Introduction à la robotique mobile
Cahier d'exercices version 0.52

Philippe Giguère

19 septembre 2017

1 Introduction

Ce cahier d'exercice a pour but de vous aider à vous familiariser avec certains concepts très utiles pour le cours. Les réponses aux questions sont dans un deuxième document. Notez que je n'offre pas de support pour Python (incluant les problèmes d'installations). Ainsi, pour certaines des questions, je ne pourrai pas vous fournir le code accessoire. Notez que j'accepte quand même Python pour vos travaux, mais à vos risques et périls.

2 Révision algèbre linéaire

2.1 Équations linéaire

a) Traduisez ce système d'équations en un format matriciel

$$\begin{aligned} y_1 &= 2x_1 - 5x_2 + 4x_3 \\ y_2 &= 3x_1 + 9x_3 \\ y_3 &= -x_2 + 17x_3 \end{aligned}$$

Notez que $\vec{x}$ et $\vec{y}$ sont des vecteurs colonnes. L'indice i dans y_i donne l'entrée dans le vecteur.

b) Quelles sont les commandes matlab ou Python [1] pour calculer la valeur $\vec{y}$, si $\vec{x} = [3\ 4\ 5]^T$?, avec la matrice système définie précédement ?

2.2 Calcul sur vecteurs

Calculez l'angle entre les vecteurs $\vec{c}$ et $\vec{d}$:

$$\vec{c} = \begin{bmatrix} 4 \\ 2 \\ -3 \end{bmatrix} \text{ et } \vec{d} = \begin{bmatrix} -1 \\ 7 \\ 3 \end{bmatrix}$$

2.3 Matrices orthogonales

Soit une matrice $\mathbf{Q}$ orthogonale [2]. Quelle est son inverse $\mathbf{Q}^{-1}$? (note : cette propriété sera très utile pour les matrices de rotation, qui sont orthogonales).

1. Pour fin de simplicité, nous assumons toujours la libraire Numpy dans les réponses
2. Elle est parfois aussi nommée orthonormale, mais c'est un abus de langage car par définition une matrice orhtogonale contient des vecteurs orthgonaux et unitaires.

3 Produit de deux fonctions Gaussiennes

Ces questions (parfois très élémentaires !) ont pour pour but de m'assurer que vous êtes tous familier avec les fonctions Gaussiennes. Comme nous allons les utiliser très souvent, il est donc bon de bien les connaître.

Soient les deux fonctions Gaussiennes suivantes :

$$f(x) = \frac{1}{\sigma_f\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu_f}{\sigma_f})^2} \qquad \text{et} \qquad g(x) = \frac{1}{\sigma_g\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu_g}{\sigma_g})^2} \tag{1}$$

avec les paramètres suivants : $\sigma_f = 1.25$, $\mu_f = 3.6$, $\sigma_g = 0.6$, $\mu_g = 6.7$.

3.1

Quel est la moyenne et l'écart-type de $f(x)$? Pas besoin de faire la démonstration : la réponse seule suffit.

3.2

Il est souvent bénéfique de visualiser les problèmes. Commencez donc par tracer ces deux fonctions Gaussiennes dans matlab ou Python. Pour ce faire, évaluez les à l'aide du vecteur x suivant `x = 0:0.01:10;` . Pour faire un tracé superposé de f et g calculés avec x dans une figure matlab, faites les commandes suivantes :

```
clf;
plot(x,f,'r');
hold on;
plot(x,g,'b');
hold off;
```

Pour Python, utilisez `matplotlib`.

3.3

Dans une figure séparée, tracez le produit de ces fonctions, i.e. $h(x) = f(x)g(x)$. Toujours avec l'aide d'un tracé, observez que $h(x)$ est une gaussienne, à un facteur de normalisation près, possédant les paramètres suivants :

$$\mu_h = \frac{\mu_f\sigma_g^2 + \mu_g\sigma_f^2}{\sigma_f^2 + \sigma_g^2} \qquad \text{et} \qquad \sigma_h^2 = \frac{\sigma_f^2\sigma_g^2}{\sigma_f^2 + \sigma_g^2} \tag{2}$$

Vous pouvez faire ce constat en traçant une troisième courbe représentant $h(x)$ sur votre graphique.

3.4

Soient a, b, c et d des constantes. Lorsqu'on prend une fonction quadratique quelconque

$$q(x) = ax^2 + bx + c \tag{3}$$

et qu'on applique une fonction exponentielle négative à cette quadratique

$$p(x) = de^{-q(x)}, \tag{4}$$

nous avons que la fonction p(x) est une fonction Gaussienne. En quelques lignes, démontrez que le produit des équation $f(x)g(x)$ engendre bel et bien une équation similaire à l'équation 4, où l'exposant est une quadratique. Ainsi, cela démontre que ce produit $f(x)g(x)$ résulte en une gaussienne aussi. Pas besoin de démontrer la valeur d.

4 Distribution Gaussienne multivariée dans matlab

Cette question vous permettra de vous familiariser avec les distributions Gaussiennes (ou normales) multivariées.

4.1

Génerez `n=1000` échantillons aléatoirement avec la fonction `numpy.random.multivariate_normal` dans Python ou `mvnrnd` dans matlab ou, pour une distribution gaussienne multivariée centrée à (0,0) et avec la covariance suivante :

$$\mathbf{S} = \begin{bmatrix} 7 & 3 \\ 3 & 2 \end{bmatrix} \tag{5}$$

Vou pouvez entrer cette matrice dans matlab avec la commande `S = [7 3; 3 2];` ou `S = np.matrix('7 3; 3 2')` dans Pyton. Tracez cette distribution à l'écran avec les commandes suivantes dans matlab :

```
plot(pts(:,1),pts(:,2),'.'); % tracer avec des points
axis equal; % pour avoir les memes echelles en x et y
hold on; % pour pouvoir ajouter (plus tard) d'autres elements
```

et dans Python

```
import matplotlib.pyplot as plt
plt.plot(pts[:,0],pts[:,1], 'b.')
plt.show()
```

où `pts` contient les points générés par votre code.

4.2

Estimez les paramètres de covariance de cette distribution, à partir des points `pts` avec la commande `Sest = cov(pts)` ou `np.cov(np.transpose(pts))` dans python. Que constatez-vous comme différence entre $\mathbf{S}$ et `Sest` ? Comment est-ce que cette différence évolue lorsque vous changez le nombre `n` de points générés ?

4.3

Superposez les vecteurs propres de `Sest` (commande `eig()` ou `numpy.linalg.eig()`) sur l'image produite à la Section 4.1, en les représentant par des lignes rouges avec la commande matlab

```
line(?,?,'LineWidth',2,'Color','r');
```

et en Python

```
line(?,?,'LineWidth',2,'Color','r');
```

où les ? sont à remplacer par vos valeurs. Assurez-vous que la longueur de ces vecteurs représentent bien la forme de la distribution, à partir de l'information extraite par `eig()`. Vous devriez obtenir quelque chose de semblable à la Figure 1 a). Notez que la direction importe peu ; c'est plutôt l'orientation qui nous intéresse ici.

4.4

Utilisez la fonction `ellipse.m` ou `plotEllipse.py` sur le site du cours pour tracer une ellipse rouge qui représente la covariance de cette distribution, à partir de l'information contenue dans les vecteurs propres et valeurs propres de `Sest`. Au besoin, faites `help ellipse` dans matlab pour connaitre les arguments passés à cette fonction ou inspectez le code `plotEllipse.py`. Vous devriez obtenir une image similaire à celle montrée à la Figure 1 b). La commande devrait être similaire à

```
h = ellipse(?,?,?,0,0,'r',200);
set(h,'LineWidth',2);
```

où les ? sont à remplacer par vos valeurs.

(Ceci nous sera utile plus tard dans le cours quand nous allons vouloir représenter des distributions gaussiennes à partir d'un filtre à particule.)

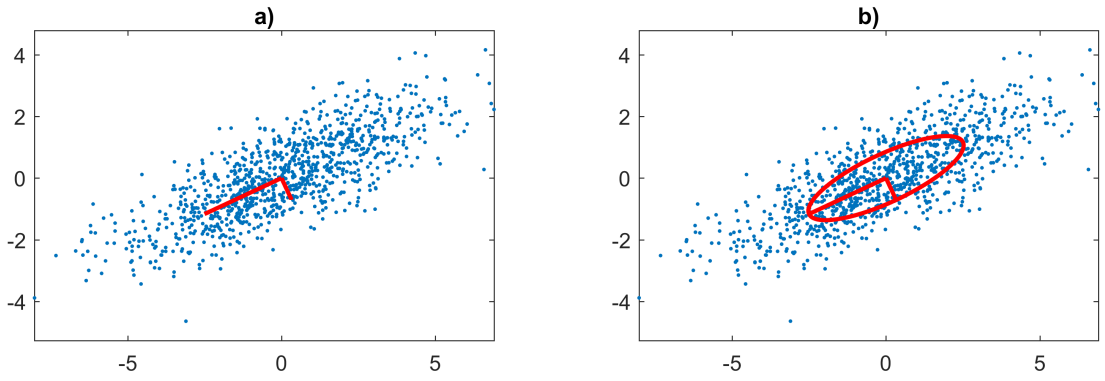

FIGURE 1 – Exemple de réponse pour le problème sur les distribution gaussiennes multivariées (Sec. 4).

5 Révision trigonométrie

Exprimez la hauteur h du triangle à la Fig. 2 en fonction des paramètres a, θ et ϕ.

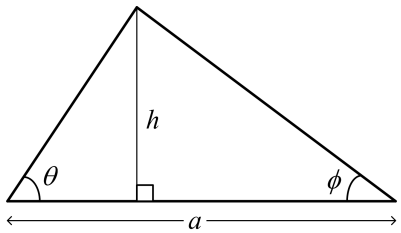

FIGURE 2 – Je suis un triangle.

6 Révision calcul différentiel et intégral

Les systèmes physiques sont souvent décrits à l'aide d'équations différentielles et intégrales. Mais n'ayez pas peur ! Dans le cadre du cours, nous allons utiliser matlab pour faire ces intégrales, de façon numérique. Soit un signal x échantillonné à des instants discrets à intervalle de Δt. Ainsi, le signal sera un vecteur

$$x = [x_0\ x_1\ x_2\ ...\ x_n]$$

Pour calculer sa dérivée dans le temps, vous pouvez utiliser l'équation suivante

$$\dot{x(t)} = \frac{d}{dt}x(t) = \frac{x_t - x_{t-1}}{\Delta t} \tag{6}$$

(Note : L'utilisation du point ˙ au dessus d'une variable marque la dérivée simple dans le temps, alors que le double point ¨ indique la double dérivée dans le temps (*notation de Newton*).

L'intégration peut se faire numériquement aussi. De nombreuses techniques existent, avec différents degrés de précisions. Par exemple, le méthode de Runge-Kutta. Cependant, en robotique mobile on aime faire les choses simples ! Par conséquent nous allons utiliser un intégrateur d'ordre zéro, qui utilise des rectangles. L'intégrale pour le signal x échantillonné à chaque Δt s'exprimera comme :

$$\int_{t_0}^{t_i} x(t)dt = x_0\Delta t + x_1\Delta t + ... + x_i\Delta t = (x_0 + x_1 + ... + x_i)\Delta t \tag{7}$$

où x_i est l'échantillon à l'instant t_i. Bien que moins précise que d'autres, cette méthode d'intégration a l'avantage d'être linéaire et d'être facilement exprimable par des équations matricielles.

6.1 Dérivée

Soit la fonction $y = sin(2t)$, avec Δt =`dt`=0.01, et la fonction évaluée (échantillonnée) à `t=0:dt:10` en notation matlab. Tracez sur un même graphique (commande `hold on` dans matlab, pas nécessaire dans `matplotlib`) :

— y ;
— la dérivée de y, calculée analytiquement ;
— la dérivée de y, calculée numériquement avec l'équation 6 ;

6.2 Intégrale

En utilisant la fonction `cumsum` ou `numpy.cumsum` , faites la même choses qu'en a) mais en remplaçant la dérivée par l'intégrale.

7 Linéarisation d'une fonction de capteur non-linéaire

7.1

Soit la fonction de capteur $v = f_{capteur}$ suivante (en Volt), et tracée à la Fig. 3.

$$f_{capteur}(x) = \frac{120}{x} e^{(-(ln(x)-4)^2)} \tag{8}$$

Notez que cette fonction est arbitraire mais ajustée de façon à approximer la forme de la réponse du télémètre infrarouge *Sharp GP2Y0A02YK0F*. Linéarisez cette fonction aux points suivants : $a_1 = 20$ cm, $a_2 = 50$ cm et $a_3 = 140$, ce qui correspond à n'utiliser que la première dérivée de la série de Taylor :

$$f_{capteur}(x)|_{a_i} \approx f_{capteur}(a_i) + f'_{capteur}(a_i)(x - a_i) \tag{9}$$

Utilisez un site en ligne (comme `http://www.derivative-calculator.net/`) ou n'importe quel logiciel de mathématique symbolique pour trouver la dérivée $f'_{capteur}(a_i)$. Indiquez votre réponse finale dans un format similaire à l'équation suivante :

$$f_i(x) = 0.274 + 90(x - 7) \tag{10}$$

(bien entendu, cette réponse n'est pas correcte !)

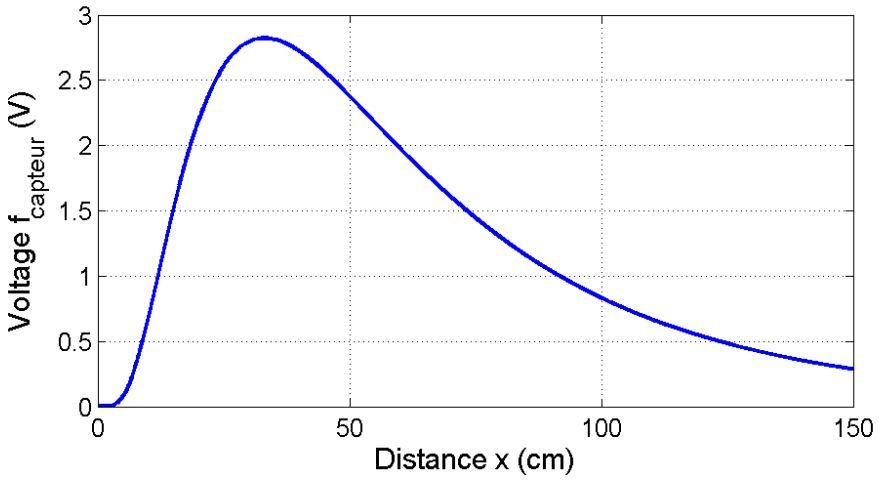

FIGURE 3 – Fonction du capteur $f_{capteur}$ relatant la distance x entre un robot et un mur, et le voltage produit par le capteur.

7.2

Le convertisseur analogue-numérique qui sera utilisé avec le robot a une précision de 0.0195 V. Quelle sera la précision sur l'estimation de la position x quand le robot se trouve près de a_1, a_2 et a_3 ? Basez-vous sur la sensibilité du capteur établi à partir des pentes (i.e. dérivées) trouvées à la partie 7.1 pour calculer cette précision.