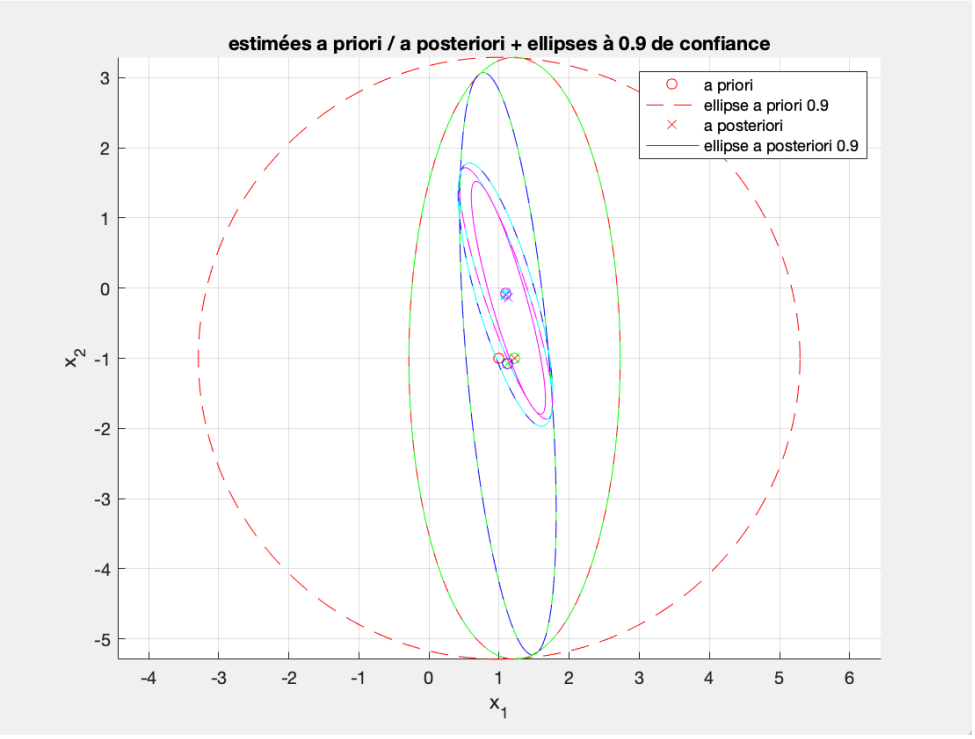
Cela justifie le fait que le point a posteriori de la mesure n soit égale au point a priori de la mesure n+1.

On peut remarquer le même mécanisme pour l'ellipse de confiance a posteriori et a priori parce que on a dans les équations initiales :

$$P(n+1|n)=F(n+1,n)\cdot P(n|n)\cdot F(n+1,n)^{H}+R_{u}(n)$$

F est toujours une matrice identité de taille (2x2) et on rappelle que R_u est une matrice nulle. Ainsi on se retrouve avec l'égalité suivante qui explique théoriquement pourquoi ces deux ellipses sont identiques :

$$P(n+1|n)=P(n|n)$$



Code couleur : [Rouge : première itération , Vert : deuxième itération , Bleu : troisième itération, Cyan : quatrième itération, Rose : cinquième itération]

Sur cette figure finale, on visualise l'ensemble de l'évolution des estimations du filtre de Kalman dans le plan (x1,x2). Le grand cercle en pointillés correspond à l'ellipse de confiance a priori initiale, très large, ce qui traduit une forte incertitude au départ sur les paramètres. À mesure que les observations sont intégrées, les estimations a posteriori successives se déplacent progressivement et se regroupent autour d'une zone restreinte, montrant la convergence du filtre vers une valeur stable des paramètres. En parallèle, les ellipses a posteriori deviennent de plus en

plus petites et plus allongées, ce qui indique une diminution progressive de l'incertitude. Leur orientation révèle une corrélation résiduelle entre x1et x2, liée à la nature du modèle de mesure On remarque aussi que les ellipses sont imbriquées les une dans les autres. Cela s'explique théoriquement car les coefficients x_1 et x_2 sont des coefficients constants donc l'algorithme converge donc vers ces coordonnées. Cela explique donc pourquoi les ellipses de confiance ne se croisent pas.

$$x(n)=\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \text{ avec } x_1 \text{ et } x_2 \text{ des coefficients constants}$$

Pour expliquer cette inclinaison on peut prendre un exemple :

$$P=\begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix} \text{ covariance 2D}$$

Si on a $\sigma_{12}=0$ alors P est diagonale, les vecteurs propres sont alignés avec les axes et donc l'ellipse est non inclinée. Au contraire si $\sigma_{12}\neq 0$ alors les variables sont corrélées, et les vecteurs propres ne sont plus alignés avec les axes donc l'ellipse est inclinée.

Sur nos figures les ellipses sont donc inclinées car $P_{12}\neq 0$ cela signifie que l'erreur sur x_1 et celle sur x_2 sont corrélées, mathématiquement, la mesure $y(n)=\alpha(n)\cdot x_1+\beta(n)\cdot x_2+w(n)$ contraint une combinaison linéaire de x_1 et x_2 pas chaque paramètre séparément. Cette contrainte crée alors une direction privilégiée d'incertitude, qui apparaît comme l'axe principal de l'ellipse.

Concernant l'épaisseur des ellipses de confiance elle est déterminé par la plus petite valeur propre de la matrice de covariance. Sa diminution au fil des 5 itérations montre la réduction de l'incertitude , signe que le filtre de Kalman affine son estimation.

III – Filtrage de Kalman : application à l'estimation de la position et de la vitesse d'un mobile

Le première objectif de cette troisième partie est donc d'écrire un fonction mettant en œuvre le filtre de Kalman récursivement sur une séquence de mesure en alternant correction et prédiction, tout en stockant l'évolution des estimations, des covariances et du gain à chaque instant. Pour cela on s'est appuyé sur l'algorithme mis en place précédement dans la fonction *inter_kalman* présenté dans la partie II.

```
function [G_list, x_post_list, P_post_list, x_prio_list, P_prio_list] = filtre_kalman(x_prio_1, P_prio_1, y_list, F,
    C, Ru, Rw)
```

Cette première partie définit la fonction *filtre_kalman*. Cette fonction comme expliqué auparavant applique le filtre de Kalman sur toute une séquence de mesures. Elle prend en entrée :

- le vecteur `x_prio_1` correspondant à $\underline{\hat{x}}(1|0)$,
- la matrice `P_prio_1` correspondant à $\underline{\underline{P}}(1|0)$,
- la matrice `y_list` contenant la suite des vecteurs d'observation (autrement dit, la $n^{\text{ième}}$ colonne de cette matrice contient le vecteur d'observation à l'instant n, noté $\underline{y}(n)$; si T désigne le nombre total d'instants, la matrice `y_list` sera donc de taille $N \times T$),
- les matrices `F`, `C`, `Ru` et `Rw` correspondant respectivement aux matrices $\underline{\underline{F}}$, $\underline{\underline{C}}$, $\underline{\underline{R}}_u$ et $\underline{\underline{R}}_w$.

Et elle fournie en sortie :

- la matrice `x_prio_list` contenant la suite des estimées a priori (autrement dit, la $n^{\text{ième}}$ colonne de cette matrice devra contenir l'estimée a priori à l'instant n, notée $\underline{\hat{x}}(n|n-1)$; la matrice `x_prio_list` sera donc de taille $M \times (T+1)$),
- le tableau tridimensionnel `P_prio_list` contenant la suite des matrices de covariance de l'erreur a priori (autrement dit, `P_prio_list(:,:,n)` contiendra la matrice $\underline{\underline{P}}(n|n-1)$; le tableau `P_prio_list` sera donc de taille $M \times M \times (T+1)$),
- le tableau tridimensionnel `G_list` contenant la suite des gains de Kalman (autrement dit, `G_list(:,:,n)` contiendra la matrice $\underline{\underline{G}}(n)$; le tableau `G_list` sera donc de taille $M \times N \times T$),
- la matrice `x_post_list` contenant la suite des estimées a posteriori (autrement dit, la $n^{\text{ième}}$ colonne de cette matrice devra contenir l'estimée a posteriori à l'instant n, notée $\underline{\hat{x}}(n|n)$; la matrice `x_post_list` sera donc de taille $M \times T$),
- le tableau tridimensionnel `P_post_list` contenant la suite des matrices de covariance de l'erreur a posteriori (autrement dit, `P_post_list(:,:,n)` contiendra la matrice $\underline{\underline{P}}(n|n)$; le tableau `P_post_list` sera donc de taille $M \times M \times T$).

```
4       [N, T] = length(y_list);
5       M = length(x_prio_1);
```

Ces lignes servent à récupérer les dimensions du problèmes. T correspond au nombre d'itérations (le nombre de mesures), N est la dimension d'une mesure et M est la dimension de l'état. On cherche ici à connaître la taille de l'état et des observations pour allouer les tableaux par la suite.

```
7       x_prio_list = zeros(M, T+1);
8       P_prio_list= zeros(M, M, T+1);
9       G_list= zeros(M, N, T);
10      x_post_list= zeros(M, T);
11      P_post_list= zeros(M, M, T);
```

On fait ici une pré-allocation mémoire des résultats. On a *x_prio_list* et *P_prio_list* qui stockent les estimations a priori. Il y en a T+1 pour stocker les valeurs d'initialisations. On a après *x_post_list* et *P_post_list* stockent les estimations a posteriori et pour finir *G_list* qui stocke le gain de Kalman à chaque itération. Cela va nous permettre d'avoir un bel historique pour l'analyse et les tracés.

```
13      x_prio_list(:,1)= x_prio_1;
14      P_prio_list(:,:,1) = P_prio_1;
```

Ici on vient stocker les conditions initiales $x_{1|0}$ et $P_{1|0}$ respectivement l'estimation a priori avant toute mesure et l'incertitude associée. C'est le point de départ du filtre.

```
16      x_prio= x_prio_1;
17      P_prio= P_prio_1;
```

On initialise proprement les variables sur lesquels on va travailler. *x_prio* et *P_prio* contiennent l'estimation a priori courante et elles seront mises à jour à chaque itération.

```
18
19      for n = 1:T
```

On lance simplement une boucle temporelle. Le filtre va traiter les mesures une par une de n=1 à T.

```
20          y = y_list(:,n);
21
22          [G, x_post, P_post, x_prio_suiv, P_prio_suiv] =iter_kalman(x_prio, P_prio, y, F, C, Ru, Rw);
23
24          G_list(:,:,n) = G;
25          x_post_list(:,n) = x_post;
26          P_post_list(:,:,n)= P_post;
```

On extrait la mesure à l'instant n avec la première ligne. Ensuite on appelle la fonction présenté partie II et on stock les valeurs retours intéressantes dans les listes crées quelques lignes au dessus

```
28          x_prio_list(:,n+1)= x_prio_suiv;
29          P_prio_list(:,:,n+1)= P_prio_suiv;
30
```

On stocke les résultats a priori pour l'instant suivant n+1.

```
31          x_prio = x_prio_suiv;
32          P_prio = P_prio_suiv;
33
34      end
35  end
```

Puis on finit par mettre à jour les variables courantes. Ce qui était la prédiction devient le nouvel o priori comme ça le filtre est prêt pour la mesure suivante !

Maintenant que l'on a rédigé cette fonction on peut donner le modèle d'état associé au problème.

On cherche à estimer la position et la vitesse d'un mobile se déplaçant sur une ligne droite, avec une période d'échantillonnage T_e=1s.

L'état du système à l'instant n est donc :

$$x(n)=\begin{pmatrix} x_n \\ v_n \end{pmatrix}$$

Le mouvement est modélisé par :

$$x(n+1)=\begin{pmatrix} x_{n+1}=x_n+v_n+u_{1n} \\ v_{n+1}=v_n+u_{2n} \end{pmatrix}$$

Sous forme matricielle on obtient :

$$x(n+1)=F\cdot x(n)+u(n)$$

On en déduit donc :

$$F=\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \text{ et } u(n)=\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

On déduit ça car :

$$x(n+1)=F\cdot x(n)+u(n)=\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\cdot\begin{bmatrix} x_n \\ v_n \end{bmatrix}+\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}=\begin{bmatrix} x_n+v_n+u_1 \\ v_n+u_2 \end{bmatrix}$$

Les variances sont données par l'énoncé on en déduit donc la matrice de covariance R_u :

$$R_u=\begin{bmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_u^2 \end{bmatrix}$$

On nous donne aussi la variance du bruit donc on en déduit :

$$R_w=[\sigma_w^2]$$

On souhaite observer la position y(n) dans un premier temps :

$$y(n)=C(n)\cdot x(n)+w(n)$$

On en déduit C(n) :

$$C(n)=\begin{pmatrix} 1 & 0 \end{pmatrix} \text{ et w(n) est un scalaire au même titre que y}$$