**Лабораторная работа №4**

**Использование преобразования Фурье в обработке изображений**

**Титов Виктор Викторович**

**e-mail: vtitov@rtc.ru**

***Цель:*** изучить применение преобразования Фурье к обработке и .

Задачи:

1. Ознакомиться с базовой теорией преобразования Фурье
2. Ознакомиться с методами вычисления преобразования Фурье
3. Выполнить практическое задание в конце

*Ключевые слова*: ряд Фурье, преобразование Фурье, образ, прообраз, БФП.

**ТЕОРИЯ**

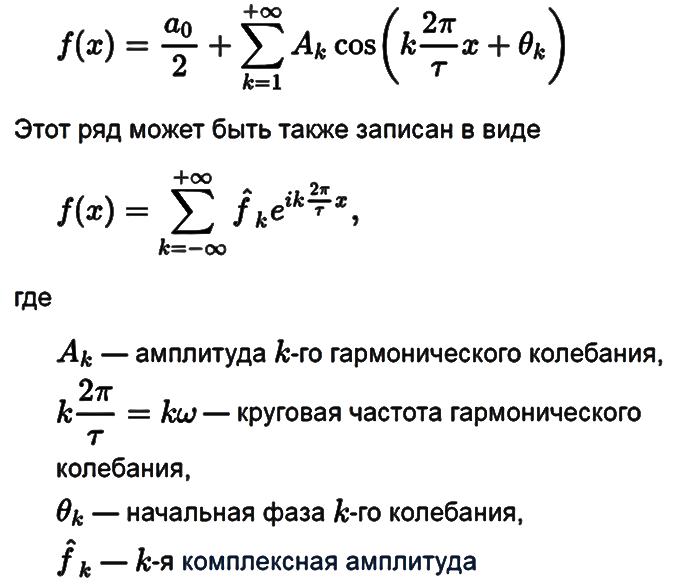
Ряд Фурье назван в честь французского математика [Жана-Батиста Жозефа Фурье](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D1%80%D1%8C%D0%B5,_%D0%96%D0%B0%D0%BD-%D0%91%D0%B0%D1%82%D0%B8%D1%81%D1%82_%D0%96%D0%BE%D0%B7%D0%B5%D1%84) (1768—1830), внесшего важный вклад в изучение тригонометрических рядов после предварительных сследований [Леонарда Эйлера](https://ru.wikipedia.org/wiki/%D0%AD%D0%B9%D0%BB%D0%B5%D1%80,_%D0%9B%D0%B5%D0%BE%D0%BD%D0%B0%D1%80%D0%B4), [Жана Лерона д’Аламбера](https://ru.wikipedia.org/wiki/%D0%94%E2%80%99%D0%90%D0%BB%D0%B0%D0%BC%D0%B1%D0%B5%D1%80,_%D0%96%D0%B0%D0%BD_%D0%9B%D0%B5%D1%80%D0%BE%D0%BD) и [Даниила Бернулли](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%80%D0%BD%D1%83%D0%BB%D0%BB%D0%B8,_%D0%94%D0%B0%D0%BD%D0%B8%D0%B8%D0%BB). Фурье представил ряд с целью решения [уравнения теплопроводности](https://ru.wikipedia.org/wiki/%D0%A3%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D1%82%D0%B5%D0%BF%D0%BB%D0%BE%D0%BF%D1%80%D0%BE%D0%B2%D0%BE%D0%B4%D0%BD%D0%BE%D1%81%D1%82%D0%B8) в металлической пластине, написав свои первоначальные результаты в своем «Воспоминании о распространении тепла в твердых телах» («Трактат о распространении тепла в твердых телах») в 1822 году. Благодаря исследованиям Фурье был установлен факт того, что **произвольная (непрерывная) функция может быть представлена ​​тригонометрическим рядом**. Первое объявление об этом великом открытии было сделано Фурье в 1807 году перед Французской академией. Ранние идеи разложения периодической функции на сумму простых осциллирующих функций относятся к 3 веку до нашей эры, когда древние астрономы предложили эмпирическую модель движения планет, основанную на семействах и эпициклах.

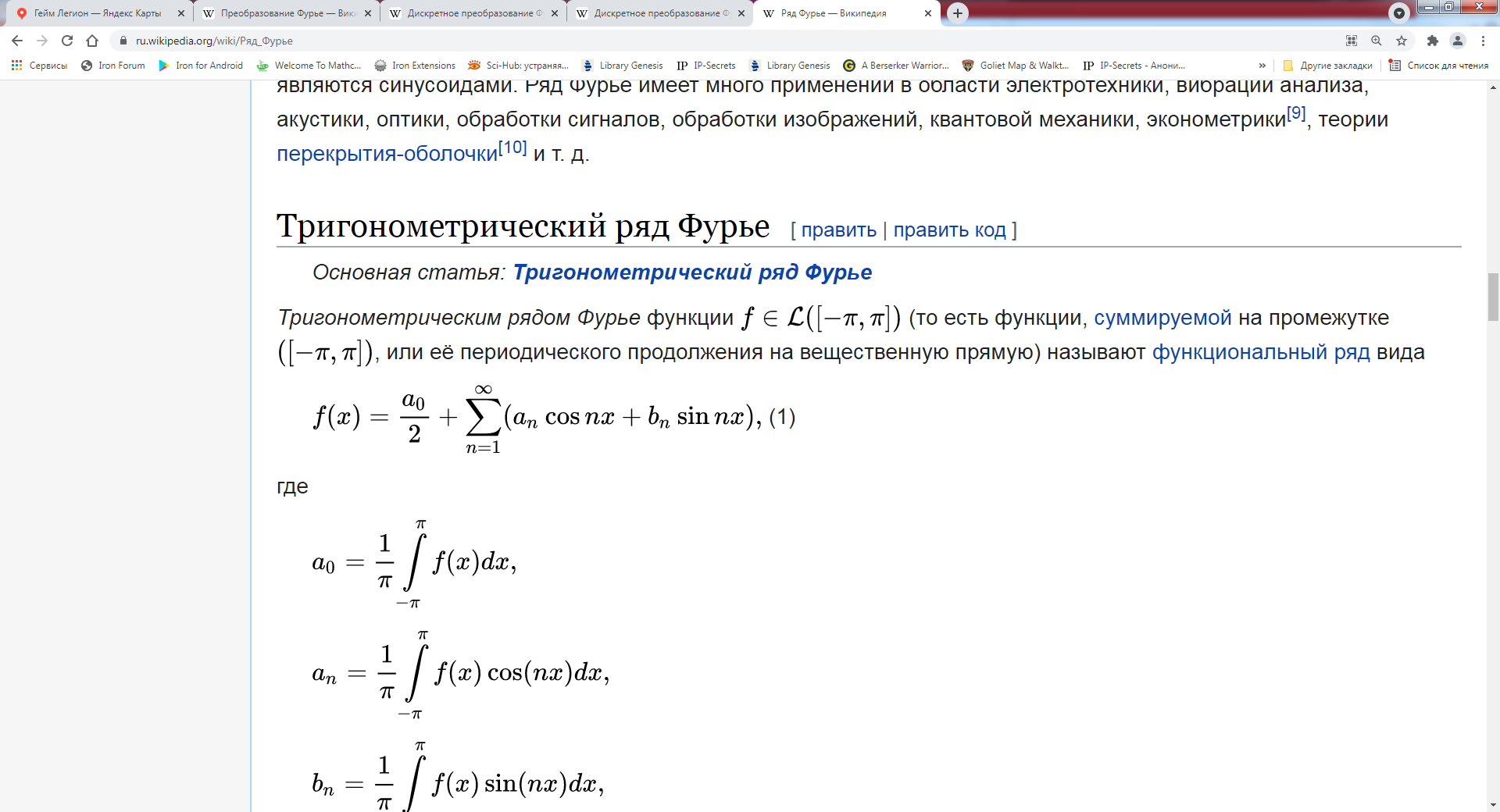
Уравнение теплопроводности является уравнением в частных производных. До работы Фурье в общем случае не было известно решение уравнения теплопроводности, хотя были известны конкретные решения, если бы источник тепла вел себя простым образом, в частности, если источником тепла была волна синуса или косинуса. Эти простые решения теперь иногда называют **собственными решениями**. Идея Фурье состояла в том, чтобы **смоделировать сложный источник тепла как суперпозицию (или линейную комбинацию) простых синусоидальных и косинусных волн и записать решение как суперпозицию соответствующих собственных решений**. Эта суперпозиция или линейная комбинация называется рядом Фурье.

С современной точки зрения, результаты Фурье несколько неформальны из-за отсутствия точного понятия функции и интеграла в начале девятнадцатого века. Позднее [Петер Густав Лежён Дирихле](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%80%D0%B8%D1%85%D0%BB%D0%B5,_%D0%9F%D0%B5%D1%82%D0%B5%D1%80_%D0%93%D1%83%D1%81%D1%82%D0%B0%D0%B2_%D0%9B%D0%B5%D0%B6%D1%91%D0%BD) и [Бернхард Риман](https://ru.wikipedia.org/wiki/%D0%A0%D0%B8%D0%BC%D0%B0%D0%BD,_%D0%91%D0%B5%D1%80%D0%BD%D1%85%D0%B0%D1%80%D0%B4" \o "Риман, Бернхард) выразили результаты Фурье с большей точностью и формальностью.

Хотя первоначальной мотивацией было решение уравнения теплопроводности, позже стало очевидно, что те же методы можно применять к широкому кругу математических и физических задач, особенно тех, которые включают линейные дифференциальные уравнения с постоянными коэффициентами, для которых собственные решения являются синусоидами [https://ru.wikipedia.org/wiki/%D0%A0%D1%8F%D0%B4\_%D0%A4%D1%83%D1%80%D1%8C%D0%B5].

Согласно теории разложения в ряд Фурье





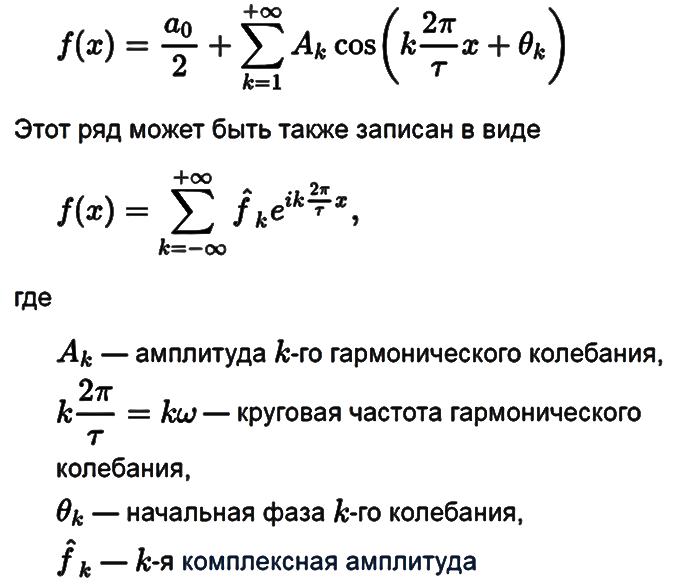
 - амплитуды гармоник

– частоты гармоник

 – фазы гармоник

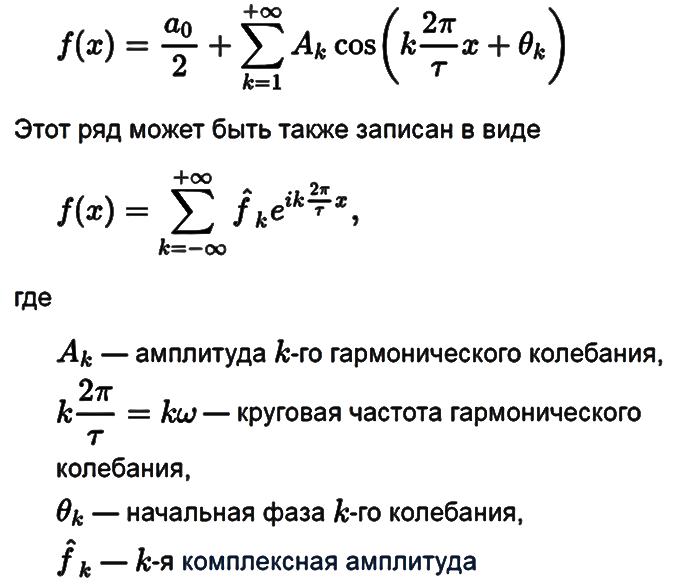


или в комплексной форме



 – комплексная форма коэффициентов ряда Фурье

Следует отметить, что не все функции могут быть разложены в ряд Фурье, а только:

* периодические (в формулах выше с периодом π) [подумать, как изменить период в формулах выше]
* функции принадлежат множеству интегрируемых функций 
* доказано, что ряд  сходится для функций, интегрируемых в квадрате 

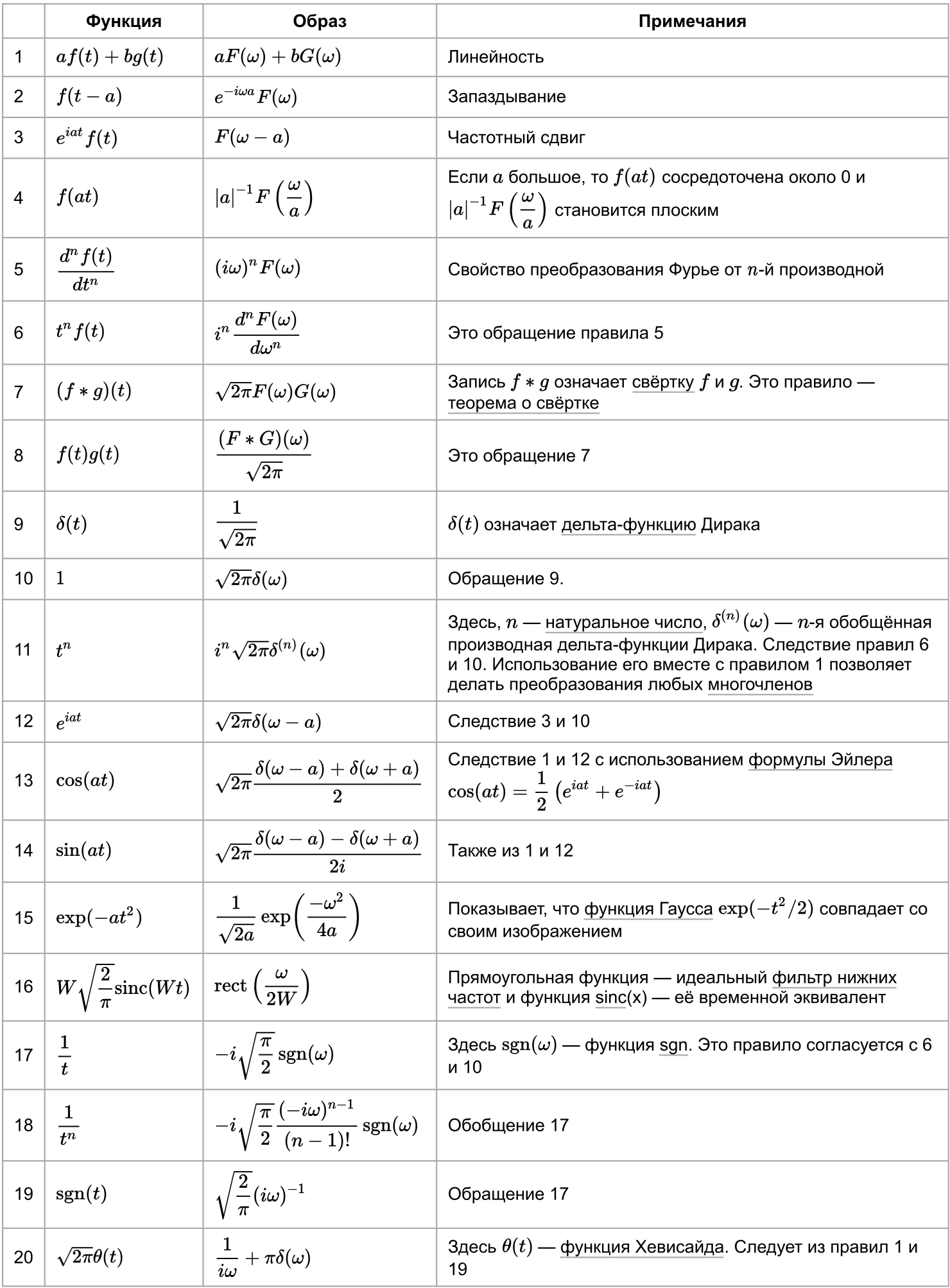
Предельный переход при стремлении периода функции Т к бесконечности, а разницы между соседними частотами к нулю, дает преобразование Фурье

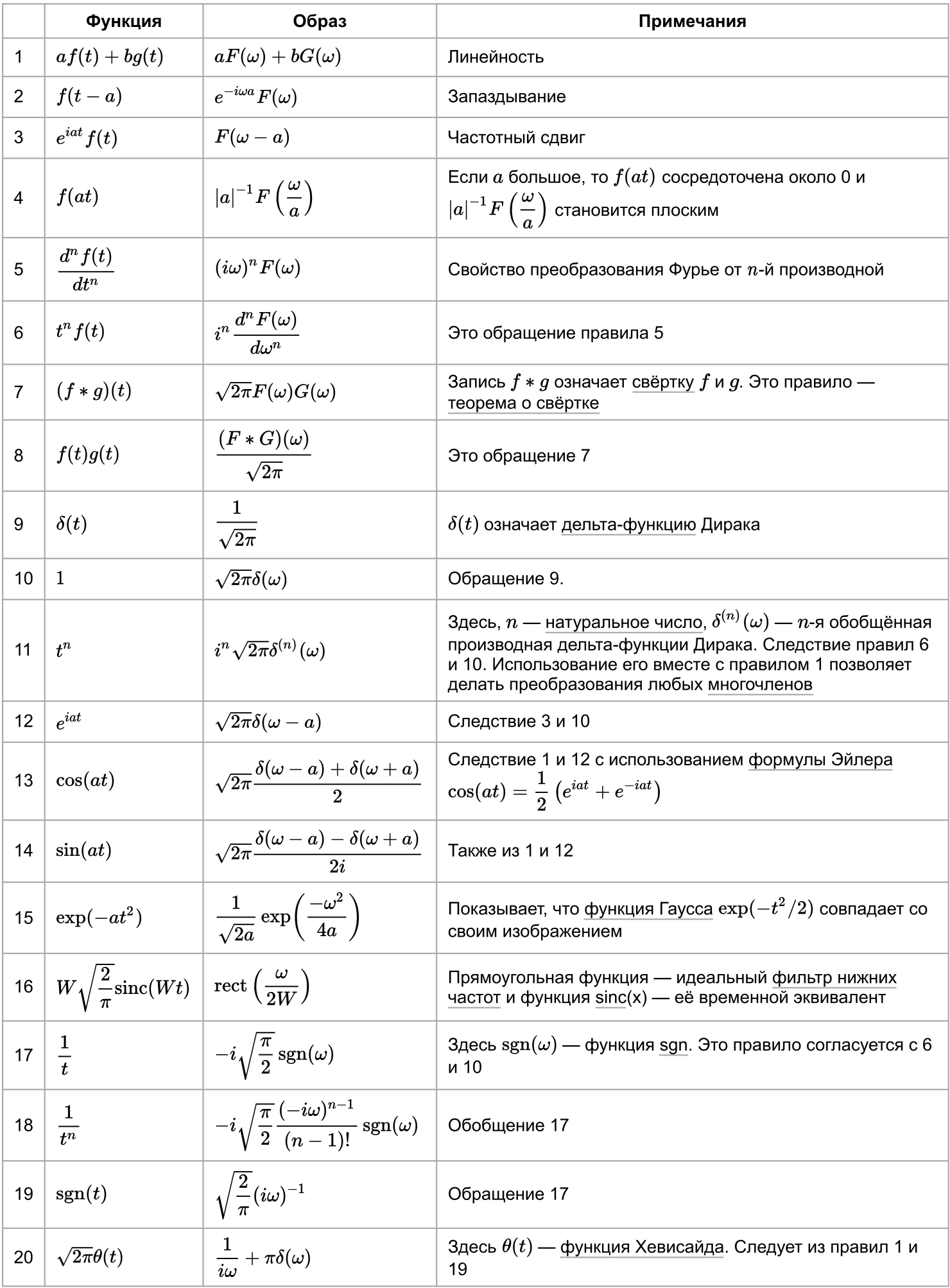


С обратным преобразованием



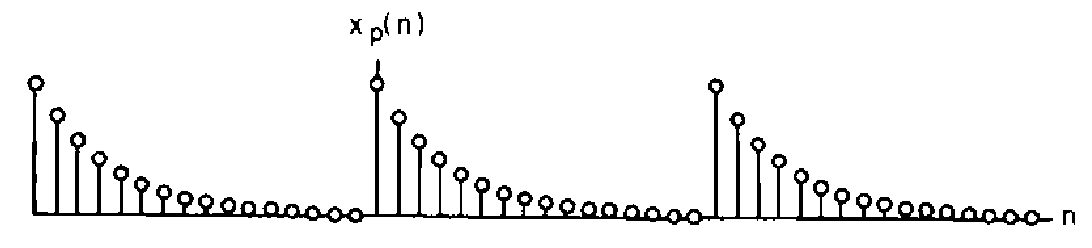
Основные свойства преобразования Фурье [https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5\_%D0%A4%D1%83%D1%80%D1%8C%D0%B5]





Данное преобразование назовем непрерывным во времени преобразованием кусочно-непрерывных функций (или просто **непрерывным преобразованием Фурье**).

Кроме непрерывных сигналов, часто встречающимися сигналами являются дискретные во времени сигналы, которые могут быть как периодическими, так и нет.



Рассмотрим упрошенный вывод формулы для преобразования Фурье дискретных во времени сигналов. Рассмотрим сигнал, представленный его обратным преобразованием Фурье



И возьмем его в дискретные моменты времени 



 – представляет собой образ Фурье непрерывного сигнала 

Заменим интеграл на бесконечную сумму интегралов с интервалом 



Введем (пастулативно) преобразование Фурье дискретного во времени сигнала, как величину



И обратное к нему



Если дополнительно перейти к безрамерной частоте , то выражения для прямого и обратного преобразований примут вид



И



Из этих двух формул следует одно из главных свойств преобразования Фурье дискретного во времени сигнала. Как можно заметить из зависимости , преобразование Фурье дискретного во времени (или дискритизованного) сигнала является суммой преобразований Фурье его непрерывной версии , сдвинутые друг относительно друга на  (см. рис.1).

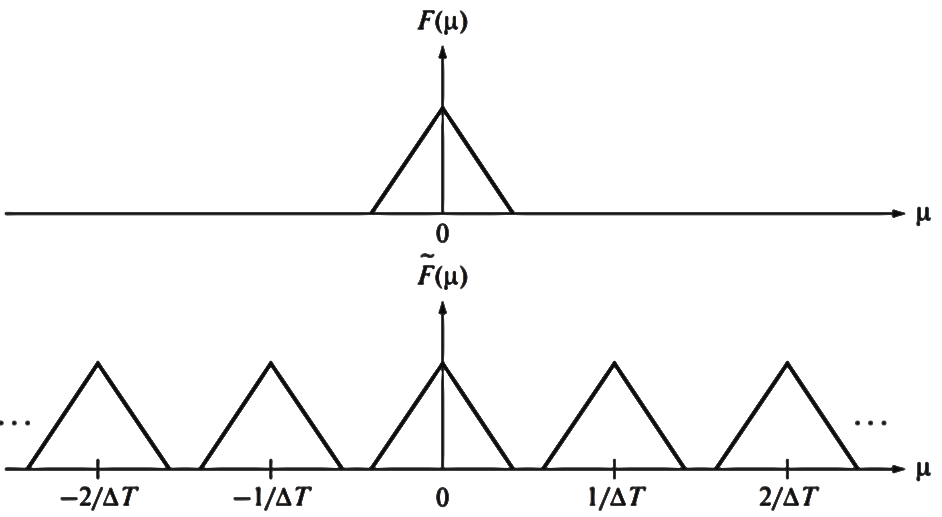


Рисунок 1 – спектр дискретного сигнала

Заметим, что величина  - есть частота квантования во времени.

Рассмотрим два важным момента:

* Теорема сэмплирования или теорема восстановления
* являение альясинга (aliasing)

Теорема сэмплирования или (она же) теорема Котельникова-Найквиста или теорема воссатновленния гласит, что любой сигнал с ограниченным спектром (т.е. модуль преобразования Фурье которого стремиться к нули при возрастании частоты, так быстро, что оказывается принебрежимо мал выше некоторой частоты ) может быть восстановлен (переведен в аналоговую форму) пропусканием через идеальный фильтр низких частот , тогда и только тогда, когда .

Графически данный процесс демонстрируется на рис. 2. Формально же это значит, что если мы возьмем дискретный сигнал  с образом фурье , то после пропускания его через идеальный фильтр нижних частот его спектр должен соответсвовать спектру непрерывного сигнала  (т.е. ). Но такое возможно только в том случае, если спекрты от соседних копий спектра аналогового сигнала  не пересекаются .

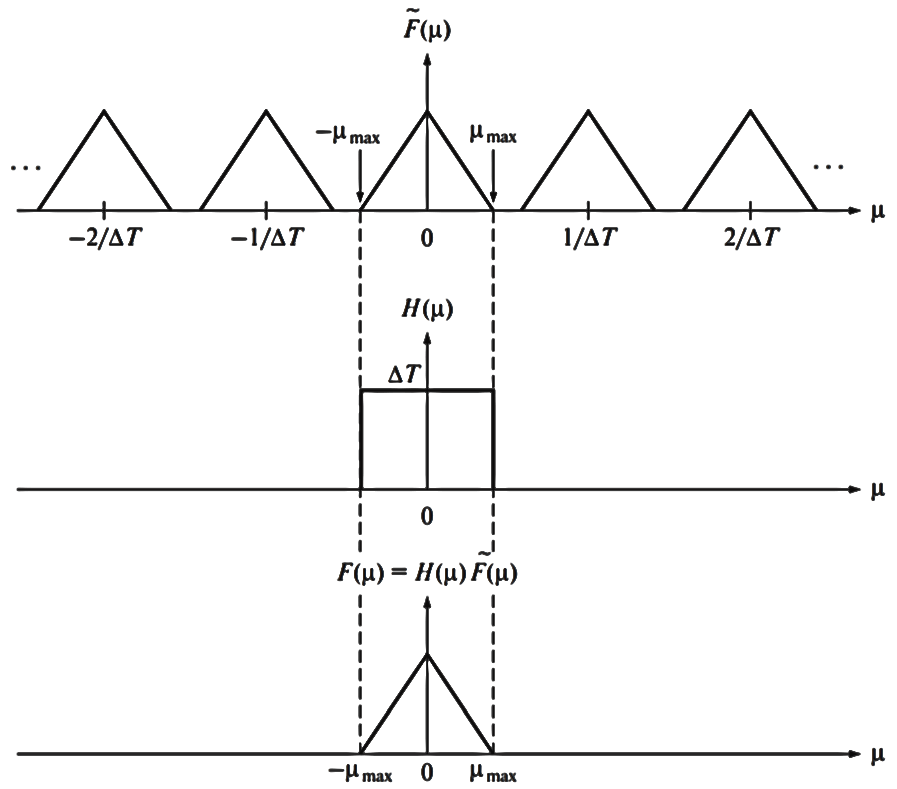


Рисунок 2 – Получение непрерывного сигнала из дискретного пропусканием через идеальный фильтр низких частот

Рассмотрим случай, когда условия теоремы не выполнены (рис. 3).

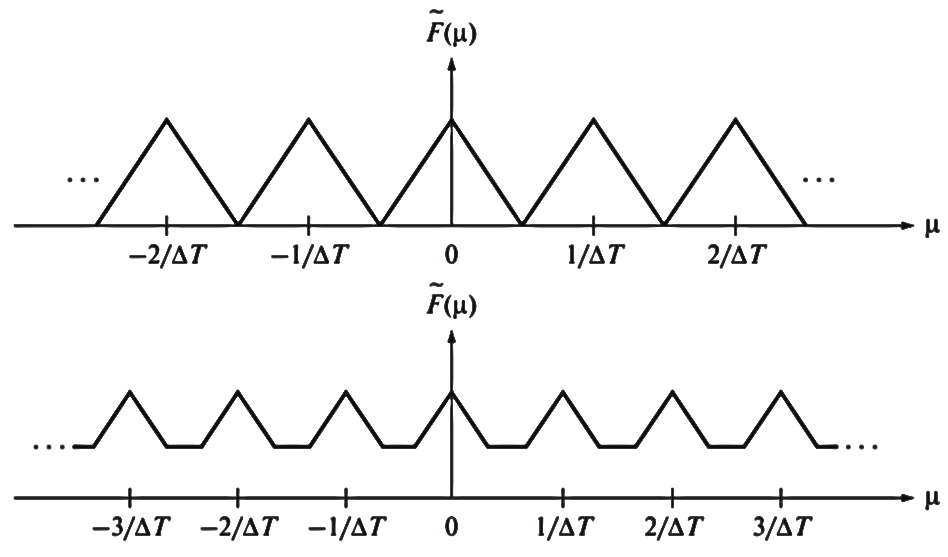


Рисунок 3 – Наложение спектров

Попытка восстановить аналоговый сигнал пропусканием через идеальный фильтр низких частот приведет к эффекту на рис. 4.

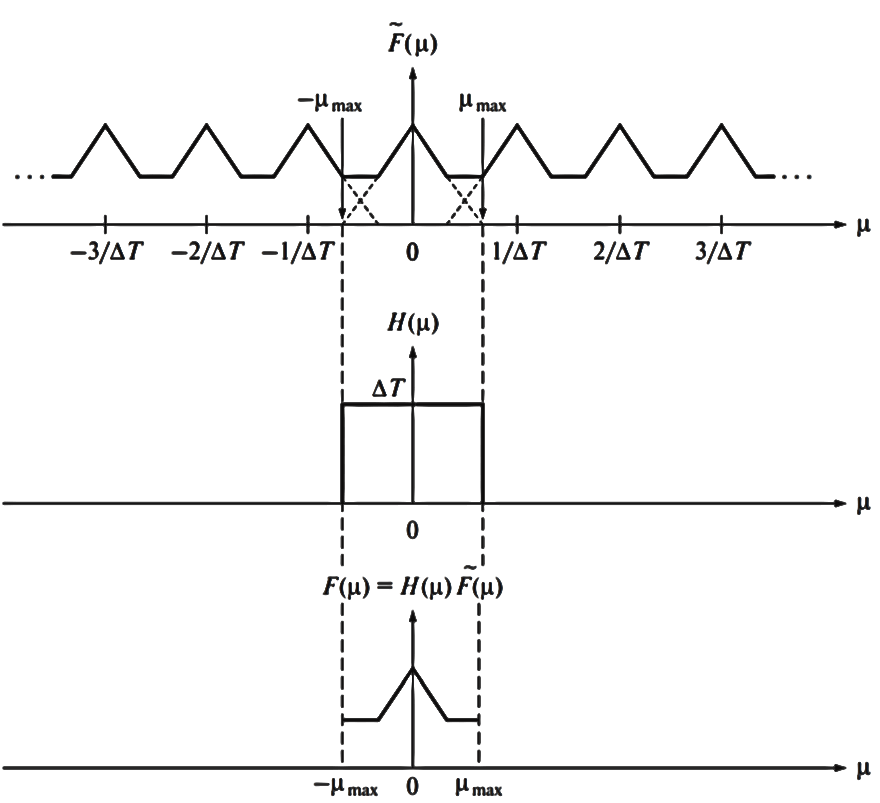


Рисунок 4 – Фильтрация наложения спектра

Как можно видеть из рис. 4, спектр сигнала оказывается искаженным. В данном случае появляется высокочастотные компоненты. Если наложение спектров было небольшим ( условие нарушается не сильно), то ситуацию еще можно исправить применив фильтр низких частот с меньшей полосой пропускания.

Рассмотрим теперь более сложный случай, когда в сигнале присутсвует высокочастотный шум, который существенно больше, чем половина частоты дискретизации. На рис. 5 показан случай, когда высокочастотный шум оказывается на столько больше частоты дискретизации, что накладывается на область низких частот. Такой шум уже не удасться убрать обычными линейными фильтрами не исказив полезного сигнала, который (как правило) располагается в области низких частот. Это особенно важно при проектировании систем автоматического управления.

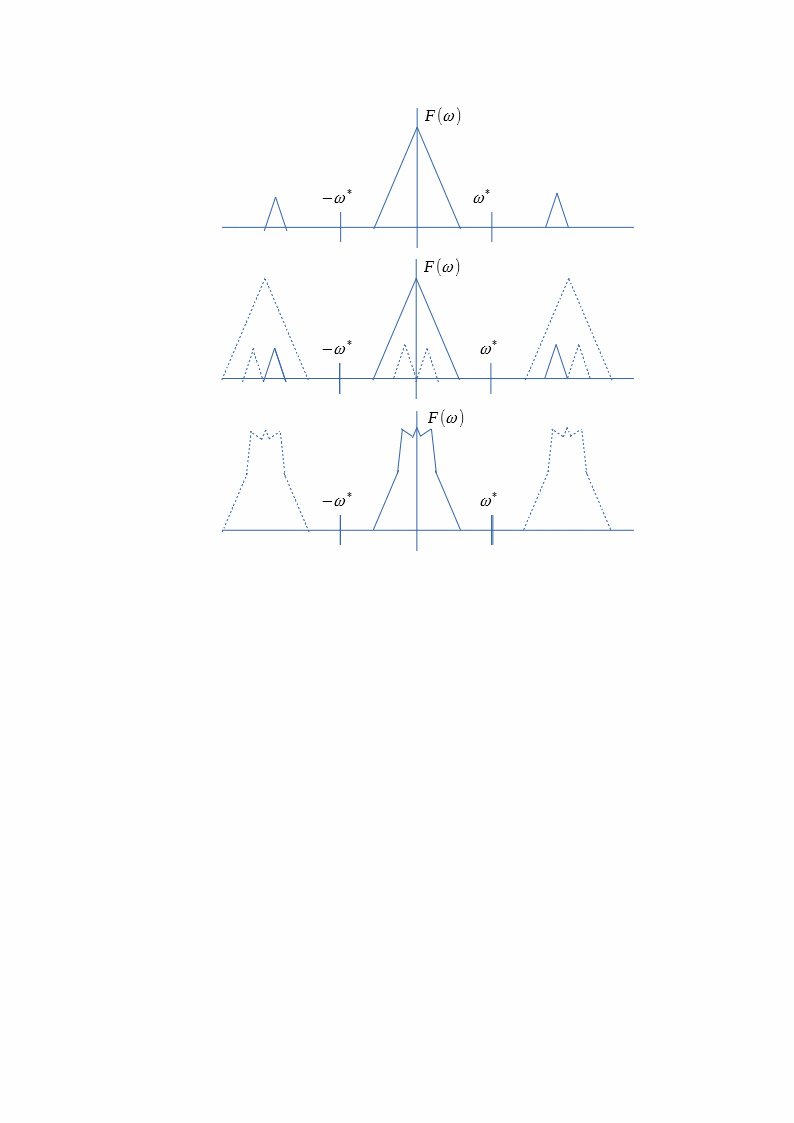


Рисунок 5 – Фильтрация наложения спектра (высокочастотный шум)

Данный эффект (альясинг) хорошо иллюстрируется во временной области рис. 6

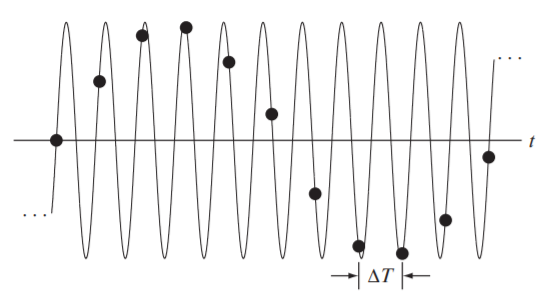


Рисунок 5 – Эффект наложения частот (пояснение во временой/пространственной) области

Немного о получении преобразования Фурье дискретного во времени сигнала. В литературе можно найти преобразование в следующей форме



где 

Это преобразование часто называю не преобразованием Фурье, а комплексной частотной характеристикой, но суть от этого не меняется.

Следует различать два понятия:

* преобразование Фурье дискретного сигнала - это непрерывное (с бесконечных количеством коэффициентов) преобразование Фурье от решетчатой функции
* дискретное преобразование Фурье (ДПФ) - это преобразование Фурье для сигнала конечной длительности с ограниченным количеством коэффициентов разложения (равного длине исходного сигнала)

Дискретное преобразование Фурье вычисляется по формулам



и обратное



где M - число отсчетов сигнала.

Отметим некоторые факты о ДПФ:

1. Число коэффициентов разложения конечно и равно числу отсчетов преобразуемого сигнала. Это лишь очевидно, т.к. для того, чтобы не терять информацию (иметь возможность точно восстанавливать исходный сигнал) необходимо, чтобы количество информации в исходном и преобразованном сигналах было одинаковым.
2. В дискретных системах пропадает информация о реальных (физических) единицах шага по пространственной координате (период дискретизации Т) и по частоте ().
3. Наим частотаеньшая в последовательности длиной M составляет , а наибольшая 
4.  является периодической функцией с периодом М. .
5. Функция , полученная из обратного преобразования, также является периодической с периодом М.
6. С помощью ДПФ можно выполнять операцию свертки

Пусть операция свертки представлена в виде



Заметим, что в этом случае функции  сами по себе считаются преодическими с пеоридом M, т.е. 

Выполняя обратное преобразование от произведения ДПФ обоих функций получим



Рассмотрим две последние формулы. Для вычисления первой, для каждого n необходимо вычислять всю сумму из M значений сдвинутой функции, что требует для вычисления всего преобразования:

–  операций сложения

–  операций умножения

– в сумме  операций

В случае второй формулы можно один раз вычислить произведения , что потребует М операций комплексного умножения (каждая из который составляет 2 операции сложения и 4 операции умножения). После чего необходимо выполнить

–  операций комплексного сложения (2 операции сложения)

–  операций комплексного умножения

Кажется, что ситуация еще ухудшилась... и это мы еще не учли "стоимость" самого ДПФ и вычисление комплексных экспонент... но пока рано.

До этого мы считали, что длительность функций  одинакова. Пусть теперь их длительность отличается , . В этом случае операция свертки



и необходимо учесть то, как ведут себя функции за границами свой длительности. Варианты:

– за пределами длительности функции обнуляются

– за пределами длительности функции продолжаются периодически

В теории линейных систем выбирается 1й вариант. Поэтому длительность результирующей последовательности будет составлять  (т.к. отклик фильтра продолжается после окончания входного сигнала). Поэтому для того, чтобы можно было свернуть два сигнала каждый из них должен быть дополнен нулями до размера  (минимум).

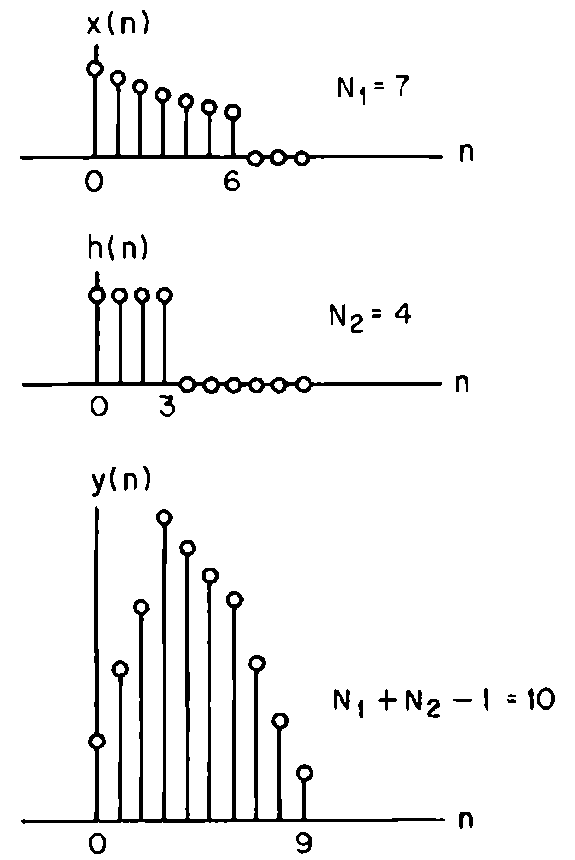


Рисунок 6 – Свертка сигналов разной длительности.

**Способы вычисления ДПФ**

Рассмотрим прямой ДПФ



где величина  постоянна.

Заметим также, что , т.е. вычисление комплексных экспонент можно сделать рекурсивным через простое умножение (без вычисления синусов и косинусов). Учтем также, что комплексная экспонента является периодической функцией .

Заметим теперь, что операция ДПФ может быть представлена в матричной форме









Заметим, что элементы матрицы  могут вычислять эффективно рекурсивно. Также сама матрица является симметричной . Кроме того, для заданного размера матриц, сама матрица  является полностью определенной и не меняется в зависимости от входного сигнала (т.е. может быть сохранена и использована повторно).

Схемотично, вычисление ДПФ "лобовым" методом представлено на рис. 7.

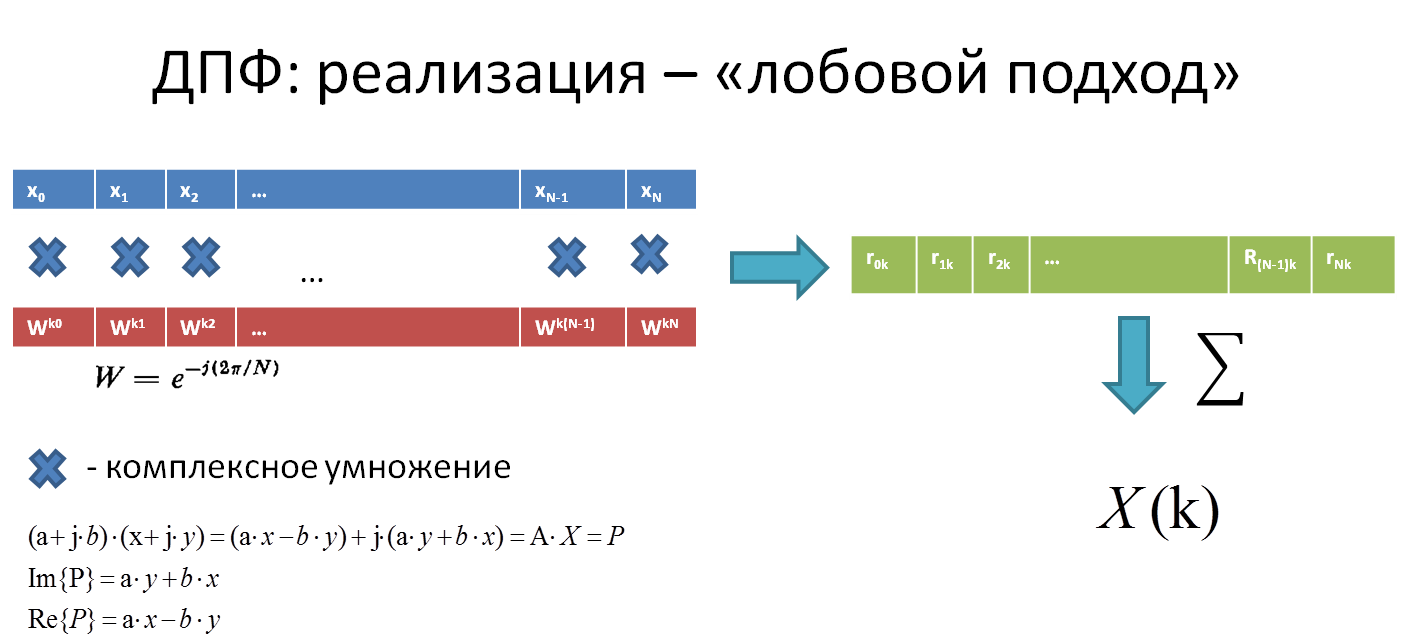


Рисунок 7 – "Лобовой" подход к вычислению ДПФ: x – прообраз, X – образ

Несмотря на кажущиеся улучшения в способе вычисления, такая реализация ДПФ все еще очень медленная.

**Быстрое преобразование Фурье (БФП): Алгоритм Cooley-Tukey или алгоритм БПФ с основанием 2**

*Основная идея:*

Пусть исходная последовательность , M – **четное**. Разобьем исходную последовательность на две с числом точек N/2 в каждой. Для вычисления ДПФ исходной требуется (N-1)2 комплексных умножений и N(N-1) комплексных сложений. Для вычисления ДПФ двух полу-последовательностей требуется (N)2/2 комплексных умножений и (N/2)(N/2-1) комплексных сложений.

Разобьем исходную последовательность на четную и нечетную под-последовательности

,

, 

Запишем ДПФ исходной через ДПФ под-последовательностей



Введем следующую нотацию и учтем свойства комплексных экспонент



получим следующее выражение





где  – ДПФ с M/2 точками. Если вычислять  обычным способом, то для из вычислений потребуется  комплексных умножений (вместо , что при больших M () может дать сокращение до 50% времени.

Т.к.  определено для , а  для  необходимо доопределить формулу  для . Для этого используем свойство периодичности преобразования Фурье



(можно учесть зависимость  и переписать выражения немного по-другому).

Пример вычисления 8-ми точечного ДПФ по двум 4-х точечным рис.8.

* сначала разбивается на две подпосдедовательности (четная, нечетная)
* расчитываются посделовательности 
* вычисляется рузльтирующая

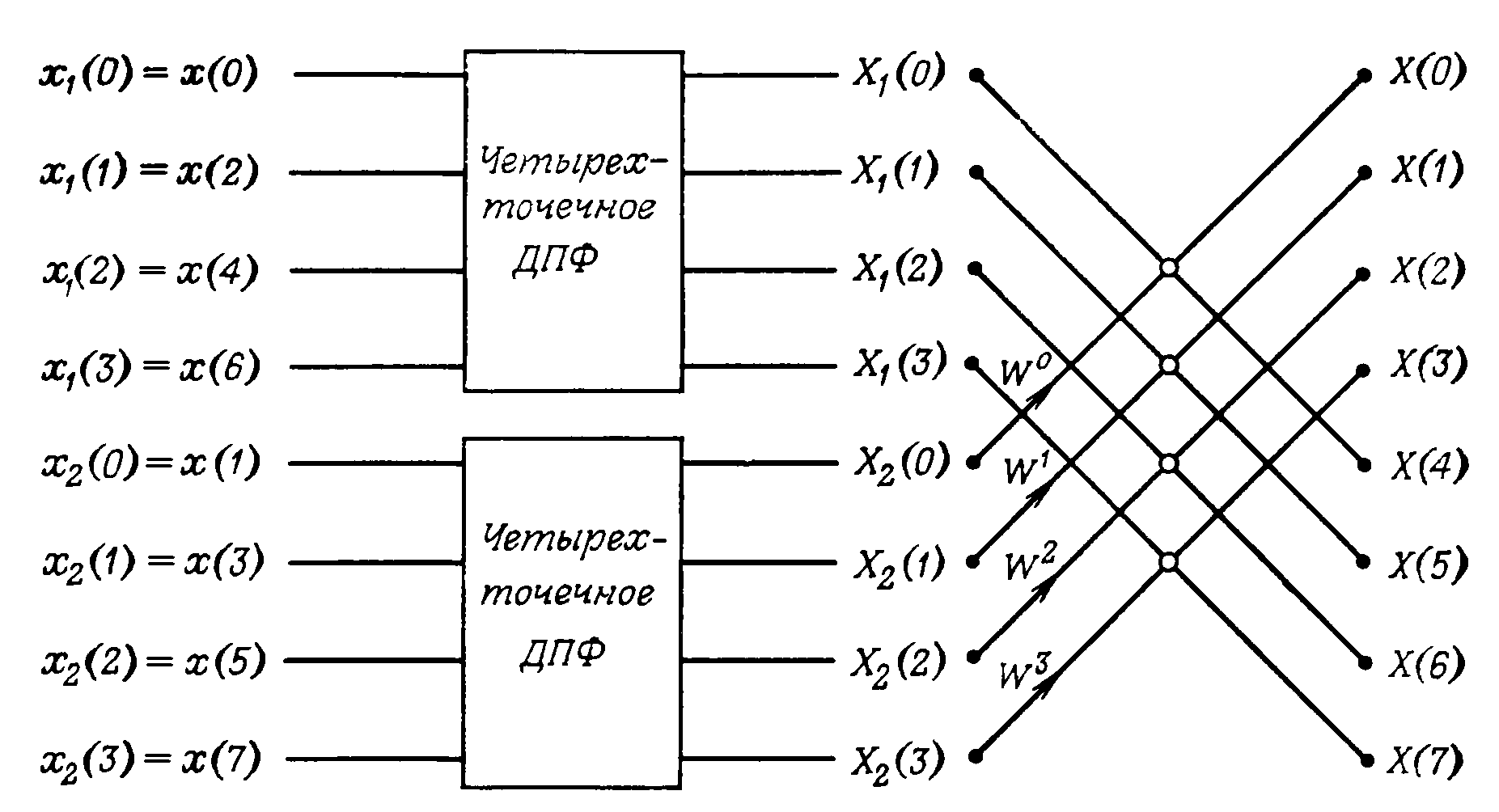


Рисунок 7 – 8-ми точечное ДПФ по двум 4-х точечным:  – умножение на значение множителя,  – можно интерпретировать как двухточечное ДПФ

Туже процедуру можно сделать и для подпоследовательностей

,

Где  – M/4 точечные ДПФ четных и нечетных челенов .

Процесс дробления пополам должен быть продолжен, пока не останутся только двухточечные ДПФ.



Поскольку , то умножения как такового не происходит.

Пример полного 8-ми точечно БПФ представлен на рис. 9

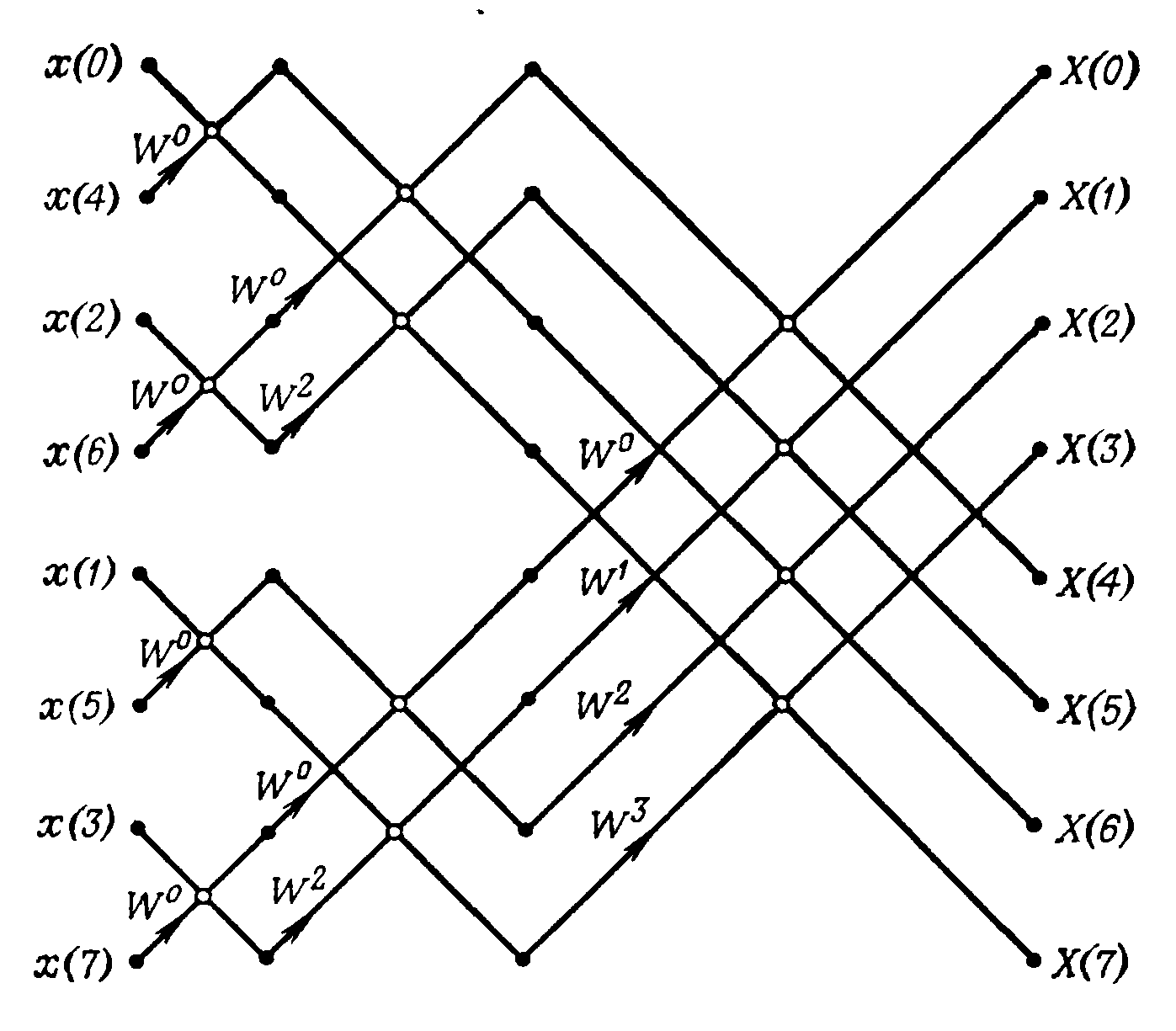


Рисунок 9 – 8-ми точечное ДПФ

В итоге, для осуществления M точечного ДПФ необходимо приблизительно комплексных умножений.

Описанный выше алгоритм был назван алгоритмом с прореживанием по времени, поскольку на каждом этапе входная (т. е. временная) последовательность разделяется на две обрабатываемые последовательности меньшей длины, т. е. входная последовательность прореживается на каждом этапе.

Базовая операция алгоритма с прореживанием по времени (так называемая „бабочка") состоит в том, что два входных числа А и В объединяются для получения двух выходных чисел X и Y следующим образом:



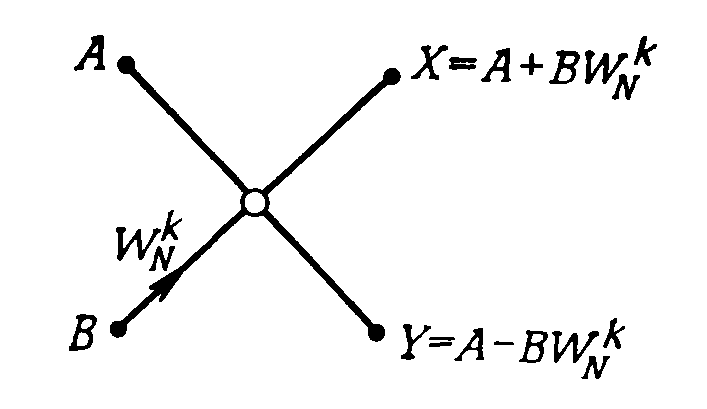


Рисунок 10 – Базовая операция БПФ

Каждый из этапов БПФ содержит M/2 базовых операций. Результаты всех промежуточных этапов БПФ можно размещать в те же ячейки памяти, где находились исходные данные. Поэтому для хранения и входной, и выходной последовательностей можно использовать один и тот же массив ячеек памяти. Алгоритм, в котором для размещения входной и выходной последовательностей используются одни и те же ячейки памяти, называется алгоритмом БПФ с замещением. В случае когда множитель  — нетривиальный, для каждой базовой операции необходимо выполнить только одно умножение, поскольку величину  можно вычислить и запомнить.

Еще одной особенностью алгоритма с прореживанием по времени (как, впрочем, и большинства других алгоритмов БПФ) является необходимость такой перестановки элементов входной последовательности, чтобы выходная последовательность имела естественный (прямой) порядок расположения, т. е. n = 0, 1, ...,M-1.

В случае, когда M является степенью 2, входная последовательность должна быть расположена в памяти в двоично-инверсном порядке для того, чтобы выходная последовательность получалась в прямом порядке. Двоично-инверсный порядок определяется следующим образом. Если записать порядковые номера элементов входной последовательности в двоичном коде, используя L двоичных разрядов, причем , а затем инвертировать порядок следования разрядов, то получаемые при этом числа и будут номерами элементов входной последовательности после их перестановки.

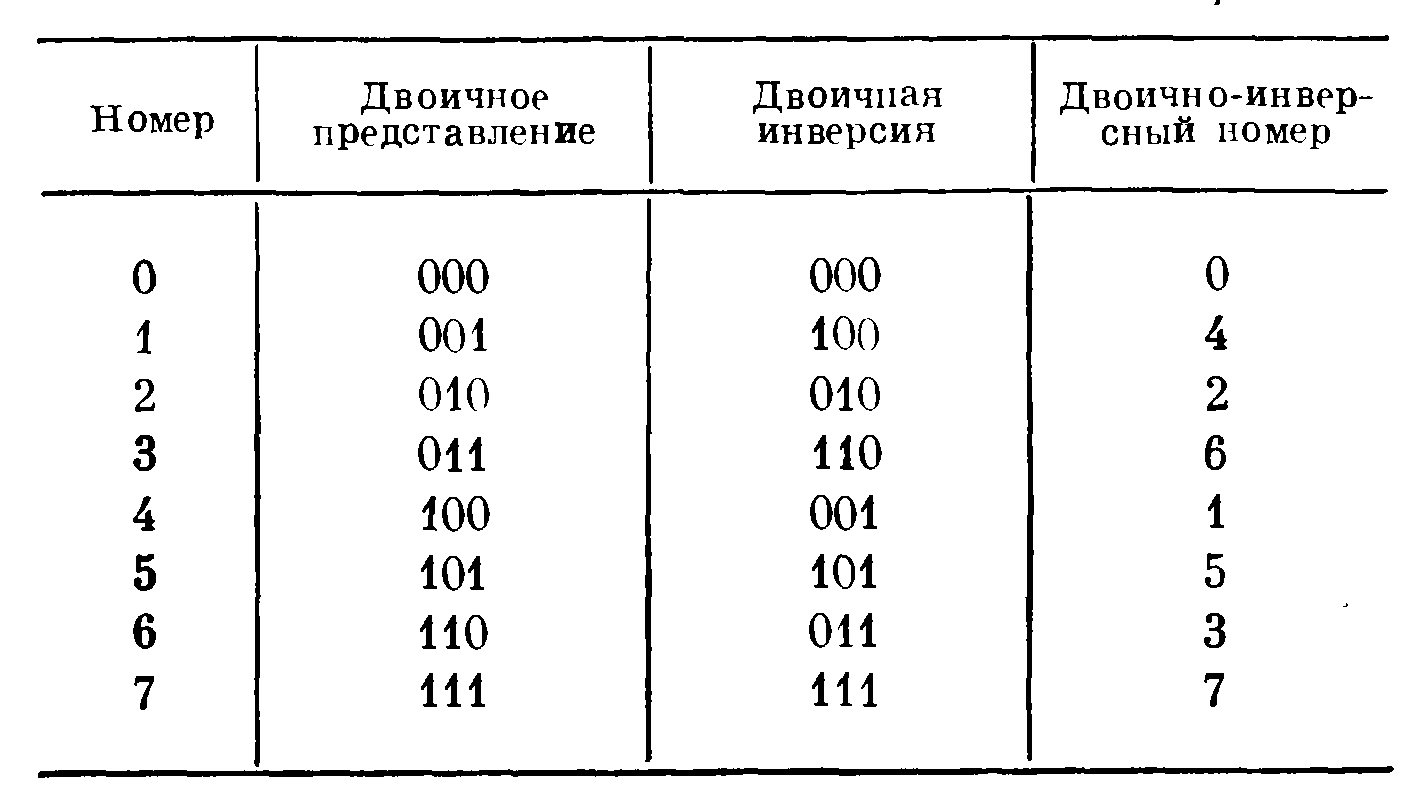


Рисунок 11 – Пример двоичной инверсии

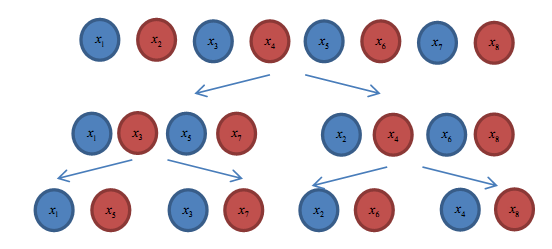


Рисунок 12 – Иллюстрация двоичной инверсии

Ясно, что перестановку входной последовательности можно произвести с замещением, меняя в парах местами числа с прямым и двоично-инверсным номерами и используя для этого лишь одну вспомогательную ячейку памяти. Другими словами, можно сначала переставить входную последовательность, а после БПФ результирующая последовательность будет в правильном порядке.

Существует несколько способов вычисления значений :

* табулирование (много памяти, но быстро)
* прямой вычисление 
* итеративное вычисление: 

**Вычисление обратного преобразования**

Получить обратное ДПФ можно выполняя прямое ДПФ с некоторыми модификациями. Таким образом, нет необходимости писать и отлаживать две функции, а достаточно правильно спроектировать одну.

Рассмотрим обратное преобразование



Найдем к нему комплексно сопряженное и домножим на М



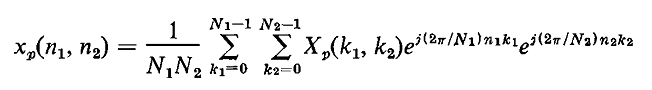
Можно заметить что правая часть соответствует прямому ДПФ для последовательности.

Таким образом

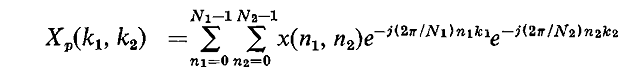


**Двумерное преобразование Фурье**

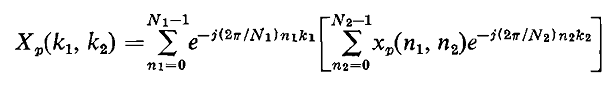
Изображения представляют собой двумерный сигнал и также могут быть представлены через разложение во взвешенную сумму комплексных экспонент. Формулы аналогичны представленным выше. Для обратного преобразования



Для прямого преобразования



Заметим следующее. Формулу прямого преобразования можно переписать в следующем виде



Данное преобразование показывает, что для вычисления прямого (так и обратного) ДПФ можно не вычислять двумерное преобразование (и соответственно не писать под него отдельную функцию), а выполнить подряд два преобразования сначала для одной координаты изображения, потом для другой. Кроме того, для вычисления двумерного ДПФ достаточно иметь функцию для одномерного ДПФ и правильно ее применить (подумать как).

*Краткие итоги и сведения из библиотеки OpenCV*

1. Функция для проведения ДПФ. С ней надо уметь обращаться, тут не всё так просто.

void dft(InputArray src, OutputArray dst, int flags = 0);

InputArray src – входное изображение (или одномерный массив). Если хотим осуществить прямое ДПФ, может быть одноканальным с вещественным числами, пусть оно будет CV\_32FC1. Если обратное - двухканальным с вещественным числами, где в одной хранятся вещественные числа, а в другой мнимые, пусть оно будет CV\_32FC2. Зависит от флага.

OutputArray dst – выходное изображение. Прямое – двухканальное. Обратное – одноканальное.

int flags – флаг режима работы.

DFT\_COMPLEX\_OUTPUT **–** для прямого ДПФ с выводом в виде полной комплексной формы

DFT\_INVERSE|DFT\_REAL\_OUTPUT **– такая совокупность флагов используется для обратного ДПФ**

1. **Ускорение работы функции ДПФ. Из-за особенностей алгоритма ДПФ, производительность ДПФ не имеет линейную зависимость от размера входного массива. Массивы, размер которых представляет собой двойку в степени (2,4,8,16…), обрабатываются быстрее всего. Так же обрабатываются достаточно эффективно массивы, имеющие размер равный произведению 2-ек, 3-ек, 5-ок (например, 300 = 5\*5\*3\*2\*2).**

int getOptimalDFTSize(int vecsize)

int vecsize **– размер массива(вектора) , для которого требуется просчитать оптимальный размер.**

**Функция возвращает значение размера, которое больше исходного, но при этом является оптимальным. Если значение входящего массива слишком велико (близко к INT\_MAX), то возвращает отрицательное число.**

**Как правильно использовать для создания размера холста при перемножении спектров:**

Size dftSize;

dftSize.width = getOptimalDFTSize(img.cols + Gy.cols - 1);

dftSize.height = getOptimalDFTSize(img.rows + Gy.rows - 1);

1. **Функция для перемножения спектров.**

void mulSpectrums(InputArray a, InputArray b, OutputArray c, int flags=0, bool conjB = false);

**Для получения адекватного результата перемножения следует использовать изображения одинаковые по размеру.**

**Для этого, до начала проведения преобразований с ДПФ, у исходных изображений увеличивают холст. Увеличение холста – это когда создают пустое изображение, которое больше по размеру, чем исходное, и копируют туда исходное, совмещая его с холстом по левому верхнему углу. Кто не понял:** <http://photoshop.e-publish.ru/p15aa1.html> **(только там по центру совмещение).**

**Данную операцию проводят для получения одинакового числа гармоник, число которых равняется площади изображения.**

**Если два изображения разные по размеру, то размер холста равен их суммарному размеру (колонки\_а+колонки\_б, строки\_а+строки\_б). С классом Size можно использовать оператор +.**

bool conjB **– сопрягать ли массив b до умножения. *0* – если хотим провести свёртку, *1* – корреляцию (сравнение). Сопряжение это когда меняют знак на обратный у мнимой части членов ряда Фурье.**

1. **Как «красиво» нарисовать Фурье-образ. Красиво нарисовать Фурье образ нужно не только для того, чтобы красиво нарисовать, но и чтобы было удобно анализировать и обрабатывать изображение. Вот порядок использования нужных функций.**
2. void split(const Mat& src, Mat\* mvbegin)

**Разбивает исходное многоканальное изображение на массив одноканальных.**

Mat\* mvbegin **– выходной массив 1-канальных изображений. Должен быть заранее определён и иметь размер, равный количеству каналов входного изображения.**

1. **Теперь найдём магнитуду для каждого элемента образа Фурье:**

void magnitude(InputArray x, InputArray y, OutputArray magnitude)

1. **Дальше нам надо перейти к логарифмическому масштабу (пример):**

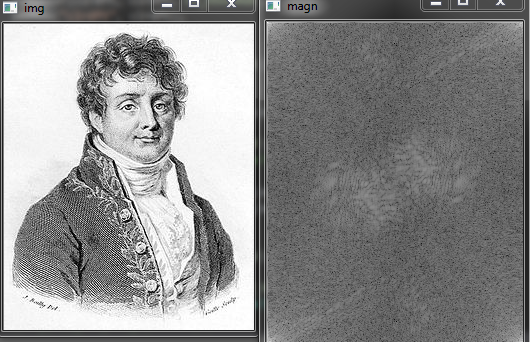
magn += Scalar::all(1);

log(magn, magn); //Натуральный логарифм для каждого элемента

1. **Нормализуем изображение, чтобы масштабировать значения его элементов до диапазона между 0 и 1 (для** CV\_32F**). В противном случае функция imshow(…) покажет нам подобие двоичного изображения(Это в случае изображений с вещественными числами).**

normalize(magn, magn, 0, 1, NormTypes::NORM\_MINMAX);

**Получим Фурье в 2-ух мерном образе самого себя:**



**Но картинка с магнитудой вышла не особо понятной и не удобной для анализа частотного (низкие частоты расположены по углам), зато красивой и похожа на космическую непонятную фигню. Чтобы привести её к принятой везде форме сделаем следующее. Данная функция перемещает квадранты (1\4 изображения) так, чтобы низкие частоты оказались в центре:**

void krasivSpektr(Mat &magI){

// rearrange the quadrants of Fourier image so that the origin is at the image center

int cx = magI.cols / 2;

int cy = magI.rows / 2;

Mat q0(magI, Rect(0, 0, cx, cy)); // Top-Left - Create a ROI per quadrant

Mat q1(magI, Rect(cx, 0, cx, cy)); // Top-Right

Mat q2(magI, Rect(0, cy, cx, cy)); // Bottom-Left

Mat q3(magI, Rect(cx, cy, cx, cy)); // Bottom-Right

Mat tmp; // swap quadrants (Top-Left with Bottom-Right)

q0.copyTo(tmp);

q3.copyTo(q0);

tmp.copyTo(q3);

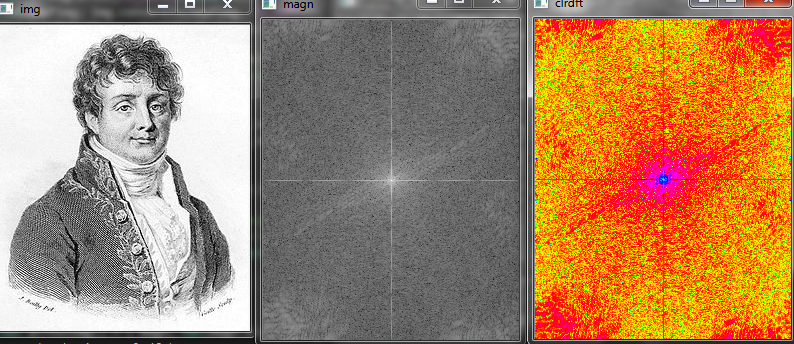
q1.copyTo(tmp); // swap quadrant (Top-Right with Bottom-Left)

q2.copyTo(q1);

tmp.copyTo(q2);

}

**И теперь можно получить красивую «понятную» картинку образа Фурье (особые молодцы могут сделать её цветной при помощи цветовой модели HSV).**



**!!! Функции** mulSpectrums **и** dft **работают с образами Фурье только в их первоначальном виде, не приведённом к «красивому».**

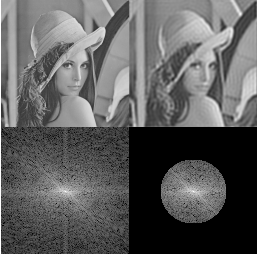
**ЗАДАНИЕ**

1. Написать свою функцию преобразования Фурье (прямое и обратное) используя «лобовой подход».
2. Написать преобразование Фурье используя алгоритм Radix-2 (по основанию 2, или «Бабочка»)
3. Сравнить быстродействие из функций из первой и второй части, а также с преобразованием из OpenCV
4. Загрузку изображений в программу (т.е. получение их имён и т.д.) крайне желательно осуществлять при помощи функций, указанных в 3-ей лаб.работе.
5. Операции свёртки и корреляции выполнять можно только при помощи преобразований Фурье.
6. Работаем с одноканальным изображением (желающие могу и с цветным, на свой страх и риск).
7. Произвести по отдельности свёртку какого-либо изображения с ядром фильтров: Собеля (по горизонтали и вертикали), усредняющего (BoxFilter), Лапласа . Необходимо «красиво» вывести магнитуду образа Фурье исходного изображения и ядра свёртки.

Полученные образы Фурье в результате выполнения свёртки следует обратно преобразовать в изображение. Сначала обрезаем полученное изображение (при помощи ROI) до первоначального размера. Полученное изображение нормализуем (с теми же параметрами, как и в примере, приведённом в теории) и выводим.

Функция **filter2D** [http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html#filter2d] использует алгоритм, основанный на свёртке через ДПФ в случае ядер от 11х11 и больше.

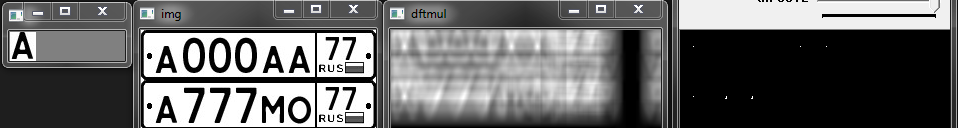
1. Взять какое-нибудь изображение и в его спектре обрезать в одном случае элементы спектра с высокими частотами, в другом – низкими. А потом выполнить обратное преобразование на основе полученных спектров:

Не забудьте после обрезания выполнить обратное перемещение квадрантов. Т.е. применить ещё раз krasivSpektr.

Такой подход позволит вам удобно создавать собственные ядра свёртки для фильтров. А отбрасывание гармоник (они же элементы образа Фурье) с малой магнитудой позволит вам хранить данные в сжатом виде. Наглядный тому пример, где в «Лене» убрали высокие частоты.

1. Провести корреляцию (сравнение) изображений автомобильных номеров по очереди с 3-мя символами. Полученный образ Фурье обратно преобразовать в обычное изображение. Найти на нём наибольшее значение, которое принимают элементы. Отнять от этого значения небольшое число (около 0.01). Использовать полученное число в качестве порога для пороговой фильтрации от полученного изображения.



Если честно, то я поленился обрезать 3-е изображение, но вам надо это сделать.

Данный метод позволяет примерно производить поиск по шаблону.

Подобный алгоритм (но более точный) использует функция [matchTemplate](http://docs.opencv.org/modules/imgproc/doc/object_detection.html?highlight=matchtemplate#matchtemplate)

[http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template\_matching/template\_matching.html]

Не забываем про материалы лекций… думаю проблемы с корреляцией будут, но надеюсь не у всех…