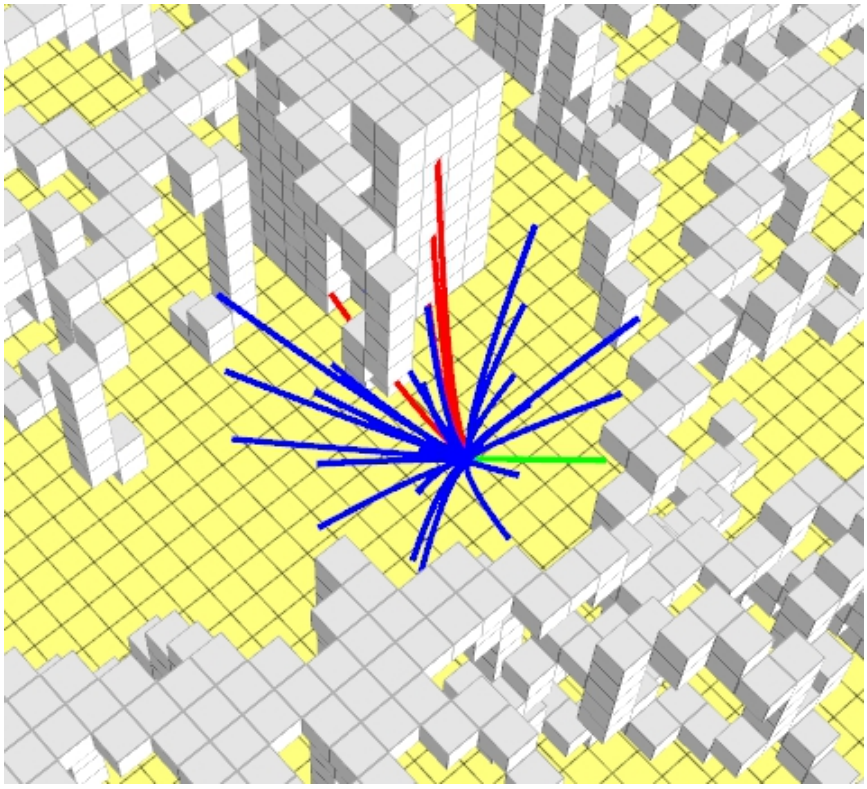
HomeWork 4

一、成果展示



二、公式推导

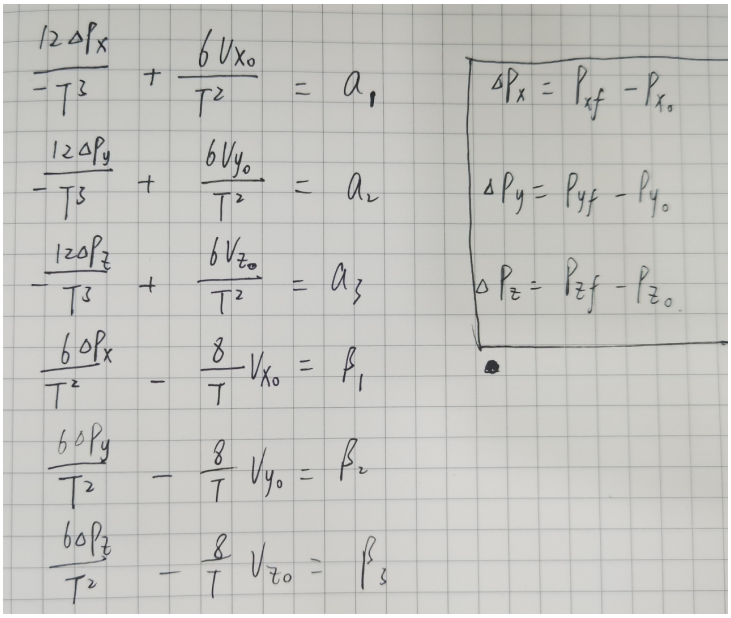
$$J = \int_0^T (1 + a_x^2 + a_y^2 + a_z^2) dt$$

$$J = T + \left(\frac{1}{3}{\alpha_1}^2 T^3 + \alpha_1\beta_1 T^2 + {\beta_1}^2 T\right) + \left(\frac{1}{3}{\alpha_2}^2 T^3 + \alpha_2\beta_2 T^2 + {\beta_2}^2 T\right) + \left(\frac{1}{3}{\alpha_3}^2 T^3 + \alpha_3\beta_3 T^2 + {\beta_3}^2 T\right)$$

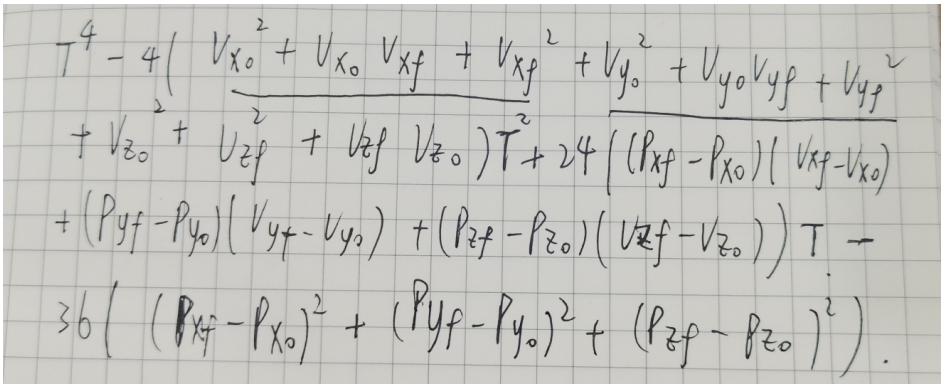
$$\begin{pmatrix} \Delta p_x \\ \Delta p_y \\ \Delta p_z \\ \Delta v_x \\ \Delta v_y \\ \Delta v_z \end{pmatrix} = \begin{pmatrix} p_{xf} - v_{x0}T - p_{x0} \\ p_{yf} - v_{y0}T - p_{y0} \\ p_{zf} - v_{x0}T - p_{z0} \\ v_{xf} - v_{x0} \\ v_{yf} - v_{y0} \\ v_{zf} - v_{z0} \end{pmatrix}$$

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} -\frac{12}{T^3} & 0 & 0 & \frac{6}{T^2} & 0 & 0 \\ 0 & -\frac{12}{T^3} & 0 & 0 & \frac{6}{T^2} & 0 \\ 0 & 0 & -\frac{12}{T^3} & 0 & 0 & \frac{6}{T^2} \\ \frac{6}{T^2} & 0 & 0 & -\frac{2}{T} & 0 & 0 \\ 0 & \frac{6}{T^2} & 0 & 0 & -\frac{2}{T} & 0 \\ 0 & 0 & \frac{6}{T^2} & 0 & 0 & -\frac{2}{T} \end{pmatrix} \begin{pmatrix} \Delta p_x \\ \Delta p_y \\ \Delta p_z \\ \Delta v_x \\ \Delta v_y \\ \Delta v_z \end{pmatrix}$$

由上式可推出：



α和β是 T 的函数，将其带入 J 中，得出 J 是只与 T 有关的函数。为了令 J 最小，对 T 求导，让求导表达式=0，得到如下多项式



多项式求解采用伴随矩阵求特征值法：

In linear algebra, the Frobenius **companion matrix** of the monic polynomial

$$p(t) = c_0 + c_1 t + \cdots + c_{n-1} t^{n-1} + t^n ,$$

is the square matrix defined as

$$C(p) = \begin{bmatrix} 0 & 0 & \dots & 0 & -c_0 \\ 1 & 0 & \dots & 0 & -c_1 \\ 0 & 1 & \dots & 0 & -c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -c_{n-1} \end{bmatrix}.$$

对矩阵求解特征值，然后将结果带入 J 中。

三、代码分析

1. forward integration 前向积分

```
*/
//pos vel delta

pos(0) = pos(0)+vel(0)*delta_time+0.5*acc_input(0)*delta_time*delta_time;  // x=x0+vt+0.5at^2
pos(1) = pos(1)+vel(1)*delta_time+0.5*acc_input(1)*delta_time*delta_time;
pos(2) = pos(2)+vel(2)*delta_time+0.5*acc_input(2)*delta_time*delta_time;
vel(0) = vel(0)+acc_input(0)*delta_time;   //v=v0+at
vel(1) = vel(1)+acc_input(1)*delta_time;
vel(2) = vel(2)+acc_input(2)*delta_time;


Position.push_back(pos);
Velocity.push_back(vel);
```

前向积分公式为 $V=V_0+at$ 和 $X=X_0+V_0t+0.5at^2$

2. OBVP

```
    */
    //初始的位置和速度
    double p_x0=_start_position(0);
    double p_y0=_start_position(1);
    double p_z0=_start_position(2);
    double v_x0=_start_velocity(0);
    double v_y0=_start_velocity(1);
    double v_z0=_start_velocity(2);
   //终点的位置和速度
    double p_xf=_target_position(0);
    double p_yf=_target_position(1);
    double p_zf=_target_position(2);
    double v_xf=0 ,v_yf=0 ,v_zf=0;
```

设置初始位置和速度，终点位置和速度，终点速度指定为 0，加速度为自由量。

```
Eigen::Matrix<double,4,4> m;

double c0 = -36*( (p_xf-p_x0)*(p_xf-p_x0) + (p_yf-p_y0)*(p_yf-p_y0) + (p_zf-p_z0)*(p_zf-p_z0)) ;
double c1 = 24*((p_xf-p_x0)*(v_xf+v_x0) + (p_yf-p_y0)*(v_yf+v_y0) + (p_zf-p_z0)*(v_zf+v_z0));
double c2 = -4*(v_x0*v_x0 + v_x0*v_xf + v_xf*v_xf + v_y0*v_y0 + v_y0*v_yf + v_yf*v_yf + v_z0*v_z0 + v_z0*v_zf + v_zf*v_zf);
double c3 = 0.0;

m << 0,0,0,-c0,
     1,0,0,-c1,
     0,1,0,-c2,
     0,0,1,c3;

double J;

Eigen::Matrix<complex<double>,Eigen::Dynamic,Eigen::Dynamic> eigenValules;
eigenValules = m.eigenvalues(); //返回特征值
```

多项式求根采用伴随矩阵求特征值方法，损失函数 J 是关于 T 的函数，所以求 J 关于 T 的导数，令其为 0，得到 T，在带入 J 中得出 J 的最小值

```
for(int i=0; i<4; ++i){
    double T = std::real(eigenValules(i)); //返回实数部分
    double img = std::imag(eigenValules(i));//返回虚数部分
     //对T进行筛选，不要负数和虚数
    if(T < 0 || std::abs(img) >= 0){
        continue;
    }

    Eigen::Vector3d alpha,beta;

    alpha(0)=-12*(p_xf-p_x0)/T/T/T + 6*v_x0/T/T;
    alpha(1)=-12*(p_yf-p_y0)/T/T/T + 6*v_y0/T/T;
    alpha(2)=-12*(p_zf-p_z0)/T/T/T + 6*v_z0/T/T;

    beta(0)=6*(p_xf-p_x0)/T/T - 8*v_x0/T;
    beta(1)=6*(p_yf-p_y0)/T/T - 8*v_y0/T;
    beta(2)=6*(p_zf-p_z0)/T/T - 8*v_z0/T;

    J=T+(1/3*alpha(0)*alpha(0)*pow(T,3)+alpha(0)*beta(0)*T*T+beta(0)*beta(0)*T)
       +(1/3*alpha(1)*alpha(1)*pow(T,3)+alpha(1)*beta(1)*T*T+beta(1)*beta(1)*T)
       +(1/3*alpha(2)*alpha(2)*pow(T,3)+alpha(2)*beta(2)*T*T+beta(2)*beta(2)*T);
```

将α和β带入求 J。