使用欧拉法推导差速机器人的里程计和误差传导模型:

Part1 里程计模型

记 X，Y，θ为参考坐标系下的横坐标，纵坐标和夹角。
则 X_k，Y_k，θ_k，分别代表参考坐标下，第 k 时刻的横坐标，纵坐标和夹角。
记 v 为机器人在机器人坐标下，沿着机器人坐标的 x 轴运动的速度。
记ω为机器人在机器人坐标下，瞬时旋转角速度。
则有:

$$\dot{X} = v\ cos\theta$$

$$\dot{Y} = v\ sin\theta$$

$$\dot{\theta} = \omega$$

由于是差速机器人，有

$$v = \frac{r(\dot{\phi}_R + \dot{\phi}_L)}{2}$$

$$\omega = \frac{r(\dot{\phi}_R - \dot{\phi}_L)}{2\ d}$$

其中$\dot{\phi}_R$、$\dot{\phi}_L$左右轮子的转的角度的导数。d 表示左右轮子间距的一半。
记T_s为从 k 到 k+1 时刻所经历的时间。
则有:

$$X_{k+1} = X_k + \Delta X = X_k + T_s *\ \dot{X}_k$$

而

$$\dot{X}_k = v_k\ cos\theta_k$$

因此有:

$$X_{k+1} = X_k + T_s *\ v_k * cos\theta_k$$
$$Y_{k+1} = Y_k + T_s *\ v_k * sin\theta_k$$
$$\theta_{k+1} = \theta_k + T_s *\ \omega_k$$

记$T_s *\ v_k = \Delta s$，$T_s *\ \omega_k = \Delta\theta$，
则最终有:

$$X_{k+1} = X_k + \Delta s * cos\theta_k$$
$$Y_{k+1} = Y_k + \Delta s * sin\theta_k$$
$$\theta_{k+1} = \theta_k + \Delta\theta$$

即为差速机器人的里程计模型。
其中有:

$$\Delta s = \frac{r(\Delta\phi_R + \Delta\phi_L)}{2}$$
$$\Delta\theta = \frac{r(\Delta\phi_R - \Delta\phi_L)}{2d}$$

Part2 误差传导模型

由上述推导，有：

$$X_{k+1} = X_k + \frac{r(\Delta\phi_R + \Delta\phi_L)}{2} * cos\theta_k$$
$$Y_{k+1} = Y_k + \frac{r(\Delta\phi_R + \Delta\phi_L)}{2} * sin\theta_k$$
$$\theta_{k+1} = \theta_k + \frac{r(\Delta\phi_R - \Delta\phi_L)}{2d}$$

记 p = $\begin{bmatrix} X_k \\ Y_k \\ \theta_k \end{bmatrix}$, $\phi = \begin{bmatrix} \Delta\phi_R \\ \Delta\phi_L \end{bmatrix}$

记 p' = $f(X_k, Y_k, \theta_k, \Delta\phi_R, \Delta\phi_L) = \begin{bmatrix} X_{k+1} \\ Y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} X_k + \frac{r(\Delta\phi_R+\Delta\phi_L)}{2} * cos\theta_k \\ Y_k + \frac{r(\Delta\phi_R+\Delta\phi_L)}{2} * sin\theta_k \\ \theta_k + \frac{r(\Delta\phi_R-\Delta\phi_L)}{2d} \end{bmatrix}$

里程计误差传导函数为：

$$\Sigma_{p\prime} = \frac{\partial f}{\partial p} * \Sigma_p * (\frac{\partial f}{\partial p})^T + \frac{\partial f}{\partial \phi} * \Sigma_\phi * (\frac{\partial f}{\partial \phi})^T$$

其中 Σ_p代表起始点的初始协方差矩阵，Σ_ϕ代表运动增量的协方差矩阵，也即：

$$\Sigma_\phi = \begin{bmatrix} k_R * |\Delta\phi_R| & 0 \\ 0 & k_L * |\Delta\phi_L| \end{bmatrix}$$

其中k_R，k_L为常数。

计算可得：

$$\frac{\partial f}{\partial p} = \begin{bmatrix} 1 & 0 & \frac{r(\Delta\phi_R + \Delta\phi_L)}{2} * -sin\theta_k \\ 0 & 1 & \frac{r(\Delta\phi_R + \Delta\phi_L)}{2} * cos\theta_k \\ 0 & 0 & 1 \end{bmatrix}$$

$$\frac{\partial f}{\partial \phi}=\begin{bmatrix} \frac{r}{2}*cos\theta_k & \frac{r}{2}*cos\theta_k \\ \frac{r}{2}*sin\theta_k & \frac{r}{2}*sin\theta_k \\ \frac{r}{2d} & \frac{-r}{2d} \end{bmatrix}$$

Part3 仿真实验验证：

1.对里程计正确性的验证：

编写 python 代码：

```python
def x_k_1(x_k,delta_R,delta_L,theta_k):
    new_x = x_k + r*(delta_R+delta_L)*cos(theta_k)/2
    return new_x
def y_k_1(y_k,delta_R,delta_L,theta_k):
    new_y = y_k + r*(delta_R+delta_L)*sin(theta_k)/2
    return new_y
def theta_k_1(theta_k,delta_R,delta_L):
    new_theta = theta_k + r*(delta_R-delta_L)/(2*d)
    return new_theta
```

上述代码定义了 x,y,z 和左右轮子转速时间的关系。

接下来模拟运行 3000 秒：

模拟运行 3000 秒：

```python
x = []
y = []
theta = []
x.append(0)
y.append(0)
theta.append(0)
degree = 0.1
s = 0

for k in range(3000):
    deltaL = 0.1
    deltaR = 0.1
    s += r*(deltaR+deltaL)/2
    theta_t += (deltaR-deltaL)/(2*d)
    x.append(x_k_1(x[len(x)-1],deltaR,deltaL,theta[len(theta)-1]))
    y.append(y_k_1(y[len(y)-1],deltaR,deltaL,theta[len(theta)-1]))
    theta.append(theta_k_1(theta[len(theta)-1],deltaR,deltaL))
```

其中

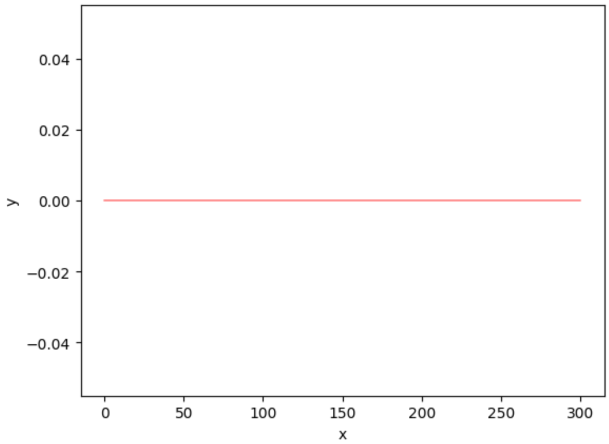
```python
deltaL = 0.1
```

```
deltaR = 0.1
```

是我们能够控制的左右轮子的转速。s 为里程计的路径累计。theta_t 为里程计的角度累计

首先设置$\Delta\phi_R = 0.1$，$\Delta\phi_L = 0.1$,预计其路径为沿着 x 轴平行的直线，并且预期 s 等于 x，theta_t 等于 0：

运行结果为：



确实为一条直线，并且 x 与 s 均为 `299.9999999999997`，`theta_t` 为 `0`
符合预期。

接下来考虑使其运动轨迹为 $y = k*x^2$

左右两边同时微分，得到：

$$\Delta Y = 2*k*X*\Delta X$$

代入前文得到的运动学公式：

$$\frac{r(\Delta\phi_R + \Delta\phi_L)}{2} * sin\theta_k = 2*k*X*\frac{r(\Delta\phi_R + \Delta\phi_L)}{2} * cos\theta_k$$

化简得到：

$$tan\theta = 2*k*X$$

再次左右两边同时微分，得到：

$$\frac{\Delta\theta}{(cos\theta)^2} = 2*k*\Delta X$$

代入$\Delta\theta 和 \Delta X$，有：

$$\frac{r(\Delta\phi_R - \Delta\phi_L)}{2d\,(cos\theta)^2} = 2*k*\frac{r(\Delta\phi_R + \Delta\phi_L)}{2} * cos\theta$$

因式相乘得到：

$$[1 - 2*\mathrm{d}*\mathrm{k}*(cos\theta)^3\,] * \Delta\phi_R = [1 + 2*\mathrm{d}*\mathrm{k}*(cos\theta)^3\,] * \Delta\phi_L$$

因此，只要$\Delta\phi_R$和$\Delta\phi_L$满足上式，即可使得轨迹为抛物线。

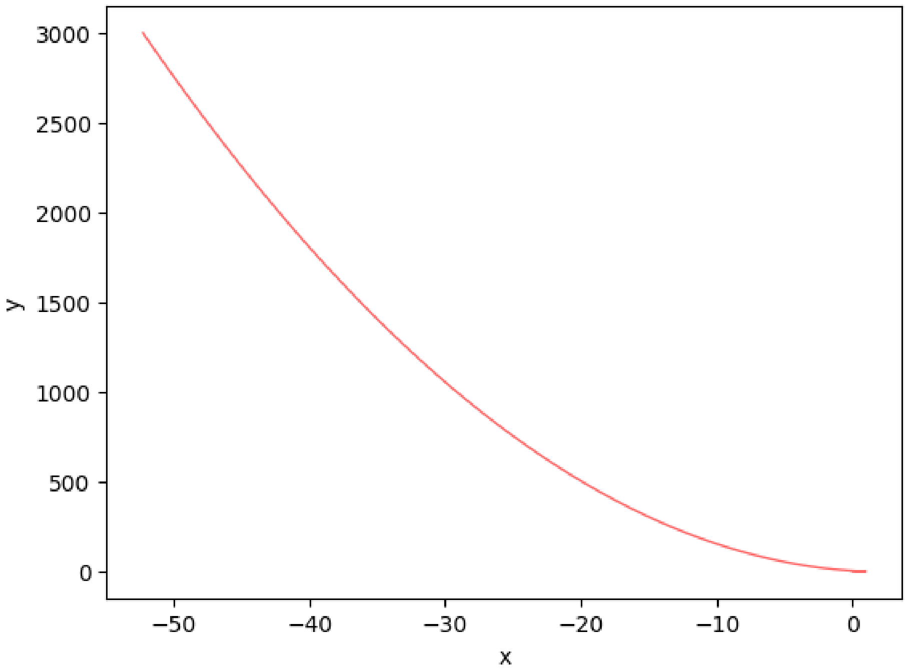
取$\Delta\phi_R = [1 + 2*\mathrm{d}*\mathrm{k}*(cos\theta)^3\,]$，$\Delta\phi_L$=$[1 - 2*\mathrm{d}*\mathrm{k}*(cos\theta)^3\,]$，

也即是:

```
deltaL = 1- 2*d*degree*pow(cos(theta[len(theta)-1]),3)
deltaR = 1+ 2*d*degree*pow(cos(theta[len(theta)-1]),3)
```

其中 degree 可以使得轨迹为 y = (degree) x^2

先设置 degree = 1，运行得:



可见确实呈抛物线状。

其中末端点坐标为（-52.24064354601165，2998.0553295598916），

52.24*52.24 = 2729,近似满足 y = x^2

再对两个里程计进行验证。

查看得到 s 的实际值为：3000，theta_t 的实际值为 1.5799167210464078

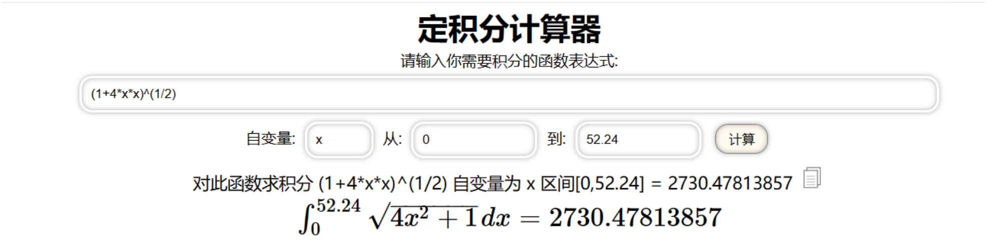
计算 s 的理论值 s_ideal:

$$s_{ideal} = \int_0^{52.24} \sqrt{1 + \left(\frac{dy}{dx}\right)^2} \ dx$$

也即是

$$s_{ideal} = \int_0^{52.24} \sqrt{1 + 4x^2} \ dx$$

使用定积分计算得到:

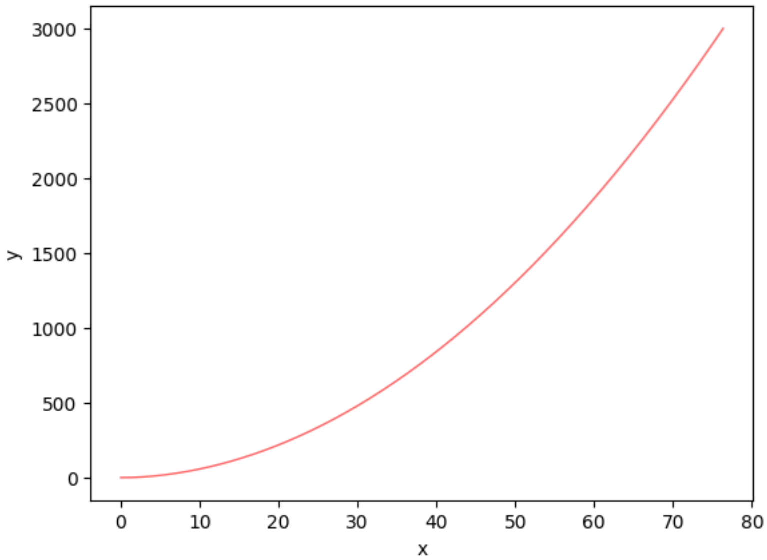


与实际测量里程计的误差为（3000-2730）/3000 *100% = 9%
认为有较小的误差。

再验证角度里程计：

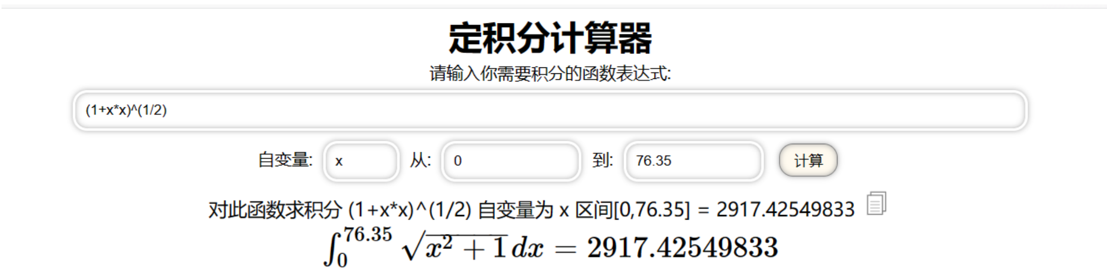
角度里程计的实际值 theta_t 为 1.5799167210464078
而理论值为：arctan(2998.05/52.24) = 1.55337343，
角度里程计的误差为 1.6%。
因此可以认为，在这种设置下，角度里程计较为准确，有较小的误差。

再将 degree 设置为 0.5：



可见仍然为抛物线，并且最终点坐标为：
（76.35812574726639，2997.1118989273637）
s = 3000, theta_t = 1.5578959596429287

并且可以验证（76.35*76.35）/2 = 2914.66，也近似满足预定轨迹
其距离里程计和角度里程计同上文，可以得到：



距离里程计误差为：2.7 %

角度里程计误差为： 0.8％

较小的误差验证了里程计方法的正确性。

2.对误差传导：

定义各个误差项：

```
def delta_R_L(delta_R,delta_L):
    delta_R_L_martrix =
np.array([kR*abs(delta_R),0],[0,kL*abs(delta_L)])
    return delta_R_L_martrix

def f_p(delta_R,delta_L,theta_k):
    f_p_martrix = np.array([1,0,-
r*(delta_R+delta_L)*sin(theta_k)/2],[0,1,r*(delta_R+delta_L)*cos(theta_
k)/2],[0,0,1])
    return f_p_martrix

def f_phai(theta_k):
    f_phai_martrix =
np.array([r*cos(theta_k)/2,r*cos(theta_k)/2],[r*sin(theta_k)/2,r*sin(th
eta_k)/2],[r/(2*d),r/(2*d)])
    return f_phai_martrix
```

模拟进行 25 误差传递：

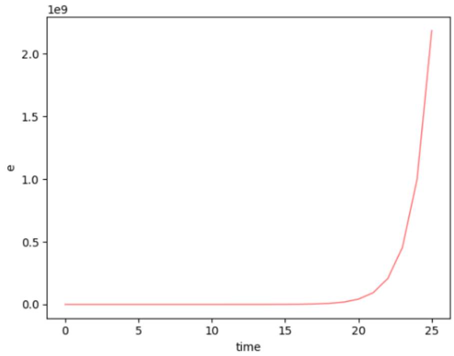
```
p += f_p(deltaR,deltaL,theta[len(theta)-1]) @ p @
f_p(deltaR,deltaL,theta[len(theta)-1]).T

p += f_phai(theta[len(theta)-1]) @ delta_R_L(deltaR,deltaL) @
f_phai(theta[len(theta)-1]).T
```

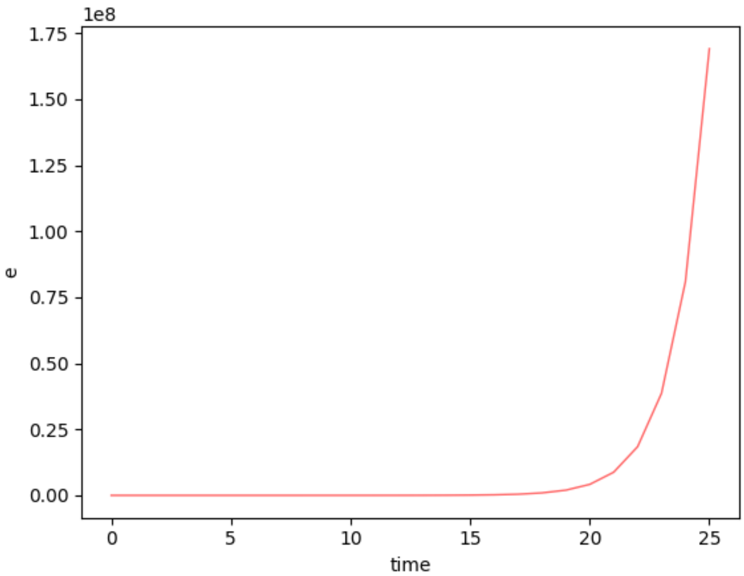
可以得出：

x,y θ 的误差关于时间的表现为：

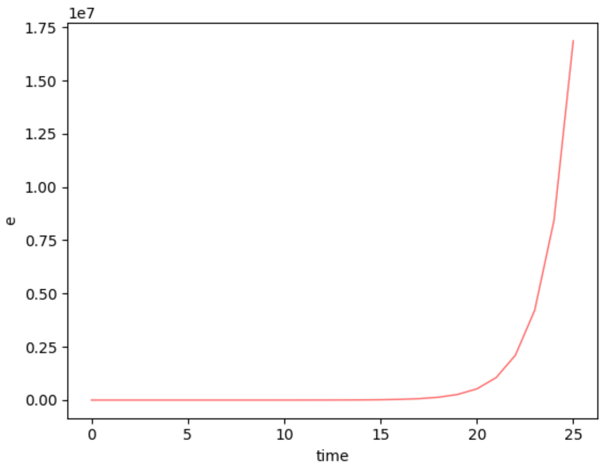
x：


1e9
2.0
1.5
1.0
0.5
0.0
e
0
5
10
15
20
25
time

Y:


1e8
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
e
0
5
10
15
20
25
time

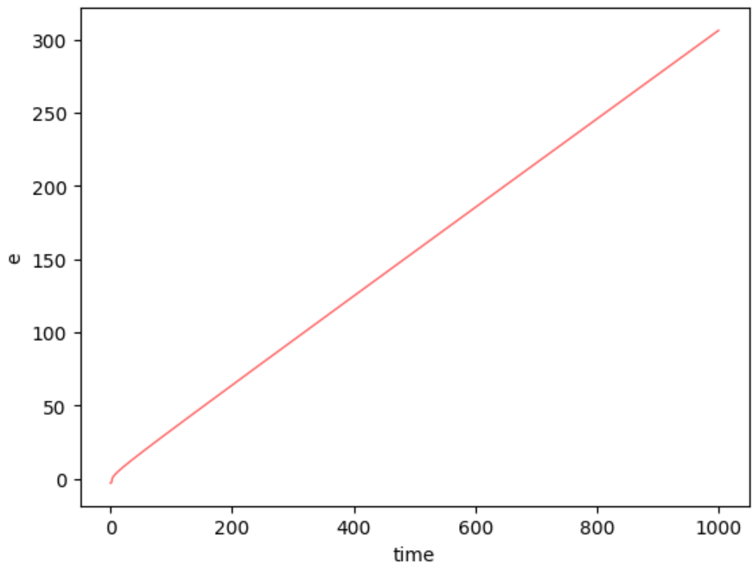
θ:


1e7
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
e
0
5
10
15
20
25
time

可见，误差传递大约呈现出指数关系

如果将时间延长到 1000 秒，并且取 log10，会得到：

误差关于时间的关系：



近似为一条直线。因而认为对于“误差关于时间成指数”的猜想是合理的。