第四次任务提交

低通滤波器

我模拟了对匀加速直线运动的低通滤波器

```
Eigen::Matrix<T, x, 1> x0;
double variance;
double v;//速度
double a;//加速度
```

分别设置其位移和速度

```
 //匀加速直线运动模型
 theoretical[0] = v * t+0.5*a*t*t;//位置
 theoretical[1] = a * t;//速度

 noise = Eigen::Matrix<T, x, 1>::Zero();
 for (int i = 0; i < x; i++) {
     noise(i, 0) = gaussianRandom(0, this->variance);
 }
 measurement = theoretical + noise;
 return measurement;
```

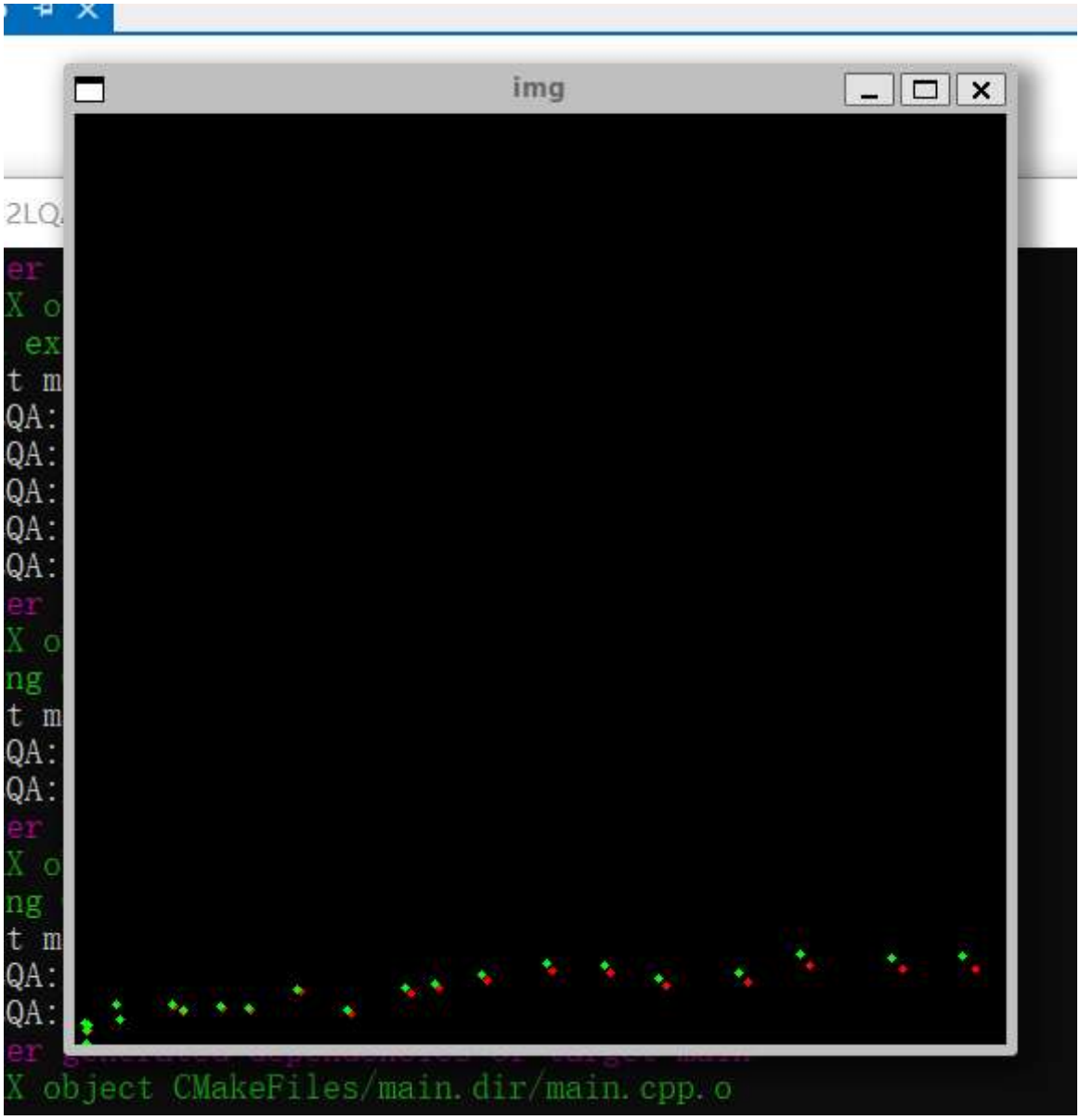
低通滤波器设置采样率和截止频率对其进行低通滤波,并实时更新

```
// 低通滤波器类
class LowPassFilter
{
public:
    LowPassFilter(double sample_rate, double cutoff_frequency)
    {
        double dt = 1.0 / sample_rate;//设置截止频率以及采样率
        double RC = 1.0 / (cutoff_frequency * 2.0 * M_PI);
        alpha_ = dt / (dt + RC);
        prev_output_ = 0.0;
    }

    // 更新滤波器输出
    double update(double input)
    {
```

```
        double output = alpha_ * input + (1.0 - alpha_) * prev_output_;
        prev_output_ = output;
        return output;
    }

private:
    double alpha_;
    double prev_output_;
};
```

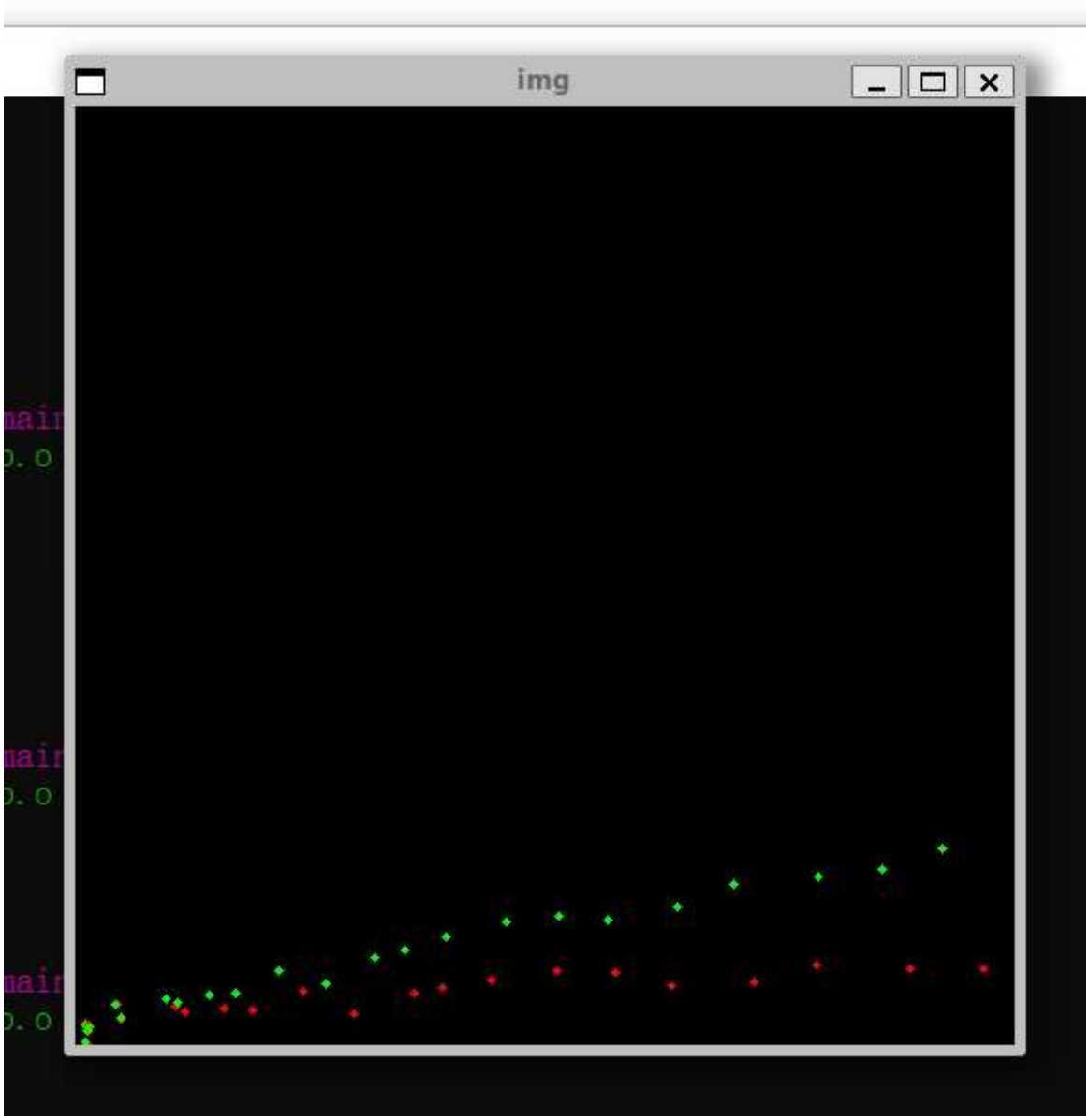
滤波结果如下

- 截止频率为1000



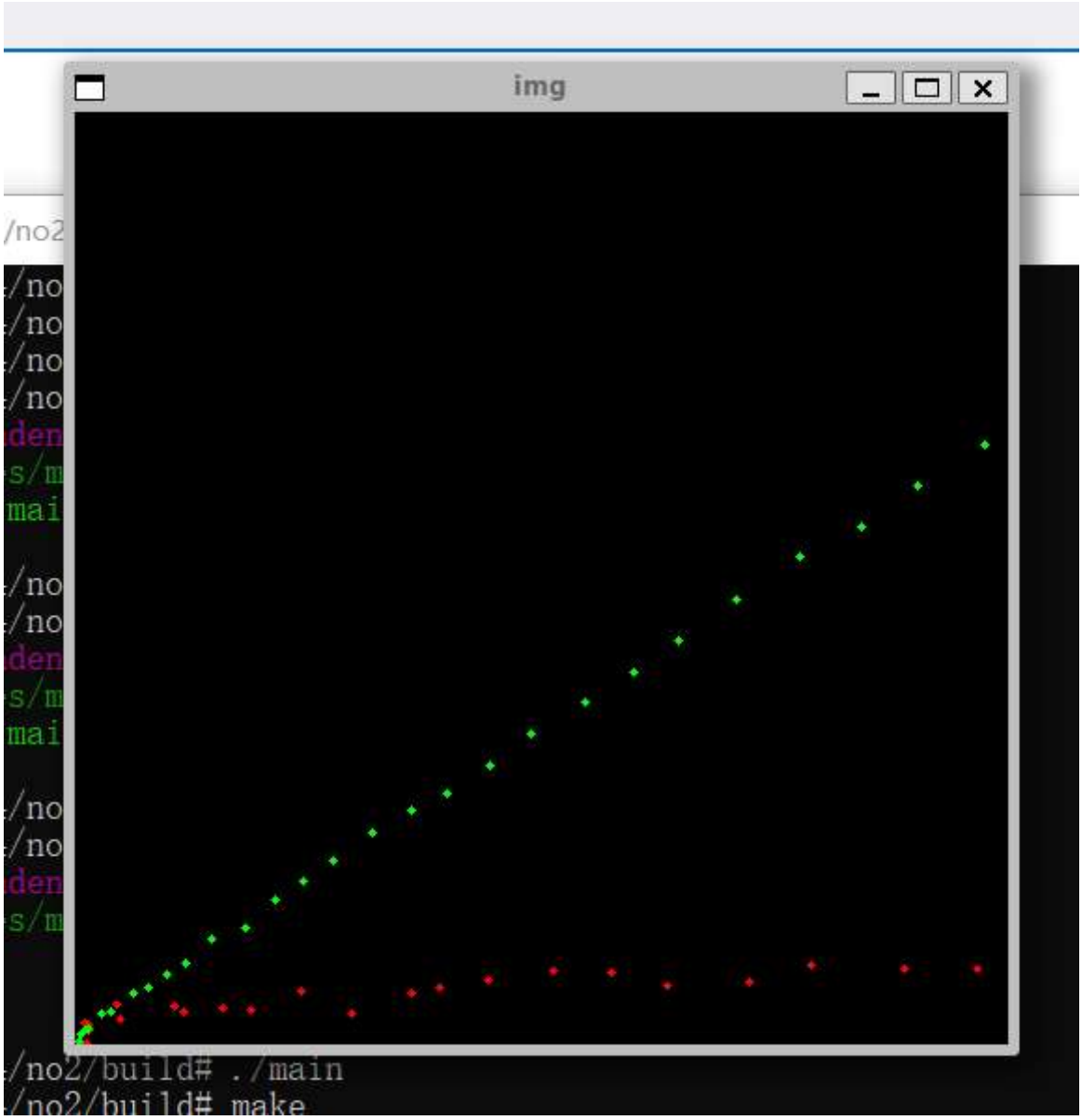
近乎拟合真实情况

- 截止频率为100



有些许失真，但是基本形状类似

- 截止频率为10



完全失真，滤去了相当大部分的噪声，基本接近匀加速直线运动模型

卡尔曼滤波

我设计了一个正弦函数的运动模拟：

```
double variance;//方差
double amplitude; // 振幅
double frequency; // 频率
double velocity; //速度
```

x轴方向为匀速直线运动，y方向作正弦函数的模拟。分别定义其振幅，频率，速度

```
public:

    Eigen::Matrix<T, x, 1> getMeasurement(double t) {
        Eigen::Matrix<T, x, 1> theoretical;
        Eigen::Matrix<T, x, 1> measurement;
```

```
        Eigen::Matrix<T, x, 1> noise;

        theoretical(0) = x0(0) + t * velocity; // x方向匀速直线运动
        theoretical(1) = x0(1) + amplitude * sin(t * frequency); // y方向正弦函数运动
        theoretical(2) = x0(2);
        theoretical(3) = x0(3);
        noise = Eigen::Matrix<T, x, 1>::Zero();
        for (int i = 0; i < x; i++) {
            noise(i, 0) = gaussianRandom(0, this->variance);
        }
        measurement = theoretical + noise;
        return measurement;
    }

    explicit Simulator(Eigen::Matrix<T, x, 1> x0, double variance, double amplitude, double
        this->x0 = x0;
        this->variance = variance;
        this->amplitude = amplitude;
        this->frequency = frequency;
        this->velocity = velocity;
    }
```

接着，初始化仿真器，并设置卡尔曼矩阵

```
// 仿真器初始化
Simulator<double, 4> *simulator;
simulator = new Simulator<double, 4>(Eigen::Vector4d(0, 0,0,0), 1,10,50,3); // 输入为起始

// 2. 设置状态转移矩阵
kf->transition_matrix << 1, 0, 1, 0,
    0, 1, 0, 1,
    0, 0, 1, 0,
    0, 0, 0, 1;

// 3. 设置测量矩阵
kf->measurement_matrix << 1, 0, 0, 0,
    0, 1, 0, 0,
    0, 0, 1, 0,
    0, 0, 0, 1;

// 4. 设置过程噪声协方差矩阵
kf->process_noise_cov << 0.01, 0, 0, 0,
    0, 0.01, 0, 0,
```

```
    0, 0, 0.01, 0,
    0, 0, 0, 0.01;

// 5. 设置测量噪声协方差矩阵
kf->measurement_noise_cov << 1, 0, 0, 0,
    0, 1, 0, 0,
    0, 0, 1, 0,
    0, 0, 0, 1;

// 6. 设置控制向量
kf->control_vector << 0, 0, 0, 0;
```

取t从0-100，展示如下的追踪结果：
绿色代表真实值，红色代表预测值