Обработка сигналов. Лекция 2.

Алексей Орлов

9 января 2021 г.

Содержание

1 Использование MATLAB для обработки сигналов

В нашем курсе для работы с сигналами будет активно применяться система компьютерной математики MATLAB. Она является ведущей системой

для проведения численных расчётов, обладает большим количеством расширений, в том числе специализированными наборами инструментов Signal Processing Toolbox и DSP System Toolbox для обработки сигналов. Также стоит упомянуть другие полезные для обработки сигналов расширения. При анализе сигналов и математическом моделировании устройств обработки сигналов особое значение имеет выбор приближений для различных сигналов и функций, представленных кривыми. Для такого приближения в MATLAB имеется пакет расширения Curve Fitting Toolbox. Один из быстро развивающихся пакетов расширения MATLAB – пакет расширения по вейвлетам. Будучи новым математическим базисом приближения произвольных сигналов и функций набором масштабируемых и сдвигаемых вейвлет-функций, похожих на «маленькие волны», вейвлеты открывают обширные возможности в анализе функций и сигналов (в том числе нестационарных), выявлении их тонких закономерностей, очистке от шума, компрессии и т. д. Однако и без этих расширений возможности MATLAB позволяют выполнять широкий спектр операций над сигналами, используя векторную парадигму программирования и ряд встроенных функций. При этом система хорошо пригодна как для пакетной обработки данных, когда сигнал целиком загружается в память, так и для потоковой обработки, когда сигнал поступает в память порционно.

Signal Processing Toolbox предоставляет функционал для анализа, предварительной обработки и извлечения характерных особенностей из сигналов с однородной и неоднородной выборкой. Набор включает инструменты для создания фильтров и анализа, передискретизации, сглаживания, удаления тренда и оценки спектра мощности. Signal Processing Toolbox также обеспечивает нас возможностями для нахождения пиковых значений и устойчивых рисунков поведения сигнала, определения количества общих черт и выполнения измерений, таких как оценка отношения сигнал/шум.

Вот только некоторые из задач, решаемых с помощью этого расширения:

- задание сигналов различного типа, в том числе модулированных;
- создание окон фильтрации и спектрального анализа;
- реализация прямого и обратного преобразований Фурье (в том числе быстрого преобразования Фурье (БПФ);
- реализация дискретного косинусного и других преобразований сигналов;
- оценка спектральной плотности мощности (СПМ) сложных сигналов;
- статистическая обработка сигналов;
- анализ линейных систем и цепей;
- фильтрация сигналов (в том числе цифровая);
- моделирование работы различных фильтров и вычисление их характеристик и др.

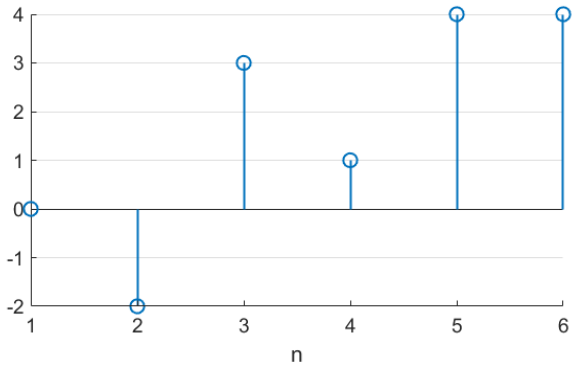

Рис. 1: Дискретный сигнал $x_1[n]$.

В целом пакет содержит около 150 функций, реализующих решение ряда задач обработки и фильтрации сигналов с помощью самых современных численных методов.

Благодаря использованию приложения Signal Analyzer можно предварительно обработать и проанализировать несколько сигналов одновременно во временной и частотной области без написания кода; исследовать длинные сигналы и извлекать из них необходимые части. С приложением Filter Designer можно проектировать и анализировать цифровые фильтры путем выбора из множества алгоритмов и откликов таких систем. Оба приложения генерируют код MATLAB.

Возможноти MATLAB исчерпывающи для того, чтобы построить компьютерную модель (soft-продукт) системы ЦОС. Если далее её необходимо реализовать в виде устройства (hard-продукт), то в ней должны учитываться эффекты квантования исходных данных и результатов арифметических операций. Далее возможно произвести тестирование модели с помощью виртуальных приборов на реальных сигналах в среде графического программирования LabVIEW, интегрированной с MATLAB. Реализация модели в виде hard-продукта может быть программной, аппаратной или аппаратно-программной на базе цифрового процессора обработки сигналов (ЦПОС), программируемой логической интегральной схемы (ПЛИС), системы на кристалле (System on Chip, SoC) и т. п. В системе MATLAB предусмотрены целевые средства реализации hard-продукта на основе модели Simulink (являющейся подсистемой MATLAB).

Инструкция, как получить дистрибутив MATLAB для студентов ИТМО, может быть найдена в приложениии A на странице 30.

1.1 Базовые операции над сигналами

Элементарно, для записи дискретного сигнала $x_1[n]$, представленного на рис. 1 мы можем воспользоваться вектор-строкой, где номер отсчёта будет

совпадать с номером элемента в вектор-строке:

```
>> x1 = [0 -2 3 1 4 4];
>> x1(1)

ans =

     0

>> x1(3)

ans =

     3
```

Задав теперь второй сигнал $x_2[n]$,

```
x2 = [4 0 2 1 0 -5];
```

выполним операции их сложения и умножения. Из первой лекции вы помните, что они осуществляются поэлементно. Сложение определяется как $w[n] = x_1[n] + x_2[n]$, а умножение как $z[n] = x_1[n] \cdot x_2[n]$. В MATLAB'е это легко реализуемо, так как система хорошо приспособлена для поэлементных манипуляций с массивами. Так, сложение последовательностей даёт

```
>> w = x1 + x2

w =

     4    -2     5     2     4    -1
```

а их умножение

```
>> z = x1 .* x2

z =

     0     0     6     1     0   -20
```

Здесь для поэлементного умножения была применена характерная для M-языка операция с точкой.

Далее, если стоит задача использовать произвольную точку начала отсчёта n, а не только $n = 1, 2, 3 \ldots$, можно поступить следующим образом. Допустим, что сигнал x_1 был перенумерован, и n теперь принимает значения от -4 до 1, по-прежнему принадлежа множеству целых чисел. В таком случае вектор-строка n может быть задана отдельно:

```
>> n = -4:1

n =
```

```
−4    −3    −2    −1     0     1
```

Теперь для обращения к элементам последовательности, таким как, например, $x_1[-2]$, применим следующую конструкцию:

```
>> x1(find(n == -2))

ans =

     3
```

Для извлечения всех элементов с $n > -2$:

```
>> x1(find(n > -2))

ans =

     1     4     4
```

Пусть теперь отсчёты сигналов x_1 и x_2 пронумерованы различным образом. Для $x_1[n_1]$ положим $n_1 = -4 \dots 1$, для $x_2[n_2]$ – $n_2 = -1 \dots 4$. Предположим, мы хотим сложить только отсчёты n с -2 по 1, которые сформируют новый сигнал $p[n]$. Тогда для этого потребуется выполнить следующую команду:

```
p = x1(find((n1 <= 1)&(n1>=-2))) +...
    x2(find((n2 <= 2)&(n2>=-1)))

p =

     7     1     6     5
```

Графическое изображение полученного дискретного сигнала, подобное представленному на рис. 1, позволяет вывести функция `stem(n,p)`. Её можно вызвать с одним аргументом, передав только последовательность значений без номеров отсчётов, тогда будет считаться, что нумерация значений сигнала начинается с единицы.

1.2 Аналого-цифровое преобразование сигнала

Рассмотрим преобразование синусоидального сигнала $x = \cos(\pi t/4), 0 \leqslant t \leqslant 16$ сек. в цифровой вид x_{digital} средствами MATLAB. Для этого необходимо произвести операции дискретизации и квантования по уровню.

Дискретизация требует задания интервала выборки T. Сделаем его равным 1.2 сек. Тогда отсчёты будут располагаться через каждые T секунд, и в интервал времени от 0 до 16 сек. уложится 13 отсчётов.

Таким образом, вычисление дискретного сигнала x_{discr}, изображенного на рис. 2 осуществляется следующим кодом:

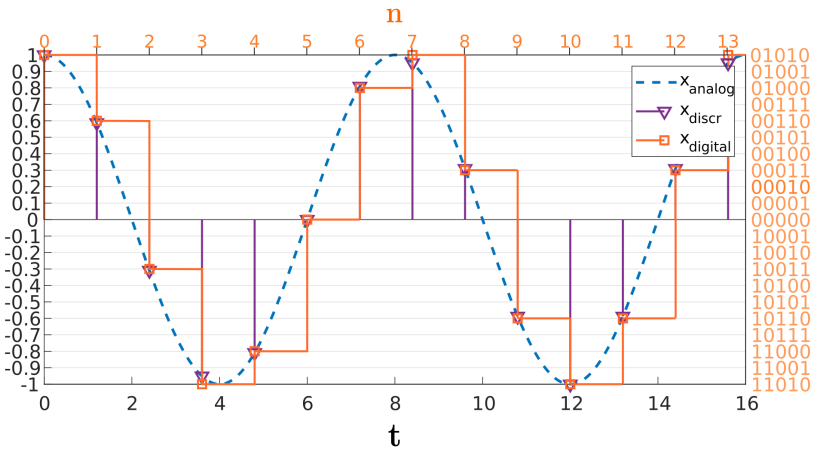

Рис. 2: Графики исходного, дискретного и цифрового сигналов, раскрывающие процесс аналого-цифрового преобразования.

```
A = 0; B = 16; %интервал времени
T = 1.2; %интервал дискретизации

tdiscr = A:T:B;
xdiscr = cos(pi*tdiscr/4);
```

Теперь нам предстоит выполнить квантование по уровню. Квантование, как вам уже известно, производится с целью представления точных значений отсчетов $x[n]$ в виде двоичных чисел конечной разрядности – квантованных отсчётов $x_{\text{digital}}[n]$. Для этого динамический диапазон дискретного сигнала $x[n]$ разбивается на конечное число дискретных уровней – уровней квантования, и каждому отсчету по определенному правилу присваивается значение одного из ближайших уровней, между которыми он оказывается. Уровни квантования кодируются двоичными числами разрядности b, зависящей от числа уровней квантования R:

$$R \leqslant 2^b \tag{1}$$

откуда $b = \text{ceil}(\log_2 R)$ – ближайшее целое в сторону увеличения. Например, выбрано 6 уровней квантования (без учета знака), поэтому $b = 3$ и отсчёты $x_{\text{digital}}[n]$ кодируются четырехразрядными двоичными числами: один разряд знаковый, три значащих.

Итого, полученная совокупность квантованных отсчётов $x_{\text{digital}}[n]$, $n = 0, 1, \ldots,$ является цифровым сигналом.

Для выполнения квантования положим, что сигнал может принимать только значения, кратные 0.1. Это будет означать, что мы задаём шаг квантования, равный 0.1, и общее число уровней для рассматриваемого сигнала, значения которого не превышают по модулю 1, составит 21 уровень. Для

применения операции квантования к имеющемуся дискретному сигналу мы просто можем округлить его с точностью до десятых:

```
xdigital = round(xdiscr, 1);
```

Далее мы могли бы получить двоичное представление значений полученного цифрового сигнала:

```
xdigital_int64 = typecast(xdigital, 'int64');
xdigital_bin = dec2bin(xdigital_int64);
```

Этот код осуществляет преобразование данных типа `double` в символьное представление набора кодирующих эти данные бит. Здесь необходимо пояснить, что функция `dec2bin` переводит числа типа `integer` в массив символов «0» и «1», соответствующих тому, как это число записывается в двоичной системе счисления. По этой причине мы не можем применить её напрямую к переменной `xdigital`. Сначала необходимо тип “double” привести к типу ”int64”, то есть к типу ”integer”, для хранения которого выделяется 64 бита, включая бит со знаком. Также допустимо приведение к беззнаковому типу ”uint64”. После этого уже можно выполнять команду `dec2bin`. В результате мы получим массив, каждый ряд которого это набор из 64 символов «0» или «1». Да, несмотря на то, что мы имеем сейчас дело с числами точностью 0.1, не превышающими по модулю 1, это по-прежнему числа с плавающей точкой, и мы всё равно используем в MATLAB тип данных “double” для их хранения. По этой причине мы получаем для каждого ряда массива `xdigital` строку длиной 64 символа. По этой же причине в двоичном формате лучше представлять не непосредственно значения уровней, представленных числами с двойной точностью, а двоичный код номеров уровней.

Вычислить номера уровней квантования можно следующим образом:

```
>> [1:length(-1:.1:1)] - find(-1:.1:1==0)
```

что даст нам значения от -10 до 10 с шагом 1. Здесь следует пояснить, что команда `[1:length(-1:0.1:1)]` вычисляет порядковые номера уровней. Далее, принимая во внимание, что уровни расположены центрально-симметрично относительно нуля, найдём его (нуля) положение в векторе порядковых номеров командой `find(-1:.1:1==0)`. Таким образом мы можем получить номера уровней со знаком, если вычтем полученный номер нулевого значения из порядкового номера элементов вектора `[-1:.1:1]`. Как видно, число уровней квантования может будет получено командой `length(-1:0.1:1)`, что составляет 21 уровень. Учитывая, что значения уровней включают ноль и расположены симметрично относительно него, без учета знака число уровней составляет $R = 11$. Теперь остаётся преобразовать номера уровней в двоичный формат. Для этого потребуется первый бит, кодирующий знак, и ещё `ceil(log(11)/log(2))`, что составляет 4, значащих бита. Значащие биты в рассматриваемом случае будут следующие:

```
>> dec2bin([0:10])
```

```
ans =

  11x4 char array

    '0000'
    '0001'
    '0010'
    '0011'
    '0100'
    '0101'
    '0110'
    '0111'
    '1000'
    '1001'
    '1010'
```

Видно, что тип данных, возвращаемых вызовом функции `dec2bin` – это двумерный массив символов типа “char”, где каждая строка содержит вектор, состоящий из набора отдельных символов «0» и «1», не являющихся, однако, цельной строкой.

Вычисление полного набора значащих бит, содержащих двоичные номера уровней квантования без знака, будет следующим:

```
levels_bin_wosign = dec2bin([10:-1:1  0  1:10]);
```

Полученный массив содержит все номера уровней, взятых по модулю.

Далее, необходимо добавить первый бит, которым кодируется знак. Одним из способов добавления вычисленного бита может быть классический подход с использованием цикла:

```
jj=1;
for ii = 10:-1:-10
    if (ii<0)
        sign_chr = '1';
    else
        sign_chr = '0';
    end
    levels_bin_sign(jj,:) = [sign_chr, levels_bin_wosign(
        jj,:)];
    jj=jj++1;
end
```

Преимущество MATLAB состоит в том, что, воспользовавшись векторной парадигмой программирования, данная операция может быть выполнена в две строки с присущим MATLAB изяществом:

```
sign_bits = num2str((sign(10:-1:-10)>0)');
```

```
levels_binary_wsign = cat(2, sign_bits, levels_bin_wosign);
```

Здесь функция sign выводит знак выбранных уровней квантования в формате -1 для отрицательных значений, 0 для нулевого значения, и 1 для положительных значений. Далее, выражение `sign(10:-1:-10)>0` производит строку, содержащую логические значения: 1 для положительных значений знака и логическое «НЕТ» для отрицательных. Апостроф выполняет операцию транспонирования, переводя столбец в строку. Последнее, что остаётся сделать, это преобразовать логический тип данных в тип “char”, что и выполняется функцией `num2str`. Получив её, можно приступить к тому, чтобы добавить знаковые биты к массиву значащих бит. Для этого потребуется присоединить столбец к массиву символов. Это несложно осуществить при помощи функции `cat`, которая «склеивает» массивы по указанному в первом аргументе направлению. Таким образом во второй строке выполняется объединение столбца, содержащего бит знака, с массивом значащих бит.

Наличие массива с двоичными значениями номеров уровней позволяет добавить на график вторую второй ось ординат, содержащую эти значения:

```
yyaxis right;
ylim([-10 10]);
yticks([-10:10])l
yticklabels(levels_bin_wsign);
```

Полный текст программы может быть найден в Приложении B.

1.3 Использование инструмента Signal Analyzer

Рассмотрим на практике, как может быть выполнена операция скользящего усреднения сигнала средствами приложения Signal Analyzer. В качестве примера возьмём данные об изменениях температуры воздуха в городе Санкт-Петербург.

Мы можем получить данные метеорологических наблюдений за любой точкой мира, воспользовавшись специальным онлайн-приложением, доступным на сайте одного из проектов NASA. Давайте посмотрим, как менялась температура воздуха у поверхности Земли в Санкт-Петербурге в период с 1 января 2015 года по 1 октября 2020 года. Для этого воспользуемся указанным веб-приложением и скачаем файл, содержайщий интересующие нас данные за выбранный промежуток времени. Выполнение данной операции вы можете наблюдать в соответствующем видеоролике лекции.

Мы получим файл в формате .csv. Он представляет данные в текстовом виде, используя для разделения значений переменной, как правило, запятую или точку с запятой. Загруженный файл помимо полей, содержащих год, месяц, день и значение температуры в этот день, будет также содержать координаты точки, для которой мы выгрузили данные наблюдений. Помимо этого в файле присутствует обширная шапка с информацией о проекте

YEAR	MO	DY	TS
2015	1	1	-0.08
2015	1	2	0.1
2015	1	3	0.03
2015	1	4	-6.02
2015	1	5	-11.86
2015	1	6	-15.02
2015	1	7	-13.69
2015	1	8	-10.14
2015	1	9	-2.95
2015	1	10	-5.79
2015	1	11	-4.25

Таблица 1: Часть CSV-файла, содержащего данные среднесуточных температур в Санкт-Петербурге.

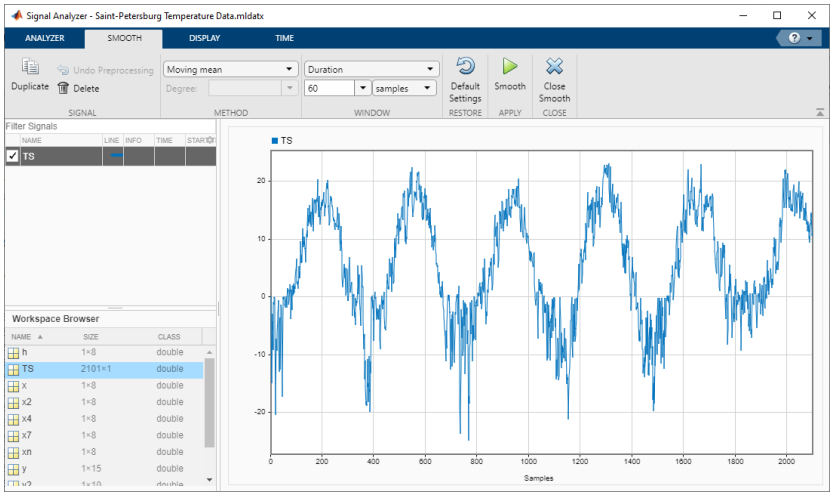

Рис. 3: Окно инструмента Signal Analyzer с загруженными данными о суточных температурных наблюдениях в Санкт-Петербурге в период с января 2015 года по октябрь 2020 года.

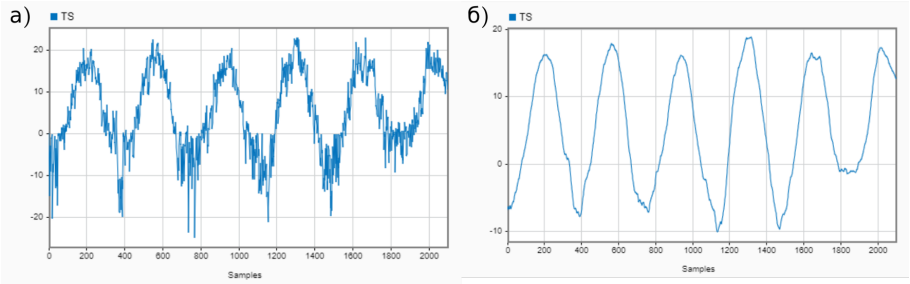

Рис. 4: Суточные температуры в Санкт-Петербурге в период с января 2015 года по октябрь 2020 года (а) до и (б) после применения операции “Smooth” с окном в 60 отсчётов в приложении Signal Analyzer.

NASA, в рамках которого реализовано задействованное нами для получения температуры приложение. Открыв этот файл в табличном редакторе, мы можем избавиться от ненужных полей с координатами, а также убрать шапку, скопировав значения четырех столбцов с датой и значением температуры в новый CSV-файл. В итоге полученный файл с данными будет иметь вид, показанный в таблице 1.

Импортировать представленные таким образом данные в MATLAB совершенно несложно. Для этого на панели инструментов необходимо нажать “Ìmport Data”. Появится диалоговое окно импорта данных, где мы выберем полученный CSV-файл. В следующем диалоговом окне будет видно, как были распознаны столбцы. Для завершения процесса импортирования нажмите на зелёную галочку в панели инструментов. После этого в рабочее пространство будет загружена таблица. Открыв её, запишем столбец TS, содержащий данные о температуре, в отдельную переменную с таким же названием. Для этого кликнем правой кнопкой мыши на заголовке столбца, далее в появившемся меню выберем “New Workspace Variable From Selection” -> “New Numeric Array”. После этого появится вектор TS, содержащий 2101 значение суточной температуры.

Для запуска приложения Signal Analyer откройте вкладку “Apps” главной панели инструментов MATLAB. Там в списке приложений можно обнаружить иконку нужного нам приложения и запустить его. Появится окно, изображённое на рис. 3, но ещё без загруженного сигнала. Как раз это и предстоит сделать – загрузить сигнал для анализа. В панели “Workspace Browser” будет видно название переменной TS. Перетащите его в панель “Filter Signals” и отметьте галочкой появившуюся там запись. После этого сигнал будет отображен в области построения графиков, как показано на рис. 3.

Операция скользящего усреднения в Signal Analyzer называется “Smooth”. Она доступна в группе “Preprocessing” верхней панели инструментов. Вызвав её, остаётся только задать окно усреденения и нажать на кнопку запуска операции. Также можно оставить автоматический выбор размера окна.

Оптимальным будет задание этого параметра в диапазоне 60-120 отсчётов в зависимости от того, какой уровень детализации мы предпочтём сохранить. На рис. 4 представлен исходный сигнал и сигнал после сглаживания. Отметим также, что скользящее среднее – это только один из методов, которым может действовать “Smooth”.

2 Дискретные системы

С точки зрения математики система с дискретным временем определяется как преобразование, или оператор, переводящий входную последовательность (сигнал) $x[n]$ в выходную последовательность $y[n]$ (отклик, или реакцию системы), что можно обозначить как

$$y[n] = T\{x[n]\} \tag{2}$$

и изображать графически, как показано на рис. 5. Далее для графического изображения реализации операций цифровой обработки сигналов мы будем использовать подобные блок-схемы. Эти блок-схемы будут содержать набор базовых условных обозначений основных операций.

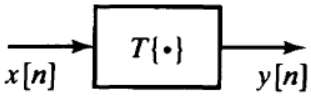

Рис. 5: Графическое представление дискретной системы

Соотношение (2) — это правило, или формула, по которому вычисляются значения реакции системы через отсчеты сигнала, поданного на её вход. Необходимо подчеркнуть, что отсчет реакции системы с индексом n может зависеть от всех отсчетов входного сигнала $x[n]$. Следующие примеры знакомят с некоторыми простыми, но полезными системами.

2.1 Пример: Идеальная система задержки

Идеальная система задержки (ИСЗ) определяется по формуле

$$y[n] = x[n - n_d], \quad -\infty < n < \infty, \tag{3}$$

где n_d — фиксированное натуральное число, называемое задержкой системы. Иными словами, ИСЗ сдвигает входную последовательность вправо на n_d отсчетов. Если в формуле (3) взять в качестве n_d фиксированное отрицательное целое число, то система будет сдвигать входную последовательность влево на $|n_d|$ отсчетов, что соответствует опережению времени.

Единичная задержка является важной операцией в цифровой обработке сигналов. Символ единичной задержки обозначает операцию, в результате которой выходная последовательность $a[n]$ равна задержанной входной последовательности $b[n]$ при $n_d = 1$. Например, $a[5] = b[4]$, $a[6] = b[5]$,

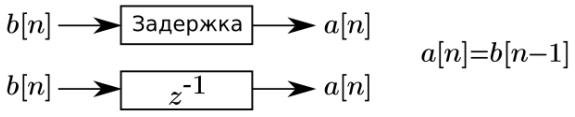

Рис. 6: Блок-схема системы, выполняющей операцию единичной задержки.

$a[7] = b[6]$ и т. д. Благодаря математическим методам, используемым для анализа цифровых фильтров, единичная задержка часто обозначается как z^{-1}.

Классы систем определяются наложением определенных ограничений на свойства преобразования T. Этот процесс, как мы сможем убедиться далее, часто ведет к самым общим математическим представлениям.

2.2 Системы без запоминания

Систему, n-й отсчет $y[n]$ реакции которой при каждом n зависит только от одного отсчета (с тем же самым индексом n) входного сигнала $x[n]$, называют *системой без запоминания*.

2.2.1 Пример: Система без запоминания

Примером может служить система, в которой

$$y[n] = (x[n])^2 \quad \forall n. \tag{4}$$

Система примера ИСЗ относится к системам без запоминания, только если $n_d = 0$. В частности, говорят, что система ИСЗ имеет «память», если n_d положительно (задержка времени) или отрицательно (опережение времени). Система из прошлой лекции, где мы разбирали операцию скользящего среднего, не имеет памяти, только когда $M_1 = M_2 = 0$.

2.3 Линейные системы

Класс линейных систем определяется по принципу суперпозиции. Если $y_1[n]$ и $y_2[n]$ — отклики системы на сигналы $x_1[n]$ и $x_2[n]$, то систему называют *линейной* тогда и только тогда, когда

$$T\{x_1[n] + x_2[n]\} = T\{x_1[n]\} + T\{x_2[n]\} \quad \text{и} \quad T\{ax[n]\} = aT\{x[n]\}, \tag{5}$$

где a – произвольная константа. Первое из свойств называют *аддитивностью*, а второе – *однородностью*. Оба свойства можно записать одной формулой по принципу суперпозиции:

$$T\{ax_1[n] + bx_2[n]\} = aT\{x_1[n]\} + bT\{x_2[n]\}, \tag{6}$$

где a и b — произвольные константы. Последнее соотношение легко может быть переписано для нескольких сигналов, а именно

$$\text{если} \quad x[n] = \sum_k a_k x_k[n], \quad \text{то} \quad y[n] = \sum_k a_k y_k[n], \tag{7}$$

где $y_k[n]$ — реакция системы на поданный сигнал $x_k[n]$.

Используя определение, легко проверить, что системы примеров ИСЗ и скользящего среднего линейны.

2.3.1 Пример: Сумматор

Система, определяемая соотношением

$$y[n] = \sum_{k=-\infty}^{n} x[k], \tag{8}$$

называется *сумматором*, поскольку значение её реакции в момент времени n равно сумме всех предыдущих отсчетов входной последовательности вплоть до n-го. Мы начнем использовать операцию суммирования всерьез, когда мы будем обсуждать цифровые фильтры.

Чтобы доказать линейность сумматора, нам необходимо показать, что он удовлетворяет принципу суперпозиции при любых входных сигналах, не ограничиваясь частными случаями. Подадим на вход сумматора две произвольные последовательности $x_1[n]$ и $x_2[n]$ и вычислим соответствующие отклики:

$$y_1[n] = \sum_{k=-\infty}^{n} x_1[k], \tag{9}$$

$$y_2[n] = \sum_{k=-\infty}^{n} x_2[k]. \tag{10}$$

Если теперь на вход сумматора подать сигнал $x_3[n] = ax_1[n] + bx_2[n]$, то по принципу суперпозиции вне зависимости от выбранных констант a и b мы должны получить равенство $y_3[n] = ay_1[n] + by_2[n]$. Проверим это, отталкиваясь от определения (8).

$$\begin{aligned} y_3[n] = \sum_{k=-\infty}^{n} x_3[k] = \sum_{k=-\infty}^{n} (ax_1[n] + bx_2[n]) = \\ = a \sum_{k=-\infty}^{n} x_1[k] + b \sum_{k=-\infty}^{n} x_2[k] = ay_1[n] + by_2[n]. \end{aligned} \tag{11}$$

Итак, сумматор действительно удовлетворяет принципу суперпозиции, т. е. является линейной системой.

В общей ситуации доказать, что данная система не является линейной (если она и вправду нелинейна), гораздо проще, чем обосновать ее линейность (если она все-таки линейна). Для этого достаточно предъявить пару входных последовательностей и констант, для которых нарушается принцип суперпозиции.

2.3.2 Пример: Нелинейная система

Рассмотрим систему

$$w[n] = \log_{10} |x[n]|. \tag{12}$$

Эта система нелинейна. Для доказательства нам достаточно найти один контрпример, т. е. пару сигналов, на которой нарушается принцип суперпозиции (6). Возьмем сигналы $x_1[n] = 1$ и $x_2[n] = 10$. Соответствующие отклики – $w_1[n] = 0$ и $w_2[n] = 1$. Требование однородности в линейных системах в данной ситуации диктует соотношение $w_2[n] = 10w_1[n]$, поскольку $x_2[n] = 10x_1[n]$. Однако у нас это не выполнено. Значит, система (12) нелинейна.

2.4 Стационарные системы

К *стационарным* (или *инвариантым во времени*) относят системы, для которых временной сдвиг (или задержка) входной последовательности индуцирует соответствующий сдвиг выходной последовательности. Более формально определение выглядит так. Пусть дискретная система определена соотношением $y[n] = T\{x[n]\}$. Она называется стационарной, если для любой входной последовательности $x[n]$ и произвольного целого числа n_0 выполнено соотношение $T\{x[n-n_0]\} = y[n-n_0]$. Стационарные системы иногда еще называют системами, *инвариантными относительно сдвигов*.

Как и в случае с линейными системами, доказательство стационарности данной системы требует общего рассуждения, не допускающего каких-либо специфических предположений относительно входных сигналов. Все системы из приведённых ранее примеров стационарны.

2.4.1 Стационарность сумматора

Рассмотрим сумматор из примера раздела 2.3.1. Положим $x_1[n] = x[n - n_0]$. Для доказательства стационарности сумматора нам следует вычислить $y[n - n_0]$, $y_1[n]$ и сравнить результаты. По определению сумматора имеем

$$y[n - n_0] = \sum_{k=-\infty}^{n-n_0} x[k]; \tag{13}$$

$$y_1[n] = \sum_{k=-\infty}^{n} x_1[k] = \sum_{k=-\infty}^{n} x[k - n_0]. \tag{14}$$

Поменяв в последней сумме параметр суммирования на $k_1 = k - n_0$, получим

$$y_1[n] = \sum_{k=-\infty}^{n-n_0} x[k_1] = y[n - n_0]. \tag{15}$$

Таким образом, сумматор действительно является стационарной системой.

Следующий пример знакомит с нестационарной системой.

2.4.2 Пример: Компрессор

Система, определенная соотношением

$$y[n] = x[Mn], \quad -\infty < n < \infty, \tag{16}$$

где M — натуральное число, называется *уплотнителем* или *компрессором*. Вольно говоря, эта система отбрасывает $M-1$ из каждых M отсчетов входной последовательности, оставляя только M-й. Она, конечно, нестационарна, в чем легко убедиться, рассмотрев реакцию $y_1[n]$ системы на входной сигнал $x_1[n-n_0]$. Если бы наша система была стационарна, то выполнялось бы равенство $y_1[n] = y[n - n_0]$. Однако

$$y_1[n] = x_1[Mn] = x[Mn - n_0] \neq y[n - n_0] = x[M(n - n_0)]. \tag{17}$$

Другой способ проверки нестационарности системы состоит в предъявлении контрпримера, т. е. входной последовательности, на которой нарушается условие стационарности. Пусть, например, $M = 2$, $x[n] = \delta[n]$ и $x_1[n] = \delta[n - 1]$. Тогда $y[n] = \delta[Mn] = \delta[n]$, но $y_1[n] = \delta[Mn - 1] = 0$. Следовательно, $y_1[n] \neq y[n-1]$, что противоречит условию стационарности.

2.5 Детерминированность

Систему называют *детерминированной*, если член входной последовательности с номером n_0 зависит только от тех членов входной последовательности, номер которых не превышает n_0. Это условие влечет тот факт, что если $x_1[n] = x_2[n]$ при $n \leqslant n_0$, то $y_1[n] = y_2[n]$ при $n \leqslant n_0$. Система примера ИСЗ детерминированная, если $n_d \geqslant 0$, и не является таковой при $n_d < 0$. Система примера скользящего среднего детерминированная при $-M_1 \geqslant 0$ и $M_2 \geqslant 0$. В противном случае она недетерминированная. Система примера без запомнинания (раздел 2.2.1) детерминированная, как и сумматор из раздела 2.3.1 и нелинейная система примера из раздела 2.3.2. А вот система примера из раздела 2.4.2 — недетерминированная при $M > 1$, поскольку $y[1] = x[M]$. Еще одна недетерминированная система приведена в следующем примере.

2.5.1 Правая и левая разностные системы

Рассмотрим *правую разностную систему*, определенную по правилу

$$y[n] = x[n + 1] - x[n]. \tag{18}$$

Она не является детерминированной, так как каждый из членов выходной последовательности с номером n вычисляется как по $x[n]$, так и по $x[n+1]$. Нарушение детерминированности легко заметить, взяв в качестве входных последовательностей $x_1[n] = \delta[n-1]$ и $x_2[n] = 0$. Соответствующие отклики – $y_1[n] = \delta[n] - \delta[n-1]$ и $y_2[n] = 0$. Заметим, что $x_1[n] = x_2[n]$ при $n \leqslant 0$. Поэтому, по определению детерминированности, должно выполняться равенство $y_1[n] = y_2[n]$ для всех $n \leqslant 0$, которое нарушается при $n = 0$. Итак, предъявив контрпример, мы показали, что правая разностная система недетерминированная.

Левая разностная система определяется как

$$y[n] = x[n] - x[n-1]. \tag{19}$$

Каждый член выходной последовательности этой системы зависит от члена с тем же номером входной последовательности и одного предыдущего. Значит, она является детерминированной.

2.6 Устойчивость

Говорят, что система *устойчива*, если и только если её реакция на любой ограниченный по амплитуде сигнал ограничена. Напомним, что последовательность $x[n]$ называется ограниченной, если найдется такое конечное положительное число B_x, что

$$\forall n \qquad |x[n]| \leqslant B_x < \infty. \tag{20}$$

Таким образом, в устойчивой системе для каждой ограниченной входной последовательности найдется такая положительная константа B_y, что

$$\forall n \qquad |y[n]| \leqslant B_y < \infty. \tag{21}$$

Важно осознать, что свойство, которое мы здесь определили, – это свойство системы, а не входных последовательностей. Иначе говоря, мы вполне можем предъявить пары сигнал-отклик, обладающие указанным свойством. Но наличие таких пар еще не означает устойчивости системы. Данная система будет устойчива только в том случае, когда этим свойством обладают все пары. Например, для неустойчивой системы мы можем найти ряд ограниченных сигналов, на которые наша система дает ограниченные отклики, однако результат применения устойчивой системы к любой ограниченной входной последовательности должен быть ограниченным. Поэтому, как только нам удалось найти хотя бы одну ограниченную последовательность, отклик на которую будет неограниченным, можно с уверенностью заключить, что эта система неустойчива. В следующем примере проверяется устойчивость уже знакомых нам систем.

2.6.1 Пример: Проверка устойчивости систем

Система примера из раздела 2.2.1 устойчива. Для доказательства этого факта предположим, что на вход системы подается сигнал $x[n]$ с верхней

границей B_x, т.е. $|x[n]| \leqslant B_x$ для всех n. Тогда $|y[n]| = |x[n]|^2 \leqslant B_x^2$. Таким образом, $B_y = B_x^2$ – верхняя граница реакции системы $y[n]$, доказывающая ее ограниченность. Следовательно, проверяемая система устойчива.

Используя аналогичные рассуждения, можно доказать устойчивость систем примеров ИСЗ, скользящего среднего, примеров 2.4.2 и 2.5.1.

С другой стороны, легко убедиться в неустойчивости сумматора, введённого формулой (8). Рассмотрим, например, последовательность $x[n] = u[n]$. Ее ограниченность не вызывает сомнений, поскольку $B_x = 1$. Отклик сумматора на такой сигнал имеет вид

$$y[n] = \sum_{k=-\infty}^{n} u[k] = \begin{cases} 0, & n < 0, \\ n+1, & n \geqslant 0 \end{cases}. \tag{22}$$

Последовательность $y[n]$ неограничена, поскольку при любом выборе константы B_y неравенство $n+1 \leqslant B_y < \infty$ нарушается при достаточно больших n. Следовательно, сумматор является неустойчивой системой.

3 Линейные стационарные системы

Особое значение имеет класс систем, являющихся одновременно как линейными, так и стационарными. Наличие этих свойств позволяет представить системы в удобном виде. Более того, они играют ведущую роль в приложениях обработки сигналов. Класс линейных систем определяется с помощью принципа суперпозиции уравнением (6). Учитывая свойство линейности и представление общей последовательности в виде линейной комбинации сдвинутых импульсов, можно заметить, что линейная система полностью определяется своей реакцией на сдвинутые импульсные последовательности. Более точно, пусть $h_k[n]$ – реакция системы на $\delta[n-k]$. Тогда согласно представлению дискретных сигналов через сумму единичных импульсов:

$$y[n] = T\left\{\sum_{k=-\infty}^{\infty} x[k]\delta[n-k]\right\}. \tag{23}$$

По принципу суперпозиции (6) можно записать

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]T\{\delta[n-k]\} = \sum_{k=-\infty}^{\infty} x[k]h_k[n]. \tag{24}$$

Итак, мы получили, что реакция линейной системы на любую входную последовательность выражается в терминах откликов системы на сигналы $\delta[n-k]$. Если система всего лишь линейна, то отсчёты $h_k[n]$ зависят как от k, так и от n, и помощь, оказываемая уравнением (24) при вычислениях, относительно невысока. Однако можно получить более полезный результат, если к свойству линейности добавить условие стационарности.

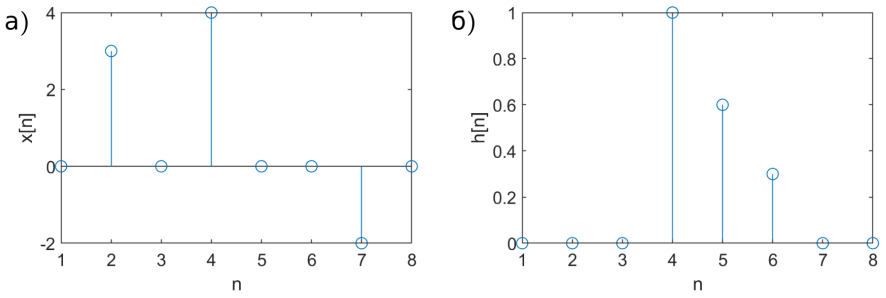

Рис. 7: Исходный сигнал $x[n]$ и импульсная характеристика дискретной системы $h[n]$

Свойство стационарности влечет, что если $h[n]$ — реакция системы на $\delta[n]$ (называемая *импульсной характеристикой системы*), то ее реакция на сигнал $\delta[n-k]$ равна $h[n-k]$. Опираясь на этот факт, уравнение (24) можно переписать в виде

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{25}$$

Как следствие этой формулы, отметим, что линейная стационарная система (ЛС-система) полностью определяется своей импульсной характеристикой $h[n]$ в том смысле, что при известной последовательности $h[n]$, опираясь на (25), можно вычислить отклик $y[n]$ на *любой* поданный сигнал $x[n]$.

Если отсчеты последовательности $y[n]$ зависят от отсчетов $h[n]$ и $x[n]$ по правилу (25), то последовательность $y[n]$ называют (дискретной) сверткой последовательностей $h[n]$ и $x[n]$ и употребляют обозначение:

$$y[n] = x[n] * h[n]. \tag{26}$$

Операция дискретной свёртки строит последовательность $y[n]$ по двум данным последовательностям $x[n]$ и $h[n]$. Уравнение (25) выражает каждый отсчёт выходной последовательности через все отсчёты входной последовательности и импульсную характеристику.

Формула (25) наводит на мысль, что отсчет входной последовательности с номером $n = k$, представленный как $x[k]\delta[n-k]$, преобразуется системой в выходную последовательность $x[k]h[n-k]$ для $-\infty < n < \infty$, и что для каждого k эти последовательности комбинируются для формирования абсолютно всех выходных последовательностей.

Рассмотрим простой входной сигнал с тремя ненулевыми отсчётами, изображенный на рис. 7(а).

```
1 x = [0  3  0  4  0  0  -2  0];
```

Импульсная характеристика системы представлена на рисунке 7(б) и имеет три ненулевых отсчета 4, 5 и 6:

```
h = [0  0  0  1  0.6  0.3  0  0];
```

Согласно представлению сигнала в виде разложения по единичным импульсам, $x[n]$ можно представить как сумму трёх последовательностей: $x[2]\delta[n-2]$, $x[4]\delta[n-4]$ и $x[7]\delta[n-7]$, представляющих три ненулевых отсчета последовательности $x[n]$. Соответствующие векторы в изложении на М-языке:

```
x2 = [0  3  0  0  0  0  0  0];
x4 = [0  0  0  4  0  0  0  0];
x7 = [0  0  0  0  0  0 -2  0];
```

Последовательности $x[2]h[n-2]$, $x[4]h[n-4]$ и $x[7]h[n-7]$, изображенные на рис. 8(б,г,е), – отклики системы на входные сигналы $x[2]\delta[n-2]$, $x[4]\delta[n-4]$ и $x[7]\delta[n-7]$ соответственно. В коде мы вычисляем эти последовательности как

```
y2 = x(2)*[0  0  h];
y4 = x(4)*[0  0  0  0  h];
y7 = x(7)*[zeros(1,7)  h];
```

обеспечивая сдвиги в h помещением соответствующего сдвигу числа нулей в начало вектора, на который производится умножение. После этого реакция системы на сигнал $x[n]$ получается в виде суммы этих индивидуальных откликов:

```
y = [y2  zeros(1,5)]  + [y4  zeros(1,3)] + y7;
```

Здесь конкатенация с дополнительными нулями используется для того, чтобы выровнять длины складываемых векторов.

Для вычисления свертки двух последовательностей MATLAB располагает встроенной функцией `conv`, на вход которой передаются два вектора. Так, рассмотренное выше вычисление осуществимо строкой

```
conv(x,h)
```

Длина получаемой свёртки составляет `length(x)+length(h)-1`, что в нашем случае равно 15.

Рассматривая процесс вычисления свертки и обращая внимание на индексы, можно отметить:

- в общем случае, при свертке двух последовательностей неважно, какую мы называем сигналом и какую импульсной характеристикой, поэтому неважно какую последовательность дополним нулями;
- всегда можно переиндексировать последовательность, дополненную нулями, так, чтобы не было отрицательных индексов;
- суммирование произведений соответствует умножению вектора на строку матрицы.

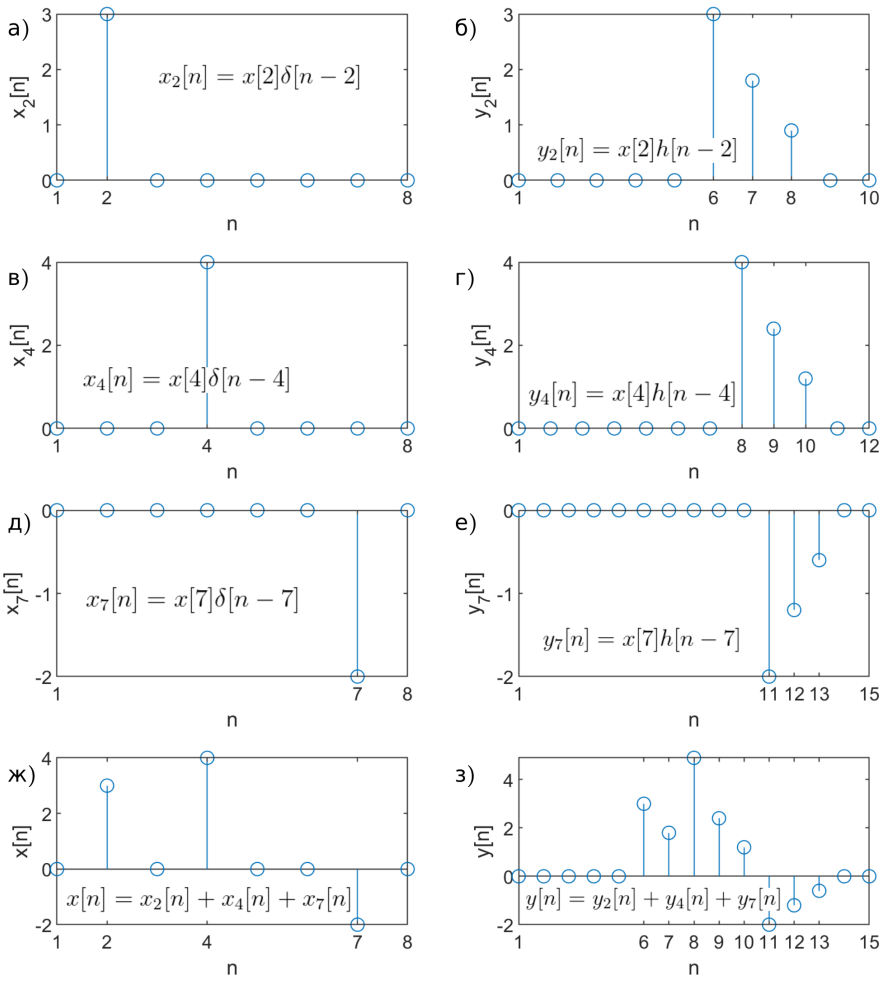

Рис. 8: Представление входной последовательности линейной стационарной системы в виде суперпозиции ответов на индивидуальные отсчёты и вычисление выходного сигнала.

Хотя свёртка из теории непрерывных линейных систем определяется с помощью интеграла, дискретную свёртку не следует воспринимать как аппроксимацию интегральной свертки. Последняя свертка играет в основном теоретическую роль в непрерывных линейных системах, в то время как дискретная свёртка, в дополнение к своей теоретической ценности, часто служит для явной реализации дискретных линейных систем. Поэтому очень важно наработать некоторую интуицию в понимании свойств дискретной свёртки в реальных вычислениях.

Изложенная интерпретация уравнения (25) заключается в том, что дискретная свёртка является прямым следствием линейности и стационарности системы. Однако несколько иной взгляд на эту формулу подводит нас к весьма важной вычислительной интерпретации. Когда мы смотрим на соотношение (25) как на формулу, вычисляющую отдельный отсчет выходной последовательности, замечаем, что $y[n]$ (т. е. n-й член входной последовательности) получается в результате умножения входной последовательности (записанной как функция от k) на последовательность $h[n-k]$, $-\infty < k < \infty$, а затем при каждом фиксированном n суммируются все произведения $x[k]h[n-k]$ с параметром k в качестве параметра суммирования. Следовательно, при свёртке двух последовательностей для вычисления n-го члена результата используются все отсчёты обеих последовательностей. Ключ к полному пониманию формулы (25) состоит в осознании процесса образования последовательности $h[n-k]$, $-\infty < k < \infty$, для всех значений n, которые представляют интерес. Чтобы закончить это осмысление свёртки, полезно заметить, что

$$h[n-k] = h[-(k-n)]. \tag{27}$$

Интерпретацию этой формулы лучше всего рассмотреть на примере.

Предположим, что $h[k]$ — последовательность, изображенная на рис. 9(а), а нам нужно найти $h[n-k] = h[-(k-n)]$. Определим $h_1[k]$ как $h[-k]$ (рис. 9(б). Затем зададим $h_2[k]$ как последовательность $h_1[k]$, задержанную на n отсчетов по оси k, т.е. $h_2[k] = h_1[k-n]$. На рис. 9(в) изображена последовательность, полученная из последовательности рисунка 9(б) сдвигом на n отсчетов вправо. Опираясь на связь между $h_1[k]$ и $h[k]$, можно показать, что $h_2[k] = h_1[k-n] = h[-(k-n)] = h[n-k]$, и, таким образом, нижняя последовательность рисунка является искомым сигналом. Подводя итог, скажем, что для вычисления $h[n-k]$ нам нужно обратить по времени последовательность $h[k]$ относительно $k=0$, а затем сдвинуть полученный результат на n отсчетов вправо.

Из рассмотренного примера должно стать ясно, что в общей ситуации последовательность $h[n-k]$, $\infty < k < \infty$, получается в два этапа:

1. зеркальным отражением последовательности $h[k]$ относительно нуля для получения $h[-k]$

2. сдвигом отраженной последовательности вправо на n отсчетов.

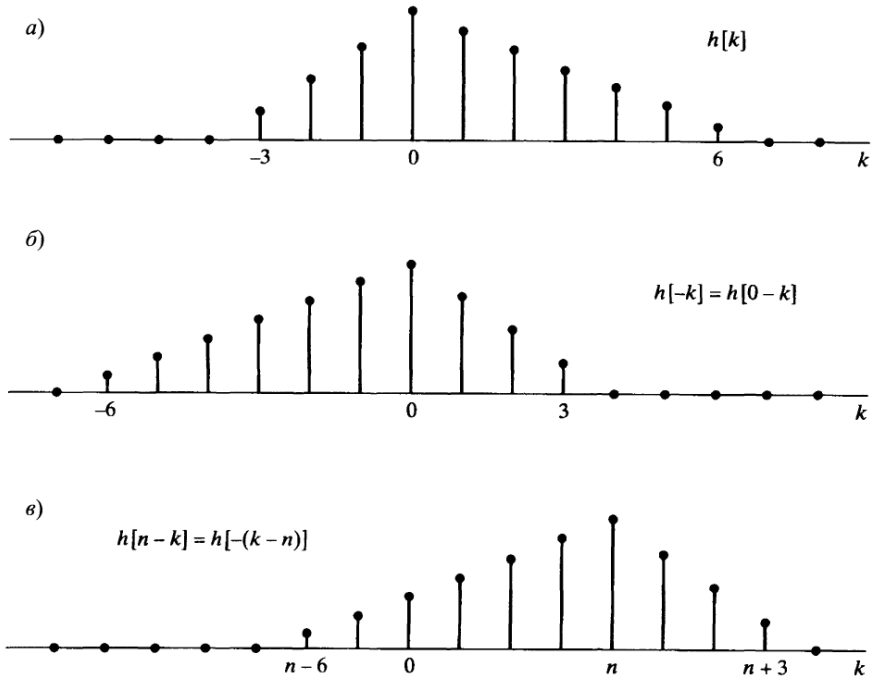

Рис. 9: Получение последовательности $h[n-k]$: а) последовательность $h[k]$; б) последовательность $h[-k]$; в) последовательность $h[-(k-n)]$ при $n=4$.

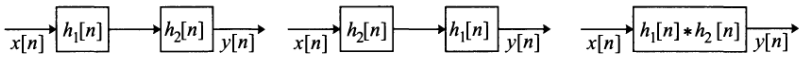

Рис. 10: Три линейных стационарных системы с одной и той же импульсной характеристикой.

Как уже отмечалось, для вычисления свертки последовательности $x[k]$ и $h[n-k]$ перемножаются, а затем произведения суммируются и получается отсчет $y[n]$ выходной последовательности. Чтобы найти другой отсчёт свёртки, начало отсчёта (нулевой момент времени) последовательности $h[-k]$ сдвигается на новую позицию и процесс повторяется. Эта процедура применяется как при численной обработке отсчётов, полученных при дискретизации сигналов, так и при аналитических вычислениях.

4 Свойства линейных стационарных систем

Поскольку линейные стационарные системы описываются свёрткой, свойства этого класса систем определяются свойствами дискретной свертки. Следовательно, импульсная характеристика конкретной системы содержит всю полноту информации.

Некоторые общие свойства класса линейных стационарных систем можно обнаружить, изучая свойства операции свертки. Например, свертка – коммутативная операция:

$$x[n] * h[n] = h[n] * x[n]. \tag{28}$$

Таким образом, порядок последовательностей в свертке несущественен, т. е. реакция системы не изменится, если поменять местами входную последовательность и отклик системы на еди- ничный импульс. Иными словами, отклик системы на сигнал $x[n]$ при импульсной характеристике $h[n]$ будет тем же, что и реакция на сигнал $h[n]$ при импульсной характеристике $x[n]$.

Свёртка удовлетворяет свойству дистрибутивности относительно сложения, а именно

$$x[n] * (h_1[n] + h_2[n]) = x[n] * h_1[n] + x[n] * h_2[n] \tag{29}$$

Это свойство непосредственно вытекает из формулы (6) и является прямым следствием линейности и коммутативности свертки.

При *каскадном соединении* (последовательном) систем реакция первой системы подается на вход второй, отклик второй — на вход третьей и т. д. Отклик последней системы служит реакцией всей цепочки систем. Две линейные стационарные системы, подключенные последовательно, образуют одну линейную стационарную систему, чья импульсная характеристика совпадает со свёрткой импульсных характеристик обеих систем, что иллюстрирует рис. 10. На первом блоке диаграмм рисунка реакция первой

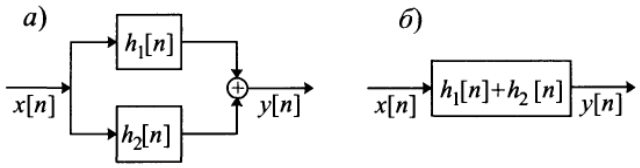

Рис. 11: а) параллельное соединение линейных стационарных систем; б) система, эквивалентная системе а).

системы на $x[n] = \delta[n]$ равна $h_1[n]$. Поэтому выходная последовательность из второй системы (и, по определению, импульсная характеристика каскада систем) должна быть равна

$$h[n] = h_1[n] * h_2[n]. \tag{30}$$

Как следствие коммутативности свертки импульсная характеристика каскада линейных стационарных систем не зависит от порядка, в котором они подключаются друг к другу. Этот факт отражен на рис. 10, где три системы имеют одну и ту же импульсную характеристику.

При параллельном соединении системы имеют общий вход, а их выходные последовательности складываются и дают реакцию всего соединения. Как следует из дистрибутивности свёртки, параллельное соединение двух линейных стационарных систем можно заменить одной линейной стационарной системой, чья импульсная характеристика равна сумме характеристик компонент соединения (рис. 11), т. е.

$$h[n] = h_1[n] + h_2[n]. \tag{31}$$

Требования линейности и стационарности выделяют класс систем с весьма специфическими свойствами. Устойчивость и детерминированность предоставляют вспомогательные свойства, и часто бывает важно знать, является ли данная линейная стационарная система устойчивой или детерминированной. Напомним (раздел 2.6), что устойчивой системой называют систему с ограниченной реакцией на каждый ограниченный входной сигнал. Линейная стационарная система является устойчивой тогда и только тогда, когда её импульсная характеристика — абсолютно суммируемая последовательность, т.е. если

$$S = \sum_{k=-\infty}^{\infty} |h[k]| < \infty. \tag{32}$$

Это является достаточным и необходимым условием устойчивости.

Детерминированная система определена в подразделе 2.5 как система, отсчёт $y[n_0]$ реакции которой зависит только от отсчетов $x[n]$ сигнала с $n \leqslant n_0$. Ввиду 6 и из $y[n] = h[n] * x[n]$ это определение влечет утверждение: импульсная характеристика детерминированной линейной стационарной системы должна удовлетворять условию

$$h[n] = 0, \quad n < 0. \tag{33}$$

По этой причине последовательность с нулевыми членами $h[n]$ при $n < 0$ удобно иногда называть детерминированной, подразумевая, что она является откликом на единичный импульс детерминированной линейной стационарной системы.

Хотя для нелинейных или нестационарных систем импульсная характеристика тоже может быть вычислена, она представляет лишь ограниченный интерес, поскольку формула свертки, а также условия (6) и (32)), обеспечивающие устойчивость и детерминированность, не применимы к таким системам.

Для иллюстрации свойств линейной стационарной системы в ответе на единичный импульс вновь обратимся к рассмотренным ранее примерам таких систем. Найдём сначала реакцию систем на единичный импульс. Сделать это можно, опираясь непосредственно на формулы, задающие соответствующие системы.

Идеальная система задержки:

$$h[n] = \delta[n - n_d], \tag{34}$$

где n_d - фикисированное натуральное число.

Скользящее среднее:

$$h[n] = \frac{1}{M_1 + M_2 + 1} \sum_{k=-M_1}^{M_2} \delta[n-k] = \begin{cases} \dfrac{1}{M_1 + M_2 + 1}, & M - 1 \leqslant n \leqslant M_2, \\ 0, & \text{в других случаях.} \end{cases} \tag{35}$$

Сумматор:

$$h[n] = \sum_{k=-\infty}^{n} \delta[k] = \begin{cases} 1, & n \geqslant 0, \\ 0, & n < 0 \end{cases} = u[n]. \tag{36}$$

Правая разностная система:

$$h[n] = \delta[n+1] - \delta[n]. \tag{37}$$

Левая разностная система:

$$h[n] = \delta[n] - \delta[n-1]. \tag{38}$$

Зная импульсные характеристики стационарных систем, мы можем проверить системы на устойчивость, вычисляя сумму S.

Ясно, что для систем идеальной задержки, скользящего среднего, правой и левой разностных систем указанная сумма меньше бесконечности, поскольку в этих случаях импульсная характеристика имеет лишь конечное число ненулевых отсчетов. Такие системы называют *системами с конечной импульсной характеристикой* (КИХ-системами). Ясно, что КИХ-системы будут устойчивыми тогда и только тогда, когда их отклики на единичный импульс конечны по абсолютной величине. С другой стороны, сумматор не является устойчивой системой, так как

$$S = \sum_{n=0}^{n} u[n] = \infty. \tag{39}$$

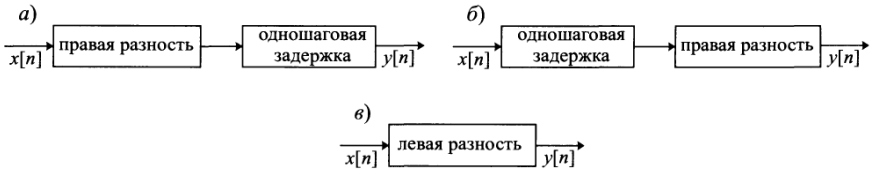

Рис. 12: Эквивалентные системы, получающиеся друг из друга благодаря коммутативности свертки.

Ранее при проверки устойчивости мы уже доказали неустойчивость сумматора, предъявив пример ограниченного сигнала (единичный скачок), реакция на который неограничена.

Импульсная характеристика сумматора неограничена по длительности. Сумматор – пример класса систем, которые называют системами с *бесконечной импульсной характеристикой* (БИХ-системами). Примером устойчивой БИХ-системы служит такая система, импульсная характеристика которой имеет вид $h[n] = a^n u[n]$ с $|a| < 1$. В этом случае

$$S = \sum_{n=0}^{\infty} |a|^n. \tag{40}$$

Если $|a| < 1$, то формула суммы членов бесконечной геометрической прогрессии даёт

$$S = \frac{1}{1 - |a|} < \infty \tag{41}$$

С другой стороны, если $|a| \geqslant 1$, то сумма соответствующего ряда бесконечна и система неустойчива.

Для проверки детерминированности линейных стационарных систем нам нужно убедиться, что $h[n] = 0$ при $n < 0$. Как было выяснено, ИСЗ детерминирована при $n_d \geqslant 0$. Если $n_d < 0$, эта система недетерминирована. В случае скользящего среднего свойство детерминированности накладывает ограничения: $-M_1 \geqslant 0$ и $M_2 \geqslant 0$. Сумматор и левая разностная система являются детерминированными системами, а правая разностная система недетерминирована.

Концепция свертки как операции над двумя последовательностями подводит нас к упрощению многих задач, связанных с системами. Особенно полезный результат можно сформулировать относительно ИСЗ. Поскольку отклик этой системы имеет вид $y[n] = x[n - n_d]$, а её импульсная характеристика равна $h[n] = \delta[n - n_d]$, то

$$x[n] * \delta[n - n_d] = \delta[n - n_d] * x[n] = x[n - n_d]. \tag{42}$$

Иными словами, свертка сдвинутого на n_d позиций единичного импульса с сигналом $x[n]$ приводит к задержке сигнала на то же число n_d.

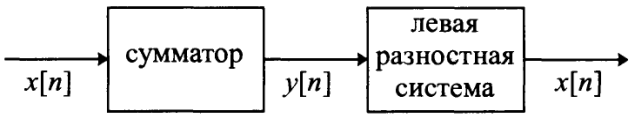

Рис. 13: Каскад сумматора и левой разностной системы.

Так как задержка является одной из основных операций при реализации линейных систем, предыдущий результат полезен при анализе и упрощении комплексов линейных стационарных систем. В качестве примера рассмотрим систему, изображенную на рис. 12(а), которая состоит из правой разностной системы, последовательно соединенной с одношаговой задержкой.

Ввиду коммутативности свёртки порядок подключения систем в последовательной цепи несущественен, поскольку они линейны и стационарны. Следовательно, поменяв местами одношаговую задержку и правую разностную систему (рис. 12(а)), мы получим ту же результирующую систему (рис. 12(б)). Кроме того, формула (30) влечет, что импульсная характеристика всей цепи систем совпадает со свёрткой импульсных характеристик каждой из составных частей. Поэтому

$$h[n] = (\delta[n+1] - \delta[n]) * \delta[n-1] = \delta[n-1] * (\delta[n+1] - \delta[n]) \tag{43}$$

$$= \delta[n] - \delta[n-1]. \tag{44}$$

Мы видим, что $h[n]$ тождественна импульсной характеристике левой разностной системы, т.е. цепи систем, изображенные на рис. 2.13, а) и б), можно заменить левой разностной системой, как показано на рис. 12(в).

Заметим, что недетерминированная правая разностная система, будучи последовательно соединена с задержкой (рис. 12(а) и (б)), превращается в детерминированную систему. В общем случае недетерминированная КИХ-система может быть преобразована в детерминированную систему, если к ней присоединить достаточно большую задержку.

Другой пример каскада систем вводит понятие инверсивной системы. Рассмотрим каскад систем (рис. 13). Его импульсная характеристика вычисляется по правилу:

$$h[n] = u[n] * (\delta[n] - \delta[n-1]) = u[n] - u[n-1] = \delta[n]. \tag{45}$$

То есть каскадная комбинация сумматора с левой разностной системой (в любом порядке) дает систему, чья импульсная характеристика совпадает с единичным импульсом. Таким образом, выход этой каскадной комбинации будет всегда совпадать со входом, поскольку $x[n]*\delta[n] = x[n]$. В этом случае действие левой разностной системы полностью компенсируется (или инвертируется) действием сумматора. При этом говорят, что левая разностная система — инверсивная система к сумматору.

Благодаря коммутативности свёртки сумматор служит инверсивной системой для левой разностной системы. В общем случае, если импульсная

характеристика линейной стационарной системы – $h[n]$, то её инверсивная система (если она существует) реагирует на единичный импульс последовательностью $h_i[n]$, определенной соотношением

$$h[n] * h_i[n] = h_i[n] * h[n] = \delta[n]. \tag{46}$$

Инверсивные системы полезны в ситуациях, когда необходимо компенсировать эффект линейной системы. В общем случае разрешить уравнение (46) относительно $h_i[n]$ очень трудно. Однако в следующих лекциях мы увидим, что z-преобразование даёт прямой метод построения инверсивной системы.

A Получение дистрибутива MATLAB

Университетом ИТМО была приобретена единая кампусная лицензия на MATLAB и Simulink для всех студентов, преподавателей и научных сотрудников, позволяющая работать как в режиме онлайн, так и в режиме офлайн с возможностью установки на любое устройство.

Получить дистрибутив довольно просто. Как это сделать?

1) Необходимо зарегистрироваться на сайте Mathworks.com, также полезной будет регистрация и на сайте официального дистрибьютора Mathworks в России — exponenta.ru, указав в качестве адреса электронной почты корпоративный адрес на одном из доменов Университета: @itmo.ru, @niuitmo.ru или @ifmo.ru.

2) После регистрации ваша учётная запись будет автоматически аффилирована с Университетом ИТМО, и у вас появится возможность скачать уже активированный установочный файл дистрибутива, запросив лицензию нажатием на главной странице на кнопку "Get MATLAB а далее на "Get Campus Software".

3) Выданную лицензию можно будет увидеть, перейдя в раздел "My Account". Там же можно будет скачать дистрибутив или запустить MATLAB онлайн.

Каждый зарегистрированный и прошедший процедуру подтверждения пользователь может устанавливать на свои устройства до 5 копий MATLAB — установочные файлы индивидуальны для каждого пользователя. Установка большего числа копий приведет к блокировке вашей учётной записи, поэтому не рекомендуема.

B Листинги программ

1. Код для осуществления аналого-цифрового преобразования синусоидального входного сигнала.

```
A = 0; B = 16; %интервал времени
T = 1.2; %интервал дискретизации

t = A:0.01:B;      %для построения
x = cos(pi*t/4); %входного сигнала

tdiscr = A:T:B;
xdiscr = cos(pi*tdiscr/4);

xdigital = round(xdiscr, 1);

%преобразование чисел
xdigital_int64 = typecast(xdigital, 'int64'); % сдвойной
xdigital_bin = dec2bin(xdigital_int64);        %точностью в
%двоичные числа
```

```
16
17 %получение двоичных
18 levels = (1:length(-1:0.1:1)) - find(-1:.1:1==0); %
       номеров уровней
19 levels_bin_wosign = dec2bin([10:-1:1 0 1:10]); %
       квантования
20 %со знаком
21
22 sign_bit = num2str((sign(10:-1:-10)>0)'); %добавление
       знакового
23 levels_bin_wsign = cat(2, sign_bit, levels_bin_wosign); %
       бита в
24 %массив
25
26 figure(); %построение графика
27 hold on; box on;
28 plot(t,x, '--');
29 stem(tdiscr,xdiscr);
30 stem(tdiscr,xdigital);
31
32 yyaxis right; %добавление
33 ylim([-10 10]) %второй осиординат ,
34 yticks([-10:10]) %содержащей двоичные
35 yticklabels(levels_bin_wsign) %значения
36 %уровней квантования
```

2. Полный листинг программы для поэтапного вычисления и демонстрации операции дискретной свёртки:

```
1 x = [0 3 0 4 0 0 -2 0];
2 h = [0 0 0 1 0.6 0.3 0 0];
3
4 x2 = [0 3 0 0 0 0 0 0];
5 y2 = x(2)*[0 0 h];
6
7 x4 = [0 0 0 4 0 0 0 0];
8 y4 = x(4)*[0 0 0 0 h];
9
10 x7 = [0 0 0 0 0 0 -2 0];
11 y7 = x(7)*[zeros(1,7) h];
12
13 xn = x2 + x4 + x7;
14 y = [y2 zeros(1,5)] + [y4 zeros(1,3)] + y7;
15
16 figure;
17 subplot(1,2,1);
18
```

```
stem(x);
xlabel('n');
ylabel('x[n]');
xlim([1 inf]);

subplot(1,2,2);
stem(h);
xlabel('n');
ylabel('h[n]');
xlim([1 inf]);

figure;
subplot(4,2,1);
stem(x2);
xlabel('n');
ylabel('x_2[n]');
xlim([1 inf]);

subplot(4,2,2);
stem(y2);
xlabel('n');
ylabel('y_2[n]');
xlim([1 inf]);

subplot(4,2,3);
stem(x4);
xlabel('n');
ylabel('x_4[n]');
xlim([1 inf]);

subplot(4,2,4);
stem(y4);
xlabel('n');
ylabel('y_4[n]');
xlim([1 inf]);

subplot(4,2,5);
stem(x7);
xlabel('n');
ylabel('x_7[n]');
xlim([1 inf]);

subplot(4,2,6);
stem(y7);
xlabel('n');
ylabel('y_7[n]');
```

```
65 xlim([1 inf]);
66
67 subplot(4,2,7);
68 stem(xn);
69 xlabel('n');
70 ylabel('x[n]');
71 xlim([1 inf]);
72
73 subplot(4,2,8);
74 stem(y);
75 xlabel('n');
76 ylabel('y[n]');
77 xlim([1 inf]);
```