检测系统与自动化误差部分大作业

院（系）名称 自动化科学与电气工程学院

专业名称 自动化

学生学号 SY2303526

学生姓名 杨和鹭

北京航空航天大学

二〇二三年十月

目 录

1 引言

1.1 问题简介

产品定量包装广泛应用于各行各业，并借助与微机系统进行同步计量。尤其是小称量定量包装，要求计量准确度高、包装出接近或略大于标定值的最佳计算值作为物料的输出量。传统的单体称量包装秤，包装速度与精度都有很大缺陷，组合秤采用分料斗称重，并组合最优的设计，大大提高了称量的精度，料斗数一般为 10 斗、14 斗。组合秤称重时，物料通过振动从储料斗经料斗门进入各编号的称料斗，通过称重各称料斗重量，然后按照组合原理进行组合计算，每次下料都是在很多个合格组合中选择最接近目标重量的组合。但是，由于系统固有的非线性特性和系统低频元件以及工业环境中的机械振动、物料下落的冲击力、电磁辐射等干扰存在，动态称重的快速性和准确性很难同时满足高性能指标要求。如何有效抑制和降低这些干扰和噪声引起的动态测量误差，是提高动态测量性能必须面对的问题。

1.2 研究现状

经过各大网站上查找论文，发现对于动态称重问题中如何提高快速性与稳定性的研究中，常用的方法就是各种滤波算法，包括对于信号数据的时域及频域上的处理，不过这些论文都是基于已经有了一定的扰动模型，比如在对车俩的动态秤重过程中，对车辆行驶到

地秤的过程中产生的冲击、电机的扰动建模，这样通过建立好的模型就可以更好滤除掉原信号中的各种扰动，得到更接近理论真值的动态测量数据，以保证准确性与快速性。

但在研究本问题的过程中，由于我自身对于电机产生的各种电磁扰动、豆子下坠对秤的机械元件产生冲击等过程的知识过于匮乏，因此无法对冲击模型进行抽象，建立模型（我看给的两篇论文也没有对产生扰动的根源进行建模，只进行数据处理），因此只能从数据本身入手，提取出数据中扰动因素的特征，在所给数据中滤掉这一部分，得到相对准确的实验数据。

1.3 技术路线

先对实验数据进行数据处理，去除粗大误差等强扰动因素。利用滤波算法对数据进行初步滤波，得到滤波后的数据，此时这些滤波后的数据已经有了一部分的稳定性。为了进一步提升快速性并进一步提升稳定性，利用这些滤波数据制作标签值，利用 BP 是神经网络拟合一个滤波函数，输入数据是这些实验数据，期望输出数据是标签值。这样，只要输入数据是测得的实验数据（不需要去除粗大误差），把数据作为神经网络的输入，就可以利用神经网络拟合滤波函数 f（t），得到期望的目标数据，同时目标数据具有更好的快速性与稳定性。

1.4 软件系统

使用的软件：

- MWORKS.syslab 2023a
- Pyhton 解释器 3.11

使用的编程语言：

- Julia 脚本语言
- Python

使用到的库：

Keras、matplotlib 等

训练环境：

GPU	NVIDIA RTX 4070
CPU	Intel(R) Core(TM) i5-12600K CPU @ 3.70GHz
内存	16G * 2

1.5 文档结构

本报告的第一部分是引言，主要介绍了问题的引出与期望的实现效果。

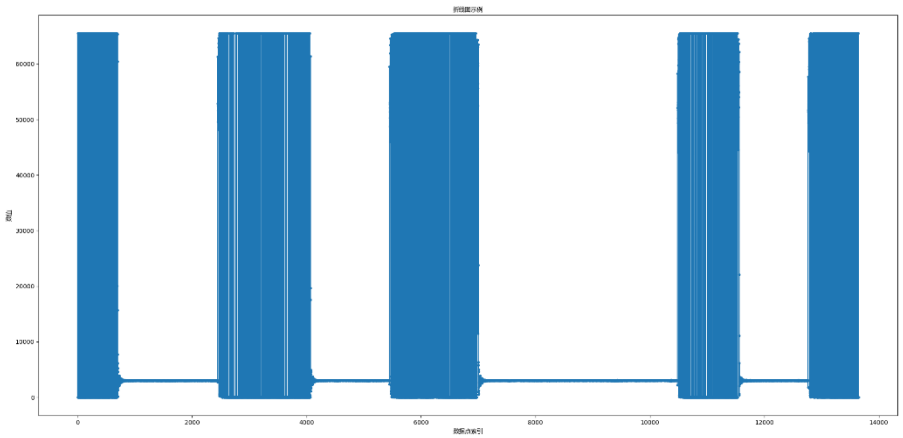
本文档的第二部分是具体的实现过程，包括数据的处理过程，滤波算法的使用，神经网络的搭建与训练效果。

本文档的第三部分是结论与感悟。

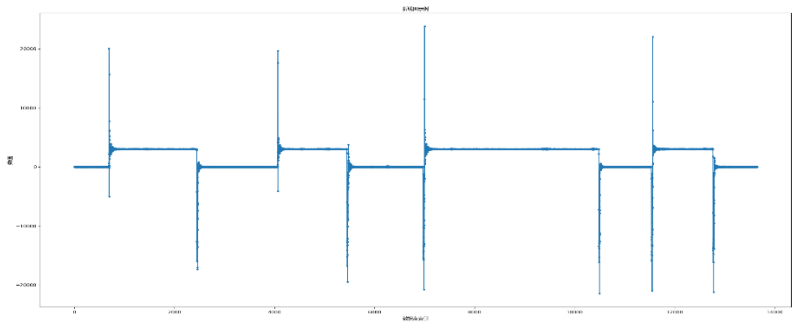
2 软件系统设计

2.1 数据预处理

通过给出的 300g 豆子的实验数据，可以绘制出最初始的实验数据，利用 MWORKS.sysploer 绘制如下：

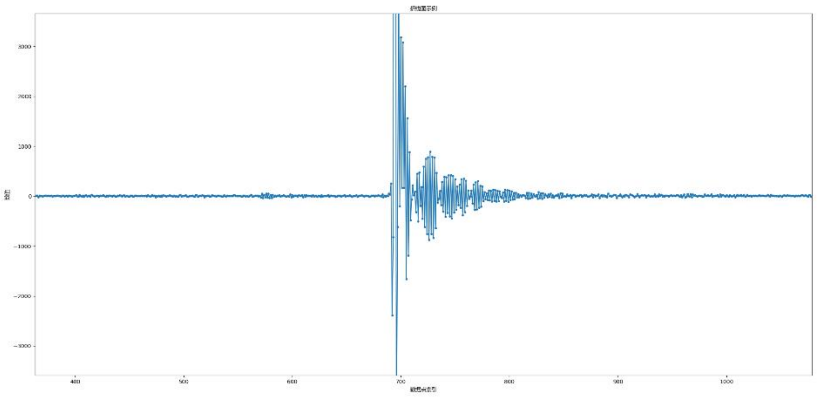


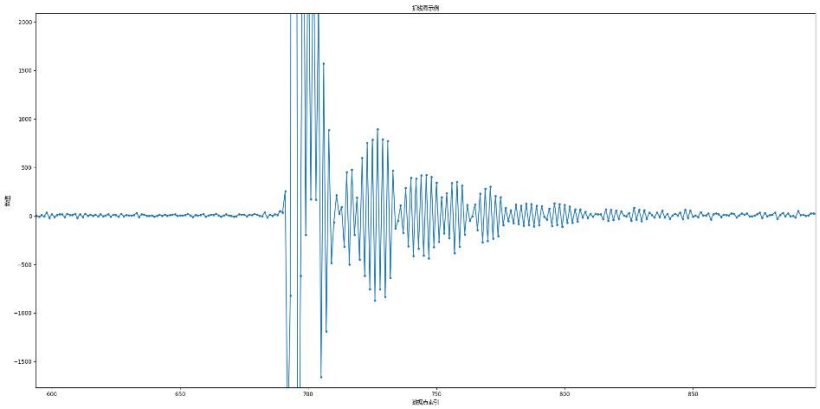
显然数据中 FFXX 这种数据是不正确的，原始数据中掺杂了大量 FFXX 这种十六进制数字，通过仔细观察 FFXX 这种十六进制数字两边的十六进制数，会发现这种数据的某一边的正十六进制数字总是接近于 0，因此猜测所有的初始数据都是一种基于计算机补码的表示形式（0+表达为十六进制为 00XX，0-表达为 FFXX）。因此将其转化为补码重新绘制如下图：



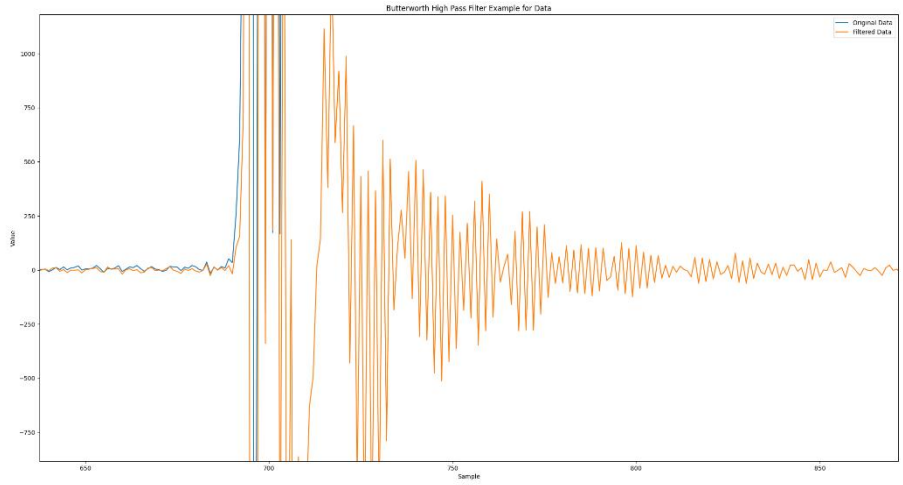
为了进一步证实原始数据是基于补码表示的，这里用两种方式求取误差数据：

1、用处理好的补码数据减去标签值（期望的测量真值），标签值设定为数据范围在负无穷到 300 之间的数据重定义为 0，剩余数据定义为 3000，得到的纯误差数据及其放大的图片如下图：





2、利用高通滤波器滤除掉真实的稳定测量数据（相比高频系统误差，测量误差算是低频），这里选用巴特沃斯滤波器，阶数为 10 阶，滤除的低频截至频率为10Hz，采样点频率为60Hz，得到的干扰误差图像如下所示：



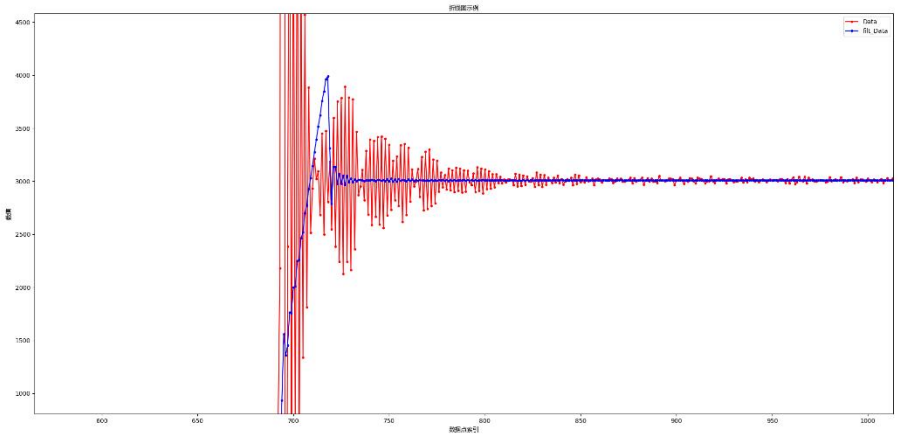
通过以上两种方法可以证明误差信号的基本震荡特征，也能够说明数据的编码形式是补码输出。同时也能说明：相对期望测量的数据来说，误差信号是高频信号，因此因该采用某种低频滤波器来滤波。

上图仅截取了一个方波上升沿产生的误差，虽然是近似得到的结果，但很明显原始的实验数据都应该当作补码处理，而不是将负数数据直接归零，FFXX 这类十六进制补码转化为数字就是接近于 0 的负数，这样处理下来能发现数据具有很明显的震荡特性，因此把初始数据中的负值按归零处理显然不是很合理的，这样就相当于人为去掉了一半的误差特征，即相当于把在 0 附近震荡的系统误差中的一半负值误差归零了，这样处理得到的结果也肯定是欠缺准确性的。

2.2 滤波算法设计

为了使原始数据获得稳定性，这里采用限幅-滑动滤波的方法，滑动滤波的方法实际上相当于低通滤波，观察上面的纯误差曲线可以看出产生的主要扰动就是一些高频信号，因此限幅滑动滤波会有效抑制。

这里滑动窗口设定为20，可以得到滑动均值处理得出的数据效果：



可以看到经过滑动滤波的曲线很好地滤除了高频信号，但同样损失了快速性，为了恢复系统的快速性，这里打算利用 BP 神经网络拟合整个滤波函数，同时被拟合的滤波函数除了要拥有上方传统滑动平均滤波函数的稳定性，还要具备一定的快速性。

2.3 神经网络设计

这里利用神经网络来拟合整个滤波函数，由于滑动平均滤波使得得到的结果损失了一部分快速性，因此以实验数据作为网络输入，已处理过的标签值作为网络反向迭代的训练集来训练网络。

F(x1,x2,x3) = 滑动滤波器的值。

在训练神经网络的过程中，利用“300g 蚕豆”这组数据时域上连续的 n 个实验数据$(x_{t-1}, x_{t-2}, x_{t-3}...x_{t-n})$作为 BP 神经网络的输入，经过隐藏层的运算在输出层会输出一个预测值 y_t，这个预测值就是神经网络提取前 n 个测量点的特征预测得到的。因此在训练神经网络时，神经网络的输入是时域上连续的 n 个测量数据，神经网络所拟合的函数可以写成：

$$f(x_{t-1}, x_{t-2}, x_{t-3}....x_{t-n}) = y_t \text{。}$$

构建数据集：为了提高整个系统的快速性，我将上一节滤波算法的数据做一些处理成为训练网络所用的数据，将所有实验数据以每五个时域上连续的数据作为 BP 神经网络一次训练的输入，即由原来的：

$$[x_1, x_2, x_3, x_4, x_5, x_6...]$$

每五个一组：

$$\left[[x_1, x_2, x_3 \ldots x_m], [x_2, x_3, x_4 \ldots x_{m+1}], [x_3, x_4, x_5 \ldots x_{m+2}] \ldots\right]$$

每个 m 元组都会对应一个标签值，“m 元组”+“标签值”构成了一个训练数据。标签值的构建需要用到上面滑动滤波得到的结果，设时间 t 的标签值为s_t，这样利用$s_t - y_t$反向迭代，训练出来的 BP 网络就会继承滑动滤波的稳定性。

由于滑动滤波在滤波的过程中损失了一部分的快速性，因此对标签值做进一步处理，以提高神经网络的快速性。

这里对滑动平均滤波的结果也做 m 元组分割，$[s_1, s_2, s_3 \ldots]$为 3.2 节中的滑动平均滤波得到的结果：

$$[[s_1, s_2, s_3, s_4, \ldots s_m], [s_2, s_3, s_4, s_5 \ldots s_{m+1}] \ldots.],$$

并且为每个元组中的数据乘以系数$(k_1, k_2, k_3, k_4 \ldots . k_m)$，其中

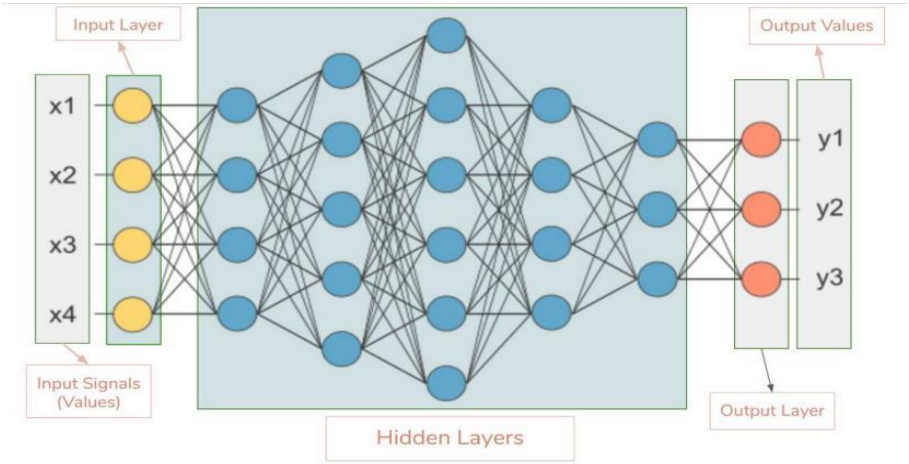
$$k_1 + k_2 + k_3 + \ldots k_m = 1$$

$$k_m > \ldots k_2 > k_1 > 0$$

元组中越后面的权重越大，标签数据

$$z_t = k_1 * s_{t-\frac{m}{2}} + k_2 * s_{t-1} + \ldots k_4 * s_{t+1} + k_m * s_{t+\frac{m}{2}}$$

，每个标签值z_t都是由 t 之前几个时域点和 t 之后的几个时域点乘系数加权得到，即$\varphi(s_{t-\frac{m}{2}}, \ldots s_{t-1}, s_t \ldots . s_{t+\frac{m}{2}}) = z_t$，并且时域上越靠后的点权重越大，因此可以显著提高快速性。



这样就构成了形如$[[[x_1,x_2,x_3,x_4,x_5],z_1],[[x_2,x_3,x_4,x_5,x_6],z_2]....]$的训练集，每组数据都可以作为神经网络每一次迭代的输入数据和反向传递的标签值。将输入数据的运算结果减标签值进行反向迭代。在实际训练的过程中将整个数据集做归一化并打乱，50%作为训练集，50%作为测试集，迭代次数为 1000 次进行训练。最终得到的结果如下：

可以看到，训练结果具有有效地提升了快速性与稳定性，在训练过程中的 loss 函数变化如下图：

3 硬件系统设计

3.1 系统硬件功能结构组成