为了符合MPC的线性模型方程表达式， 且满足下面的微分方程，我们分别构建对应的横向模型和纵向模型。

$$\dot{X} = AX + Bu$$

横向模型

横向模型我们使用第四章节中的横向动力学模型，具体推导过程见第四章课件第52页，

$$\frac{d}{dt}\begin{bmatrix} e_1 \\ \dot{e_1} \\ e_2 \\ \dot{e_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_{af}+2C_{ar}}{mV_x} & \frac{2C_{af}+2C_{ar}}{m} & \frac{-2C_{af}l_f+2C_{ar}l_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2C_{af}l_f-2C_{ar}l_r}{I_zV_x} & \frac{2C_{af}l_f-2C_{ar}l_r}{I_z} & -\frac{2C_{af}l_f^2+2C_{ar}l_r^2}{I_zV_x} \end{bmatrix}\begin{bmatrix} e_1 \\ \dot{e_1} \\ e_2 \\ \dot{e_2} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2C_{af}}{m} \\ 0 \\ \frac{2C_{af}l_f}{I_z} \end{bmatrix}\delta$$

纵向模型

纵向方向使用的模型很简单，可以简单的理解为一个纵向的PID, 状态方程如下

$$\frac{d}{dt}\begin{bmatrix} e_s \\ e_v \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} e_s \\ e_v \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix}u_a$$

那么该如何理解整个状态方程呢?

首先,

$$\dot{e_s} = e_v$$

这个很好理解，路径的导数等于速度, 并且该等式主要是为了能够帮助整个MPC状态方程去凑对应的标准矩阵范式而存在的，而主要需要理解的是下面的等式,

$$\dot{e_v} = -u_a$$

在这里更多的体现出来的是一种PID的思想, 我们将其离散化会更好说明这一点

$$\frac{e_v(k+1) - e_v(k)}{\Delta t} = -u_a$$

$$e_v(k) = v_k - v_{set}$$

当 $e_v(k+1) - e_v(k) == v_{k+1} - v_k$ > 0的时候，说明此时的速度误差在正向的变大，也就是我们需要控制器施加一个负向的控制力，使得速度误差降下来，反之 则需要施加一个正向的控制力，使得我们的速度增加，从而降低我们的速度误差, 从而最终实现我们的一个控制效果。当然我们这里只是使用了一种非常简陋的纵向建模，但是整体上MPC的状态方程原理是这样的, 也能在我们的作业示例中获得一个可视化的demo效果。

MPC约束和代价函数

代价函数

$$\sum_{i=0}^{n-1} x^TQx + u^TRu$$

我相信看到这里很多同学会疑惑为什么前面的$x_N^T Q x_N$ 没了,因为这一项是常数，在我们对cost_function求最值的时候，不会影响到我们的结果,所以代价函数如上，且在具体实现mpc代码的时候，也只要看上述的公式即可

约束

等式约束

$$x_{k+1} = ((I + TA/2)(I - TA/2) - 1)x_k + TBu_k$$

当然在这里我们使用的只是其中的一种模型离散化的方法，大家可以尝试使用其他的方法。

不等式约束

主要由变量约束组成

X约束

$$-Max <= e_d <= Max$$
$$-Max <= \dot{e_d} <= Max$$
$$-PI <= \psi <= PI$$
$$-Max <= \dot{\psi} <= Max$$
$$-Max <= s <= Max$$
$$-Max <= v <= Max$$

U约束

$$-PI/6 <= \delta <= PI/6$$
$$-acclimt <= u_a <= acclimt$$

这里的acclimt可以你自己来定义大小