牛哥哥，QQ:769985518

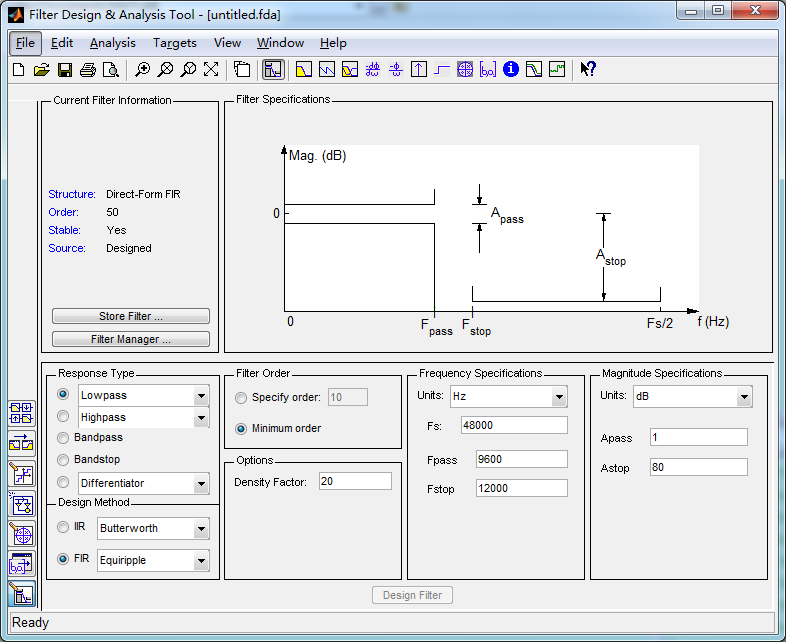
具体理解不透，只会使用。

我们常用的也就是一阶或者二阶，这些都是可以通过c语言代码来实现的。其过程包括分母A和分子B参数的求取，再者通过公式得出想要的输出结果。

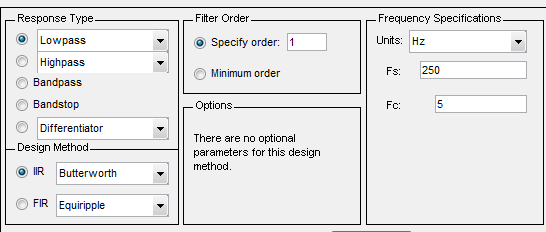
1. **一阶巴特沃斯低通滤波器设计**

第一步：其参数A和B的求取可以通过Matlab的滤波器工具箱

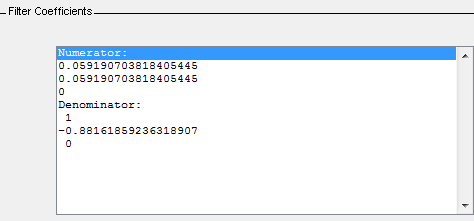
来获得（因为这个c语言代码还不知道怎么写出来），打开Matlab软件，依次Start(在Matlab界面的左下角)🡪Toolboxes🡪Filter Design🡪Filter Design & Analysis Tool (fdatool)，打开的界面如下：



可以针对自己想要的滤波器，这里都可以进行设置，得到相应的参数，本人需要一阶巴特沃斯低通滤波器，采样频率：250Hz，截止频率5Hz，设置如下：



然后，点Design Filter即可，接着在菜单栏Analysis🡪Filter Coefficients，可以看到分母A和分子B相应的参数：



第二步：参数得到，代入公式即可了。

差分方程公式为：，为当前输入，c语言代码为：

float LPF\_Butter\_(float curr\_input)

{

static float input[2];

static float output[2];

/\*获取最新输入\*/

input[1] = curr\_input;

/\* Butterworth \*/

output[1] =B[0] \* input[1] +B[1] \* input[0] - A[1] \* output[0];

/\* 输入序列保存 \*/

input[0] = input[1];

/\*输出序列保存 \*/

output[0] = output[1];

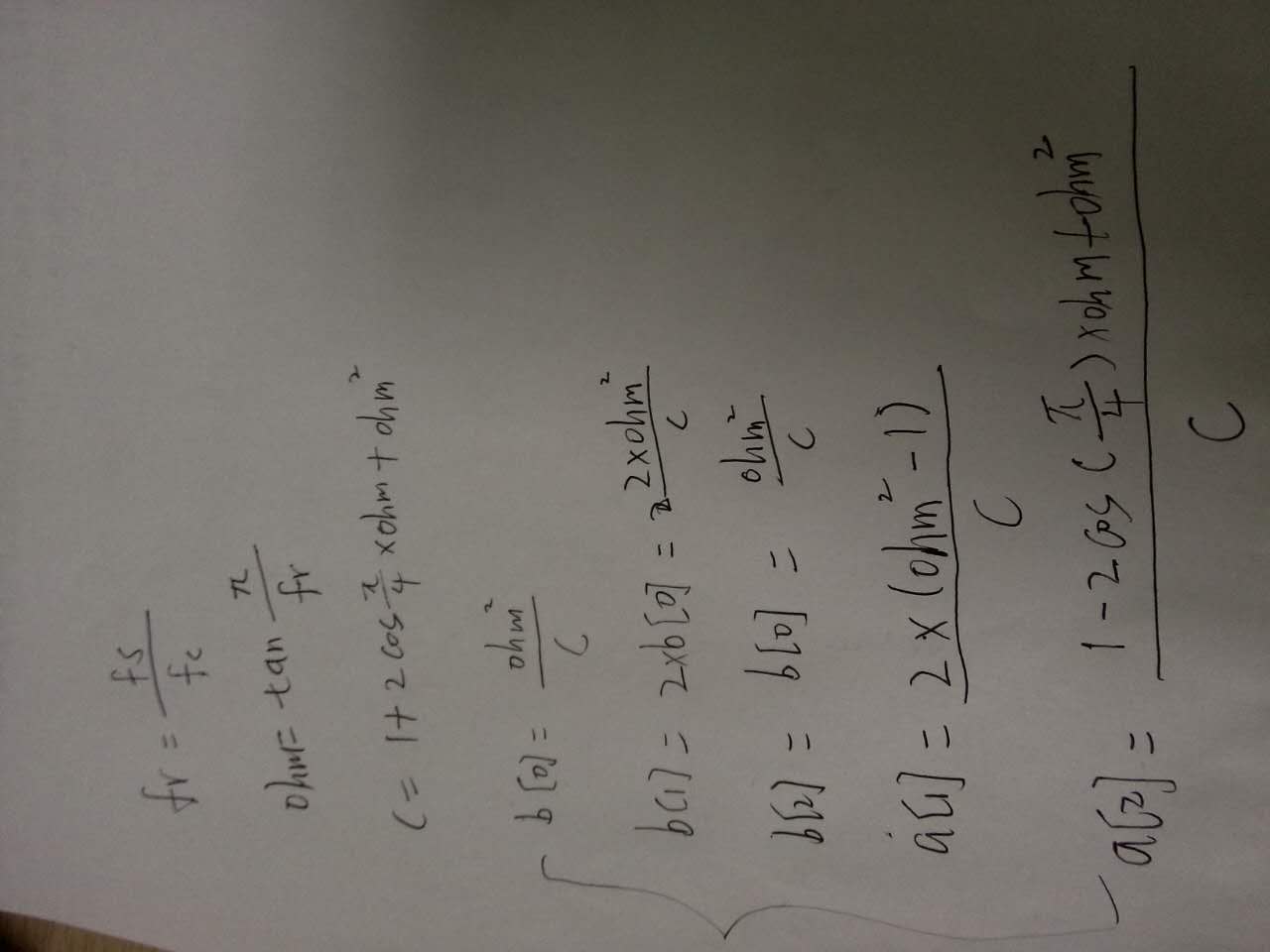
return output[1];

}

1. **二阶巴特沃斯低通滤波器设计**

第一步：二阶的话，其用Matlab获取参数和c语言代码得到参数都可以，Matlab获取参数跟上述类似，现在使用c语言代码得到分子B和分母A的参数。

设置采样频率为fs，截止频率为fc，则：



代码为：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

通过此函数可以得到二阶巴特沃斯的系数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LPF2pSetCutoffFreq(float sample\_freq, float cutoff\_freq)

{

float fr =0;

float ohm =0;

float c =0;

fr= sample\_freq/cutoff\_freq;

ohm=tanf(M\_PI\_F/fr);

c=1.0f+2.0f\*cosf(M\_PI\_F/4.0f)\*ohm + ohm\*ohm;

// \_cutoff\_freq1 = cutoff\_freq;

// if (\_cutoff\_freq1 > 0.0f)

// {

b\_acc[0] = ohm\*ohm/c;

b\_acc[1] = 2.0f\*b\_acc[0];

b\_acc[2] = b\_acc[0];

a\_acc[1] = 2.0f\*(ohm\*ohm-1.0f)/c;

a\_acc[2] = (1.0f-2.0f\*cosf(M\_PI\_F/4.0f)\*ohm+ohm\*ohm)/c;

// }

}

第二步：代入公式

一个二阶结构是这样的一个表达式：



这个表达式中需要过去的4个历史值：，

float LPF\_Butter\_(float curr\_input)

{

static float input[3];

static float output[3];

/\*获取最新输入\*/

input[2] = curr\_input;

/\* Butterworth \*/

output[2] =B[0] \* input[2] +B[1] \* input[1] +B[2] \* input[0] - A[1] \* output[1] - A[2] \* output[0];

/\* 输入序列保存 \*/

input[0] = input[1];

input[1] = input[2];

/\*输出序列保存 \*/

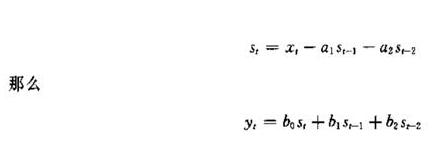
output[0] = output[1];

output[1] = output[2];

return output[2];

}

有一个简单的技巧，可以把上面的计算简化，使得历史状态由4减为2，定义下面的表达式：



那么代码就可以写为：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

二阶巴特沃斯低通滤波函数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static float \_delay\_element\_11; // buffered sample -1

static float \_delay\_element\_21; // buffered sample -2

float LPF2pApply\_1(float sample)

{

float delay\_element\_0 = 0, output=0;

delay\_element\_0 = sample - \_delay\_element\_11 \* a\_acc[1] - \_delay\_element\_21 \* a\_acc[2];

if (isnan(delay\_element\_0) || isinf(delay\_element\_0)) {

// don't allow bad values to propogate via the filter

delay\_element\_0 = sample;

}

output = delay\_element\_0 \* b\_acc[0] + \_delay\_element\_11 \* b\_acc[1] + \_delay\_element\_21 \* b\_acc[2];

\_delay\_element\_21 = \_delay\_element\_11;

\_delay\_element\_11 = delay\_element\_0;

// return the value. Should be no need to check limits

return output;

// }

}

两种方法都可使用。，看个人选择