Лабораторная №12

1) В классе PROCESSING реализовать функцию расчета импульсной реакции или весовой функции (весов *lpw[]) фильтра низких частот (ФНЧ) со сглаживающим окном Поттера lpf(fc, m, dt, *lpw) на основе предоставленного кода (см. в конце).
   Дополнить код расчета (m+1) весов расчетом (2m+1) весов путем зеркального отражения асимметричной весовой функции относительно нулевого веса. Отобразить полученную симметричную весовую функцию.
2) Используя функцию lpf() в классе PROCESSING реализовать функции для расчета соответственно весов:
   а) hpf() - фильтра высоких ФВЧ;
   б) bpf() - полосового фильтра (ПФ);
   в) bsf() - режекторного фильтра (РФ);
   на основе предоставленных кодов (см. в конце).
   Отобразить все функции графически в разных окнах.
3) Используя функции для расчета амплитудного спектра Фурье в классе ANALYSIS реализовать функцию для расчета частотной характеристики фильтров путем поэлементного умножения рассчитанных значений спектра на длину преобразования, т.е. на (2*m+1).
   Отобразить все функции графически в разных окнах.

Рекомендуемые значения: fc = 50, dt=0.002, m=64; fc1=35, fc2=75.

Код для расчета весов ФНЧ фильтра Поттера

```
lpf(fc, dt, m, *lpw)
{
    const double d[4] = {0.35577019, 0.2436983, 0.07211497, 0.00630165};
   // rectangular part weights
   fact = float(2 * fc * dt);
   lpw[0] = fact;
   arg = fact * M_PI;
   for ( i=1; i <= m; i++ )  lpw[i] = sin(arg*i)/(M_PI*i);
   // trapezoid smoothing at the end
   lpw[m] /= 2.;
   // P310 smoothing window
   sumg = lpw[0];
   for ( i=1; i <= m; i++ ) {
        sum = d[0];
        arg = M_PI * i /m;
        for ( k=1; k <= 3; k++ ) sum += 2. * d[k] * cos(arg*k);
        lpw[i] *= sum;
        sumg += 2 * lpw[i];
   }
        for (i=0; i <= m; i++)  lpw[i] /= sumg;
}
/*** get (2*m+1) weights from (m+1) ***/
```

Коды для расчета весов трех фильтров ФВЧ, ПФ и РФ на основе весов ФНЧ фильтра Поттера

Фильтр ФВЧ

```
// *hpw - weights for HPF;
lpf(fc, dt, m, *lpw);
Loper = 2*m+1;
for (k = 0; k <= Loper; k++)
        if k==m then hpw[k] = 1. - lpw[k] else hpw[k] = - lpw[k];
```

Полосовой фильтр ПФ

```
// *bpw - weights for BPF;       fc1 < fc2;
lpf(fc1, dt, m, *lpw1);
lpf(fc2, dt, m, *lpw2);
Loper = 2*m+1;
for (k = 0; k <= Loper; k++) bpw[k] = lpw2[k] - lpw1[k];
```

Режекторный фильтр РФ

```
// *bsw - weights for BSF;       fc1 < fc2;
lpf(fc1, dt, m, *lpw1);
lpf(fc2, dt, m, *lpw2);
Loper = 2*m+1;
for (k = 0; k <= Loper; k++)
     if k==m then bsw[k] = 1. + lpw1[k] - lpw2[k] else bsw[k] = lpw1[k] - lpw2[k];
```