A localization Study with Exponential Coordinates

Wang Longfei A0263224M e1010582@u.nus.edu

***Abstract*—Observing the distribution of the robot is the key for robot localization and SLAM problem. The Gaussian distribution defined in Cartesian coordinates is mainly used for the algorithm in localization. However, for a 2-wheeled robot with stochastic noise, the probability density function of position and orientation is often not in elliptical shape which means the Gaussian distribution in Cartesian coordinates cannot express it behavior well. What's more, as the uncertainty coefficient increase, the feature become more curve and the inconsistency is more obvious to that problem. In this paper we will derive a new kind of Gaussian distribution in exponential coordinates to fit for the 'banana' shape figure. And propagate new mean and covariance for Gaussian distribution in exponential coordinates.**

Keywords— Localization, SLAM, Gaussian

I. Introduction

In robotics research, there is a heated topic which is Simultanious Localization and Mapping (SLAM) to solve the problem about robot mapping. The robot starts to move from an unknown position in an unknown environment, and locates itself according to the position and map during the movement process. At the same time, an incremental map is built on the basis of its own positioning to realize the autonomous positioning and navigation of the robot. Due to the existence of noise, it's hard to determine the position and orientation of robots or veihcles as they move. One way is to calculate all the possible poses of them of evey frame with a probility density function (pdf).

For almost all the SLAM algorithm, the Guassian distributions are always defaultly in Cartesian coordinates (x, y) and orientation heading angle (θ). This Guassian representation can be fully parameterized by a mean and covariance. However, if we let the differential-drive robot whose two wheels' speed disturbed by stochastic noise move along a straight path. After plenty of experiements, the final poses distribution may look like a banana shape (Fig. 1) which can not match with the Guassian distribution in Cartesian coordinates [1]. As the nosie coeffienct increase, the curvature will become larger and inconsist with Gaussian distribution in Cartesian cooredinates.

Here is the outline of this paper. In Section II, we will talk about the literature related to this problem. Next, we will derive the stochastic differential equation for the drive robot in the plane with two methods in Section III and review rigid body motion and its relationship with exponential cooedinates in Section IV. In Section V, we will talk about the Guassian distribution in Cartesian and exponential coordinates and the propagation method will be used to obtain the mean and covariance in Section VII. Use Quantile-Quantile Plot (QQ plot) to find the fitter distribution in Section VIII. Finally, we apply a Kalman-Bucy Filter to this question in Section IX.

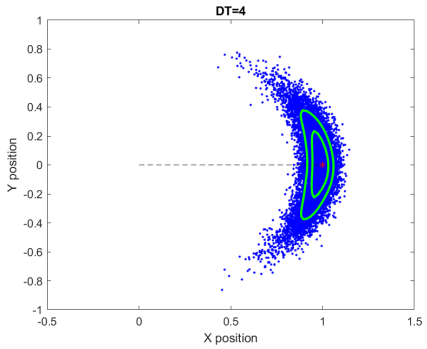

Fig. 1 Banana-shaped distribution for position of a differential-drive robot moving along a straight line with noisy wheel speeds.

II. Literature Review

As the robot move, the pdf will be stored and update which is overwhelmingly numeric for timely SLAM. Therefore, there are two main tools used to increase the effeiceny for SLAM algorithms which are extended Kalman filter (EKF) and the Rao-Black wellized (RB) particle filter [2]. There is another kind of filter called Unscented Kalman filter (UKF) [3] which considers Bayesian fusion problem in a matrix Lie group and proposed to tackle it using the unscented transform. It can be applied to more genrall case like nonlinear measurement model by a second-order approximation of a distribution defined directly on the lie group configuration space [4]. There is a improvement of the EKF which is Invariant Extended Kalman filter (IEKF) [5]. IKEF approach extends the state space where convergence of states and covariance are guarantees. It transforms a nonlinear process and measurement system to a locally linear system. And a new robust linearly constrained InEKF is introduced, together with particular parametric mismatched models and mitigation strategies through linear constraints, which has a better performance [6].

As we can see from the development of filter, the transformation of rigid body with the Lie group theory has drawn some attention for the SLAM problem. That is because of the inconsistency of Guassian distribution in Cartesian coordinates. Bailey et al. shows that the inconsistencies are almost certainly due to frowing heading uncertainty, not time. The EKF-SLAM algorithm is not so applicable to the real situation because of the poor linearization and observation model [7]. Shoudong Huang and Gamin Dissanayake also proved that the inconsistency may occur in EKF SLAM and when the robot orientation uncertainty is large, the estimator in consistency can result in highly optimistic confidence limits [8]. Therefore, in this paper we want to figure out the way can reduce the error when the

uncertainty coeffeicent increases.

III. Stochastic Differential Equation

The example we considered here is a two-wheeled cart robot or differential-drive robot. This kind of robot has two wheels with radius r sharing a common rotation axis with length l. The wheels are assumed to only roll but not slip. Vector $\mathbf{x} = [x\ y\ \theta]^T$ is given to describe its configuration. (x, y) is the position coordinates of Cartesian, θ and is the heading angle as shown in Fig. 2.

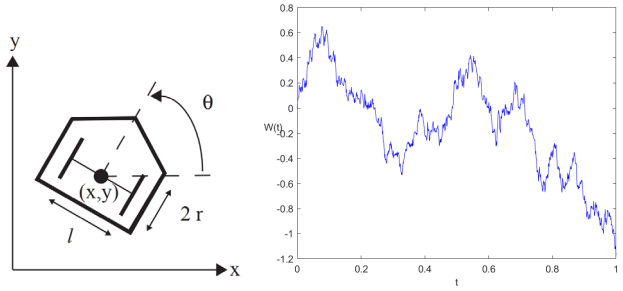

Fig. 2. (a) Notation for two-wheeled differential drive robot. (b) Discretized Brownian noise.

The angular velocity of each wheel is given by $\omega_i = d\phi_i/dt$ for i = 1 or 2. From Fig.2, we can derive the differential equations of the robot movement.

$$dx = \frac{r\cos\theta(d\phi_1 + d\phi_2)}{2}, \qquad dy = \frac{r\sin\theta(d\phi_1 + d\phi_2)}{2},$$

$$d\theta = \frac{r}{l}(d\phi_1 - d\phi_2) \tag{1}$$

If the wheels speeds are governed by the stochastic differential equations (SDE) as

$$d\phi_i = \omega_i dt + \sqrt{D}dw_i \quad \text{for} \quad i = 1 \text{ or } 2 \tag{2}$$

where $d\omega_i$ are unit-strength Wiener processes and D is a noise coefficient.

The unit-strength Wiener processes is also called scalar standard Brownian motion [9]. Over [0, T] is a random variable $W(t)$ that depends continuously on $t \in [0, T]$ and satisfies the following three conditions.

1. W (0) = 0 (with probability 1).
2. For 0 ≤s < t ≤T the random variable given by the increment W (t)−W (s) is normally distributed with mean zero and variance t −s; equivalently, W (t+s) −W (s) ∼ $\sqrt{t}$ N (0, 1), where N (0, 1) denotes a normally distributed random variable with zero mean and unit variance.
3. For 0 ≤s < t < u < v ≤T the increments W (t) −W(s) and W (v) −W (u) are independent.

For the discrete problem, $d\omega_i$ can be written as

$$dw = \sqrt{dt}\ N(0, 1) \tag{3}$$

After multiply different noise coefficients, it means add different random noise to the speeds of two wheels. The noise which is discretized Brownian is shown in Fig. 2(b) Combine the equations (1) (2), the SDE can be rewritten as

$$dx = \begin{pmatrix} dx \\ dy \\ d\theta \end{pmatrix} = \begin{pmatrix} \frac{r}{2}(\omega_1 + \omega_2)\cos\theta \\ \frac{r}{2}(\omega_1 + \omega_2)\sin\theta \\ \frac{r}{l}(\omega_1 - \omega_2) \end{pmatrix} dt + \sqrt{D}\begin{pmatrix} \frac{r}{2}\cos\theta & \frac{r}{2}\cos\theta \\ \frac{r}{2}\sin\theta & \frac{r}{2}\sin\theta \\ \frac{r}{l} & -\frac{r}{l} \end{pmatrix}\begin{pmatrix} dw_1 \\ dw_2 \end{pmatrix} \tag{4}$$

For all the example in the remainder of the paper, the parameters are given in table 1:

Symbol	*Definition*	*Value*	*Units*
l	The wheel base	0.2	m/s
r	Radius of the wheel	0.033	m/s
v	Driving forward speed	1	m/s
a	Radius of the arc	1	m
$\dot{\alpha}$	Angular acceleration	$\pi/2$	rad/s^2
dt	Time step	0.001	s
T	Total time	1	s

Table 1 The parameters of the example problem

In this paper we mainly consider two situation of the robot movements which would be driving straight or driving along an arc of constant curvature. For driving forward, the wheels speeds can be shown to be

$$\omega_1 = \omega_2 = \frac{\mathrm{v}}{r} \tag{5}$$

For driving along an arc

$$\omega_1 = \frac{\dot{\alpha}}{\mathrm{r}}(R + \frac{l}{2}), \qquad \omega_2 = \frac{\dot{\alpha}}{\mathrm{r}}(R - \frac{l}{2}) \tag{6}$$

From the Euler-Maruyama (EM) method, SDE equation can be written like

$$dX(t) = f(X(t))dt + g(X(t))dW(t),\ X(0) = X_0,\ 0 \leqslant t \leqslant T. \tag{7}$$

We want to apply equation (7) to discrete interval in the simulation, the equation for each path is

$$X_i = X_{i-1} + f(X_{i-1})\Delta t + g(X_{i-1})\Delta W,\ i = 1, 2, ..., L. \tag{8}$$

where in initial position and orientation is set to be $[0\ \ 0\ \ 0]^T$.

If we simulated the SDE for 10000 times in two cases which means there are 10000 possible paths for the robot. The results with diffusion constant DT equals to 1 and 7 are shown in Fig. 3 and Fig. 4 (here T=1). The dash line shows the ideal path of the robot without disturbance and the blue dot is the final pose of 10000 simulations while the red one is the ideal final pose. We can see that the distribution is in a banana shape and as the DT increase the curvature become larger

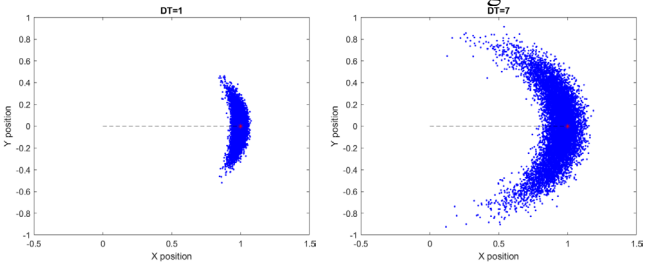

Fig. 3 Distribution of differential-drive robot moving along a straight line with DT=1,7.

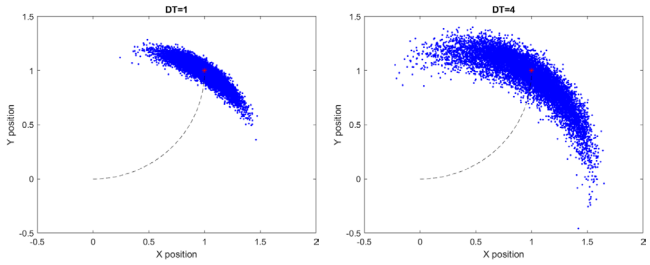

Fig. 4 Distribution of differential-drive robot moving along an acr line with DT=1,4.

From [9], there is a concept of strong convergence to measure the Euler-Maruyama method which means as Δt decrease, the EM estimated solution matches the true solution more closely. In our numerical test, we will focus on the error at the end point t= T, so we have below equation to measure the convergence

$$e_{\Delta t}^{\text{strong}} := \mathbb{E}\,|X_L - X(T)|, \quad \text{where } L\Delta t = T \tag{9}$$

It is possible to raise the strong order of EM by adding a correction to the stochastic increment, giving Milstein's Higher order method. The correction arises because the traditional Taylor expansion must be modified in the case of Itˆo calculus. A so-called Itˆo–Taylor expansion can be formed by applying Itˆo's result, which is a fundamental tool of stochastic calculus. Truncating the Itˆo–Taylor expansion at an appropriate point produces Milstein's method for the SDE.

$$X_i = X_{i-1} + f(X_{i-1})\Delta t + g(X_{i-1})\Delta W + \frac{1}{2}g(X_{i-1})g'(X_{i-1})\,(\Delta W^2 - \Delta t),\ i = 1, 2, ..., L. \tag{10}$$

If we simulate the results with Milstein's method, the plot cannot show significant different in Fig.5.

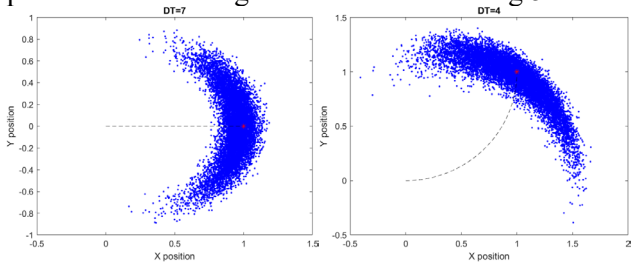

Fig. 5 Distribution of differential-drive robot moving along a striaght line (left) and acr line (right) with DT=7,4 by Milstein;s method.

However, we can check the strong convergence of two method to show the difference in Fig. 6. Here we only consider the straight line case while the acr line have the same conclusion.

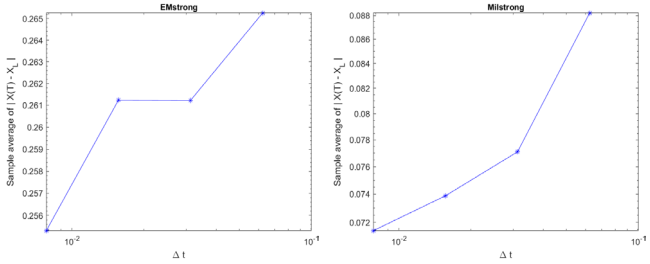

Fig. 6 Strong convergence by EM method (left) and Milstein's method (right).

From the plot we can see that as the Δt increase, the sample average will increase in two methods while Milstein's method is with a better convergence ability.

IV. Rigid Body Motion

Next we derive the rigid body motion transformation for the derivation of exponential coordinates. Form Chales theorem we know evry motion of a rigid body can be considered as a translation in space and a rotation about a point. And a homogeneous transformation matrice can be expressed as

$$g = \begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix} \in SE(2) \tag{11}$$

where $R \in SO(2)$, $t \in R^2$ and $g \in R^{3\times 3}$. $G = SE(2)$ is a special orthogonal Euclidean group where the group operator $\circ$ is matrix multiplication. For a vector in Cartesian coordinates to be $[\mathrm{t}_1 \ \ \mathrm{t}_2 \ \ \theta]$, g can be written as

$$\mathrm{g} = \begin{pmatrix} \cos\theta & -\sin\theta & \mathrm{t}_1 \\ \sin\theta & \cos\theta & \mathrm{t}_2 \\ 0 & 0 & 1 \end{pmatrix} \tag{12}$$

For a vector in exponential coordinates to be $\mathbf{x} = [v_1 \ \ v_2 \ \ \theta]^T$, an element of X the Lie algebra $se(2)$ can be expressed as

$$X = \hat{\mathbf{x}} = \begin{pmatrix} 0 & -\theta & v_1 \\ \theta & 0 & v_2 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \check{X} = \mathbf{x} \tag{13}$$

where the operators $\wedge$and $\vee$allow us to map from to and back.

By using the matrix exponential on element of $se(2)$, we can obtain group element of $SE(2)$ as

$$g(v_1, v_2, \theta) = \exp(X) \tag{14}$$

where the relationship between v and t

$$t_1 = [v_2(-1 + \cos\theta) + v_1 \sin\theta]/\theta \tag{15}$$

$$t_2 = [v_1(1 - \cos\theta) + v_2 \sin\theta]/\theta \tag{16}$$

Then we can get the vector of exponential coordinates $\mathbf{x}$ from g

$$\mathbf{x} = (\log(g))^{\vee} \tag{17}$$

where log is the matrix logarithm.

V. Guassian Distribution in Cartesian Coordinates and Exponential Corrdinates

For a vector x in mutidimension, the mean and covariance about the pdf are shown

$$\boldsymbol{\mu} = \int_{R^n} \mathbf{x} f(\mathbf{x}) d\mathbf{x} \tag{18}$$

$$\Sigma = \int_{R^n} (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T f(\mathbf{x}) d\mathbf{x} \tag{19}$$

For this discrete problem, they can be calculated as

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i \tag{20}$$

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T \tag{21}$$

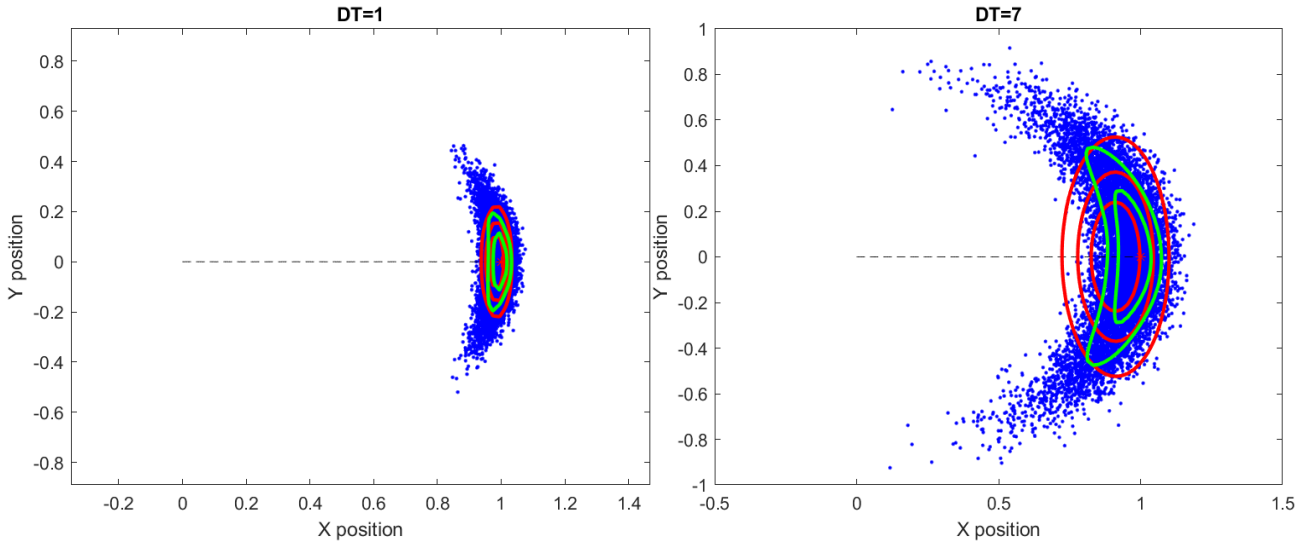

Fig. 7 Distribution of differential-drive robot moving along a straight line with DT=1 (left),7 (right). And pdf contours of Gaussians in Cartesian (red) and exponential (green) coordinates.

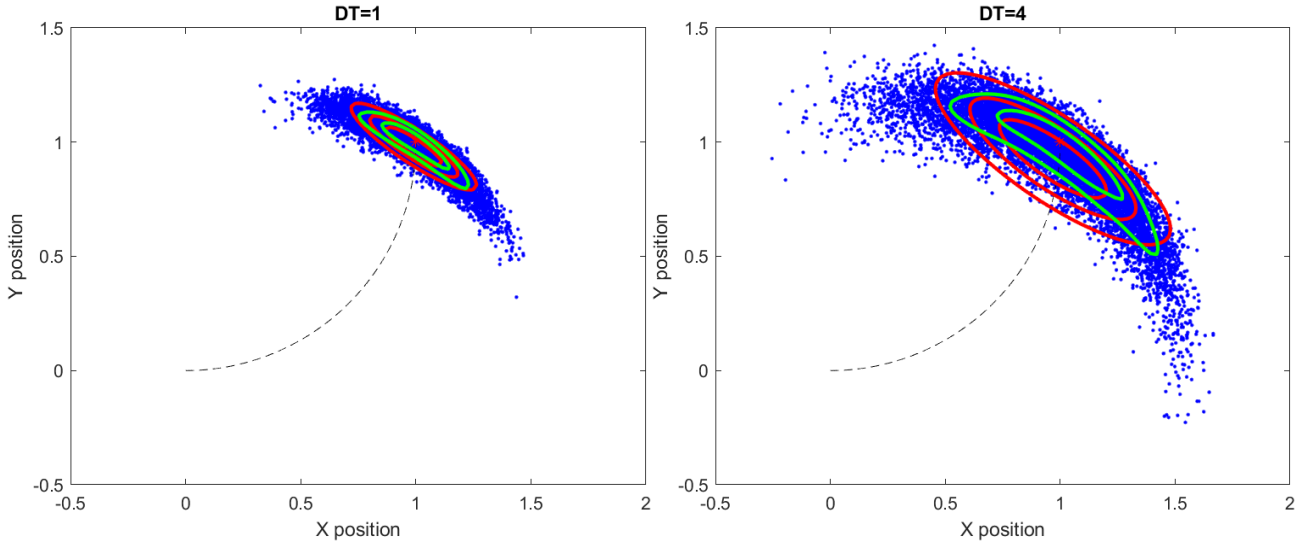

Fig. 8 Distribution of differential-drive robot moving along an arc line with DT=1 (left),4 (right). And pdf contours of Gaussians in Cartesian (red) and exponential (green) coordinates.

The mutivariate Gaussian distribution in Cartesian coordinates can be expressed as

$$f(\mathbf{x},\boldsymbol{\mu},\Sigma)=\frac{1}{(2\pi)^{n/2}|\det\Sigma|^{1/2}}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{T}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] \quad (22)$$

By the Lie algebra, we can obtain the mean and covariance for a group G

$$0=\int_{G}\log^{\vee}(\mu^{-1}\circ g)f(g)dg \quad (23)$$

$$\Sigma=\int_{G}\log^{\vee}(\mu^{-1}\circ g)[\log^{\vee}(\mu^{-1}\circ g)]^{T}f(g)dg \quad (24)$$

For the discrete problem, it can be approximated with the recursive formula

$$\mu_{k+1}=\mu_{k}\circ\exp[\frac{1}{N}\sum_{i=1}^{N}\log(\mu_{k}^{-1}\circ g_{i})] \quad (25)$$

$$\Sigma=\frac{1}{N}\sum_{i=1}^{N}y_{i}y_{i}^{T} \quad (26)$$

where $y_{i}=[\log(\mu^{-1}\circ g_{i})]^{\vee}$

After we derive all the equations, finally we can plot the Guassian distribution in Cartesian coordinates and exponential coordinates in Fig. 7 and Fig. 8. From them we can see that the Gaussian distribution in exponential coordinates fit the data more accurately. And as the uncertainty increases, the Gaussian distribution more can represents the real distribution of the poses than the Gaussian distribution in Cartesian coordinates although at the beginning this two kind of distribution can not see big difference. In Section VIII, we will use QQ plot to further invastigate which one can better match the real distribution.

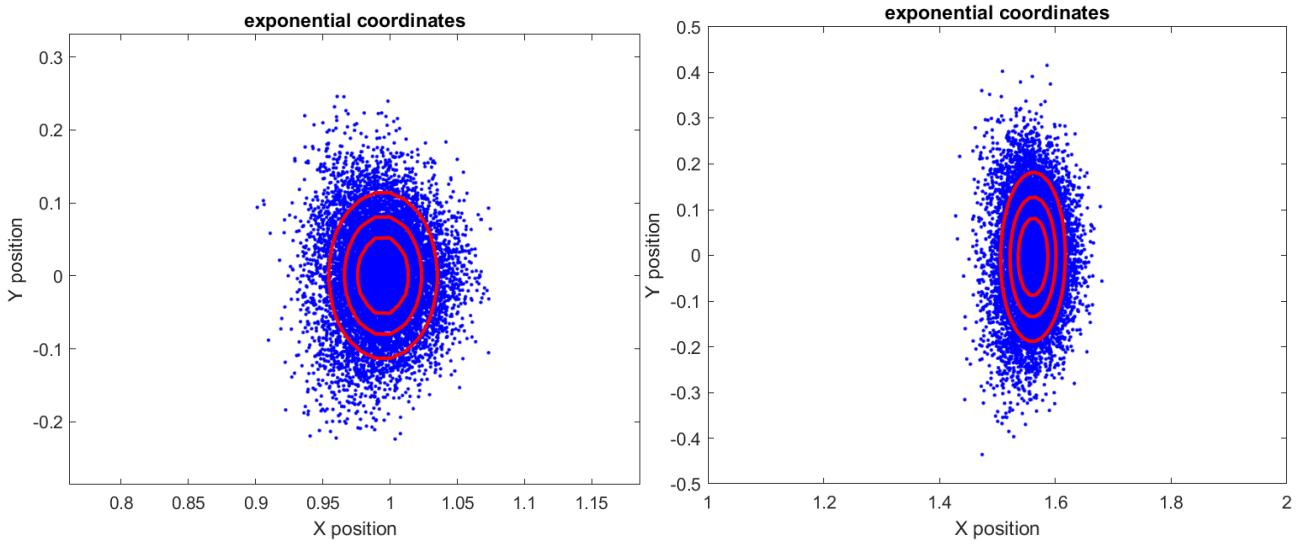

Fig. 9 Distribution as in exponential coordinates with pdf contours of Gaussian for drving straight (left) and driving along an arc (right) with diffusion rate DT=1.

VI. Sampled Data in Exponential Coordinates

From before section, we know the Guassian distribution in exponential coordinates will fit the sampled data better. And what if we change the poses of them and represent them in exponential coordinates. Will the distribution is eliptical and Guassian distribution fit them now?

The main key here is to use equation (17). Given a vector $[t_1 \ \ t_2 \ \ \theta]$, then the homogeneous transformation matrix g can be obtained. Do the log(.) to matrix will get X and $\vee$ operator to get the coordinates in exponential **x**. And then we plot those points and their Gaussian distribution in Fig. 9. We can see that the final poses are in elliptical shape and hence the Gaussian distribution fit the sampled data in exponential coordinates well.

VII. Propagation with Exponential Coordinates

There is another way to derive the mean and covariance in *SE(2)* . By using SDE in (2), we can derive another stochastic differential equation as

$$(g^{-1}g)^{\vee}dt=\begin{pmatrix}\cos\theta dx+\sin\theta dy\\ -\sin\theta dx+\cos\theta dy\\ d\theta\end{pmatrix}$$

$$=\begin{pmatrix}\frac{r}{2}(\omega_1+\omega_2)\\ 0\\ \frac{r}{2}(\omega_1-\omega_2)\end{pmatrix}dt+\sqrt{D}\begin{pmatrix}\frac{r}{2}&\frac{r}{2}\\ 0&0\\ \frac{r}{l}&\frac{r}{l}\end{pmatrix}\begin{pmatrix}d\omega_1\\ d\omega_2\end{pmatrix} \tag{26}$$

This will be written in short hand

$$\left(g^{-1}\dot{g}\right)^{\vee}dt=\mathrm{h}dt+Hdw \tag{27}$$

Then the mean and covariance in *SE(2) can be written as*

$$\mu(t)=\exp\left(\int_0^t\hat{\mathbf{h}}d\tau\right) \tag{28}$$

$$\Sigma(t)=\int_0^t Ad(\mu^{-1}(\tau))HH^TAd^T(\mu^{-1}(\tau))d\tau \tag{31}$$

For the straight-line movement ($\omega_1=\omega_2=w$), we have

$$\mu(t)=\begin{pmatrix}1&0&r\omega t\\ 0&1&0\\ 0&0&1\end{pmatrix} \tag{32}$$

$$\Sigma(t)=\begin{pmatrix}\frac{1}{2}Dr^2t&0&0\\ 0&\frac{2D\omega^2r^4t^3}{3l^2}&\frac{2D\omega r^3t^2}{l^2}\\ 0&\frac{D\omega r^3t^2}{l^2}&\frac{2Dr^2t}{l^2}\end{pmatrix} \tag{33}$$

For the arc-movement ($\omega_1\neq\omega_2$) with (6), we have

$$\mu(t)=\begin{pmatrix}\cos(\dot{\alpha}t)&-\sin(\dot{\alpha}t)&a\sin(\dot{\alpha}t)\\ \sin(\dot{\alpha}t)&\cos(\dot{\alpha}t)&a(1-\cos(\dot{\alpha}t))\\ 0&0&1\end{pmatrix} \tag{34}$$

$$\Sigma(t)=\begin{pmatrix}\sigma_{11}&\sigma_{12}&\sigma_{13}\\ \sigma_{21}&\sigma_{22}&\sigma_{23}\\ \sigma_{31}&\sigma_{32}&\sigma_{33}\end{pmatrix} \tag{35}$$

where

$$\sigma_{11}=\frac{c}{8}\left[\left(4a^2+l^2\right)\left(2\dot{\alpha}t+\sin\left(2\dot{\alpha}t\right)\right)+16a^2\left(\dot{\alpha}t-2\sin\left(\dot{\alpha}t\right)\right)\right],$$

$$\sigma_{12}=\sigma_{21}=\frac{-c}{2}\left[4a^2\left(-1+\cos\left(\dot{\alpha}t\right)\right)+l^2\right]\sin\left(\frac{\dot{\alpha}t}{2}\right)^2,$$

$$\sigma_{13}=\sigma_{31}=2ca\left(\dot{\alpha}t-\sin\left(\dot{\alpha}t\right)\right),$$

$$\sigma_{22}=-\frac{c}{8}\left(4a^2+l^2\right)\left(-2\dot{\alpha}t+\sin\left(2\dot{\alpha}t\right)\right),$$

$$\sigma_{23}=\sigma_{32}=-2ca\left(-1+\cos\left(\dot{\alpha}t\right)\right),$$

$$\sigma_{33}=2c\dot{\alpha},$$

$$c=\frac{Dr^2}{l^2\dot{\alpha}}$$

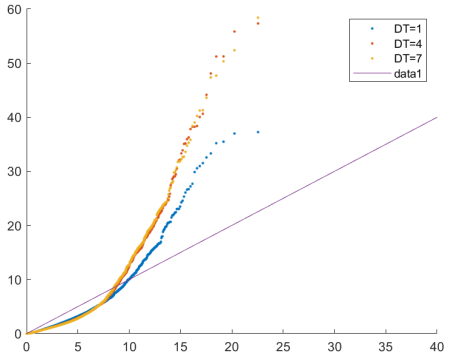

Fig. 10 Gaussian distribution in Cartesian coordinates with *DT*=1,4,7.

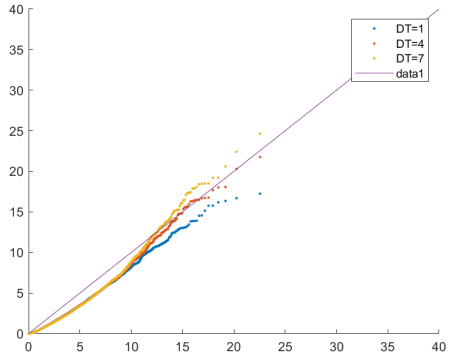

Fig. 11 Gaussian distribution in exponential coordinates with *DT*=1,4,7.

Then we can calculate the mean and covariance in *SE(2)* from 10000 sample data frames using (25) and (26) and compare them with propagation method. For the straight driving example with $r\omega = 1$ and $DT = 1$

$$\mu_{data} = \begin{pmatrix} 1 & 0.002 & 1 \\ -0.002 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{36}$$

$$\Sigma_{data} = \begin{pmatrix} 0.0007 & 0 & 0 \\ 0 & 0.0172 & 0.026 \\ 0 & 0.026 & 0.0534 \end{pmatrix} \tag{37}$$

By the propagation method

$$\mu_{prop} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{38}$$

$$\Sigma_{prop} = \begin{pmatrix} 0.0005 & 0 & 0 \\ 0 & 0.0182 & 0.0272 \\ 0 & 0.0272 & 0.0545 \end{pmatrix} \tag{39}$$

We can see the errors within the acceptable range. For the arc-movement of the robot, the auccracy between exponential coordinates and propagation method is similar.

VIII. QQ Plot

As we previouly say in Section V, we use Quantile-Quantile plot (QQ plot) here to verify whether a set of data comes from a particular distribution, including sample quantile and theoretical quantile. If the theoretical distribution is close to the sampled distribution, the point in QQ plot will approach closely to the line *y*=*x*. for the multidimensional vector here, we use the Mahalanobis Distance [10] to calculate the distance between sampled mean, covariance and theoretical mean and covariance as shown here

$$Dm = (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \tag{40}$$

For the Gaussian distribution in Cartesian coordinates, we choose *DT*=1,4,7 for the QQ plot in Fig. 10

For the Gaussian distribution in exponential coordinates, we also choose *DT*=1,4,7 for the QQ plot in Fig. 11

When *DT* is small the Guassian distribution in Cartesian coordinates line is close to *y*=*x*. However, when increase the diffusion rate it cannot match the sampled points. As for the exponential coordinates, the line is affected by little disturbance when *DT* increase. Hence , the QQ plot verify the conclusion we made in Section V that exponential coordiantes is better for the distribution here. For the arc-movement of the robot, the QQ plot of Cartesian and exponential coordinates are similar to the straight line one.

IX. Kalman Filter

For a discrete control system which can be represented by a linear stochasitc difference equation as

$$X(k) = AX(k-1) + W(k) \tag{41}$$

$$Y(k) = HX(k) + V(k) \tag{42}$$

W and V denote the strength of the noise for X and observer. Let

$$Q = W(k)W(k)^T \text{ and } R = V(k)V(k)^T \tag{43}$$

Then we can construct the Kalman filter for this problem from Algorithm 1 [11]

Algorithm 1 Kalman filter construction Algorithm

$\hat{X}_0 = \mathrm{E}[X_0]$ $P_0 = \mathrm{E}\left[\left(X_0 - \hat{X}_0\right)\left(X_0 - \hat{X}_0\right)^{\mathrm{T}}\right]$	Initialize the parameters
$\hat{X}_k = A\hat{X}_{k-1}$ $\mathrm{P_k} = \mathrm{AP_kA^T} + \mathrm{Q_{k-1}}$	Predict the state from last results
$\mathrm{G_k} = \mathrm{P_kH_k}^{\mathrm{T}}(\mathrm{H_kP_kH_k}^{\mathrm{T}} + \mathrm{R_k})^{-1}$	Calculate the Kalman gain
$\hat{\mathrm{X}}_{\mathrm{k}} = \hat{\mathrm{X}}_{\mathrm{k-1}} + \mathrm{G_k}\left(\mathrm{y_k} - \mathrm{H_k}\hat{\mathrm{X}}_{\mathrm{k-1}}\right)$	Estimate the new state
$\mathrm{P_k} = (\mathrm{I} - \mathrm{G_kH_k})\mathrm{P_k}$	New covariance matrix

For this problem the initial condition is $\hat{X}_0 = [0,0]^T$ and $P_0 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$. And the transformation matrix of state *A*=1 as the robot ideal move a straight line. We assume the observation noise *V* to be identity and *Q* simultaneously change as time changes in Fig. 12 (diffusion rate *DT*=7). Therefore, for a noise system we know the exact noise of every moment, and we apply it to the adaptive Kalman filter, it shows the filter

successfully filter the noise and estimate the path as close as to the ideal one. Then we realize the function of Kalman filter which is to minimize the prediction error.

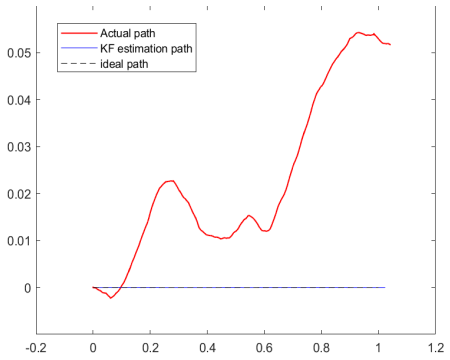

Fig.12 The real path, ideal path and estimated path.

X. Conclusion

For the robot localization problem, we establish the SDE equation for the robot movement. After derivation of rigid body motion in Lie group and Gaussian distribution, we find the banana shape distribution more fit with the Gaussian distribution in exponential coordinates. By changing the sampled data to exponential coodinates and using QQ plot prove the conclusion. We also use propagation method to calculate the exponential coordinates. Finally the adaptive kalman filter is introduced to measure its function. For the future plan, the new filter which can fusing data from different model together can be developed. And we can apply this paper theories to more diverse robot like soft robot and caterpillar robot. In addition, we can do some experienment in real life to justify our simulation.

References

[1] A. Long, K. Wolfe, M. Mashner, G. S. Chirikjian. (2012) The banana distribution is Gaussian: a localization study with exponential coordinates. In: Robotics: science and systems

[2] H. Durrant Whyte and T. Bailey. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. IEEE Robotics and Automation Magazine, 13(2):99–110, 2006.

[3] M. Brossard, S. Bonnabel and J. -P. Condomines, "Unscented Kalman filtering on Lie groups," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 2485-2491, doi: 10.1109/IROS.2017.8206066.

[4] G. Chirikjian and M. Kobilarov, "Gaussian approximation of non-linear measurement models on Lie groups," 53rd IEEE Conference on Decision and Control, 2014, pp. 6401-6406, doi: 10.1109/CDC.2014.7040393.

[5] N. Y. Ko, G. Song, W. Youn, I. H. Choi and T. S. Kim, "Improvement of Extended Kalman Filter Using Invariant Extended Kalman Filter," 2018 18th International Conference on Control, Automation and Systems (ICCAS), 2018, pp. 948-950.

[6] P. Chauchat, J. Vilà-Valls and E. Chaumette, "Robust Linearly Constrained Invariant Filtering for a Class of Mismatched Nonlinear Systems," in IEEE Control Systems Letters, vol. 6, pp. 223-228, 2022, doi: 10.1109/LCSYS.2021.3064931.

[7] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the EKF-SLAM algorithm. In Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems (IROS), pages 3562–3568, 2006.

[8] S. Huang and G. Dissanayake, "Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM," in IEEE Transactions on Robotics, vol. 23, no. 5, pp. 1036-1049, Oct. 2007, doi: 10.1109/TRO.2007.903811.

[9] Higham D J. An algorithmic introduction to numerical simulation of stochastic differential equations[J]. SIAM review, 2001, 43(3): 525-546.

[10] Mahalanobis, P. Chandra. (1936) On the generalised distance in statistics. In: Proceedings of the National Institute of Sciences of India, pages 49-55

[11] S. Akhlaghi, N. Zhou and Z. Huang, "Adaptive adjustment of noise covariance in Kalman filter for dynamic state estimation," 2017 IEEE Power & Energy Society General Meeting, 2017, pp. 1-5, doi: 10.1109/PESGM.2017.8273755.

Appendix

Part of the MATLAB code is attached here.

```
clear
clc
% initial parameters
l=0.2; r=0.033; v=1; %set the velocity
w1=v/r; w2=v/r;%two omega are same
ti=10000;% times(how many pathes)
T=1; %total time
dt=0.001; %time step
D=4; %noise coeffcient
P=T/dt; %how many points in one path
x = zeros(ti,P);
y = zeros(ti,P);
theta=zeros(ti,P);
%Brownian increments
randn('state',400)
dw1=sqrt(dt)*randn(ti,P);% two omega
increase randomly and differently
dw2=sqrt(dt)*randn(ti,P);
%plot the distribution in cartesian
figure %EM method of SDE
for i=1:ti %for the first path
    for j=2:P % from start time to end
        x(i,j) = x(i,j-1) +
0.5*r*(w1+w2)*cos(theta(i,j-1))*dt +
sqrt(D)*0.5*r*cos(theta(i,j-1))*(dw1(i,j-
1)+dw2(i,j-1));
        y(i,j) = y(i,j-1) +
0.5*r*(w1+w2)*sin(theta(i,j-1))*dt +
```

```
sqrt(D)*0.5*r*sin(theta(i,j-1))*(dw1(i,j-
1)+dw2(i,j-1));
        theta(i,j) = theta(i,j-1) +
dt*r*(w1-w2)/l + sqrt(D)*r*(dw1(i,j-1)-
dw2(i,j-1))/l;
    end
    plot(x(i,P),y(i,P),'b.')%plot every end
point of evry path
    hold on
end
save('Data',"x","y")
%ideal path
t_ideal = linspace(0,1,ti);
x_ideal=v*t_ideal;
y_ideal=zeros(1,ti);
plot(x_ideal,y_ideal,'--k')
plot(1,0,'r*')
axis([-0.5 1.5 -1 1])
xlabel('X position')
ylabel('Y position')
title(['DT=',num2str(D)])
hold on
%Cartesian pdf
%mean
mean_ca=zeros(1,3);
mean_ca(1)=sum(x(:,end))/ti;%sum all the end
point's x
mean_ca(2)=sum(y(:,end))/ti;
mean_ca(3)=sum(theta(:,end))/ti;
%covariance
multi=zeros(3);
for o=1:ti
    multi=multi+([x(o,end)-
mean_ca(1);y(o,end)-mean_ca(2);theta(o,end)-
mean_ca(3)]*[x(o,end)-mean_ca(1);y(o,end)-
mean_ca(2);theta(o,end)-mean_ca(3)]');
end
cov_ca = multi/ti;
num=100;
xlim = get(gca,'XLim');
ylim = get(gca,'YLim');
[xc,yc] =
meshgrid(linspace(xlim(1),xlim(2),num)',lins
pace(ylim(1),ylim(2),num)');
zc=zeros(num);
for i=1:num
    for j= 1:num
        zc(i,j)=exp((-
0.5)*([xc(i,j),yc(i,j)]-
mean_ca(1:2))*inv(cov_ca(1:2,1:2))*([xc(i,j)
,yc(i,j)]'-
mean_ca(1:2  )'))/(2*pi*(det(cov_ca(1:2,1:2)
))^(0.5));
    end
end
c1=contour(xc,yc,zc,3,'r','linewidth',2);
% %Exponential pdf
xx=x(:,end);     yy=y(:,end);
data=[xx, yy];
GMModel = fitgmdist(data,10);
GMMpdf=reshape(pdf(GMModel,
[xc(:),yc(:)]),num,num);
hold on
c2=contour(xc, yc,
GMMpdf,2,'g','linewidth',2);
hold off
legend([c1,c2],{'Cart pdf','Exp pdf'})
%QQ plot for Cartesian
dm=zeros(1,ti);
for i=1:ti
    dm(i)=[x(i,end)-mean_ca(1),y(i,end)-
mean_ca(2)]*inv(cov_ca(1:2,1:2))*[x(i,end)-
mean_ca(1),y(i,end)-mean_ca(2)]';
end
dm=sort(dm);
xt=0:40;
yt=xt;
pt=((1:ti)-0.5)/ti;
xm=chi2inv(pt,3);
figure;
scatter(xm,dm','.');
hold on
plot(xt,yt);
hold off
%% Express points by exponential coordinate
x_exp=zeros(1,ti);
y_exp=zeros(1,ti);
a_exp=zeros(1,ti);
%plot distribution in exponential
figure;
for i=1:ti
    H=[cos(theta(i,end)),-
sin(theta(i,end)),x(i,end);

sin(theta(i,end)),cos(theta(i,end)),y(i,end)
;
        0,0,1 ];
    N=logm(H); %exp(N)=H
    x_exp(i)=N(1,3);        % v1
    y_exp(i)=N(2,3);        % v2
    a_exp(i)=N(2,1);    % alpha
    plot(x_exp(i),y_exp(i),'b.');
    hold on
end
axis([0.4 1.5 -0.8 0.8])
xlabel('X position')
ylabel('Y position')
title('exponential coordinates')
%pdf in exponential coordinate
%mean
mean_exp=zeros(1,3);
mean_exp(1)=sum(x_exp)/ti;%sum all the end
point's x
mean_exp(2)=sum(y_exp)/ti;
mean_exp(3)=sum(a_exp(:,end))/ti;
%covariance
multi=zeros(3);
for o=1:ti
    multi=multi+([x_exp(o)-
mean_exp(1);y_exp(o)-mean_exp(2);a_exp(o)-
```

```
mean_exp(3)]*[x_exp(o)-mean_exp(1);y_exp(o)-
mean_exp(2);a_exp(o)-mean_exp(3)]');
end
cov_exp = multi/ti;
xlim_exp = get(gca,'XLim');
ylim_exp = get(gca,'YLim');
[xe,ye] =
meshgrid(linspace(xlim_exp(1),xlim_exp(2),nu
m)',linspace(ylim_exp(1),ylim_exp(2),num)');
zc_exp=zeros(num);
for i=1:num
    for j= 1:num
        zc_exp(i,j)=exp((-
0.5)*([xe(i,j),ye(i,j)]-
mean_exp(1:2))*inv(cov_exp(1:2,1:2))*([xe(i,
j),ye(i,j)]'-
mean_exp(1:2)'))/(2*pi*(det(cov_exp(1:2,1:2)
))^(0.5));
    end
end
contour(xe,ye,zc_exp,3,'r','linewidth',2);
%QQ plot for exponential
dme=zeros(1,ti);
for i=1:ti
    dme(i)=[x_exp(i)-mean_exp(1),y_exp(i)-
mean_exp(2)]*inv(cov_exp(1:2,1:2))*[x_exp(i)
-mean_exp(1),y_exp(i)-mean_exp(2)]';
end
dme=sort(dme);
xt=0:40;
yt=xt;
pt=((1:ti)-0.5)/ti;
xm=chi2inv(pt,3);
figure;
scatter(xm,dme','.');
hold on
plot(xt,yt);
hold off
%% Propagation method
% mean
t=T;
mean_prop=[1 0 r*w1*t; 0 1 0; 0 0 1];
% covariance
sigma11=0.5*D*t*r^2;
sigma22=(2*D*(w1^2)*(r^4)*(t^3))/( 3*(l^2) )
;
sigma23=D*w1*r^3*t^2/l^2;
sigma32=sigma23;
sigma33=2*D*r^2*t/(l^2);
cov_prop=[sigma11 0 0; 0 sigma22 sigma23; 0
sigma32 sigma33];
% zc_exp2=zeros(num);
% for i=1:num
%     for j= 1:num
%         zc_exp2(i,j)=exp((-
0.5)*([xe(i,j),ye(i,j)]-
mean_prop(1:2))*inv(cov_prop(1:2,1:2))*([xe(
i,j),ye(i,j)]'-
mean_prop(1:2)'))/(2*pi*(det(cov_prop(1:2,1:
2)))^(0.5));
%     end
% end
%
contour(xe,ye,zc_exp2,3,'g','linewidth',2);
%% Kalman filter
kf = [];
xk=[0;0];
Pk=zeros(2);
A=[1,0;0,0];
R=[1,0;
   0,1];
H=eye(2);
% Q=[0.1,0;
%    0,0.1];
for i = 1:P
    z= [x(1,i);y(1,i)];

N1=[sqrt(D)*0.5*r*cos(theta(1,i)),sqrt(D)*0.
5*r*cos(theta(1,i));

sqrt(D)*0.5*r*sin(theta(1,i)),sqrt(D)*0.5*r*
sin(theta(1,i))];
    Q=N1*N1';
    % prediction
    xhat = A * xk;
    Phat = A * Pk * A' + Q;
    %gain
    K = Phat * H' * inv( H*Phat*H'+R);
    xhat = xhat + K*(z-H*xhat);
    Pk = (1 - K * H) * Phat;
    % Store
    xk=xhat;
    kf = [kf,xhat];
end
figure
plot(x(1,:),y(1,:),'linewidth',
1,'color','r');
hold on
plot(kf(1,:),kf(2,:),'b');
plot(x_ideal,y_ideal,'--k');
legend('Actual path','KF estimation
path','ideal path');
hold off
```