Московский авиационный институт (национальный исследовательский университет)

Факультет компьютерных наук и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Дискретный анализ»

Студент: О. А. Мезенин
Преподаватель: Н. К. Макаров
Группа: М8О-306Б-21
Дата:
Оценка:
Подпись:

Москва, 2023

Курсовой проект

Задача: Реализуйте алгоритм быстрое преобразование Фурье для действительного сигнала.

Преобразование проводится скользящим окном на 4096 отсчёта с шагом 1024. Перед преобразованием Фурье необходимо подействовать на отсчёты окном Ханна.

1 Описание

1 Преобразование Фурье

Преобразование Фурье – операция, сопоставляющая одной функции другую функцию. Общая формула выглядит следующим образом:

$$\hat{f}(\omega) = \sqrt{\frac{|b|}{(2\pi)^{1-a}}} \int_{-\infty}^{\infty} f(x) e^{-ibx\omega} dx,$$

где a и b – параметры.
При выборе $a = 0$, $b = -2\pi$, формула становится проще:

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(x) e^{2\pi i x\omega} dx.$$

Дискретное преобразование Фурье (ДПФ) имеет вид:

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{2\pi i}{N} nk} = \sum_{n=0}^{N-1} x_n \left(\cos\left(\frac{2\pi nk}{N}\right) + i \sin\left(\frac{2\pi nk}{N}\right) \right), \ k = 0, \ldots, N-1$$

2 Применение

Рассмотрим применение преобразования Фурье для обработки аудиосигнала. Как известно, аудиосигнал состоит из нескольких одночастотных звуковых волн. При записи звука фиксируются только результирующие амплитуды этих нескольких волн. Преобразование Фурье помогает разложить сигнал на составляющие его частоты.

3 Оконное преобразование Фурье

На практике входной сигнал разделяют на интервалы, или окна. Оконное дискретное преобразование будет иметь вид:

$$X_{k,m} = \sum_{n=-\infty}^{\infty} x_n \omega[n-m] e^{2\pi i n k},$$

где $\omega[n-m]$ – некоторая оконная функция.
Если просто ограничивать интервал, то оконная функция будет прямоугольная. Результатом такого преобразования является не спектр исходного сигнала, а спектр произведения сигнала и оконной функции. В результате боковые высокие амплитуды могут маскировать присутствие меньших амплитуд. Для решения этой проблемы

используют более гладкие оконные функции.
Одной из таких функции является окно Ханна:

$$\omega(n) = 0.5 * \Big(1 - \cos\Big(\frac{2\pi n}{N-1}\Big)\Big),$$

где N – размер окна.

4 Быстрое преобразование Фурье

Быстрое преобразование Фурье (fast Fourier transform) – это метод, позволяющий вычислить дискретное преобразование Фурье (ДПФ) за время $O(n \log n)$. Основная идея БПФ заключается в разделении вектора коэффициентов на два вектора, рекурсивном вычислении ДПФ для них, и объединении результатов в одно БПФ.
Пусть имеется многочлен $A(x)$ степени n, где n – степень двойки и $n > 1$:

$$A(x) = a_0 x^0 + a_1 x^1 + \cdots + a_{n-1} x^{n-1}.$$

Разделим его на два многочлена, один — с чётными, а другой — с нечётными коэффициентами:

$$A_0(x) = a_0 x^0 + a_2 x^1 + \cdots + a_{n-2} x^{\frac{n}{2}-1},$$
$$A_1(x) = a_1 x^0 + a_3 x^1 + \cdots + a_{n-1} x^{\frac{n}{2}-1}.$$

Нетрудно заметить, что:

$$A(x) = A_0(x^2) + xA_1(x^2). \tag{1}$$

Пусть $\{y_k^0\}_{k=0}^{n/2-1} = DFT(A_0)$ и $\{y_k^1\}_{k=0}^{n/2-1} = DFT(A_1)$. Найдём выражения для $\{y_k\}_{k=0}^{n-1} = DFT(A)$.
Во-первых, вспоминая (1), мы сразу получаем значения для первой половины коэффициентов:

$$y_k = y_k^0 + \omega_n^k y_k^1,\ k = 0 \ldots n/2 - 1.$$

Для второй половины коэффициентов после преобразований также получаем простую формулу:

$$y_{k+n/2} = A(\omega_n^{k+n/2}) = A_0(\omega_n^{2k+n}) + \omega_n^{k+n/2} A_1(\omega_n^{2k+n}) = A_0(\omega_n^{2k}\omega_n^n) + \omega_n^k \omega_n^{n/2} A_1(\omega_n^{2k}\omega_n^n) =$$
$$A_0(\omega_n^{2k}) - \omega_n^k A_1(\omega_n^{2k}) = y_k^0 - \omega_n^k y_k^1.$$

В результате мы получили формулы для вычисления всего вектора $\{y_k\}$:

$$y_k = y_k^0 + \omega_n^k y_k^1,\ k = 0 \ldots n/2 - 1,$$
$$y_{k+n/2} = y_k^0 - \omega_n^k y_k^1,\ k = 0, \ldots n/2 - 1.$$

Таким образом, мы получаем алгоритм БПФ: мы рекурсивно вычисляем $DFT(A_0)$ и $DFT(A_1)$ и из них за линейное время строим $DFT(A)$ – это схема «разделяй и властвуй», которая работает за $O(n \log n)$.

2 Исходный код

Функция `main` считывает входные значения амлитуд, затем проходится окном по ним: применяет окно Ханна, затем к преобразованным данным применяет БПФ, после чего достаёт максимум из каждого такого преобразованного окна данных.

```
int main() {
    std::ios_base::sync_with_stdio(false);
    std::cin.tie(nullptr);
    std::vector<short> dataIn = decoder();

    size_t n = dataIn.size();
    for (size_t i = 0; i+WINDOW_SIZE < n; i += WINDOW_STEP) {
        std::vector<complex> transformData = hunn(dataIn, i, i+WINDOW_SIZE);
        fft(transformData);
        double answer = getMax(transformData);
        std::cout << std::setprecision(26) << answer << '\n';
    }
    return 0;
}
```

Функция `decoder` читает mp3-файл и достаёт значения амлитуд.

```
std::vector<short> decoder() {
    mp3dec_t mp3d;
    mp3dec_file_info_t info;
    if (mp3dec_load(&mp3d, "input.mp3", &info, NULL, NULL)) {
        throw std::runtime_error("Decode error");
    }
    std::vector<short> res(info.buffer, info.buffer + info.samples);
    free(info.buffer);
    return res;
}
```

Реализация окна Ханна приведена в функции `hunn`.

```
std::vector<complex> hunn(const std::vector<short>& data, size_t start, size_t end) {
    std::vector<complex> res(end-start);
    for (size_t i = start; i < end; ++i) {
        res[i-start] = data[i] * 0.5 * (1 - cos(2*M_PI*double(i-start)/double(end-start
            -1)));
    }
    return res;
}
```

Алгоритм Быстрого преобразования Фурье приведён в рекурсивной функции `fft`.

```
void fft(std::vector<complex> & data) {
    size_t n = data.size();
    if (n == 1) return;

```

```
    std::vector<complex> even (n / 2), odd (n / 2);
    for (size_t i=0; i<n/2; ++i) {
        even[i] = data[2*i];
        odd[i] = data[2*i + 1];
    }
    fft (even);
    fft (odd);

    double ang = 2*M_PI/double(n);
    complex w (1), wn (cos(ang), sin(ang));
    for (size_t i=0; i<n/2; ++i) {
        data[i] = even[i] + w * odd[i];
        data[i+n/2] = even[i] - w * odd[i];
        w *= wn;
    }
}
```

Функция `getMax` достаёт из промежутка комплексных чисел максимум.

```
double getMax(const std::vector<complex>& data) {
    double res = abs(data[0]);
    for (size_t i = 1; i < data.size(); ++i) {
        if (abs(data[i]) > res) {
            res = abs(data[i]);
        }
    }
    return res;
}
```

3 Консоль

```
aprold@SAI:~/Documents/GitHub/MAI-DA/CPfor3$ make cp
g++ -std=c++2a -pedantic -Wall -Wextra -Werror main.cpp -o cp.out
aprold@SAI:~/Documents/GitHub/MAI-DA/CPfor3$ file input.mp3
input.mp3: Audio file with ID3 version 2.4.0,contains: MPEG ADTS,layer III,v1,56
kbps,44.1 kHz,Monaural
aprold@SAI:~/Documents/GitHub/MAI-DA/CPfor3$ ./cp.out | head
3921998.1086264750920236111
5495942.4045716114342212677
5803943.0759099461138248444
5598042.5314552430063486099
5032736.2881987979635596275
4188627.4532871609553694725
3592698.5686036236584186554
5467929.2396295610815286636
7220862.0855080047622323036
6006171.4280391717329621315
```

4 Тест производительности

Будет два файла: с частотой дискретизации 44кГц и длительностью 6 секунд и с частотой дискретизации 44кГц и длительностью 264 секунды.

```
aprold@SAI:~/Documents/GitHub/MAI-DA/CPfor3$ file input.mp3
input.mp3: Audio file with ID3 version 2.4.0,contains: MPEG ADTS,layer III,v1,56
kbps,44.1 kHz,Monaural
aprold@SAI:~/Documents/GitHub/MAI-DA/CPfor3$ file "Craftor -Space Flight.mp3"
Craftor -Space Flight.mp3: Audio file with ID3 version 2.3.0,contains: MPEG
ADTS,layer III,v1,320 kbps,44.1 kHz,JntStereo
aprold@SAI:~/Documents/GitHub/MAI-DA/CPfor3$ time ./cp.out input.mp3 >/dev/null

real    0m1,015s
user    0m1,011s
sys     0m0,004s
aprold@SAI:~/Documents/GitHub/MAI-DA/CPfor3$ time ./cp.out "Craftor -Space
Flight.mp3" >/dev/null

real    1m23,736s
user    1m23,629s
sys     0m0,088s
```

5 Выводы

Выполнив курсовой проект по курсу «Дискретный анализ», узнал про преобразование Фурье и его разновидность – оконное преобразование, в котором для сглаживания спектра при делении сигнала на интервалы применяется оконная функция, например, окно Ханна. Узнал про быстрое преобразование Фурье, которое вычисляет дискретное преобразование Фурье за время $O(n \log n)$. А также узнал об одном из применении преобразования Фурье в обработки сигнала: оно позволяет разложить сигнал на составляющие его частоты.

Список литературы

[1] *Преобразование Фурье*
URL: `https://ru.wikipedia.org/wiki/Преобразование_Фурье` (дата обращения: 12.11.2023).

[2] *Дискретное преобразование Фурье*
URL: `https://ru.wikipedia.org/wiki/Дискретное_преобразование_Фурье` (дата обращения: 12.11.2023).

[3] *Оконное преобразование Фурье*
URL: `https://ru.wikipedia.org/wiki/Оконное_преобразование_Фурье` (дата обращения: 12.11.2023).

[4] *Быстрое преобразование Фурье за O (N log N)*
URL: `https://e-maxx.ru/algo/fft_multiply` (дата обращения: 12.11.2023).

[5] *Introduction to the Fast Fourier Transform Algorithm in C++*
URL: `https://cpp.helpful.codes/algorithms/FFT-Algorithm/` (дата обращения: 12.11.2023).

[6] *Проектирование оконных функций, суммирующихся в единицу с заданным уровнем перекрытия*
URL: `https://habr.com/ru/articles/430536/` (дата обращения: 12.11.2023).

[7] *Понимание аудиоданных, преобразования Фурье, БПФ, спектрограммы и распознавания речи*
URL: `https://questu.ru/articles/276549/` (дата обращения: 12.11.2023).