kalman测试样例

做简单的卡尔曼测试样例

```
//测试卡尔曼滤波器
#include <iostream>
#include <Eigen/Dense>

//构造卡尔曼滤波器类
class KalmanFilter
{
public:
    //构造函数
    KalmanFilter(const Eigen::MatrixXd &A, const Eigen::MatrixXd &B, const
Eigen::MatrixXd &C,
                 const Eigen::MatrixXd &Q, const Eigen::MatrixXd &R, const
Eigen::MatrixXd &P0)
        : A(A), B(B), C(C), Q(Q), R(R), P(P0),
x(Eigen::VectorXd::Zero(A.rows())) {}//初始化 状态向量初始化为0
    //使用控制矩阵进行预测
    void predict(const Eigen::VectorXd &u)//u为输入控制量 外部输入
    {
        x = A * x + B * u; //先验估计值
        P = A * P * A.transpose() + Q; //先验估计协方差
    }
    //更新
    void update(const Eigen::VectorXd &z)//z为测量值 观测矩阵
    {
        Eigen::MatrixXd S = C * P * C.transpose() + R; //kalmanGain的分母
        Eigen::MatrixXd K = P * C.transpose() * S.inverse();//卡尔曼增益
        x = x + K * (z - C * x);//后验估计值
        P = (Eigen::MatrixXd::Identity(P.rows(), P.cols()) - K * C) * P;//后验估计
协方差
    }
    //获取状态向量
    Eigen::VectorXd getState() const
    {
        return x;
    }
private:
    Eigen::MatrixXd A; //状态转移矩阵
    Eigen::MatrixXd B; //控制输入矩阵
    Eigen::MatrixXd C; //观测矩阵
    Eigen::MatrixXd Q; //状态噪声协方差矩阵
    Eigen::MatrixXd R; //观测噪声协方差矩阵
    Eigen::MatrixXd P; //估计误差协方差矩阵
    Eigen::VectorXd x; //状态向量
};
//测试卡尔曼滤波器
void test()
{
    //定义卡尔曼滤波器参数
```

```
    Eigen::MatrixXd A(2, 2); //状态转移矩阵
    A << 1, 1, 0, 1;

    Eigen::MatrixXd B(2, 1); //控制输入矩阵
    B << 0.5, 1;

    Eigen::MatrixXd C(1, 2); //观测矩阵
    C << 1, 0;

    Eigen::MatrixXd Q(2, 2); //状态噪声协方差矩阵
    Q << 0.1, 0, 0, 0.1;

    Eigen::MatrixXd R(1, 1); //观测噪声协方差矩阵
    R << 1;

    Eigen::MatrixXd P0(2, 2); //估计误差协方差矩阵 初始化
    P0 << 1, 0, 0, 1;

    //创建卡尔曼滤波器对象
    KalmanFilter kf(A, B, C, Q, R, P0);

    //定义测试数据
    Eigen::VectorXd measurements(10); //包含10个数据的向量 这个可以理解为测量值 状态噪声观测
    measurements << 1.1, 1.3, 1.5, 1.7, 1.9, 2.1, 2.3, 2.5, 2.7, 2.9;

    //执行卡尔曼滤波
    for (int i = 0; i < measurements.size(); ++i)
    {
        kf.predict(Eigen::VectorXd::Zero(1));//对状态进行预测,输入控制量为0 也就是建立在不计外部控制干预的情况下
        kf.update(measurements.segment(i, 1));//对状态进行更新 i为当前的测量值 1为测量值的维度
        std::cout << "Estimated state: " << kf.getState()(0) << std::endl;//输出估计值
    }
}

int main()
{
    test();
    //随着迭代次数增加，估计值会越来越接近测量值，也就是measurements中的值
    system("pause");
    return 0;
}
```

把所有的参数值都用上了，但是这个控制输入向量并没有，就赋值为0向量了

实际中可能会有控制输入向量的，从外部输入的值会对系统造成一定影响