МИНОБРНАУКИ РОССИИ

ФГБОУ ВО «ИЖЕВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ М. Т. КАЛАШНИКОВА»

Кафедра «Программное обеспечение»

Отчет

по лабораторной работе № 3

по дисциплине

«Теория цифровой обработки сигналов»

|  |  |
| --- | --- |
| Выполнил:  студент гр. Б05-191-1 | Р. И. Мусин |
| Принял:  к.т.н., доцент | И.О.Архипов |

Ижевск 2017

ЦЕЛЬ РАБОТЫ

Овладеть навыками подавления шума в цифровых сигналах.

ХОД ВЫПОЛНЕНИЯ

Сгенерированы три вида сигналов - исходный сигнал без шума (длительность сигнала 100 отсчетов; длительность импульса 20 отсчетов), этот же сигнал с импульсным и гауссовым шумами. Сигналы представлены на рисунках 1-3.

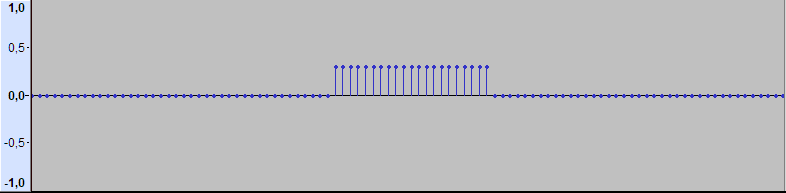


Рис.1. Сигнал без шума

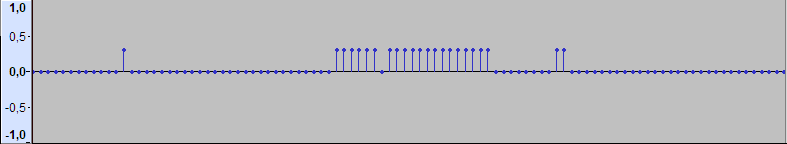


Рис.2. Сигнал с импульсным шумом

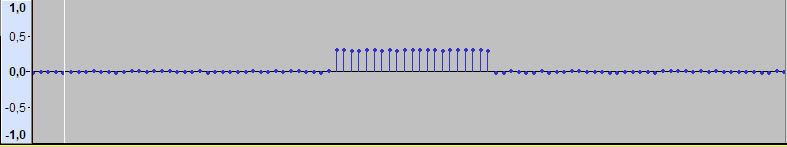


Рис.3. Сигнал с гауссовым шумом

КОД ПРОГРАММЫ

#include **<iostream>**#include **<vector>**#include **<algorithm>  
using namespace** std;  
  
FILE \*inf, \*outf;  
  
**struct** rifftype {  
 **char** id[4];  
 **long** len;  
} IDRiff;  
  
**struct** chucktype {  
 **char** id[4];  
 **char** fmt[4];  
 **long** len;  
} IDChuckWave;  
  
**struct** wavetype {  
 **short** type;  
 **short** channels;  
 **long** SamplesPerSec;  
 **long** AvgBytesPerSec;  
 **short** align;  
 **short** bits;  
} IDWave;  
  
**struct** sampletype {  
 **char** id[4];  
 **long** len;  
} IDSampleWave;  
  
**short** normalNoise(**short** s, **short** max) {  
 **short** dispersion = max / 10;  
 **return** s + rand() % dispersion - max / 20;  
}  
  
**short** pulseNoise(**int** i, **short** s) {  
 **if** (i == 12 || i == 46 || i == 69 || i == 70)  
 **return** s == 0 ? 10000 : 0;  
 **return** s;  
}  
  
**void** generateNoiseSignal() {  
 **for** (**int** i = 0;i < IDSampleWave.len / 2;i++) {  
 **short** s = 40 <= i && i <= 60 ? 10000 : 0;  
 *//short value = pulseNoise(i, s);* **short** value = normalNoise(s, 10000);  
 *//short value = s;* fwrite(&value, 2, 1, outf);  
 }  
}  
  
**void** filter(**bool** movingAverage) {  
 **int** neighbourCount = 5;  
 vector<**short**> allValues;  
 **int** signalLen = IDSampleWave.len / 2;  
 **for** (**int** i = 0; i < signalLen; i++) {  
 **short** buf;  
 fread(&buf, 2, 1, inf);  
 allValues.push\_back(buf);  
 }  
  
 **for** (**int** i = 0; i < neighbourCount / 2; i++)  
 fwrite(&allValues[i], 2, 1, outf);  
  
 **for** (**int** i = neighbourCount / 2; i < signalLen - neighbourCount / 2; i++) {  
 **short** value;  
 **if** (movingAverage) {  
 **int** sum = 0;  
 **for** (**int** j = i - neighbourCount / 2; j <= i + neighbourCount / 2; j++)  
 sum += allValues[j];  
 value = sum / neighbourCount;  
 } **else** {  
 vector<**short**> cur;  
 **for** (**int** j = -neighbourCount / 2; j <= neighbourCount / 2; j++)  
 cur.push\_back(allValues[i + j]);  
 sort(cur.begin(), cur.end());  
 value = cur[neighbourCount / 2];  
 }  
 fwrite(&value, 2, 1, outf);  
 }  
  
 **for** (**int** i = signalLen - neighbourCount / 2; i < signalLen; i++)  
 fwrite(&allValues[i], 2, 1, outf);  
}  
  
**void** readTitles(**char** \*inFileName, **char** \*outFileName) {  
 inf = fopen(inFileName, **"rb"**);  
 outf = fopen(outFileName, **"wb"**);  
 **if** (inf == **nullptr**) **return**;  
  
 fread(&IDRiff, **sizeof**(IDRiff), 1, inf);  
 fread(&IDChuckWave, **sizeof**(IDChuckWave), 1, inf);  
 fread(&IDWave, **sizeof**(IDWave), 1, inf);  
 fread(&IDSampleWave, **sizeof**(IDSampleWave), 1, inf);  
 IDRiff.len = 236;  
 IDSampleWave.len = 200;  
 fwrite(&IDRiff, **sizeof**(IDRiff), 1, outf);  
 fwrite(&IDChuckWave, **sizeof**(IDChuckWave), 1, outf);  
 fwrite(&IDWave, **sizeof**(IDWave), 1, outf);  
 fwrite(&IDSampleWave, **sizeof**(IDSampleWave), 1, outf);  
}  
  
**int** main() {  
 readTitles(**"CK.wav"**, **"mySignal.wav"**);  
 generateNoiseSignal();  
 fclose(inf);  
 fclose(outf);  
  
 readTitles(**"mySignal.wav"**, **"mySignalAverage.wav"**);  
 filter(**true**);  
 fclose(inf);  
 fclose(outf);  
  
 readTitles(**"mySignal.wav"**, **"mySignalMedian.wav"**);  
 filter(**false**);  
 fclose(inf);  
 fclose(outf);  
}

РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

Результаты работы сглаживающих фильтров представлены на рисунках 4-9.

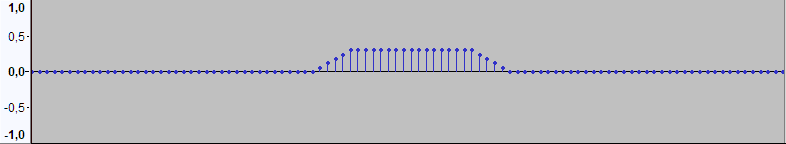


Рис.4. Усредняющий фильтр для обычного сигнала

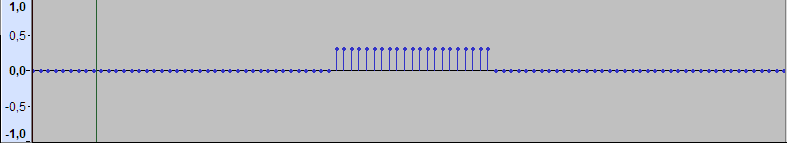


Рис.5. Медианный фильтр для обычного сигнала

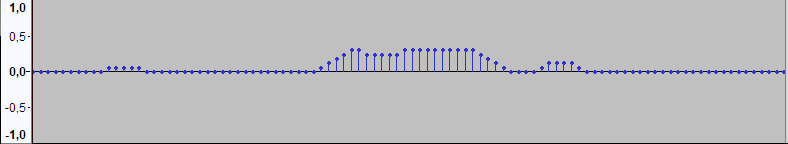


Рис.6. Усредняющий фильтр для сигнала с импульсным шумом

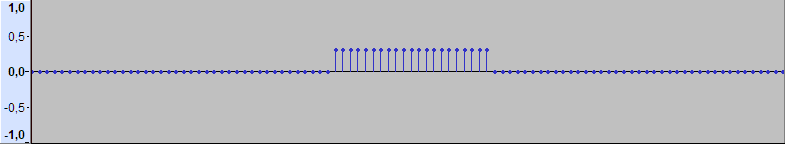


Рис.7. Медианный фильтр для сигнала с импульсным шумом

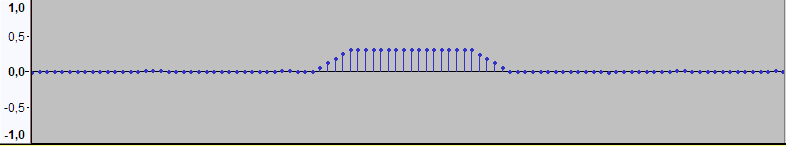


Рис.8. Усредняющий фильтр для сигнала с нормальным шумом

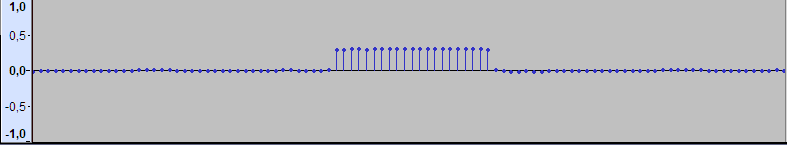


Рис.9. Медианный фильтр для сигнала с нормальным шумом

ВЫВОДЫ

В ходе выполнения лабораторной работы были изучены особенности работы сглаживающих фильтров при обработке сигналов с разными видами шумов.