|  |  |  |  |
| --- | --- | --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования  Белорусский Государственный Университет Информатики и Радиоэлектроники | | | |
| Факультет | Компьютерных сетей и систем | | |
| Кафедра | Информатики  Дисциплина: Конструирование те технологии электронных вычислительных средств | | |
|  |  | | |
| **лабораторная работа**  по курсу ОЭДвСС  на тему “Обработка сигналов и экспериментальных данных” | | | |
| Студент:  гр. 7М2332  Маркусенко Т. С. | |  | Проверил:  Бранцевич П. Ю. |
| Минск, 2017 | | | |

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc499548604)

[постановка задачи 3](#_Toc499548605)

[реализация 5](#_Toc499548606)

[заключение 6](#_Toc499548607)

[приложение 9](#_Toc499548608)

постановка задачи

Разработать проект и реализовать программное средство для обработки цифровых сигналов.

При реализации проекта обеспечить выполнение следующих функций:

1. Работа с файловой системой для выбора файла для обработки.

Файл, содержащий имена файлов, входящих в состав группы имеет расширение txt. Файлы, содержащие данные, имеют расширение bin. Если производится групповой выбор файлов, то для анализа считываются все файлы группы.

1. Отображение графиков временных реализаций.

Необходимо предусмотреть:

* возможность автоматического выбора предела шкалы;
* возможность задания (изменения) верхнего и нижнего пределов шкал для всех графиков одновременно и для каждого графика в отдельности.

На графиках отображаются шкалы.

Для отображения длинных временных реализаций должен быть реализован скользящий просмотр с возможностью задания числа точек смещения графика за одно итерацию.

Для просмотра текущих значений графика реализуется курсор, положение которого управляется оператором. Для каждого зафиксированного положения курсора отображаются значения времени и амплитуды.

1. Вычисление параметров сигналов.

Вычисление параметров сигнала производится на конечном числе дискретных точек N, выбираемых из ряда значений кратных степени двойки:

64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, и т.д. Начальное значение N

выбирается из параметров прочитанного файла.

Вычисляются максимальное и минимальное значение сигнала, размах колебаний, среднее квадратическое значение (СКЗ), пик-фактор.

Вычисление параметров производится начиная с точки, соответствующей началу отображения сигнала в поле графика (условная нулевая точка). Вычисленные параметры отображаются на графике или в специальном поле.

1. Определение амплитудного спектра сигнала.

Осуществляется на том же числе точек *N*, что и вычисление параметров сигнала. Вычисление амплитудного спектра производится с использованием алгоритма дискретного или быстрого преобразования Фурье.

Амплитудные спектры вычисляются для каждого из исследуемых сигналов и отображаются в виде графиков в отдельном окне. Для. Для просмотра и определения значений отдельных спектральных составляющих реализуется графический курсор. Для каждого зафиксированного положения курсора отображаются значения амплитуды и частоты.

1. Построение гистограммы распределения амплитуд исследуемого сигнала.

Задаются: диапазон амплитуд для анализа и количество интервалов. Дополнительно вычисляются эксцесс и асимптота.

В другом окне на графике отображается гистограмма распределения амплитуд исследуемого и параметры сигнала.

реализация

Проект разработан и реализован на языке программирования JavaScript с использованием библиотеки Highcharts.

1. Работа с файловой системой для выбора группы файлов или файла для обработки представлена на рисунках 1.1 и 1.2.



Рис. 1.1 – Поле для выбора файла

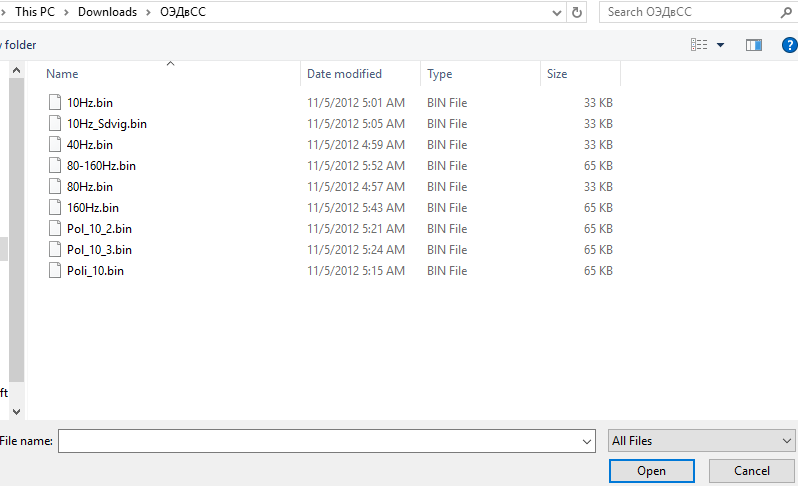


Рис. 1.2 – Окно выбора файла

1. Для отображения графиков для каждого сигнала формируется свое поле отображения. Оригинальный и масштабированный графики представлены на рисунках 1.3 и 1.4 соответственно.

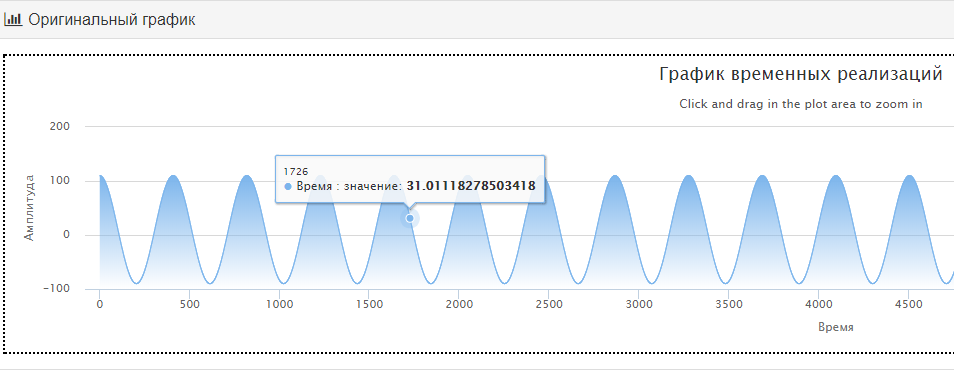
****

Рис. 1.3 – Оригинальный график временных реализаций

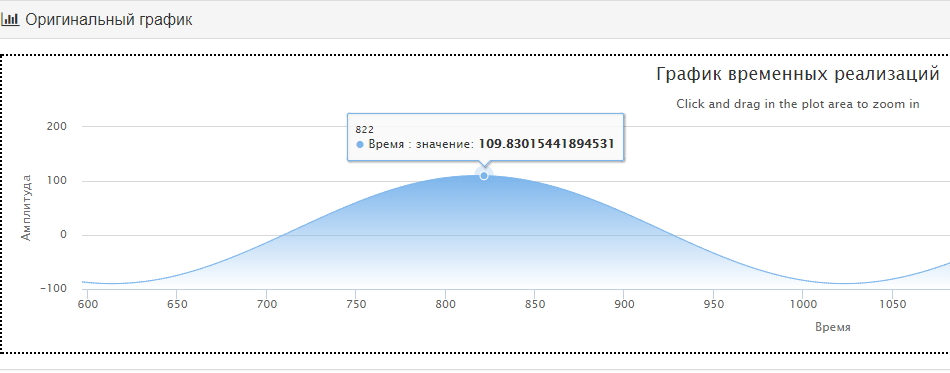


Рис. 1.4 – Масштабированный график временных реализаций

1. Вычисленные параметры сигнала представлены на рисунке 1.5.

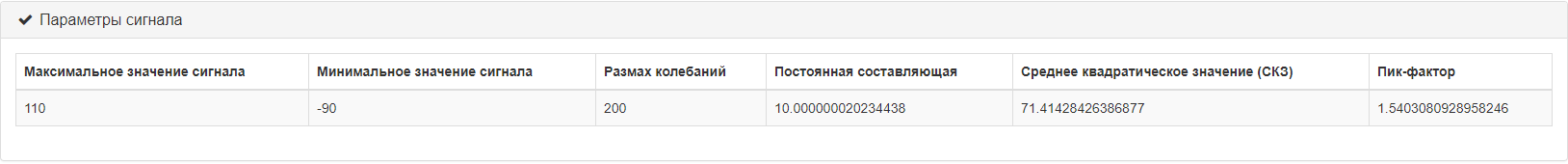


Рис. 1.5 – Параметры сигнала

1. Совокупность всех гармонических составляющих негармонического сигнала называют спектромэтого сигнала. Различают фазовый и амплитудный спектр сигнала:

* фазовый спектр сигнала — совокупность начальных фаз всех гармоник;
* амплитудный спектр сигнала — амплитуды всех гармоник, из которых складывается негармонический сигнал.

Вычисленный с помощью преобразования Фурье амплитудный спектр представлен на рисунке 1.6.

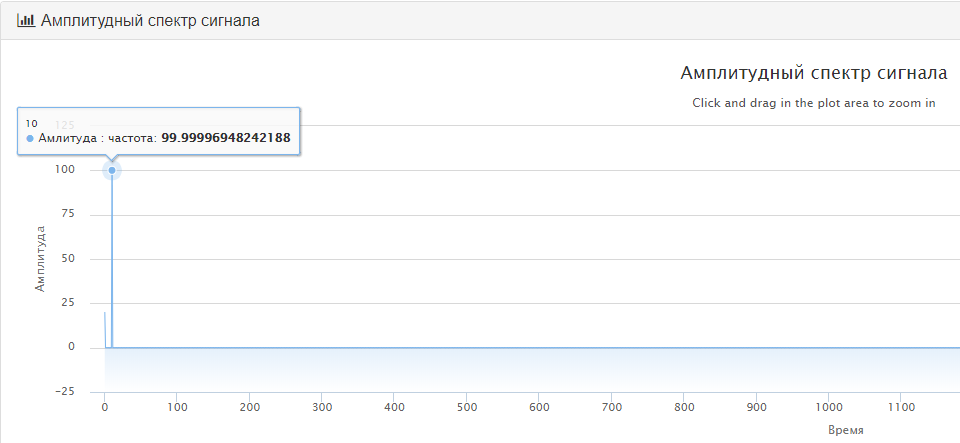


Рис. 1.6 – Амплитудный спектр сигнала

1. На рисунке 1.7 представлена гистограмма распределения амплитуд исследуемого сигнала.

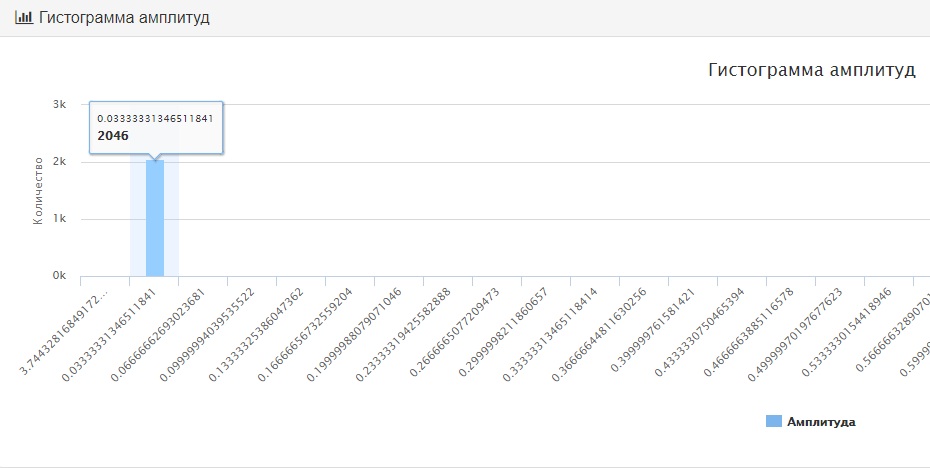


Рис.1.7 – Гистограмма распределения амплитуд исследуемого сигнала

заключение

1. Спектральный анализ – один из методов обработки сигналов, который позволяет охарактеризовать частотный состав измеряемого сигнала;
2. Математической основой спектрального анализа сигналов является преобразование Фурье;
3. Преобразование Фурье позволяет представить непрерывную функцию f(x) (сигнал), определенную на отрезке {0, T} в виде суммы бесконечного числа (бесконечного ряда) тригонометрических функций (синусоид и\или косинусоид) с определёнными амплитудами и фазами, также рассматриваемых на отрезке {0, T}. Такой ряд называется рядом Фурье;
4. Преобразование Фурье используется во многих областях науки — в физике, теории чисел, комбинаторике, обработке сигналов, теории вероятностей, статистике, криптографии, акустике, океанологии, оптике, геометрии и многих других. В обработке сигналов и связанных областях преобразование Фурье обычно рассматривается как декомпозиция сигнала на частоты и амплитуды, то есть обратимый переход от временно́го пространства (time domain) в частотное пространство (frequency domain).

приложение

**var *start*** = 0;  
**var *resultInt*** = [],  
 ***resultFloat*** = [];  
**var *flag*** = 0;  
  
  
*// Функция для чтения файла***function** *read*(i) {  
 **var** fr = **new FileReader**(),  
 file = **document**.querySelector(**'input'**).**files**[i];  
  
 **if** (!file) {  
 alert(**'Выберите, пожалуйста, файл!'**);  
 **return**;  
 }  
  
 fr.**onloadend** = **function**(e) {  
 **if** (e.**target**.**readyState** == **FileReader**.**DONE**) {  
 **var** tmp = e.**target**.**result**;  
 ***resultInt***[i] = (**new** Int32Array(tmp));  
 ***resultFloat***[i] = (**new** Float32Array(tmp));  
 }  
 ***flag***++;  
 *generateWorkObjects*();  
 }  
 fr.readAsArrayBuffer(file);  
}  
  
  
  
  
*// вычисление спектра***function** *generateSpectre*(data) {  
 **var** fft = **new** FFT(data.**selectionSize**, data.**frequencyResolution**);  
 fft.forward(data.**data**.slice(0, data.**selectionSize**));  
 **var** spectrum = fft.spectrum;  
 **var** res = [];  
 **for** (**var** i = 0; i < spectrum.**length**; i++) {  
 res[i] = [i, spectrum[i]];  
 }  
 *getHighchart*(**'#amplitude-area-chart'**, **'Амплитудный спектр сигнала'**, **'Амлитуда : частота'**, res);  
  
  
 **var** real = fft.real;  
 **var** imag = fft.imag;  
 **for** (**var** i = 0; i < real.**length** / 4; i++) {  
 real[i] = 0;  
 }  
 **for** (**var** i = real.**length** - real.**length** / 4; i < real.**length**; i++) {  
 real[i] = 0;  
 }  
  
 **var** buffer = fft.inverse(real, imag);  
 **var** res1 = [];  
 **for** (**var** i = 0; i < buffer.**length**; i++) {  
 res1[i] = [i, buffer[i]];  
 }  
 *getHighchart*(**'#original-chart'**, **'Отфильтрованный сигнал'**, **'Время : значение'**, res1);  
  
 *calculateParams*(buffer, **'#filteredParams'**, buffer.**length**);  
  
 *getHistogram*(spectrum);  
}  
  
*// вычисление параметров сигнала***function** *calculateParams*(data, selector, selectionSize) {  
 **var** min = **max** = data[0];  
 **for** (**var** i = 0; i < selectionSize && i < data.**length**; i++) {  
 **if** (min > data[i]) {  
 min = data[i];  
 }  
 **if** (**max** < data[i]) {  
 **max** = data[i];  
 }  
 }  
 **var** razmah = **max** - min;  
 **var** constant = 0.0;  
 **for** (**var** i = 0; i < selectionSize && i < data.**length**; i++) {  
 constant += data[i];  
 }  
 constant /= selectionSize;  
  
 **var** skz = 0.0;  
 **for** (**var** i = 0; i < selectionSize && i < data.**length**; i++) {  
 skz += data[i] \* data[i];  
 }  
 skz = ***Math***.sqrt(skz / selectionSize);  
  
 **var** pik = ***Math***.max(***Math***.abs(min), ***Math***.abs(**max**)) / skz;  
  
 $(selector + **' .max'**).**text**(**max**);  
 $(selector + **' .min'**).**text**(min);  
 $(selector + **' .razmah'**).**text**(razmah);  
 $(selector + **' .constant'**).**text**(constant);  
 $(selector + **' .skz'**).**text**(skz);  
 $(selector + **' .pik'**).**text**(pik);  
}  
  
*// рисование графика***function** *getHighchart*(selector, title, series, data) {  
 $(selector).highcharts({  
 **chart**: {  
 **zoomType**: **'x'** },  
 **title**: {  
 **text**: title  
 },  
 **subtitle**: {  
 **text**: **document**.ontouchstart === **undefined** ?  
 **'Click and drag in the plot area to zoom in'** : **'Pinch the chart to zoom in'** },  
 **xAxis**: {  
 **title**: {  
 **text**: **'Время'** },  
 **type**: **'number'** },  
 **yAxis**: {  
 **title**: {  
 **text**: **'Амплитуда'** }  
 },  
 **legend**: {  
 **enabled**: **false** },  
 **plotOptions**: {  
 **area**: {  
 **fillColor**: {  
 **linearGradient**: {  
 **x1**: 0,  
 **y1**: 0,  
 **x2**: 0,  
 **y2**: 1  
 },  
 **stops**: [  
 [0, Highcharts.getOptions().colors[0]],  
 [1, Highcharts.Color(Highcharts.getOptions().colors[0]).setOpacity(0).**get**(**'rgba'**)]  
 ]  
 },  
 **marker**: {  
 **radius**: 2  
 },  
 **lineWidth**: 1,  
 **states**: {  
 **hover**: {  
 **lineWidth**: 1  
 }  
 },  
 **threshold**: **null** }  
 },  
  
 **series**: [{  
 **type**: **'area'**,  
 **name**: series,  
 **data**: data  
 }]  
 });  
}  
  
**function** *getHistogram*(data) {  
 **var** min = **max** = data[0];  
 **for** (**var** i = 0; i < data.**length**; i++) {  
 **if** (min > data[i]) {  
 min = data[i];  
 }  
 **if** (**max** < data[i]) {  
 **max** = data[i];  
 }  
 }  
  
 **var** cat = [],  
 result = [];  
 **for** (**var** tmp = min; tmp < **max** + (**max** - min) / 31; tmp += (**max** - min) / 30) {  
 cat.push(tmp);  
 result.push(0);  
 }  
  
 **for** (**var** i = 0; i < data.**length**; i++) {  
 **for** (**var** j = 0; j < cat.**length**; j++) {  
 **if** (data[i] <= cat[j]) {  
 result[j]++;  
 **break**;  
 }  
 }  
 }  
  
  
 $(**'#histogramm'**).highcharts({  
 **chart**: {  
 **type**: **'column'** },  
 **title**: {  
 **text**: **'Гистограмма амплитуд'** },  
 **xAxis**: {  
 **categories**: cat,  
 **crosshair**: **true** },  
 **yAxis**: {  
 **min**: 0,  
 **title**: {  
 **text**: **'Количество'** }  
 },  
 **tooltip**: {  
 **headerFormat**: **'<span style="font-size:10px">{point.key}</span><table>'**,  
 **pointFormat**: **'<tr><td style="color:{series.color};padding:0"></td>'** +  
 **'<td style="padding:0"><b>{point.y:.0f}</b></td></tr>'**,  
 **footerFormat**: **'</table>'**,  
 **shared**: **true**,  
 **useHTML**: **true** },  
 **plotOptions**: {  
 **column**: {  
 **pointPadding**: 0.2,  
 **borderWidth**: 0  
 }  
 },  
 **series**: [{  
 **name**: **'Амплитуда'**,  
 **data**: result  
 }]  
 });  
}  
  
*// генерирование рабочих объектов***function** *generateWorkObjects*() {  
 **if** (***flag*** < **document**.querySelector(**'input'**).**files**.**length**) {  
 **return**;  
 }  
 **var** result = [];  
 **var** detailChart;  
 **for** (**var** i = 0; i < ***resultInt***.**length**; i++) {  
 result[i] = {  
 **signature**: **"TMB1"**,  
 **channelCount**: ***resultInt***[i][1],  
 **selectionSize**: ***resultInt***[i][2],  
 **spectralLineCount**: ***resultInt***[i][3],  
 **srezFreq**: ***resultInt***[i][4],  
 **frequencyResolution**: ***resultFloat***[i][5],  
 **blockTime**: ***resultFloat***[i][6],  
 **totalTime**: ***resultInt***[i][7],  
 **gotBlocksCountUser**: ***resultInt***[i][8],  
 **dataSize**: ***resultInt***[i][9],  
 **gotBlocksCountSystem**: ***resultInt***[i][10],  
 **maxValue**: ***resultFloat***[i][11],  
 **minValue**: ***resultFloat***[i][12],  
 **data**: ***resultFloat***[i].slice(13),  
 **dataXY**: []  
 }  
 **for** (**var** j = 0; j < result[i].**data**.**length**; j++) {  
 result[i].**dataXY**[j] = [result[i].**blockTime** \* j, result[i].**data**[j]];  
 }  
 }  
 **console**.log(result);  
  
 *getHighchart*(**'#dropzone'**, **'График временных реализаций'**, **'Время : значение'**, result[0].**dataXY**);  
 *calculateParams*(result[0].**data**, **'#signalParams'**, result[0].**selectionSize**);  
 *generateSpectre*(result[0]);  
}  
  
*// обработчик выбора файлов***function** *onFilesSelect*(e) {  
 **var** files = e.**target**.**files**,  
 fr,  
 file,  
 data;  
  
 ***flag*** = 0;  
 **for** (**var** i = 0; i < files.**length**; i++) {  
 file = files[i];  
 *read*(i);  
 }  
}  
  
*// отслеживание появления файлов***if** (**window**.**File** && **window**.**FileReader** && **window**.**FileList** && **window**.**Blob**) {  
 onload = **function**() {  
 **document**.querySelector(**'input'**).addEventListener(**'change'**, *onFilesSelect*, **false**);  
  
 **var** dropZone = **document**.getElementById(**'dropzone'**);  
 dropZone.addEventListener(**'dragover'**, **function**(e) {  
 e.stopPropagation();  
 e.preventDefault();  
 e.dataTransfer.**dropEffect** = **'copy'**;  
 }, **false**);  
 dropZone.addEventListener(**'drop'**, **function**(e) {  
 e.stopPropagation();  
 e.preventDefault();  
 *onFilesSelect*(e);  
 }, **false**);  
 }  
 *// если нет, то предупреждаем, что демо работать не будет*} **else** {  
 alert(**'К сожалению ваш браузер не поддерживает file API'**);  
}