数字滤波器

心电波形数据

本文中使用到的心电波形数据如下所示。

```
/*
 * 心电波形数据
 * 采样率 2kHz
 * 脉率值 60BPM
 * 单位 mV
 */
const double g_arrEcgWave[4000] =
{
-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
-0.06,-0.06,-0.06,-0.06,-0.05,-0.06,-0.05,-0.06,-0.06,-0.06,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,
-0.06,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.06,-0.06,-0.05,-0.06,-0.06,-0.05,-0.06,-0.06,-0.06,-0.06,-0.05,
-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.05,
-0.06,-0.06,-0.06,-0.06,-0.05,-0.06,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,
-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,
-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,
-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,
-0.05,-0.05,-0.06,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,
-0.06,-0.05,-0.06,-0.06,-0.06,-0.06,-0.05,-0.06,-0.06,-0.06,-0.05,-0.06,-0.06,-0.05,-0.06,-0.05,-0.05,-0.05,
```

```
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.06,-0.06,-0.06,-0.06,
-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.06,-0.06,
-0.06,-0.06,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.06,
-0.06,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
-0.06,-0.06,-0.06,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.06,-0.06,-0.06,-0.05,-0.05,-0.06,-0.06,
-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.06,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,
-0.06,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,
-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.06,-0.05,-0.06,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,
-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.05,-0.05,-0.06,-0.06,
-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
```

```
-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.04,-0.04,-0.04,-0.04,-0.04,-0.03,-0.03,-0.02,-0.02,-0.03,
-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.01,-0.01,-0.01,-0.01,0,0,0,0,0,0,0,0,0.01,0.01,0.01,0.01,0.01,0.01,
0.01,0.01,0.01,0.01,0.01,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.03,0.03,0.02,0.03,0.03,0.03,0.03,0.03,
0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,
0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.02,0.02,0.03,0.03,0.03,0.03,0.02,
0.02,0.02,0.02,0.02,0.02,0.02,0.01,0.02,0.02,0.02,0.02,0.01,0.01,0.01,0.01,0.01,0.01,0,0,0.01,0,0,0.01,0,0,0,
0,-0.01,-0.01,-0.01,-0.01,-0.01,-0.01,-0.02,-0.02,-0.02,-0.02,-0.03,-0.03,-0.02,-0.03,-0.03,-0.03,-0.03,-0.04,
-0.04,-0.04,-0.04,-0.04,-0.04,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,
-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,
-0.05,-0.06,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.06,-0.07,
-0.08,-0.08,-0.09,-0.1,-0.1,-0.1,-0.11,-0.12,-0.12,-0.12,-0.12,-0.14,-0.14,-0.14,-0.15,-0.16,-0.17,-0.17,-0.17,
-0.19,-0.19,-0.19,-0.17,-0.13,-0.12,-0.12,-0.09,-0.05,-0.05,-0.05,-0.04,0,0.02,0.02,0.03,0.08,0.09,0.09,0.11,0.15,
0.16,0.16,0.19,0.23,0.23,0.23,0.26,0.3,0.3,0.3,0.31,0.35,0.37,0.37,0.39,0.43,0.44,0.44,0.45,0.49,0.51,0.51,0.53,
0.57,0.58,0.58,0.61,0.64,0.64,0.64,0.65,0.64,0.64,0.64,0.65,0.65,0.65,0.64,0.63,0.58,0.57,0.56,0.53,0.49,0.48,0.49,
0.47,0.42,0.4,0.4,0.38,0.34,0.32,0.32,0.3,0.26,0.24,0.24,0.22,0.17,0.16,0.15,0.12,0.08,0.07,0.07,0.04,0,-0.01,-0.01,
-0.02,-0.07,-0.09,-0.09,-0.11,-0.16,-0.17,-0.17,-0.18,-0.24,-0.25,-0.25,-0.28,-0.33,-0.33,-0.33,-0.32,-0.29,-0.28,
```

```
-0.29,-0.28,-0.25,-0.23,-0.24,-0.23,-0.2,-0.2,-0.19,-0.18,-0.16,-0.15,-0.14,-0.13,-0.1,-0.1,-0.1,-0.08,-0.05,-0.06,
-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.06,-0.06,-0.06,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.06,
-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.04,
-0.05,-0.05,-0.05,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.05,-0.05,-0.05,-0.04,-0.04,
-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,
-0.04,-0.04,-0.04,-0.04,-0.04,-0.03,-0.03,-0.04,-0.04,-0.04,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,
-0.03,-0.03,-0.03,-0.02,-0.02,-0.02,-0.03,-0.03,-0.03,-0.02,-0.02,-0.02,-0.03,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,
-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.01,-0.01,-0.02,-0.02,-0.02,-0.01,-0.01,-0.02,-0.02,-0.01,
-0.02,-0.01,-0.01,-0.01,-0.01,-0.01,-0.01,-0.01,0,-0.01,-0.01,-0.01,-0.01,-0.01,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.02,0.02,0.02,0.02,
0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.03,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.03,0.03,0.03,0.03,
0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.03,0.04,0.04,
0.04,0.04,0.05,0.05,0.05,0.04,0.04,0.04,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,
0.05,0.06,0.06,0.06,0.05,0.06,0.06,0.06,0.05,0.05,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,
0.06,0.06,0.07,0.07,0.07,0.07,0.07,0.07,0.07,0.07,0.07,0.07,0.08,0.07,0.07,0.07,0.07,0.08,0.08,0.08,0.08,0.08,
0.08,0.08,0.08,0.08,0.08,0.08,0.08,0.08,0.08,0.08,0.08,0.08,0.08,0.08,0.09,0.09,0.08,0.09,0.08,0.08,0.08,0.08,
0.08,0.08,0.08,0.09,0.09,0.09,0.09,0.09,0.09,0.09,0.09,0.1,0.1,0.09,0.1,0.1,0.1,0.1,0.1,0.1,0.11,0.11,0.11,
0.11,0.11,0.11,0.11,0.11,0.12,0.12,0.12,0.12,0.12,0.13,0.13,0.13,0.13,0.13,0.14,0.14,0.14,0.14,0.14,0.14,0.14,0.14,
0.15,0.15,0.15,0.15,0.15,0.15,0.15,0.15,0.15,0.15,0.15,0.15,0.15,0.15,0.15,0.16,0.16,0.16,0.16,0.16,0.16,0.16,0.16,
0.16,0.16,0.16,0.16,0.16,0.16,0.17,0.17,0.17,0.17,0.16,0.16,0.16,0.16,0.16,0.16,0.16,0.16,0.16,0.15,0.15,0.15,0.15,
0.15,0.15,0.15,0.15,0.14,0.14,0.14,0.14,0.13,0.14,0.13,0.13,0.13,0.13,0.13,0.13,0.12,0.12,0.12,0.12,0.12,0.12,0.12,
0.12,0.12,0.12,0.11,0.12,0.12,0.11,0.11,0.11,0.11,0.11,0.11,0.11,0.11,0.11,0.11,0.11,0.11,0.11,0.11,0.11,0.11,0.11,
```

```
 92 0.1,0.1,0.1,0.1,0.1,0.09,0.1,0.09,0.08,0.08,0.08,0.08,0.07,0.07,0.06,0.06,0.06,
    0.05,0.05,0.05,0.04,0.04,0.04,
 93 0.03,0.03,0.03,0.03,0.02,0.01,0.01,0.01,0.01,0,0,0,0,-0.01,-0.02,-0.02,-0.02,-0
    .02,-0.02,-0.02,-0.02,-0.02,-0.02,
 94 -0.03,-0.03,-0.03,-0.03,-0.03,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-
    0.04,-0.05,-0.05,-0.04,-0.05,-0.05,
 95 -0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.05,-0.05,-
    0.06,-0.05,-0.05,-0.06,-0.06,-0.06,
 96 -0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-
    0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
 97 -0.06,-0.05,-0.06,-0.06,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-
    0.05,-0.05,-0.05,-0.05,-0.06,-0.06,
 98 -0.06,-0.06,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.06,-0.05,-0.06,-0.06,-0.06,-
    0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
 99 -0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-
    0.05,-0.05,-0.05,-0.05,-0.06,-0.06,
100 -0.06,-0.05,-0.05,-0.05,-0.06,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.05,-
    0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
101 -0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-
    0.06,-0.05,-0.05,-0.05,-0.05,-0.05,
102 -0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-
    0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
103 -0.06,-0.06,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
104 -0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-
    0.05,-0.06,-0.06,-0.06,-0.06,-0.06,
105 -0.06,-0.06,-0.06,-0.06,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-
    0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
106 -0.05,-0.06,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-
    0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
107 -0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.06,-
    0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
108 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-
    0.06,-0.06,-0.05,-0.06,-0.06,-0.06,
109 -0.06,-0.06,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.06,-0.06,-0.06,-
    0.06,-0.06,-0.06,-0.05,-0.05,-0.05,
110 -0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.06,
111 -0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.06,-
    0.06,-0.05,-0.05,-0.06,-0.05,-0.06,
112 -0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.05,-
    0.06,-0.06,-0.06,-0.06,-0.06,-0.06,
113 -0.06,-0.06,-0.05,-0.06,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.06,-
    0.06,-0.06,-0.05,-0.05,-0.06,-0.06,
114 -0.06,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
```

```
115 -0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.05,-0.06,-0.06,-0.05,-0.06,-
    0.06,-0.06,-0.06,-0.05,-0.06,-0.06,
116 -0.06,-0.05,-0.06,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.06,-0.05,
117 -0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.06,-0.05,-0.06,-0.05,-0.05,-0.06,-
    0.06,-0.06,-0.05,-0.05,-0.05,-0.05,
118 -0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
119 -0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-
    0.05,-0.06,-0.06,-0.05,-0.06,-0.05,
120 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
121 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.06,-0.05,-0.05,-0.05,-0.05,-0.05,
122 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
123 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
124 -0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
125 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
126 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
127 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
128 -0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
129 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.06,-0.05,-0.05,-0.05,
130 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.06,-0.06,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
131 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
132 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
133 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
134 -0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
135 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
136 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
137 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.04,-0.05,-0.05,-0.05,-0.05,
```

```
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.04,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.04,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.04,-0.04,-0.04,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.04,-0.04,-0.04,-0.04,-0.04,-0.03,-0.03,
-0.03,-0.03,-0.03,-0.03,-0.02,-0.03,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.01,-0.01,-0.01,-0.01,0,0,0,0,0.01,0.01,
0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,
0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.04,0.04,0.04,0.03,0.03,0.03,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.03,0.03,0.04,
0.04,0.04,0.04,0.04,0.04,0.03,0.03,0.04,0.04,0.04,0.04,0.04,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.04,0.03,0.03,0.03,
0.03,0.03,0.03,0.03,0.03,0.02,0.02,0.02,0.03,0.02,0.02,0.03,0.03,0.02,0.02,0.02,0.02,0.02,0.01,0.02,0.01,0.01,0.01,
0.01,0.01,0.01,0.01,0.01,0.01,0,0,0,0,0,-0.01,-0.01,-0.01,-0.01,-0.01,-0.01,-0.01,-0.02,-0.02,-0.02,-0.02,-0.02,
-0.02,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.04,-0.04,-0.04,-0.04,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.04,-0.04,-0.04,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.04,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.04,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.04,-0.04,-0.05,-0.04,-0.05,-0.05,-0.05,-0.04,
-0.04,-0.05,-0.05,-0.05,-0.05,-0.04,-0.05,-0.04,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.04,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
```

```
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.04,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.07,-0.07,-0.07,-0.08,-0.09,-0.09,
-0.1,-0.1,-0.11,-0.12,-0.12,-0.12,-0.13,-0.14,-0.14,-0.14,-0.15,-0.16,-0.17,-0.17,-0.19,-0.19,-0.19,-0.17,-0.13,
-0.12,-0.12,-0.11,-0.06,-0.04,-0.04,-0.03,0,0.02,0.02,0.03,0.06,0.09,0.1,0.11,0.15,0.16,0.16,0.18,0.22,0.23,0.23,
0.24,0.28,0.3,0.3,0.31,0.35,0.37,0.37,0.38,0.42,0.44,0.44,0.46,0.5,0.51,0.51,0.54,0.57,0.58,0.58,0.59,0.63,0.65,
0.65,0.65,0.65,0.65,0.65,0.65,0.65,0.65,0.65,0.63,0.58,0.57,0.57,0.54,0.5,0.48,0.49,0.48,0.44,0.41,0.4,0.39,0.35,
0.33,0.33,0.32,0.27,0.24,0.24,0.22,0.18,0.16,0.16,0.14,0.09,0.08,0.08,0.07,0.02,-0.01,0,-0.02,-0.06,-0.09,-0.09,
-0.1,-0.14,-0.17,-0.17,-0.19,-0.24,-0.25,-0.25,-0.28,-0.33,-0.33,-0.33,-0.32,-0.3,-0.29,-0.28,-0.28,-0.25,-0.24,
-0.24,-0.24,-0.21,-0.19,-0.19,-0.18,-0.15,-0.14,-0.14,-0.13,-0.1,-0.1,-0.09,-0.09,-0.06,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.04,-0.04,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.04,-0.05,-0.05,-0.05,-0.05,-0.04,-0.04,-0.05,-0.05,-0.05,-0.05,-0.04,-0.04,-0.04,-0.04,
-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.04,-0.03,-0.03,-0.04,-0.04,
-0.04,-0.04,-0.04,-0.03,-0.03,-0.04,-0.04,-0.04,-0.03,-0.03,-0.03,-0.03,-0.03,-0.04,-0.04,-0.03,-0.04,-0.04,-0.03,
-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.02,-0.02,-0.02,-0.03,
-0.03,-0.02,-0.03,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,
-0.02,-0.02,-0.02,-0.01,-0.01,-0.01,-0.01,-0.01,-0.01,-0.01,-0.01,-0.01,-0.01,-0.01,-0.01,-0.01,-0.01,0,0,0,
-0.01,0,0,0,-0.01,0,0,0,0,0,0,0,0,0,0.01,0.01,0,0,0,0.01,0,0.01,0.01,0.01,0,0.01,0.01,0.01,0.01,0.01,0.01,0.01,
0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.02,
0.02,0.02,0.02,0.02,0.02,0.02,0.02,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.03,0.04,0.04,0.04,
0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.05,0.05,0.05,0.04,
0.04,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.06,0.05,0.05,0.05,0.06,0.06,0.06,0.06,0.06,0.06,0.05,0.06,
0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.07,0.07,0.06,0.07,0.06,0.06,0.07,0.07,0.07,0.07,
```

```
184 0.07,0.07,0.08,0.07,0.07,0.07,0.08,0.08,0.08,0.08,0.08,0.08,0.08,0.08,0.09,0.08
    ,0.08,0.08,0.08,0.09,0.08,0.08,
185 0.08,0.09,0.09,0.09,0.08,0.08,0.09,0.08,0.08,0.09,0.09,0.09,0.09,0.09,0.09,0.09
    ,0.09,0.09,0.09,0.09,0.09,0.1,
186 0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.11,0.11,0.11,0.11,0.11,0.11,0
    .11,0.12,0.12,0.12,0.13,0.12,
187 0.12,0.13,0.13,0.13,0.13,0.13,0.13,0.13,0.13,0.13,0.14,0.14,0.14,0.14,0.15,0.15
    ,0.15,0.15,0.15,0.14,0.15,0.15,
188 0.16,0.16,0.16,0.15,0.15,0.16,0.16,0.16,0.16,0.16,0.16,0.16,0.16,0.16,0.16,0.17
    ,0.17,0.17,0.17,0.17,0.16,0.16,
189 0.17,0.17,0.17,0.17,0.17,0.17,0.17,0.17,0.16,0.16,0.16,0.16,0.16,0.15,0.16,0.15
    ,0.15,0.15,0.15,0.15,0.15,0.14,
190 0.14,0.14,0.14,0.14,0.13,0.13,0.14,0.14,0.13,0.13,0.13,0.13,0.13,0.12,0.12,0.12
    ,0.12,0.12,0.12,0.12,0.12,0.12,
191 0.12,0.12,0.12,0.12,0.12,0.12,0.12,0.11,0.11,0.12,0.11,0.11,0.11,0.11,0.11,0.11
    ,0.1,0.1,0.11,0.1,0.11,0.11,0.1,
192 0.1,0.1,0.09,0.09,0.09,0.08,0.07,0.07,0.07,0.07,0.06,0.05,0.06,0.06,0.04,0.04,0
    .04,0.04,0.04,0.03,0.03,0.03,0.02,
193 0.02,0.01,0.01,0.01,0.01,0.01,0,0,-0.01,-0.01,-0.02,-0.02,-0.02,-0.02,-0.02,-0.
    02,-0.02,-0.03,-0.03,-0.03,-0.03,
194 -0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.03,-0.04,-0.04,-0.04,-0.05,-0.05,-0.04,-
    0.04,-0.04,-0.05,-0.05,-0.05,-0.06,
195 -0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,0.05,-0.05,-0
    .05,-0.06,-0.06,-0.05,-0.05,-0.05,
196 -0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-
    0.06,-0.06,-0.05,-0.05,-0.05,-0.06,
197 -0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.06,-0.06,
198 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.06,-0.06,-0.06,-0.05,
199 -0.05,-0.06,-0.05,-0.05,-0.06,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.06,-0.06,-0.05,-0.05,-0.05,-0.05,
200 -0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-
    0.05,-0.05,-0.06,-0.06,-0.05,-0.06,
201 -0.05,-0.06,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
202 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.06,-0.06,-0.05,-0.05,
203 -0.06,-0.06,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.06,-0.05,-0.06,-0.05,-
    0.06,-0.05,-0.05,-0.05,-0.05,-0.05,
204 -0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
205 -0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.06,
206 -0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-
    0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
```

```
-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.06,-0.06,-0.06,-0.05,-0.05,-0.05,-0.06,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.06,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,-0.05,
};
```

需要包含的头文件

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <math.h>
```

随机数生成

测试过程中需要用随机数生成噪声，生成随机数代码如下所示。

```
/*****************************************************************************
*************************
* 函数名称：  Uniform
* 函数功能：  生成均匀分布的随机数
* 输入参数：  a：区间下限
*             b：区间上限
*             seed：随机数种子
* 输出参数：  void
* 返 回 值：  随机数
* 创建日期：  2023年08月10日
* 注    意：
*****************************************************************************
*************************/
double Uniform(double a, double b, long int* seed)
{
  double t;
  *seed = 2045 * (*seed) + 1;
  *seed = *seed - (*seed / 1048576) * 1048576;
  t = (*seed) / 1048576.0;
  t = a + (b - a) * t;
  return t;
```

```
}
/*****************************************************************************
**************************
* 函数名称:  Gauss
* 函数功能:  生成正态分布的随机数
* 输入参数:  mean:正态分布的均值μ
*            sigma:正态分布的均方差σ
*            seed:随机数种子
* 输出参数:  void
* 返 回 值:  随机数
* 创建日期:  2023年08月10日
* 注    意:
*****************************************************************************
**************************/
double Gauss(double mean, double sigma, long int* seed)
{
  int i;
  double x, y;
  for (x = 0, i = 0; i < 12; i++)
  {
    x += Uniform(0.0, 1.0, seed);
  }
  x = x - 6.0;
  y = mean + x * sigma;
  return y;
}
```

维纳（Wiener）数字滤波

维纳（Wiener）数字滤波器实现代码如下所示。

```
/*****************************************************************************
**************************
* 函数名称:  Levin
* 函数功能:  求解一般托布利兹方程组的莱文森算法
* 输入参数:  t:双精度实型一维数组，长度为 n。存放对称托布利兹矩阵的元素 t0,t1,...,tn-
1。
*            b:双精度实型一维数组，长度为 n。存放方程组右端的常数常量。
*            n:整形变量。方程组的阶数。
*            x:双精度实型一维数组，长度为 n。存放方程组的解。
* 输出参数:  void
* 返 回 值:  本函数的返回值若小于 0，则说明方程是病态的。
* 创建日期:  2023年09月20日
* 注    意:
```

```c
/*****************************************************************************
*************************/
int Levin(double* t, double* b, int n, double* x)
{
  int i, j, k;
  double a, beta, q, c, h, *y, *s;
  s = malloc(n * sizeof(double));
  y = malloc(n * sizeof(double));
  a = t[0];
  if ((fabs(a) + 1.0) == 1.0)
  {
    free(s);
    free(y);
    return (-1);
  }
  y[0] = 1.0;
  x[0] = b[0] / a;
  for (k = 1; k < n; k++)
  {
    beta = 0.0;
    q = 0.0;
    for (j = 0; j < k; j++)
    {
      beta += y[j] * t[j + 1];
      q += x[j] * t[k - j];
    }
    if ((fabs(a) + 1.0) == 1.0)
    {
      free(s);
      free(y);
      return (-1);
    }
    c = -beta / a;
    s[0] = c * y[k - 1];
    y[k] = y[k - 1];
    if (k != 1)
    {
      for (i = 1; i < k; i++)
      {
        s[i] = y[i - 1] + c * y[k - i - 1];
      }
    }
    s[k] = y[k - 1];
    a += c * beta;
    if ((fabs(a) + 1.0) == 1.0)
    {
      free(s);
```

```
58        free(y);
59        return (-1);
60      }
61      h = (b[k] - q) / a;
62      for (i = 0; i < k; i++)
63      {
64        x[i] += h * s[i];
65        y[i] = s[i];
66      }
67      x[k] = h * y[k];
68    }
69    free(s);
70    free(y);
71    return (1);
72  }
73  /*****************************************************************************
    ***************************
74  * 函数名称:  Wiener
75  * 函数功能:  维纳数字滤波
76  * 输入参数:  rxx:双精度实型一维数组，长度为（p+1）。信号 x(n) 的自相关函数 rxx(i)。
77  *            rdx:双精度实型一维数组，长度为（p+1）。信号 d(n) 与 x(n) 的互相关函数
    rdx(i)。
78  *            p  :整型变量。维纳滤波器的阶数。
79  *            h  :双精度实型一维数组，长度为（p+1）。维纳滤波器的系数。
80  *            e  :双精度实型变量。维纳滤波器的最小均方误差。
81  *            x  :双精度实型一维数组，长度为 n。存放输入信号 x(i)。
82  *            y  :双精度实型一维数组，长度为 n。存放输出信号 y(i)。
83  *            n  :整形变量。输入信号的长度。
84  * 输出参数:  void
85  * 返 回 值:  void
86  * 创建日期:  2023年09月21日
87  * 注    意:
88  *****************************************************************************
    ***************************/
89  void Wiener(double* rxx, double* rdx, int p, double* h, double* e, double* x,
    double* y, int n)
90  {
91    int i, k;
92    double sum;
93    Levin(rxx, rdx, p + 1, h);
94    sum = 0.0;
95    for (i = 0; i <= p; i++)
96    {
97      sum += rdx[i] * h[i];
98    }
99    *e = rdx[0] - sum;
100   for (k = 0; k < n; k++)
```

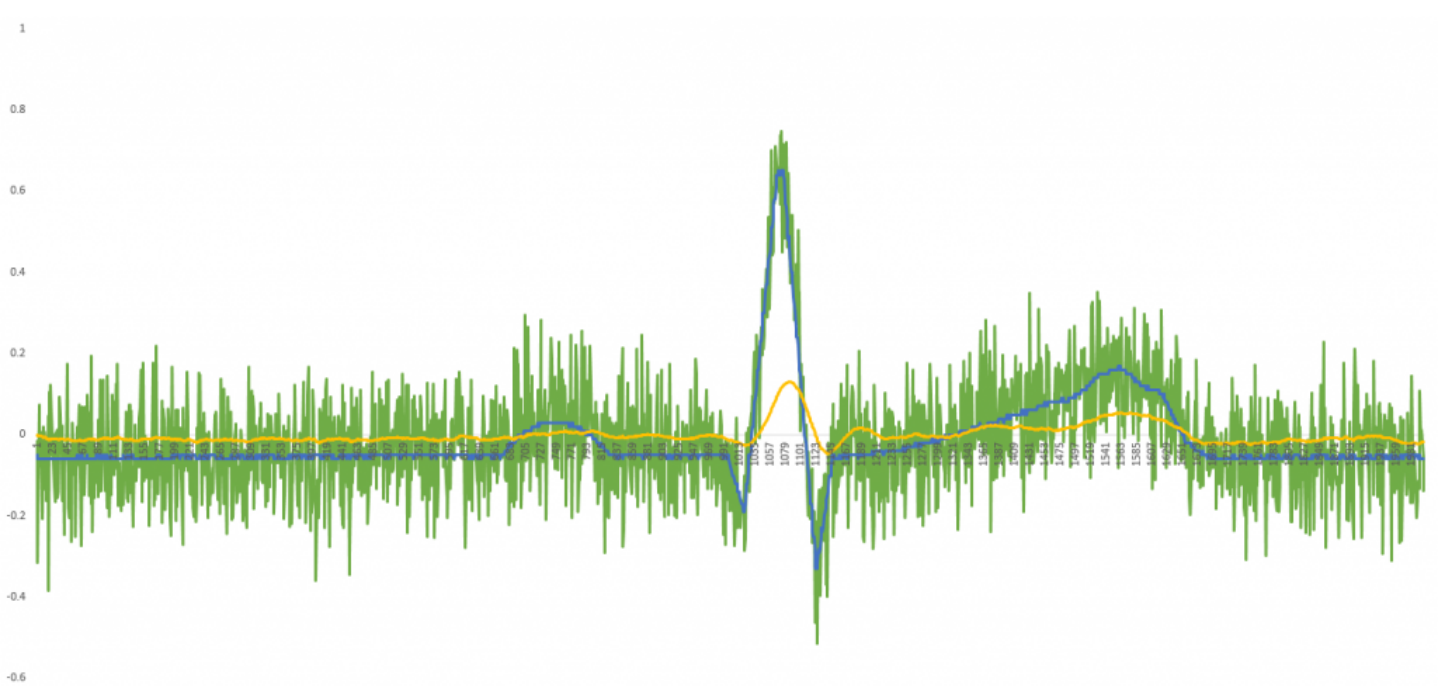
```
  {
    y[k] = 0.0;
    for (i = 0; i <= p; i++)
    {
      if ((k - i) >= 0)
      {
        y[k] += h[i] * x[k - i];
      }
    }
  }
}
```

维纳（Wiener）滤波器的使用如下所示。其中心电波形数据为模拟器标准数据，脉率值为 60BPM，采样率为 2kHz，在文章的末尾给出。

```
int main(void)
{
  extern const double g_arrEcgWave[4000];
  int i, k, n, p;
  long seed;
  double e;
  static double rxx[4000], rdx[4000];
  static double h[4000], x[4000], y[4000], s[4000];
  FILE* fp;
  //获取原始数据
  n = 2000;
  for (i = 0; i < n; i++)
  {
    s[i] = g_arrEcgWave[i];
  }
  //添加白噪声
  seed = 157l;
  for (i = 0; i < n; i++)
  {
    x[i] = s[i] + 0.1 * Gauss(0.0, 1.0, &seed);
  }
  //生成互相关函数
  p = 2000;
  for (k = 0; k <= p; k++)
  {
    rdx[k] = 0.0;
    for (i = 0; i < (n - k); i++)
    {
      rdx[k] += s[i] * s[i + k];
    }
```

```
    rdx[k] = rdx[k] / n;
  }
  //生成自相关函数
  rxx[0] = rdx[0] + 1.0;
  for (i = 1; i <= p; i++)
  {
    rxx[i] = rdx[i];
  }
  //滤波
  Wiener(rxx, rdx, p, h, &e, x, y, n);
  printf("The Minimum MSE Error = %lf\n", e);
  //输出波形数据，按照原始信号、参考信号、滤波后信号顺序
  fp = fopen("wieners.csv", "w");
  for (i = 0; i < n; i++)
  {
    fprintf(fp, "%d,%lf,%lf,%lf\n", i, x[i], s[i], y[i]);
  }
  fclose(fp);
  return 0;
}
```

实验结果如下所示。蓝色为参考信号，绿色为输入信号，黄色为输出信号。



最小均方（LMS）自适应数字滤波

实现代码如下所示。

```
/*****************************************************************************
*************************
```

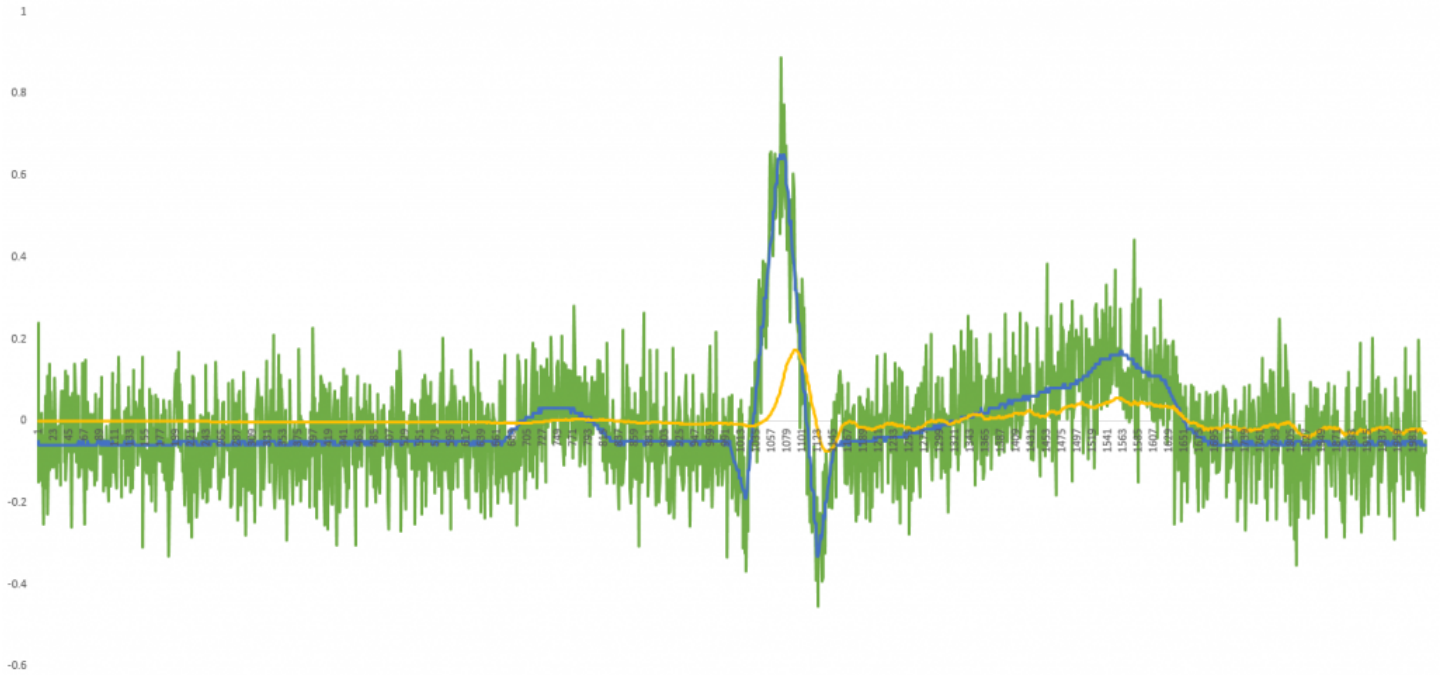
```
 2 * 函数名称:  LMS
 3 * 函数功能:  最小均方（LMS）自适应数字滤波
 4 * 输入参数:  x ：双精度实型一维数组，长度为 n。输入信号。
 5 *            d ：双精度实型一维数组，长度为 n。理想输出信号。
 6 *            y ：双精度实型一维数组，长度为 n。实际输出信号。
 7 *            n ：整型变量。输入信号的长度。
 8 *            w ：双精度实型一维数组，长度为 m。自适应滤波器的加权系数。
 9 *            m ：整型变量。自适应滤波器的长度（阶数-1）。
10 *            mu：双精度实型变量。收敛因子。
11 * 输出参数:  void
12 * 返 回 值:  void
13 * 创建日期:  2023年09月22日
14 * 注    意:
15 ********************************************************************************
   **************************/
16 void LMS(double* x, double* d, double* y, int n, double* w, int m, double mu)
17 {
18   int i, k;
19   double e;
20   for (i = 0; i < m; i++)
21   {
22     w[i] = 0.0;
23   }
24   for (k = 0; k < m; k++)
25   {
26     y[k] = 0.0;
27     for (i = 0; i <= k; i++)
28     {
29       y[k] += x[k - i] * w[i];
30     }
31     e = d[k] - y[k];
32     for (i = 0; i <= k; i++)
33     {
34       w[i] += 2.0 * mu * e * x[k - i];
35     }
36   }
37   for (k = m; k < n; k++)
38   {
39     y[k] = 0.0;
40     for (i = 0; i < m; i++)
41     {
42       y[k] += x[k - i] * w[i];
43     }
44     e = d[k] - y[k];
45     for (i = 0; i < m; i++)
46     {
47       w[i] += 2.0 * mu * e * x[k - i];
```

```
    }
  }
}
```

测试代码如下所示。其中心电波形数据为模拟器标准数据，脉率值为 60BPM，采样率为 2kHz，在文章的末尾给出。

```
int main(void)
{
  extern const double g_arrEcgWave[4000];
  int i, m, n;
  long seed;
  double mu;
  static double d[4000], x[4000], y[4000], w[4000];
  FILE* fp;
  //获取理想输出信号
  n = 2000;
  for (i = 0; i < n; i++)
  {
    d[i] = g_arrEcgWave[i];
  }
  //获取输入数据，加噪声
  seed = 13579l;
  for (i = 0; i < n; i++)
  {
    x[i] = g_arrEcgWave[i] + 0.1 * Gauss(0.0, 1.0, &seed);
  }
  //LMS 滤波
  m = 50;
  mu = 0.0005;
  LMS(x, d, y, n, w, m, mu);
  //输出波形数据，按照原始信号、参考信号、滤波后信号顺序
  fp = fopen("lms.csv", "w");
  for (i = 0; i < n; i++)
  {
    fprintf(fp, "%d,%lf,%lf,%lf\n", i, x[i], d[i], y[i]);
  }
  fclose(fp);
  return 0;
}
```

最终实验结果如下所示。蓝色为参考信号，绿色为输入信号，黄色为输出信号。



归一化（LMS）自适应数字滤波

实现代码如下所示。

```
/*****************************************************************************
*****************************
* 函数名称：  NLMS
* 函数功能：  归一化（LMS）自适应数字滤波
* 输入参数：  x     ：双精度实型一维数组，长度为 n。开始时存放输入信号，最后存放实际输出信号。
*             d     ：双精度实型一维数组，长度为 n。理想输出信号。
*             n     ：整型变量。输入信号的长度。
*             w     ：双精度实型一维数组，长度为 m。自适应滤波器的加权系数。
*             m     ：整型变量。自适应滤波器的长度（阶数-1）。
*             mu    ：双精度实型变量。学习因子，0 < mu < 1。
*             sigma2：双精度实型变量。输入信号的功率估值 σ^2。
*             a     ：双精度实型变量。遗忘因子，0 <= a <= 1。
*             px    ：双精度实型一维数组，长度为 m。在分块处理时，用于保存输入信号的过去值。
* 输出参数：  void
* 返 回 值：  void
* 创建日期：  2023年09月23日
* 注    意：
*****************************************************************************
*************************/
void NLMS(double* x, double* d, int n, double* w, int m, double mu, double sigma2, double a, double* px)
{
  int i, k;
  double e, tmp;
```

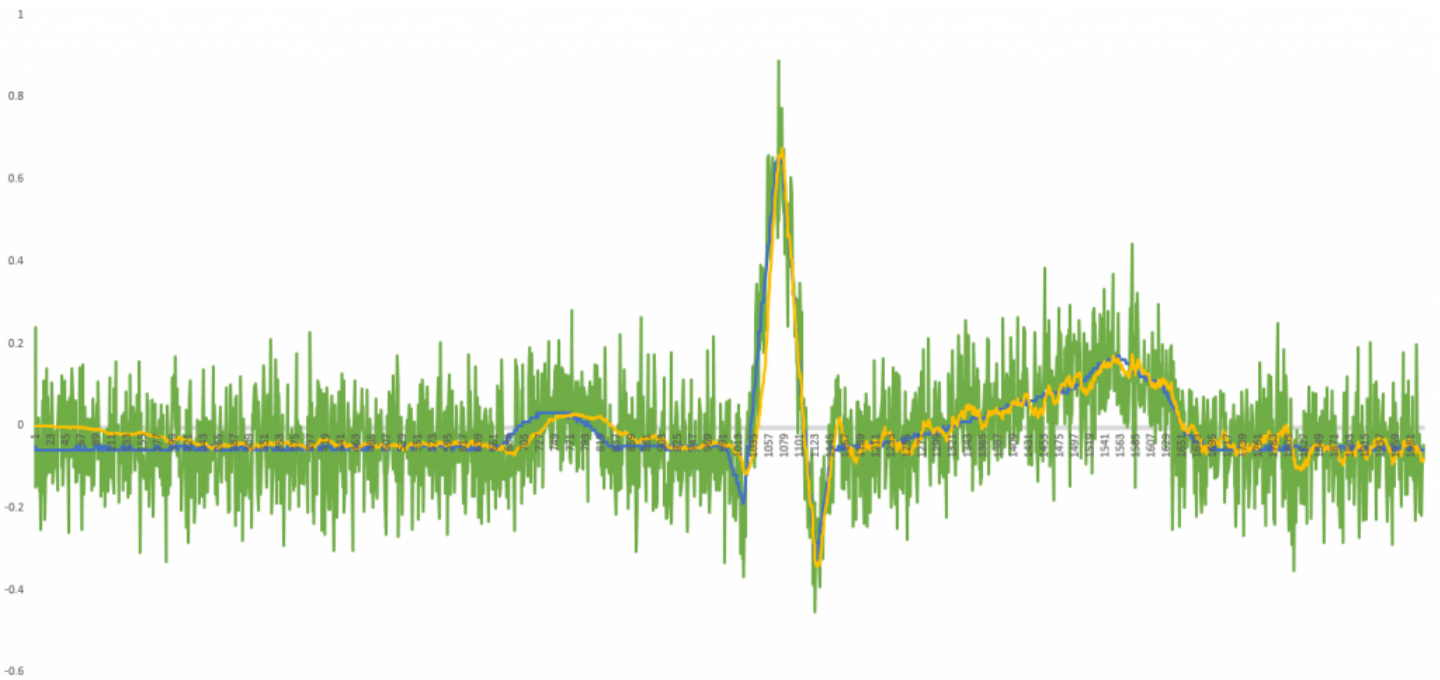
```
  for (k = 0; k < n; k++)
  {
    px[0] = x[k];
    x[k] = 0.0;
    for (i = 0; i < m; i++)
    {
      x[k] += px[i] * w[i];
    }
    e = d[k] - x[k];
    sigma2 = a * px[0] * px[0] + (1.0 - a) * sigma2;
    tmp = 2 * mu / (m * sigma2);
    for (i = 0; i < m; i++)
    {
      w[i] += tmp * e * px[i];
    }
    for (i = (m - 1); i >= 1; i--)
    {
      px[i] = px[i - 1];
    }
  }
}
```

测试代码如下所示。其中心电波形数据为模拟器标准数据，脉率值为 60BPM，采样率为 2kHz，在文章的末尾给出。

```
int main(void)
{
  extern const double g_arrEcgWave[4000];
  int i, m, n;
  double a, mu, sigma2;
  long seed;
  static double d[4000], x[4000], y[4000], w[4000], px[4000];
  FILE* fp;
  //获取理想输出信号
  for (i = 0; i < 2000; i++)
  {
    d[i] = g_arrEcgWave[i];
  }
  //获取输入数据，加噪声
  seed = 13579l;
  n = 2000;
  for (i = 0; i < n; i++)
  {
    x[i] = g_arrEcgWave[i] + 0.1 * Gauss(0.0, 1.0, &seed);
    y[i] = x[i];
```

```
    }
    //LMS 滤波
    m = 51;
    mu = 0.2;
    sigma2 = 0.2;
    a = 0;
    for (i = 0; i < m; i++){w[i] = 0.0;}
    for (i = 0; i < m; i++){px[i] = 0.0;}
    NLMS(y, d, n, w, m, mu, sigma2, a, px);
    //输出波形数据，按照原始信号、参考信号、滤波后信号顺序
    fp = fopen("nlms.csv", "w");
    for (i = 0; i < n; i++)
    {
      fprintf(fp, "%d,%10.7lf,%10.7lf,%10.7lf\n", i, x[i], d[i], y[i]);
    }
    return 0;
}
```

最终实验结果如下所示。蓝色为参考信号，绿色为输入信号，黄色为输出信号。



递推最小二乘（RLS）自适应数字滤波

实验代码如下所示。

```
/*****************************************************************************
*****************************
* 函数名称：  RLS
* 函数功能：  递推最小二乘（RLS）自适应数字滤波
* 输入参数：  x：双精度实型一维数组，长度为 n。开始时存放输入信号，最后存放实际输出信号。
```

```
 5 *             d：双精度实型一维数组，长度为 n。理想输出信号。
 6 *             n：整型变量。输入信号的长度。
 7 *             w：双精度实型一维数组，长度为 m。自适应滤波器的加权系数。
 8 *             m：整型变量。自适应滤波器的长度（阶数-1）。
 9 *             r：双精度实型变量。遗忘因子，0 < r <= 1。
10 * 输出参数：  void
11 * 返 回 值：  void
12 * 创建日期：  2023年09月23日
13 * 注    意：
14 ********************************************************************************
   ************************/
15 void RLS(double* x, double* d, int n, double* w, int m, double r)
16 {
17   int i, j, k;
18   double a, s, *g, *u, *px, *p;
19   g = malloc(m * sizeof(double));
20   u = malloc(m * sizeof(double));
21   px = malloc(m * sizeof(double));
22   p = malloc(m * m * sizeof(double));
23   for (i = 0; i < m; i++)
24   {
25     for (j = 0; j < m; j++)
26     {
27       p[i * m + j] = 0.0;
28     }
29   }
30   for (i = 0; i < m; i++)
31   {
32     p[i * m + i] = 1.0e+8;
33   }
34   for (i = 0; i < m; i++)
35   {
36     px[i] = 0.0;
37   }
38   for (k = 0; k < n; k++)
39   {
40     px[0] = x[k];
41     for (j = 0; j < m; j++)
42     {
43       u[j] = 0.0;
44       for (i = 0; i < m; i++)
45       {
46         u[j] = u[j] + (1 / r) * p[j * m + i] * px[i];
47       }
48     }
49     s = 1.0;
50     for (i = 0; i < m; i++)
```

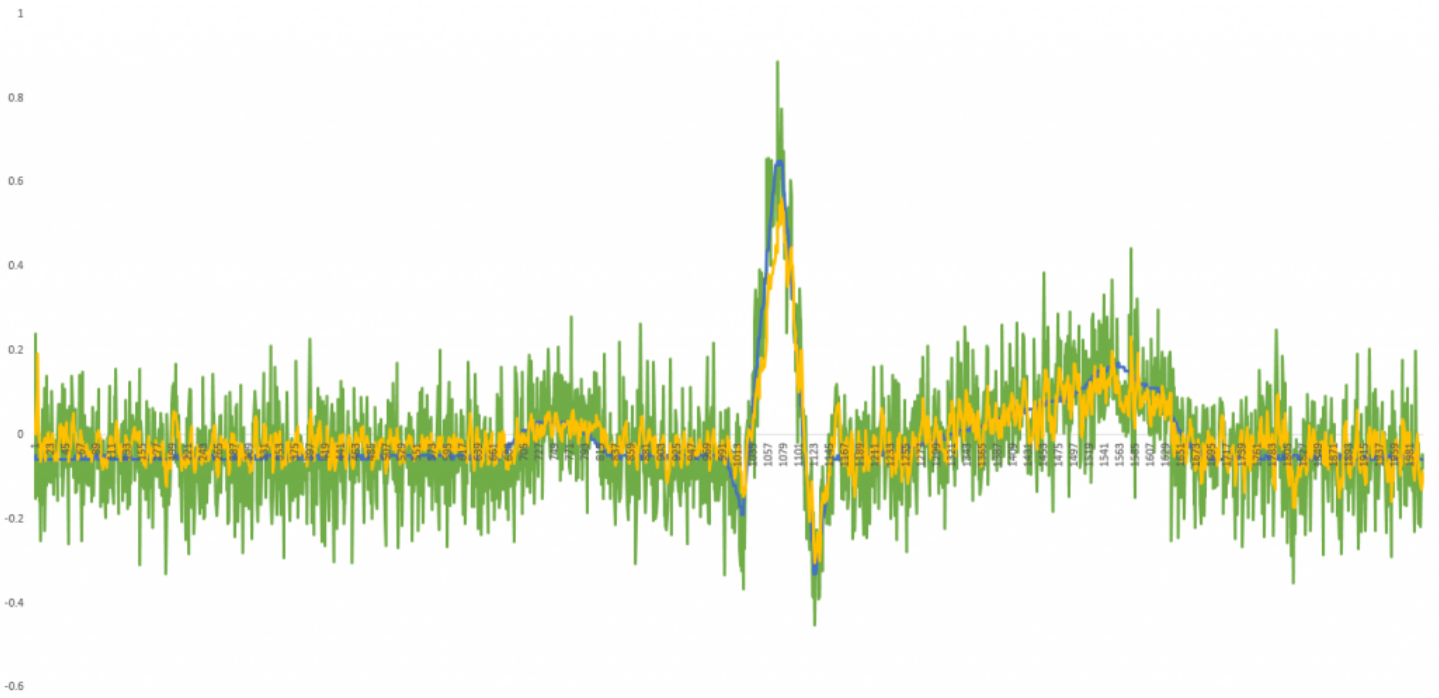
```
    {
      s = s + u[i] * px[i];
    }
    for (i = 0; i < m; i++)
    {
      g[i] = u[i] / s;
    }
    x[k] = 0.0;
    for (i = 0; i < m; i++)
    {
      x[k] = x[k] + w[i] * px[i];
    }
    a = d[k] - x[k];
    for (i = 0; i < m; i++)
    {
      w[i] = w[i] + g[i] * a;
    }
    for (j = 0; j < m; j++)
    {
      for (i = 0; i < m; i++)
      {
        p[j * m + i] = (1 / r) * p[j * m + i] - g[j] * u[i];
      }
    }
    for (i = (m - 1); i >= 1; i--)
    {
      px[i] = px[i - 1];
    }
  }
  free(g);
  free(u);
  free(px);
  free(p);
}
```

测试代码如下所示。其中心电波形数据为模拟器标准数据，脉率值为 60BPM，采样率为 2kHz，在文章的末尾给出。

```
int main(void)
{
  extern const double g_arrEcgWave[4000];
  int i, m, n;
  long seed;
  static double r, w[4000], d[4000], x[4000], y[4000];
  FILE* fp;
```

```
    //获取理想输出信号
    for (i = 0; i < 2000; i++)
    {
      d[i] = g_arrEcgWave[i];
    }
    //获取输入数据，加噪声
    seed = 13579l;
    n = 2000;
    for (i = 0; i < n; i++)
    {
      x[i] = g_arrEcgWave[i] + 0.1 * Gauss(0.0, 1.0, &seed);
      y[i] = x[i];
    }
    // RLS 滤波
    m = 4;
    r = 1.0;
    for (i = 0; i < m; i++){ w[i] = 0.0;}
    RLS(y, d, n, w, m, r);
    //输出波形数据，按照原始信号、参考信号、滤波后信号顺序
    fp = fopen("rls.csv", "w");
    for (i = 0; i < n; i++)
    {
      fprintf(fp, "%d,%10.7lf,%10.7lf,%10.7lf\n", i, x[i], d[i], y[i]);
    }
    return 0;
}
```

最终实验结果如下所示。蓝色为参考信号，绿色为输入信号，黄色为输出信号。



滑动平均滤波

实现代码如下。

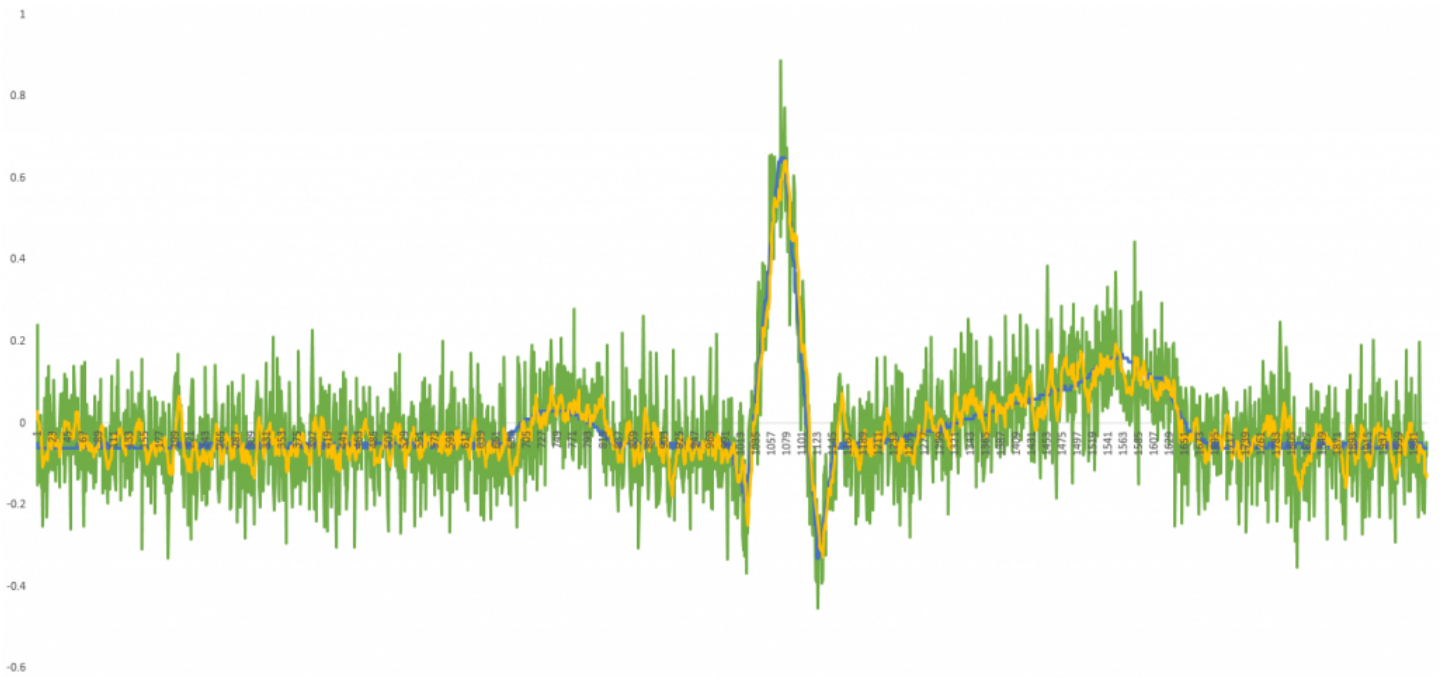
```
/*****************************************************************************
**************************
* 函数名称：  MovAverageFilter
* 函数功能：  滑动平均滤波
* 输入参数：  filter：双精度实型一维数组，长度为 oder。滤波缓冲区。
*             order ：整型变量。滤波阶数。
*             dat   ：双精度实型变量。新的采样数据
* 输出参数：  void
* 返 回 值：  滤波值
* 创建日期：  2023年09月20日
* 注    意：
*****************************************************************************
**************************/
double MovAverageFilter(double* filter, int order, double dat)
{
  int i;
  double sum;
  for (i = 0; i < order - 1; i++)
  {
    filter[i] = filter[i + 1];
  }
  filter[order - 1] = dat;
  sum = 0;
  for (i = 0; i < order; i++)
  {
    sum = sum + filter[i];
  }
  sum = sum / order;
  return sum;
}
```

测试代码如下所示。其中心电波形数据为模拟器标准数据，脉率值为 60BPM，采样率为 2kHz，在文章的末尾给出。

```
int main(void)
{
  extern const double g_arrEcgWave[4000];
  #define FILTER_ORDER 8
  static double s_arrFilter[FILTER_ORDER] = {0};
```

```
  int i, n;
  long seed;
  static double d[4000], x[4000], y[4000];
  FILE* fp;
  //获取理想输出信号
  n = 2000;
  for (i = 0; i < 2000; i++)
  {
    d[i] = g_arrEcgWave[i];
  }
  //获取输入数据，加噪声
  seed = 13579l;
  for (i = 0; i < n; i++)
  {
    x[i] = g_arrEcgWave[i] + 0.1 * Gauss(0.0, 1.0, &seed);
  }
  //滑动平均滤波
  for (i = 0; i < n; i++)
  {
    y[i] = MovAverageFilter(s_arrFilter, FILTER_ORDER, x[i]);
  }
  //输出波形数据，按照原始信号、参考信号、滤波后信号顺序
  fp = fopen("MovAverageFilter.csv", "w");
  for (i = 0; i < n; i++)
  {
    fprintf(fp, "%d,%10.7lf,%10.7lf,%10.7lf\n", i, x[i], d[i], y[i]);
  }
  return 0;
}
```

最终实验结果如下所示。蓝色为参考信号，绿色为输入信号，黄色为输出信号。

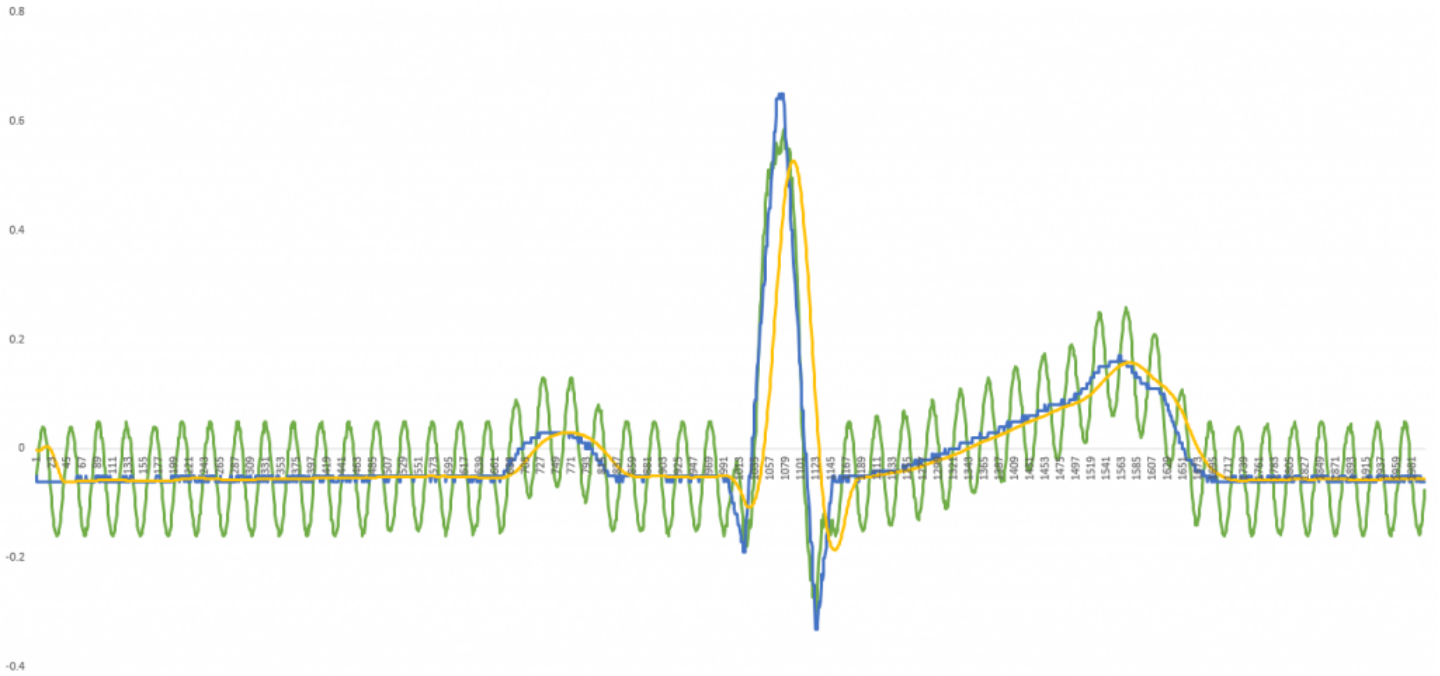


心电信号添加 50Hz 工频干扰，利用滑动平均滤波滤除工频干扰示例如下。2kHz 采样率下滤波阶数为 40，500Hz 采样率下滤波阶数为 10，其它采样率可以根据这个规律设置滤波阶数。

```
int main(void)
{
  extern const double g_arrEcgWave[4000];
  #define FILTER_ORDER 40
  static double s_arrFilter[FILTER_ORDER] = { 0 };
  int i, n;
  long seed;
  static double d[4000], x[4000], y[4000];
  double pi, time;
  FILE* fp;
  //获取理想输出信号
  n = 2000;
  for (i = 0; i < 2000; i++)
  {
    d[i] = g_arrEcgWave[i];
  }
  //获取输入数据，加 50Hz 工频干扰
  seed = 13579l;
  pi = 3.1415926535;
  time = 0;
  for (i = 0; i < n; i++)
  {
    x[i] = g_arrEcgWave[i] + 0.1 * sin(time * 2 * pi / (1.0 / 50.0));
    time = time + 0.0005;
  }
  //滑动平均滤波
  for (i = 0; i < n; i++)
```

```
  {
    y[i] = MovAverageFilter(s_arrFilter, FILTER_ORDER, x[i]);
  }
  //输出波形数据，按照原始信号、参考信号、滤波后信号顺序
  fp = fopen("MovAverageFilter.csv", "w");
  for (i = 0; i < n; i++)
  {
    fprintf(fp, "%d,%10.7lf,%10.7lf,%10.7lf\n", i, x[i], d[i], y[i]);
  }
  return 0;
}
```

最终实验结果如下所示。蓝色为参考信号，绿色为输入信号，黄色为输出信号。



中值滤波

实现代码如下。

```
/*****************************************************************************
*****************************
* 函数名称：  MedianFilter
* 函数功能：  中值滤波
* 输入参数：  filter1：双精度实型一维数组，长度为 oder。滤波缓冲区。
*             filter2：双精度实型一维数组，长度为 oder。滤波缓冲区。
*             order  ：整型变量。滤波阶数。
*             dat    ：双精度实型变量。新的采样数据
* 输出参数：  void
* 返 回 值：  滤波值
* 创建日期：  2023年09月23日
```

```
* 注    意:
*************************************************************************************
*************************/
double MedianFilter(double* filter1, double* filter2, int order, double dat)
{
  int i, j;
  double swap;
  //将最新的数据保存到缓冲区 1
  for (i = 0; i < order - 1; i++)
  {
    filter1[i] = filter1[i + 1];
  }
  filter1[order - 1] = dat;
  //将数据拷贝到缓冲区 2
  for (i = 0; i < order; i++)
  {
    filter2[i] = filter1[i];
  }
  //重新排序
  for (i = 0; i < order; i++)
  {
    for (j = i + 1; j < order; j++)
    {
      if (filter2[j] > filter2[i])
      {
        swap = filter2[j];
        filter2[j] = filter2[i];
        filter2[i] = swap;
      }
    }
  }
  //返回中位值
  return filter2[order / 2];
}
```

测试代码如下所示。其中心电波形数据为模拟器标准数据，脉率值为 60BPM，采样率为 2kHz，在文章的末尾给出。

```
int main(void)
{
  extern const double g_arrEcgWave[4000];
  #define FILTER_ORDER 8
  static double s_arrFilter1[FILTER_ORDER] = { 0 };
  static double s_arrFilter2[FILTER_ORDER] = { 0 };
  int i, n;
```

```
    long seed;
    static double d[4000], x[4000], y[4000];
    FILE* fp;
    //获取理想输出信号
    n = 2000;
    for (i = 0; i < 2000; i++)
    {
      d[i] = g_arrEcgWave[i];
    }
    //获取输入数据，加噪声
    seed = 13579l;
    for (i = 0; i < n; i++)
    {
      x[i] = g_arrEcgWave[i] + 0.1 * Gauss(0.0, 1.0, &seed);
    }
    //滑动平均滤波
    for (i = 0; i < n; i++)
    {
      y[i] = MedianFilter(s_arrFilter1, s_arrFilter2, FILTER_ORDER, x[i]);
    }
    //输出波形数据，按照原始信号、参考信号、滤波后信号顺序
    fp = fopen("MedianFilter.csv", "w");
    for (i = 0; i < n; i++)
    {
      fprintf(fp, "%d,%10.7lf,%10.7lf,%10.7lf\n", i, x[i], d[i], y[i]);
    }
    return 0;
}
```

最终实验结果如下所示。蓝色为参考信号，绿色为输入信号，黄色为输出信号。